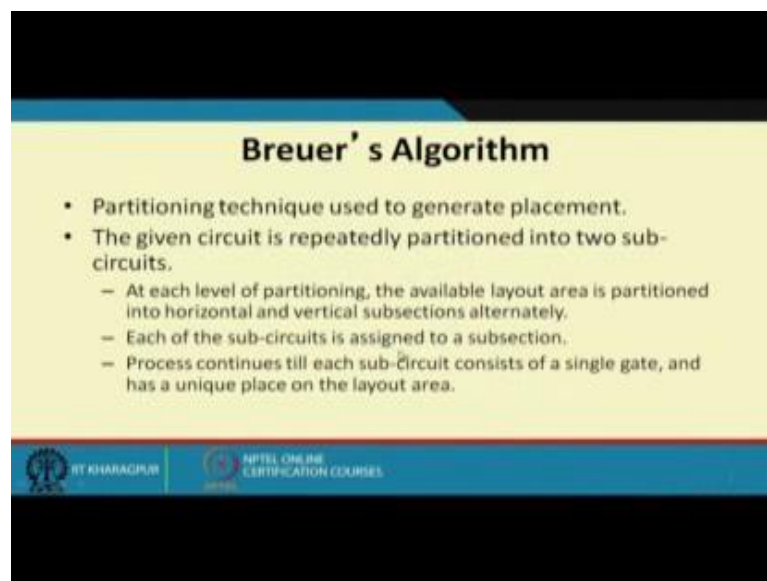


VLSI Physical Design
Prof. Indranil Sengupta
Department of Computer Science and Engineering
Indian Institute of Technology, Kharagpur

Lecture - 14
Placement (Part IV)

So, in this last lecture of the week we shall be looking at some more placement techniques instead of how they work. So, this is lecture 14.

(Refer Slide Time: 00:35)



Breuer's Algorithm

- Partitioning technique used to generate placement.
- The given circuit is repeatedly partitioned into two sub-circuits.
 - At each level of partitioning, the available layout area is partitioned into horizontal and vertical subsections alternately.
 - Each of the sub-circuits is assigned to a subsection.
 - Process continues till each sub-circuit consists of a single gate, and has a unique place on the layout area.

IIIT KHARAGPUR | NPTEL ONLINE CERTIFICATION COURSES

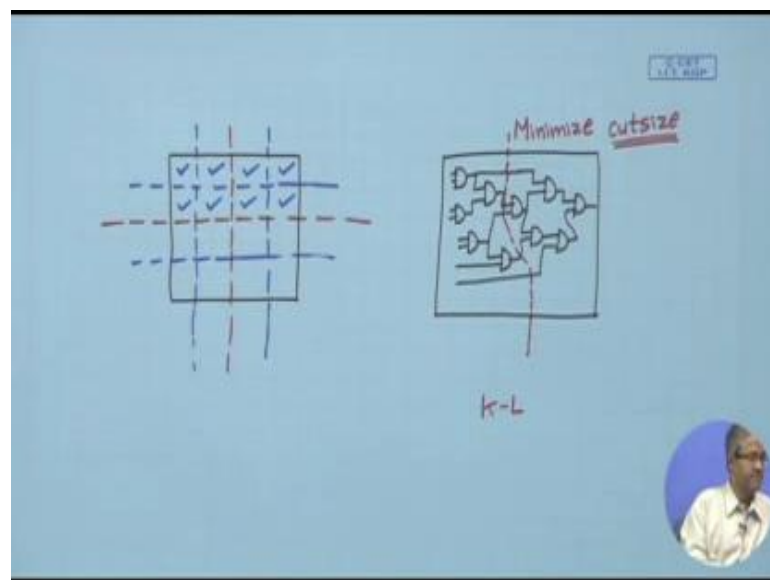
So, we start with an interesting algorithm which is proposed by Melvin Breuer. This is a placement technique which replaces partitioning technique, where the idea is very simple in concept. See normally what we do, we do circuit partitioning, we create the different modules of the blocks then we place the blocks, but here he says that when we have a netlist, we repeatedly do the partitioning that in such a way that at one stage each of the partitions is small enough, so that we can directly place it in a cell. So, partitioning in placement are going hand in hand. So, if we have a 2 dimensional area for placement, which is typically the case for a full custom design style or a gate array design style, then you can use this kind of a placement strategy very effectively.

So, this is a placement strategy, which uses partitioning technique. So, given circuit netlist is repeatedly or progressively partitioned into two or more sub circuits; two sub circuits at every stage because you are using 1 partition 1 slice in a wave (Refer Time:

02:09). So, given a circuit if you make a slice you will get 2 circuits from there. So, at every step you are doing some kind of a you can say bi partitioning divide it up into 2, but a several such partitioning you are doing repeatedly, you are doing a sequence of bi partitioning until you reach a stage where each of the pieces that you create can be directly placed.

So, some of the salient ideas is that at every level of partitioning. So, you use vertical and horizontal slices, so that the available layout area is partitioned into horizontal and vertical subsections alternately. So, what I mean is something like this.

(Refer Slide Time: 02:58)



Suppose I have layout area rectangular in nature. So, suppose I start with a vertical partition like this. I divide it up in 2 pieces, then I use a horizontal partition, I divide it up into 4 pieces, then I again use let say a vertical partition like this, so 2 more pieces are created. Let say again a vertical partition like this, 2 more pieces I created then again an horizontal partition like this, a horizontal partition like this, this I go on repeating.

Now, the sequence of this horizontal and vertical partitions can differ; so there are many such partition strategy 2 of them we shall be seeing through some examples, but the idea is this; you go on partitioning a given rectangular space using vertical and horizontal slices repeatedly, to create the partitions at the end you will be getting these partitions, which can be directly placed, placed on the silicon floor.

