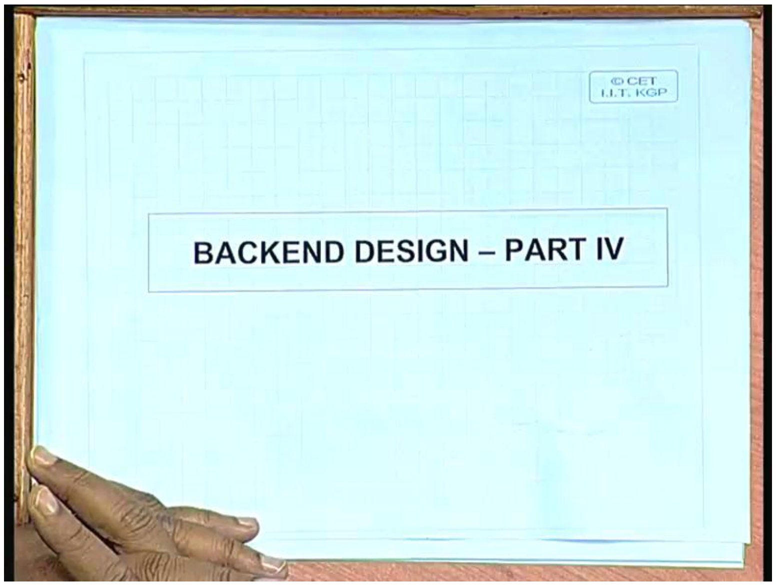
Electronic Design Automation
Prof. Indranil Sengupta
Department of Computer Science and Engineering
Indian Institute of Technology, Kharagpur

Lecture No #18
Backend Design: Part IV

In our last class we were talking about some of the problems related to floor planning. And we also looked at one of the methods or algorithms that have been proposed to solve the problem, namely using integer linear programming.

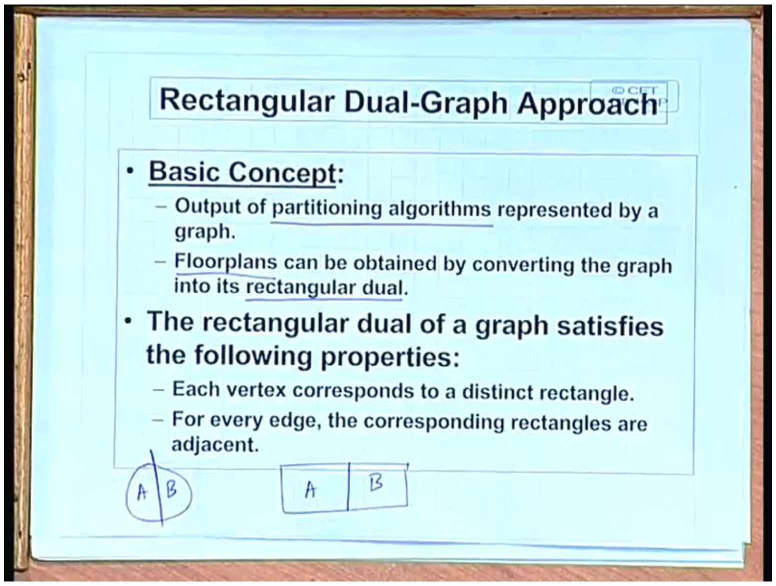
(Refer Slide Time: 01:25)



BACKEND DESIGN – PART IV

Now today to start with we talk about an interesting method which relies on some concepts of graph theory the concept of dual of the graph. And starting from the concept of slicing a given floor plan, we can we can have the concept of the slicing tree and we will see that a slicing tree and a floor plan have the one to one correspondence. So let us have a look at this method first.