So, the basic idea is that the process is continued till let say each of the partition sub circuit is single gate, which has a unique place on the layout area well. Here you can relate this problem to the gate array placement, in a gate array you have an array of gates, here also we are doing partitioning, partitioning, partitioning we arrive at gates. So, each of these gates can be placed into one particular gate in the gate array, right.

(Refer Slide Time: 04:55)

• Several cut-oriented sequences have been proposed.
– Cuts are minimized during partitioning.

• We shall illustrate two alternate cut sequences proposed by Breuer:

1. Quadrature mincut placement
2. Recursive bipartitioning mincut placement

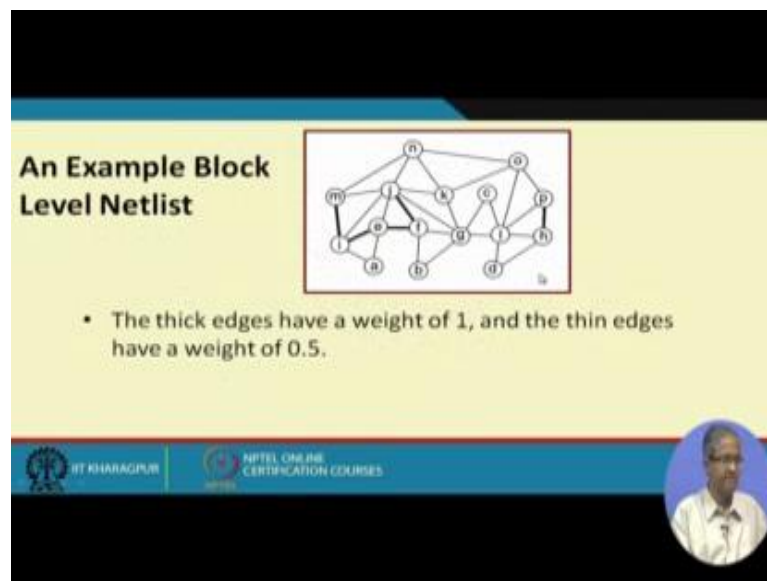
Now, Breuer has proposed several sequences of the horizontal and vertical cuts, these are called cut oriented sequences. Now when these cuts are proposed you see there is an important point, see in this diagram say I have shown these lines, well as if I am cutting the circuit in the middle, but it is not exactly like that. So, what is done is like this I am showing I am illustrating for 1 step, let say let us take a very small example I have some gates, let us take a very small example.

Let say I have a circuit netlist like this, now in this diagram I have said that I start with a vertical cut in the middle, but actually this middle is just for illustrations purposes, it is not exactly the middle I am cutting, what I am doing you see I have a 4 5 6 7 9 10; 10 gates. So, let say I am using Kernighan Lin bipartitioning algorithm, to get a good partition of these gates. So, I do not know which is the best partition, let say this is a good partition let say. So, I have 5 gates on this side, I have 5 gates on other side.

So, my objective is to minimize the size or the cuts the cuts size number of lines which are cutting. Now in this case it is 1 2 3 4 5 lines are cutting, so cuts size is 5. So, here

whenever I am showing these lines as if they are dividing it up into 2, it is not blind division, but you look at the cut size minimization sub problem, you cut it in such a way that the cutsize is minimized right. So, each of these cut oriented sequences at every step, you are actually trying to minimize the cutsize. Now 2 alternate cut sequences are means it shall be illustrated here. In fact, Breuer illustrated and stated a few more, but we are just illustrating 2 of them with example. So, that you know exactly what is being done.

(Refer Slide Time: 07:58)



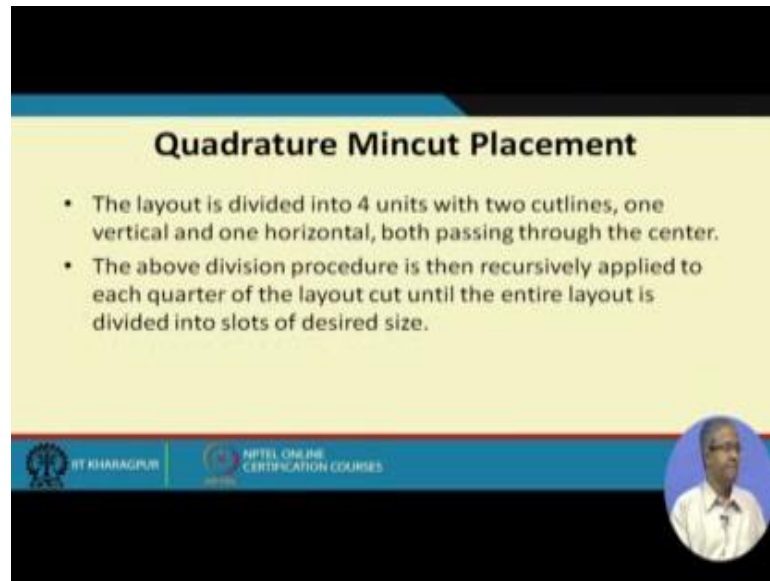
An Example Block Level Netlist

- The thick edges have a weight of 1, and the thin edges have a weight of 0.5.

The slide features a graph with nodes labeled 'a' through 's'. Nodes 'a', 'b', 'c', 'd', 'e', 'f', 'g', 'h', 'i', 'j', 'k', 'l', 'm', 'n', 'o', 'p', 'q', 'r', and 's' are arranged in a roughly circular pattern. Edges connect these nodes, with some edges being thicker than others. The thick edges represent a weight of 1, and the thin edges represent a weight of 0.5. The slide also includes the IIT Kharagpur logo and NPTEL Online Certification Courses logo at the bottom left, and a small circular portrait of a man at the bottom right.

So, this is the example block level netlist we are considering for illustrative purposes well, these each of these circuits can indicate a gate right, and we are assuming that the thick edges; there are 4 thick edges, 5 thick edges have a weight of 1, they are critical in some sense and the thin edges have a weight of half 0.5. So, when you make a cut you will have to estimate the cut size accordingly, right.


(Refer Slide Time: 08:33)



Quadrature Mincut Placement

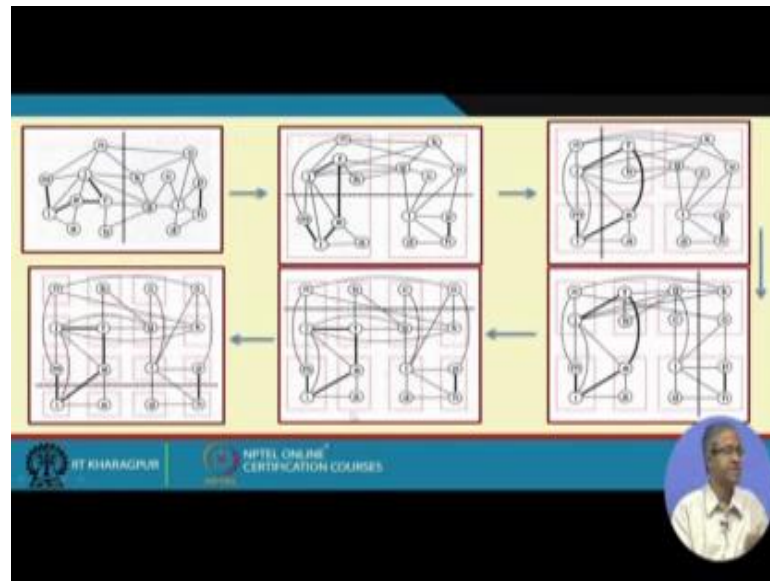
- The layout is divided into 4 units with two cutlines, one vertical and one horizontal, both passing through the center.
- The above division procedure is then recursively applied to each quarter of the layout cut until the entire layout is divided into slots of desired size.

IT KHARAGPUR | NPTEL ONLINE CERTIFICATION COURSES



So, in the first method which is called Quadrature Mincut placement; the idea is to divide the layout into 4 parts with 2 cutlines: 1 vertical, 1 horizontal both passing through the center, just like I have illustrated earlier given a layout you cut vertically, cut horizontally you get 4 pieces. Now when you make a cut as I have said, you do the cut considering the cut size minimization problem, you can use the Kernighan Lin algorithm for the purpose. Now this division procedure is recursively applied to every quarter, see means after these 2 lines you have divided the whole layout into 4 pieces, now each of these 4 pieces you again subject to vertical and horizontal cuts. So, you go on repeating till each of the small pieces are of the desired size. So, let us see with an example.

(Refer Slide Time: 09:38)



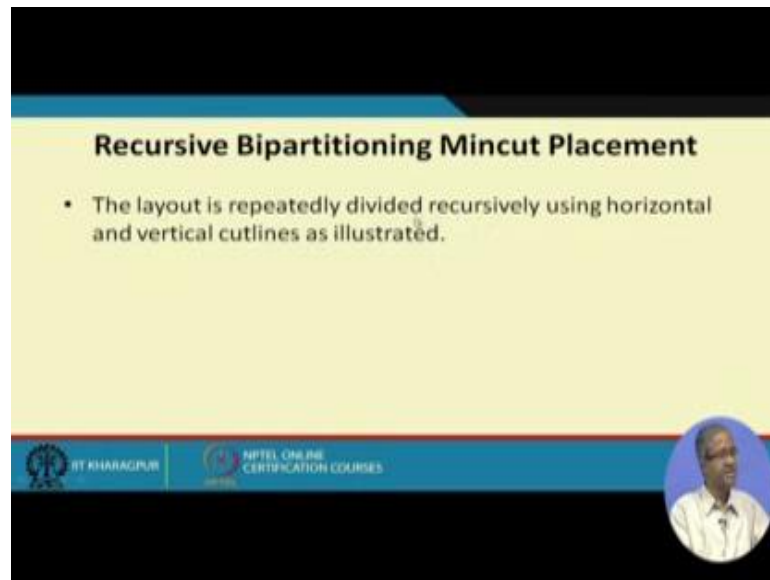
So, this was the example block level netlist. So, we start with this. So, I am showing the solutions only a. So, here these dotted line means I am starting with a vertical cut and in fact, this is the best vertical cut. If you apply Kernighan Lin algorithm this will be the best vertical cut, which is dividing the netlist into 2 parts. So, these 2 parts is shown by these 2 pink rectangles. So, I have already divided 2 netlist into 2 partitions.

Now, each of these partitions you are now subjecting to a horizontal cut, again you apply Kernighan Lin algorithm. So, this will be the best cut, this will be the best cut. So, after doing this cut you divide you have already divided the netlist into 4 parts shown by the pink rectangles. So, again you start with a vertical cut on the left side. So, these 2 rectangular blocks will be divided into 2 and 2. So, let say the cuts are like this. So, the partitions created will be like this, like this m i n e a. Now apply a vertical cut in the right hand side, so these 2 blocks will now get divided similarly; so they will become like this, like this. Similarly you apply a horizontal cut on the top to divide these 4 rectangles and a horizontal cut on the bottom to divide these 4 rectangles. So, finally, you arrive at 16 rectangles where each rectangle contains only 1 gate or 1 node. So, this is my desired final state.