(Refer Slide Time: 02:01)



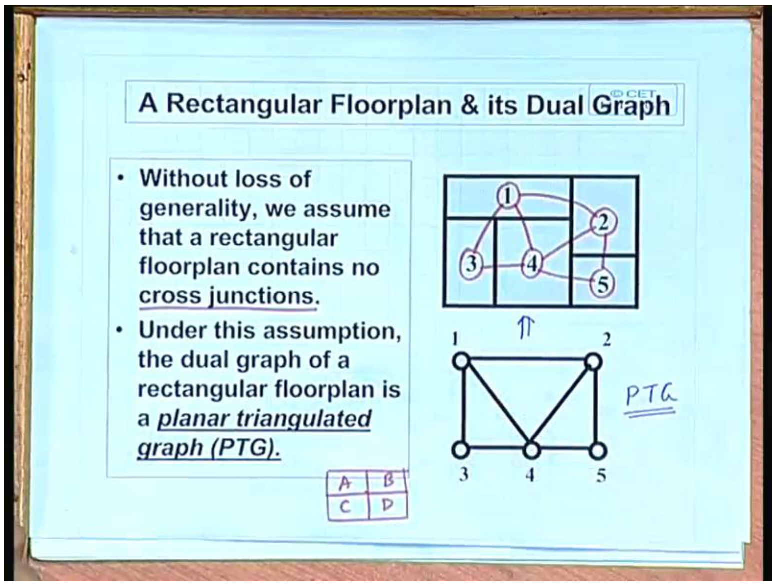
Rectangular Dual-Graph Approach

- **Basic Concept:**
  - Output of partitioning algorithms represented by a graph.
  - Floorplans can be obtained by converting the graph into its rectangular dual.
- **The rectangular dual of a graph satisfies the following properties:**
  - Each vertex corresponds to a distinct rectangle.
  - For every edge, the corresponding rectangles are adjacent.

So this method is called rectangular dual graph method or approach. Now the basic concept is quite simple. See when you are carrying out the partitioning of the circuit, because this floor planning and partitioning are typically carried out hand in hand. When you are dividing a circuit into two blocks through partitioning, so what we expect is that, with respect to the floor plan there one of partition a will be placed in some region. And the other partition would be placed in a disjoint region. So the output of the partitioning algorithm can be represented by a graph. So there will be the graph where the nodes indicate the different partitions of modules and their edges will indicate the interconnections.

And in this method starting from that partition graph we can directly get the floor plan by converting it into its rectangular dual. We will see what is rectangular dual is. Now a rectangular dual of a graph can be obtained. Well we will just show an example where each vertex in the graph will correspond to a distinct rectangular block. And if two blocks are adjacent, these two nodes these two nodes will be connected by an edge. So I am okay I am just showing an example to illustrate this point that each vertex in this graph will correspond to a rectangle of the floor plan. And two rectangles which are adjacent the corresponding nodes will be connected by an edge. Okay. So let us see an example.

(Refer Slide Time: 03:55)



A Rectangular Floorplan & its Dual Graph

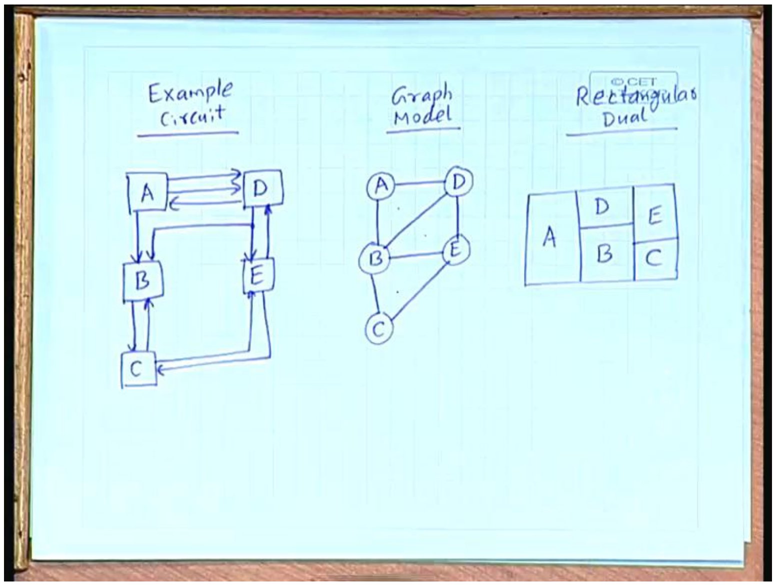
- Without loss of generality, we assume that a rectangular floorplan contains no cross junctions.
- Under this assumption, the dual graph of a rectangular floorplan is a ***planar triangulated graph (PTG).***

Suppose I have a floor plan like this and this is the so called partition graph or that you can say the triangle. Sometimes this is called planar triangulated graph also PTG. This is essentially the partition graph which well actually I have shown this after this. But actually this is the starting point and from this we want to get the floor plan. But here I am showing the correspondence. Suppose I have a floor plan like this, so corresponding to each rectangle of the floor plan there will be one vertex. One will be 1, 2, 3, 4 and 5; and the two rectangles which have some common area of intersection, region of intersection. They will be connected by an edge like one and three are adjacent.