So, you see that initially my graph was haphazard, but after completing it, I have got n here, I have got b here, I have got c here, I have got g here; you see initially b was here initially c was here, but they have got into a position which indicates minimization of the

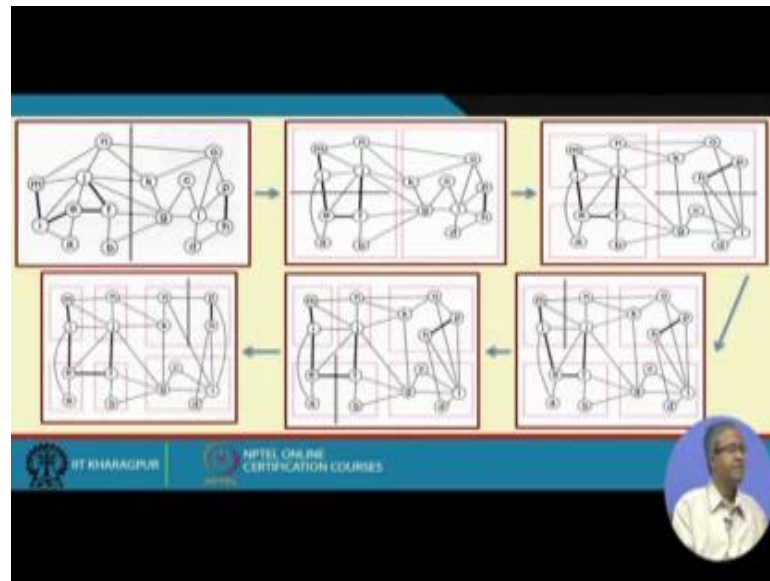
cut at every step, because c which is here this has moved here at this step, this has moved here in this step and it remained there. So, it is not that I am blindly cutting the gates are also moving around during the process and this is the final placement I will get right. So, this is the first method.

(Refer Slide Time: 12:24)



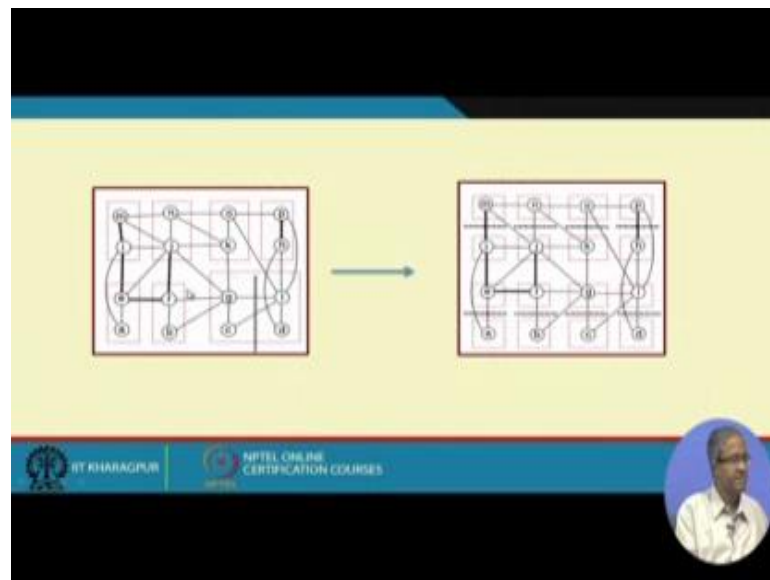
Now, the second method is called recursive Bipartitioning Mincut placement; again you see here the method is similar, but the sequence of cut lines are a little different as I will show with an example, this again is a recursive division using vertical horizontal cut lines, but I shall be explaining with the help of the example that how.

(Refer Slide Time: 12:46)



So, the same example I take. So, I start with a vertical cut dividing up into 2 rectangles, but instead of a horizontal cut across the layout, I am cutting only one of the hubs. So, I am cutting this, I get this then I am cutting the right half and then I am getting this, then I am cutting vertically one of the 4 this one, then I cut this.

(Refer Slide Time: 13:23)

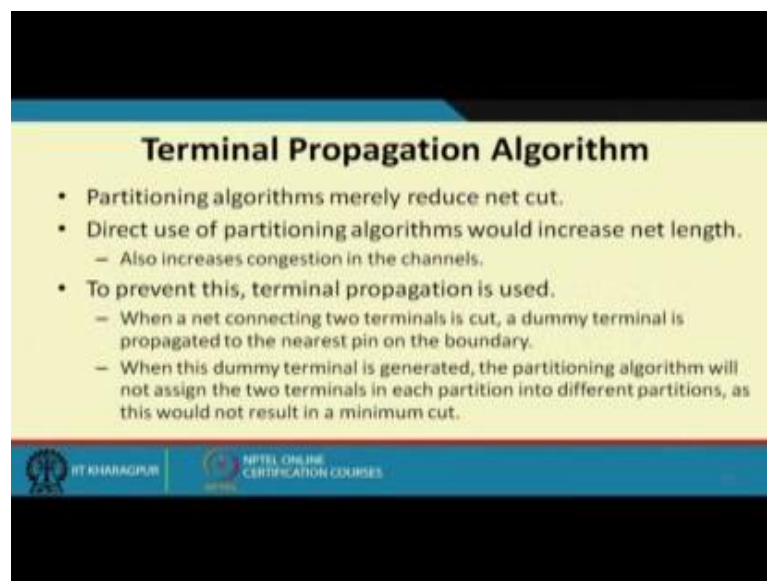


Then I cut this right, then I cut this and finally, well here I am showing in 1 diagram I cut this, I cut this, there are 8 cuts I am showing here together. So, finally, again I get a partition.

Now, one thing you see you go back to the previous design, you see the final this is your final partition. You see this partition does not look to be a very good partition, because you see there are several wires which are running pretty long N to O, O to P there are then J to K, there are many wires which are pretty long. So, if you compute the wire length if you take the Manhattan distance, then you can see very easily that this solution is worse as compared to what I get here. You see here the connections are very localized only other than 2 or 3 of the connections. So, all of them are between neighbors.

So, this method gives better solution as compared to the other one, but of course, this has happened by chance you cannot say that for any netlist that I give you this method will give you the best solution better than the other one. So, Breuer proposed 4 or 5 such sequence of cuts, and you can apply them and see which of them is giving the better result and select that better one this is the basic idea.

(Refer Slide Time: 15:05)



Terminal Propagation Algorithm

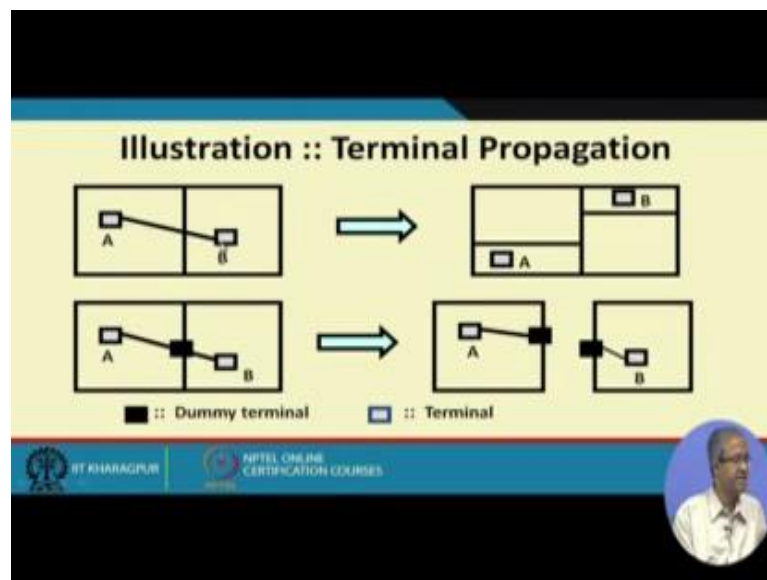
- Partitioning algorithms merely reduce net cut.
- Direct use of partitioning algorithms would increase net length.
 - Also increases congestion in the channels.
- To prevent this, terminal propagation is used.
 - When a net connecting two terminals is cut, a dummy terminal is propagated to the nearest pin on the boundary.
 - When this dummy terminal is generated, the partitioning algorithm will not assign the two terminals in each partition into different partitions, as this would not result in a minimum cut.

IT KHARAGPUR | NPTEL ONLINE CERTIFICATION COURSES

Fine now there is an associated problem with any partitioning based placement algorithm, and this can be solved by using a process called terminal propagation. So, this session illustrated with a simple example, the idea is like this. Direct use of partitioning algorithm can increase net length as the first method showed us, this can also increase congestions in channels in some areas a lot of lines will be there, this also was seen in the first method.

So, to avoid this, a method called terminal propagation is used which I shall be illustrating with an example, where the concept of a dummy terminal is used which is propagated towards the partitioning boundary. So, I will be explaining with example then you will understand what is it clear.

(Refer Slide Time: 16:08)

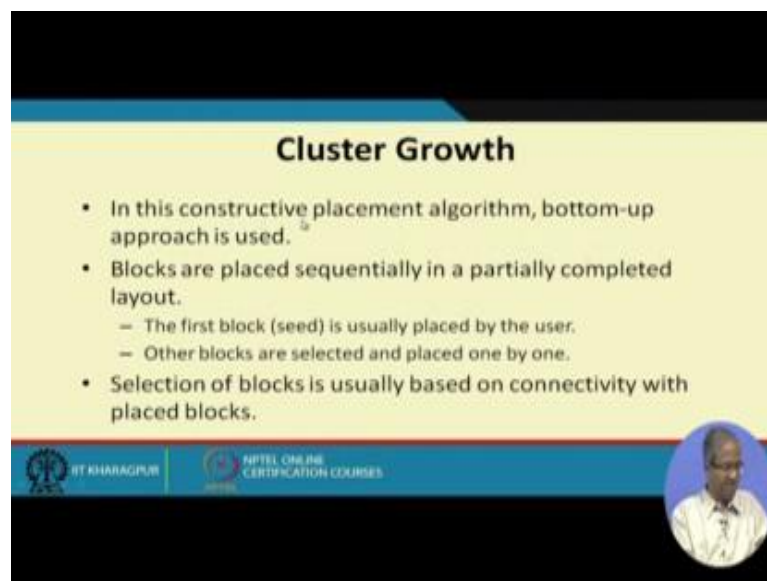


Let us take an example like this, these are very simple example; 2 blocks this A and B they are connected. So, if I use a simple partitioning based (Refer Time: 16:21) deistic like in the first method. So, I go on partitioning other blocks I am not showing, I am only showing A and B. So, as you go on partitioning it may so happen that A will land up here and B will land up there. So, connecting A and B will involve a long inter connection line it means a long delay.

But terminal propagation concept is something like this, that when you partition a layout using a horizontal thing you see this, this AB was connected, you are doing a partition. So, when you do a partition you see that this one wire or one net was cutting. So, what you do? You introduce a dummy terminal at the partitioning junction. So, when you take the 2 partitions out, this dummy junction there will be 2 copies of it, but one thing you remember, but these 2 dummy junctions actually represent the same net. So, when you move or cut the other blocks and compute the cost. So, if you move these dummy junctions too far away, that cost is also taken into account.

So, you are not allowed to move B or A too far apart like here it shows. These 2 dummy terminals will always remain closer to each other, which will also make A and B closer to each other. So, this is just a very simple illustration I have given, but actually in a practical case the terminal propagation process is very complex, because the net can be pretty complicated, large number of blocks, large number of inters connection lines, so there will be large number of dummy terminals. So, again there are lot of heuristics that are used here.