1 and 4 are adjacent, 3 and 4, 1 and 2, 4 and 2, 4 and 5, 2 and 5 you will get this graph. So from the floor from the floor plan you can get this graph very easily. Similarly from this graph if you know the sizes of the blocks, you can also go towards this and there are algorithms for doing that. Now in order to do this, one thing is clear is that you assume that there are no cross junctions in the floor plan like cross junction, means four blocks connected like this a, b, c, d. So we will see later. If there is a cross junction like this, what does this mean with respect to the dual graph? Because why we say this is that the success of this algorithm relies on a graph where all the faces of the graph, these are planar graph.

It can be laid out in a plane because this is a plane planar representation. So the vertices you are connecting by an edge, there cannot be two edges crossing. So this will also be a planar graph. So the success of this method relies on means one premises is that the faces of each region in this graph will be a triangle. That is why it is called planar triangulated graph. But the problem with cross junction is that you will get a graph which is not triangle faced the graph will be like this. A connected to b, b to d, d to c and c to a. Okay. So we do not want graphs like this. Because graphs like this will lead to some problem in translating from here to here. So we take an example to illustrate how starting from a circuit, we can just arrive at a floor plan.

(Refer Slide Time: 07:05)

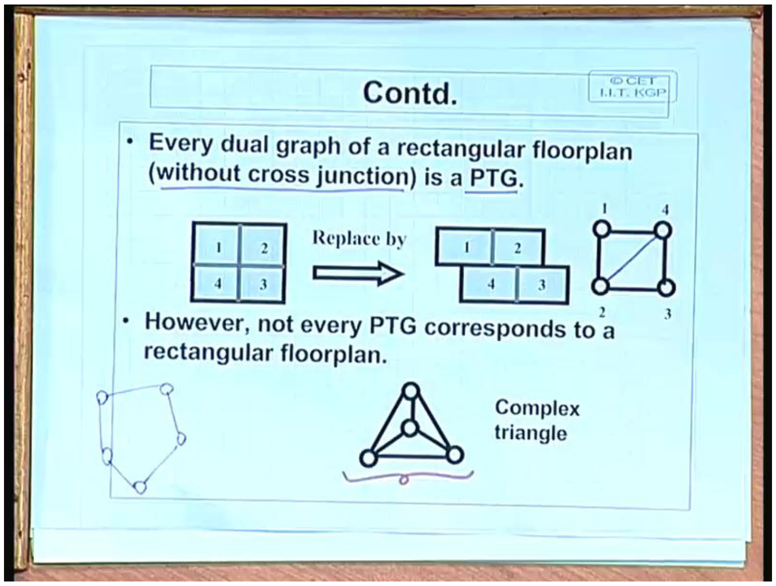


Just we start with an example circuit. Well I am not showing the exact circuit. Say in terms of gates, but rather I am saying that after partitioning is done, a, b, c, d, these are the partitions. So I am actually showing the partitioned picture after partitioning suppose you have something like this. Suppose you have an example circuit like this where a, b, c, d, are the blocks which have been obtained after partition and this shows the interconnection. Now this circuit can be represented by a graph. This is your so called graph model which you can get immediately from

the partition circuit. So here the partitions will be represented as vertices and if two partitions are connected, they will be joined by an edge.
Well this can be a weighted graph also indicating the number of connections which are there from d and b there is a connection. Okay, fine. So you can say this graph is already triangulated. So all the faces are triangles. Now from this graph model you can draw the corresponding rectangular dual. So rectangular dual is a graph where corresponding to each vertex, you will get a rectangle and two edges. I mean two vertices are connected in edge means the two corresponding rectangle will touch at least in some non-overlapping area. So one such rectangular dual this is again not unique. One such rectangular dual for this graph is a, d, b.