(Refer Slide Time: 18:25)



Cluster Growth

- In this constructive placement algorithm, bottom-up approach is used.
- Blocks are placed sequentially in a partially completed layout.
 - The first block (seed) is usually placed by the user.
 - Other blocks are selected and placed one by one.
- Selection of blocks is usually based on connectivity with placed blocks.

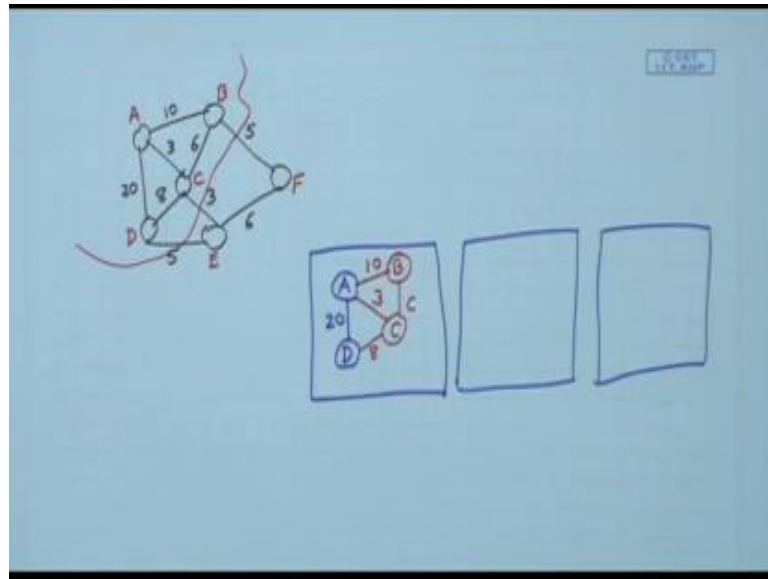
IT KHARAGPUR | NPTEL ONLINE CERTIFICATION COURSES



So, here we look at another constructive placement algorithm, which is very simple. Now we had seen earlier in our last lecture that the force directed placement algorithm can also be used in a constructive sense where you can place one block at a time, find its 0 force location and place it there that way you can make the clusters or the partitions grow.

So, you can have a much simpler approach here, this is called the cluster growth algorithm, you use some kind of a bottom up approach. So, what you do? Suppose I have a partially completed layout at any stage, suppose there are already a percentage of the blocks already placed, the other blocks you have to place. What you do is that you place one of the block initially, you call it the seed the other blocks are selected one by one and placed closer to the blocks depending on the connectivity, that how they are connected to the other blocks that creates the partitions.

(Refer Slide Time: 19:55)



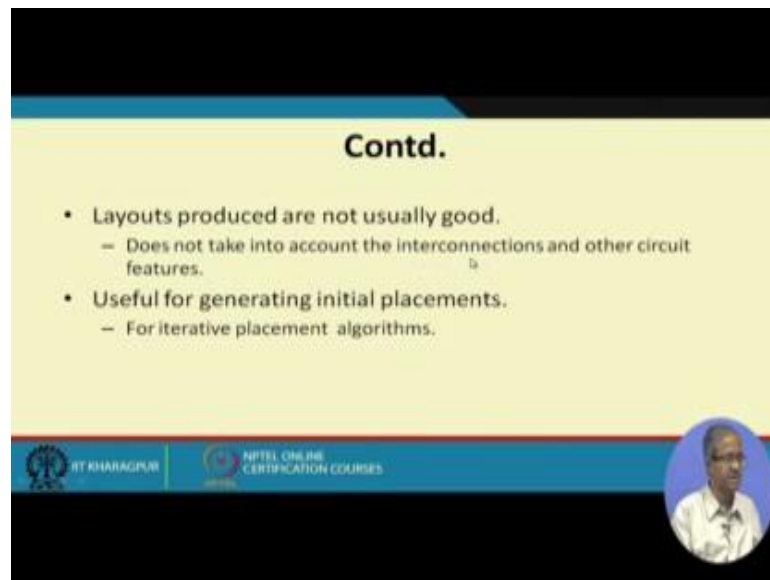
Like you see I am giving an example let say I have a netlist like this, there are 6 nodes, let say the connection weights are like this. Let us also give some names to these vertices A B C D E and F. Now let us suppose I have a partition which I am creating, a cluster which I am creating for I have already placed A and B. So, similar method we have discussed earlier also. So, here I have already placed the images most strongly connected.

So, what I am saying is that among the remaining one I place them one at a time, possibly depending on the connectivity. B is most strongly connected, so maybe in the next step I shall be placing B 10. Next step maybe I shall be placing C because it is strongly connected to the rest 8 and 6, I am sure here there is another link I missed it let say this is 3 say this is 3.

So, now if my partition size constraint is 4 let say stop here. So, A B C D these 4 already have been put in one partition. Now I repeat it for the other, there can be other vertices you repeat and you try constructing the 7 partition. So, each partition can be having a limit to the maximum number of vertices that you can place you do it, you then move on to the next partition. So, as if this is like a cluster you are starting with an initial point you are growing the cluster up to some maximum allowable size.

So, after you have done with this, you move to the next cluster in this way you create the partitions at the placement of the blocks.

(Refer Slide Time: 22:23)



The slide is titled "Contd." and contains the following text:

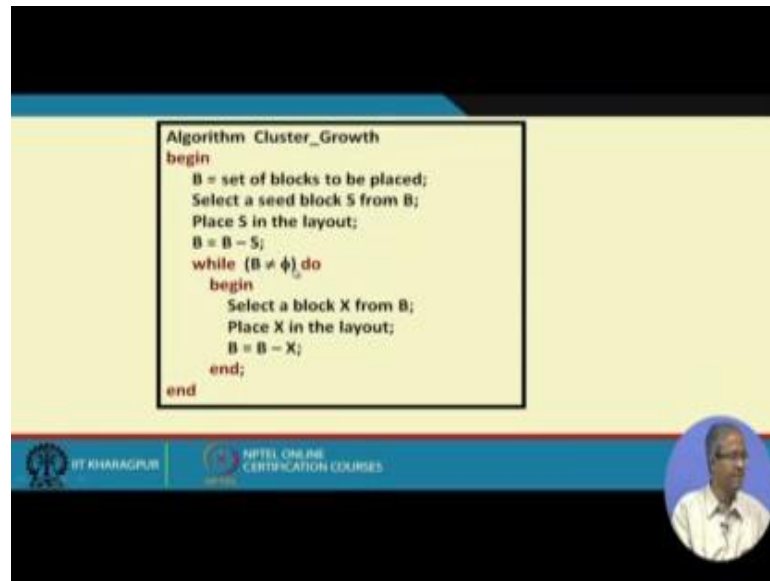
- Layouts produced are not usually good.
 - Does not take into account the interconnections and other circuit features.
- Useful for generating initial placements.
 - For iterative placement algorithms.

At the bottom of the slide, there are logos for IIT KHARAGPUR and NPTEL ONLINE CERTIFICATION COURSES, along with a circular portrait of a man in a white shirt.

So, layout produced using this kind of cluster go algorithms are not typically not good, because it does not directly take into account the inter connection details exactly how they are inter connected and what is the connection length and so on, just the number of connections are taken into account.

But this method has some utility like, it can be useful for generating initial placement; which can be used in iterative improvement algorithms like simulated aniline, that for simulated aniline we start with an initial placement, now we can use this cluster growth algorithm which is pretty fast to create that initial placement. So, instead of creating the initial placement absolutely randomly, we use some intelligence to create a reasonably good placement then we give it to the simulated aniline tone. So, it can improve upon that. So, this is the idea.

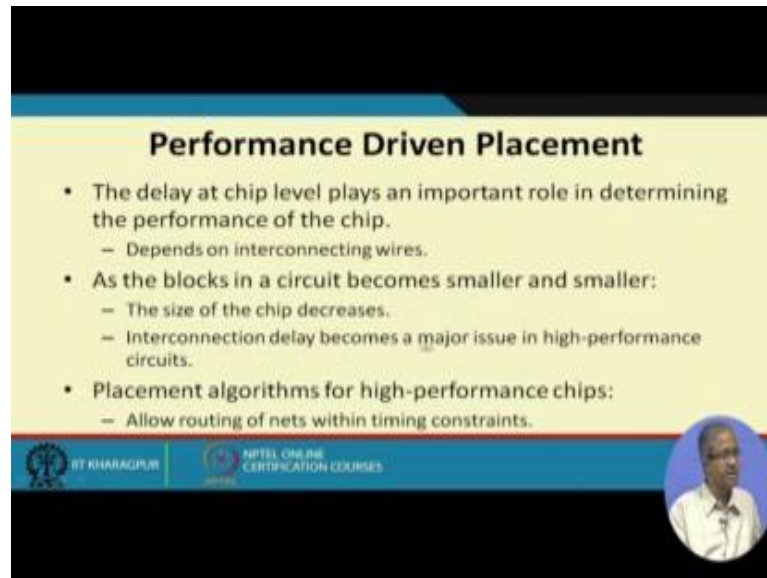
(Refer Slide Time: 23:34)



```
Algorithm Cluster_Growth
begin
  B = set of blocks to be placed;
  Select a seed block S from B;
  Place S in the layout;
  B = B - S;
  while (B ≠  $\phi$ ) do
    begin
      Select a block X from B;
      Place X in the layout;
      B = B - X;
    end;
  end;
```

So, this is the pseudo code for the cluster growth algorithm. So, B denotes the set of blocks that are to be placed, you select a block S you call it as seed from B, there can be some heuristic you select the block which is maximally connected that can be your seed you place a single layout remove S from B. So, while B is not empty; repeat select a block X from B, place X in the layout and remove X from B well. This is as if you are completing the entire thing, but there can be another restriction as I have said which is not shown here. So, each of these clusters can have a maximum size; so once that maximum size is reached, you can start and you can start the selective seed for the next cluster. So, in this way several clusters can be formed right C 1, C 2, C 3.


(Refer Slide Time: 24:40)



Performance Driven Placement

- The delay at chip level plays an important role in determining the performance of the chip.
 - Depends on interconnecting wires.
- As the blocks in a circuit becomes smaller and smaller:
 - The size of the chip decreases.
 - Interconnection delay becomes a major issue in high-performance circuits.
- Placement algorithms for high-performance chips:
 - Allow routing of nets within timing constraints.