See of course the exact sizes of these will depend on the width and height or the areas of these things in terms of the number of components there. So I am assuming that a is the biggest a is the biggest area b is bigger than d and so on. So this is one possible floor plan you can see a is adjacent to b and d, b and d, d is adjacent to a, b, e, d is adjacent to a, b, e, c, is adjacent to b and e and so on. So there are systematic algorithms to convert a graph model into the rectangular dual we are not going into that algorithm. But you can see that it can be drawn. So if it is a planar triangulated graph then this translation can be done fairly easy. Okay. But as I said if there are cross junctions then the algorithm for translation can create some problem. So we try to adjust that problem.

(Refer Slide Time: 10:53)


Contd.

- Every dual graph of a rectangular floorplan (without cross junction) is a PTG.
- However, not every PTG corresponds to a rectangular floorplan.

©CET I.I.T. KGP
Replace by
1
2
3
4
Complex triangle

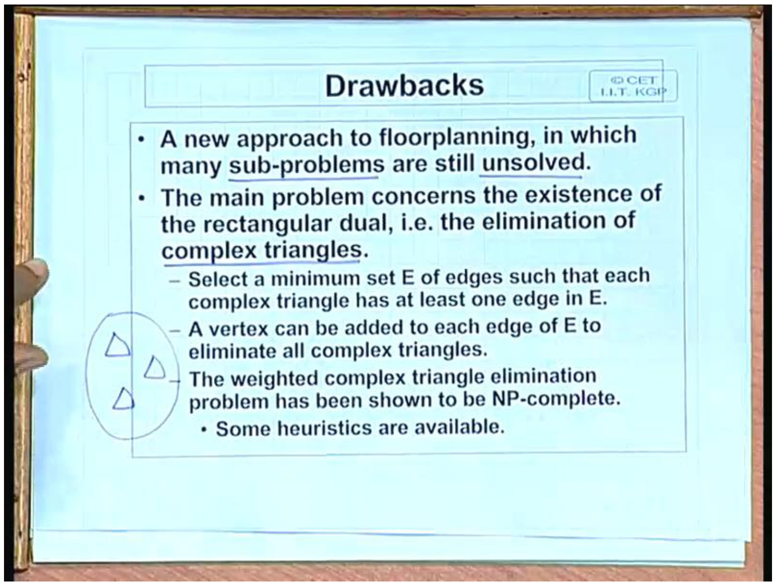
So the first thing is that we have already seen that if the floor plan does not have any cross junction, then the corresponding graph model is a planar triangulated graph and also the reverse vice versa. So since in the earlier example, for example this was a PTG planar triangulated graph. This is one triangle, this is one triangle, this is one triangle, then we get a rectangular floor plan; floor plan without any cross junctions. But if there are any cross junction like this, well then then means according to the normal thing you will get a graph like this 1, 2, 3, and 4. See although 1 and 3, and 2 and 4 are very close together, they are touching at the corners. But still you do not show any edge in the graph. See interconnecting 2 and 4 or 1 and 3 may also be very easy because they are touching. But here you are not capturing that. So normally what is done if you have a cross junction is that, say means I am showing it indirectly.

Actually it will happen from other side, say you shift this block slightly on some sides. For example if you shift it like this, we shift four a little bit on the right. So that 4 and 2 overlaps a little. So now you will get a graph like this with a new edge added. Now if we have a graph like this from your partition circuit, then you find out that among this 2 and 4 and 1 and 3 which are the ones which are most strongly connected among themselves. If you say 2 and 4 have some wires among them, well they will obviously be connected. But if not you add some dummy edge in order to make it triangulate. So means in order to mean if you make it triangulated it is easy to convert to this. That is the thing because the algorithm which is there, that can very easily convert this into a corresponding floor plan. But if there are non-triangulated faces, then it can create problem. Because in general it can be faces of sizes 4, 5, 6. So it creates a problem in making the layout.

See for four is fine. Suppose you have a graph like this, where there are five. So what do you do about this? Without living some blank space in between you cannot lay them out. You cannot have a very nice packed layout for this. So for this it is mandatory to have a triangulated form first and then you try to convert into the rectangular dual. But there are yes [student noise Time: 14:04] graph to a dual, graph to a dual like I have just shown in this one from the partition circuit you get a graph from the graph. You get a dual this is your floor plan. [student noise Time: 14:19] Interconnection arrows are the interconnections blocks which are interconnected. They are shown by an edge. Now in this example, this is already triangulated. If for example d and b were not connected, we would have forcibly added this edge or edge between a and e. Fine. But there are some exceptions. This is one exceptional structure that can occur in the graph.

You can see in this, this is a complete graph of four vertices. Here all the faces are triangles. But yet you try to draw the corresponding rectangle floor plan. You will not be able to do that there will be conflicts you will see this structure is called a complex triangle structure. And this does not correspond to a rectangular dual rectangular dual. You cannot construct the rectangular dual rectangular dual. You cannot construct a rectangular dual for this. Yes. [student noise Time: 15:24] We cannot make a shift like this. Yes. Then means we will have to have some priority means you will be taking this. Say we will be just adding some edges if required will be removing some edges in order to make all the faces triangular triangle. Yes true. So starting from the graph the first step is to have our only triangulated faces. The second step problem is that if there is a complex triangle, you will have to handle this. So now let us say if there is a complex triangle, how we can handle this.

(Refer Slide Time: 16:07)

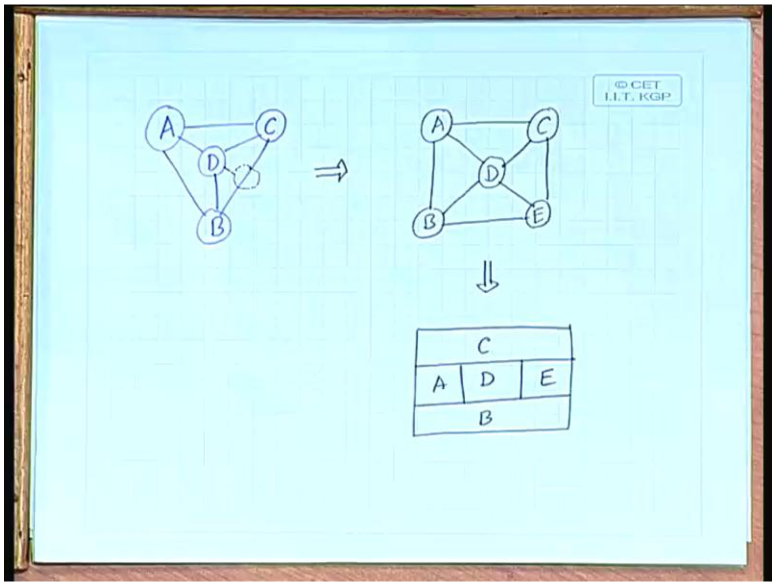


Drawbacks

- A new approach to floorplanning, in which many sub-problems are still unsolved.
- The main problem concerns the existence of the rectangular dual, i.e. the elimination of complex triangles.
  - Select a minimum set E of edges such that each complex triangle has at least one edge in E.
  - A vertex can be added to each edge of E to eliminate all complex triangles.
  - The weighted complex triangle elimination problem has been shown to be NP-complete.
    - Some heuristics are available.

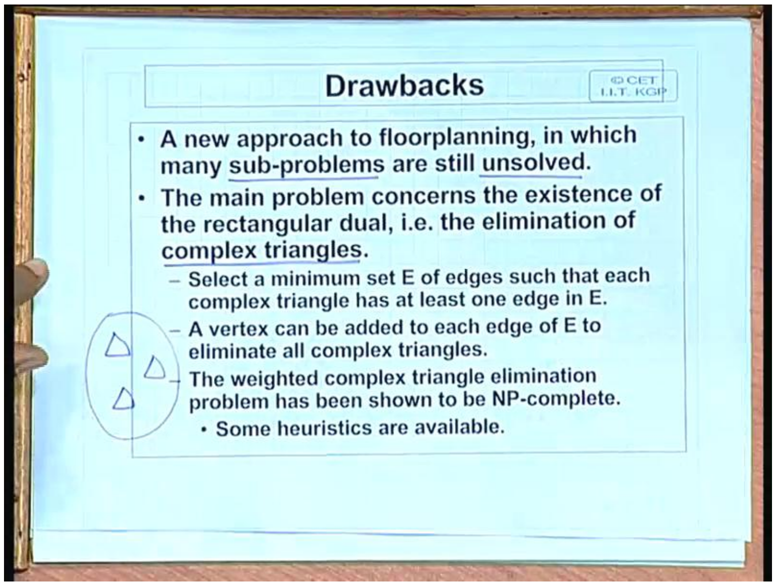
So the methods well, we will see this method is very easy to understand. But the problem is that this is based on some graph theoretic approach the some of the step of which are still not pretty easy. Like for example from a given arbitrary graph, how to convert it into triangulated one and from the triangulated one how to have a very efficient algorithm to convert it into floor plan. Well from small examples you can do it by hand. But for very complex examples it is difficult. But the main concern is if there is a complex triangle. What to do? The idea is fairly simple see your total graph may contain several such complex triangles. Say three such, so you try to break the complex triangles. How? We break the complex triangles by adding a dummy vertex to one edge of each of the complex triangle. We will show how. So we add vertex and an edge also. We will see how. To each of these complex triangles as a result of which we eliminate the complex triangles structure. Well I am showing you an example. Then again I will come back to this.