IT KHARAGPUR | NPTEL ONLINE CERTIFICATION COURSES

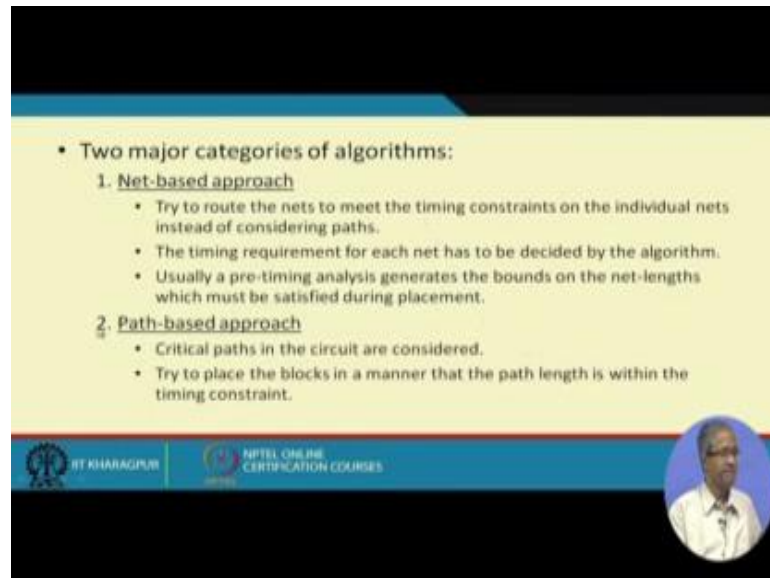


Now, lastly let us make some brief comments on performance driven placement, you see we shall again come to this performance driven issues later. So, when we talk about the timing analysis and other related issues, now the issue here is that delay at the chip level is quite important, and it plays an important role to identify the critical paths, which in turn can determine the clock frequency, which in turn can determine the overall performance of the chips. And this delay often is due to inter connections; bad wiring or bad placement can lead to large delay at the chip level.

Now, as the blocks becomes smaller and smaller; blocks in the transistor you can say the transistor is becoming smaller, the chips are also becoming smaller. So, now, interconnection delay has become a major issue because now the gate delays and the interconnection delays are almost becoming comparable. So, earlier the gate delays were much larger. So, you could afford to ignore the interconnection delay, but you cannot right.

So, for high performance circuits in chips, the placement algorithms will allow inter connection of the nets where some timing constraints maybe there, I can say that you route in such a way that your maximum delay should not exceed some delta value that can be specified by the user.

(Refer Slide Time: 26:29)



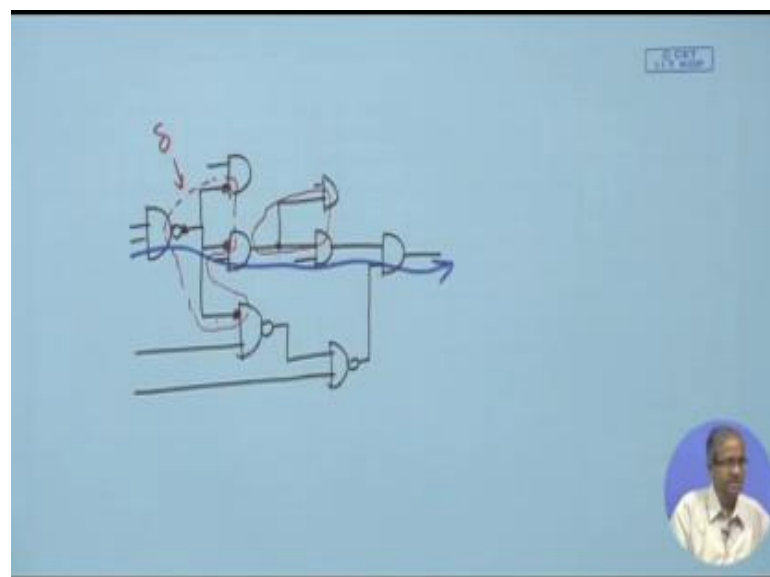
• Two major categories of algorithms:

1. Net-based approach
 - Try to route the nets to meet the timing constraints on the individual nets instead of considering paths.
 - The timing requirement for each net has to be decided by the algorithm.
 - Usually a pre-timing analysis generates the bounds on the net-lengths which must be satisfied during placement.
2. Path-based approach
 - Critical paths in the circuit are considered.
 - Try to place the blocks in a manner that the path length is within the timing constraint.

BT KHARAGPUR | NPTEL ONLINE CERTIFICATION COURSES

So, the algorithms broadly can be classified into 2 classes: 1 works on a net by net basis. So, recall what is the net? A net is a connection of pins, which have to be connected; they belong to the same means equi potential net in that sense. So, in the net based approach here we try to route the nets to meet the timing constraints individually on a net by net basis, we do not consider the paths.

(Refer Slide Time: 27:11)



Like what we mean is that let say I can have a scenario, where let us consider a small circuit at the level of the gates let us take this. So, here if you look at this, this set of

wires is 1 2 3 4 they form a net because this pin here, this pin here, this pin here and this pin here they are equipotential. So, we consider this net individually, we try to reduce the delay of this net the maximum delay from the output of this NAND gate to the input of these NAND gates, we try to reduce this delay. But we do not consider the path like from my input to my final output that is the total delay, I am not considering it totally I am looking on it net by net basis. Similarly if here I have this going here again; so I will be having another net out here, there will be another net here. So, I will be trying to minimize the delay of this, net like that on a net by net basis we shall precede.

So, the timing requirement for each net has to be decided, and usually some kind of a timing analysis tool is available which generates bounds on the net lengths which must be satisfied during placement like for example, let say I want that my circuit needs to work at some particular clock frequency. So, you can make an analysis and you can predict that well to make my circuit work at this particular clock, my maximum net length can never exceed this; if it exceeds this then this requirement may be difficult to meet. So, this kind of a constraint can be there.

The other approach maybe you consider paths from input to the output critical paths. You look at the total delays along the paths and see whether the critical paths are violating some clock timing constraints or not, and you try to place the blocks in such a manner that the path lengths particularly the critical paths they do not violate the timing constraint, not only that some of the paths which might not be critical initially, but after you have placed them maybe some other path has now become critical. You should make sure that such criticality of the paths their increasing of delay, should not adversely affect the timing constraint, they should be met at all times.

Now, we shall be looking at these things later, when we discuss this static timing analysis and maybe and such related issues. So, with this we have looked at the various placement algorithms now in this week we have looked at the initial problems of physical design the partitioning, floor planning and placement.

So, with this we come to the end of this lecture. So, in the next lecture we shall be starting our discussions on routing.

Thank you.