(Refer Slide Time: 17:33)



Suppose we have a complex triangle like this. So this is a complex triangle and means what we do that we convert this into an equivalent triangle where we introduce a dummy vertex out here and a dummy edge connecting d and this. So we get a modified graph which looks like this. So this new vertex we are calling e; e lies between b and c. Okay. And a new edge between d and e. So now the new graph becomes like this which is nicely triangulated four triangular faces. There is no complex triangle and if you convert this into the rectangular dual you get the corresponding floor plan also. So, one possible floor plan is this. So you can basically check c is adjacent to a, d, e; a, d, e, a is adjacent to b, d, c; b, d and c and so on. So if there is a complex triangle you identify the complex triangles in the graph and for each such you add a dummy vertex and an edge to it this e is actually a dummy one. It does not make any meaning. But this e you can merge finally blank space. But by introducing this, you can have a properly triangulated one for which the dual exists does not mean means you can get a floor plan. But if you do not do this, you cannot get a floor plan from this. Right. Okay.

(Refer Slide Time: 19:54)

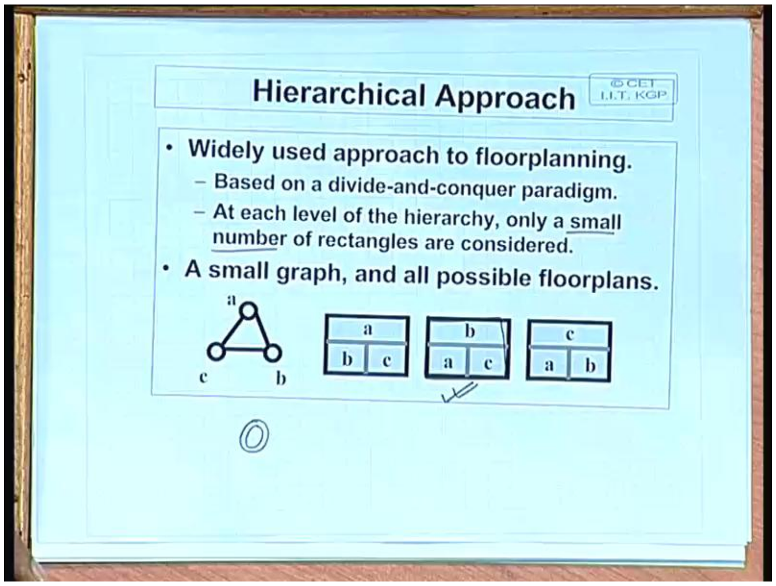


Drawbacks

- A new approach to floorplanning, in which many sub-problems are still unsolved.
- The main problem concerns the existence of the rectangular dual, i.e. the elimination of complex triangles.
  - Select a minimum set E of edges such that each complex triangle has at least one edge in E.
  - A vertex can be added to each edge of E to eliminate all complex triangles.
  - The weighted complex triangle elimination problem has been shown to be NP-complete.
    - Some heuristics are available.

But the problem is that for a given arbitrary graph identification of complex triangles in it this itself has been shown to a NP complete problem. Okay. So this method although apparently it looks fine, but implementation wise it is a complex problem. Right, fine. So what people normally do? They either go for a hierarchical approach or they go for you can say a randomized algorithm like simulated annealing.

(Refer Slide Time: 20:32)



Hierarchical Approach

© CET I.I.T. KGP

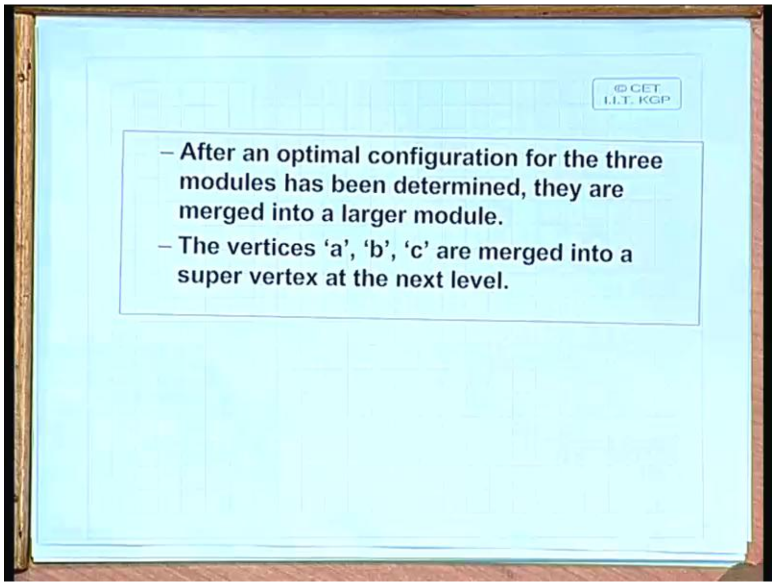
- Widely used approach to floorplanning.
  - Based on a divide-and-conquer paradigm.
  - At each level of the hierarchy, only a small number of rectangles are considered.
- A small graph, and all possible floorplans.

a
c
b
a
b c
b
a c
c
a b

So in hierarchical approach it is fairly simple in concept. This is based on a divide and conquers approach. Divide and conquer why? Because at every stage we consider only a few of the rectangles and their relative orientations and placement. It is possible that the original sub problem we will be having thousands of rectangles. But at any particular level of our means of our algorithm we consider only a very small number of rectangles. Okay. Just to show you, suppose you have a very small such interconnection graphs like this a, b, c, these are three blocks. From this graph you can have several different floor plans just to show.

So if you have a small graph which you are handling, so for this graph you generate the all possible floor plans which are feasible you estimate the costs of each and you select the one, which you feel, is the best. Now after this is done, what is done at the next level of hierarchy? These vertices a b c are merged together into a super vertex. So you start with small sub circuits, you try to obtain the best relative. Means the best relative placement or floor plan of those you merges them into one. Now this entire thing becomes a single rectangle. Now at the next level, the entire thing will act as rectangle which you want to place. So in this way, so actually from bottom, it goes really towards the top. So in this example as I have just said.

(Refer Slide Time: 22:35)

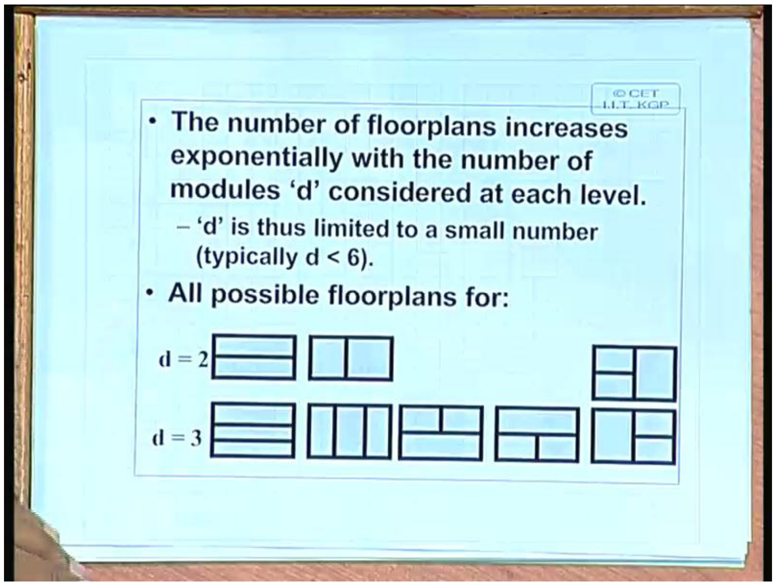


© CET
I.I.T. KGP

- After an optimal configuration for the three modules has been determined, they are merged into a larger module.
- The vertices 'a', 'b', 'c' are merged into a super vertex at the next level.

That after an optimal configuration has been determined; they can be merged into a bigger module. And similarly the vertices will be merged into a super vertex at the next level. This process will go on.

(Refer Slide Time: 23:00)



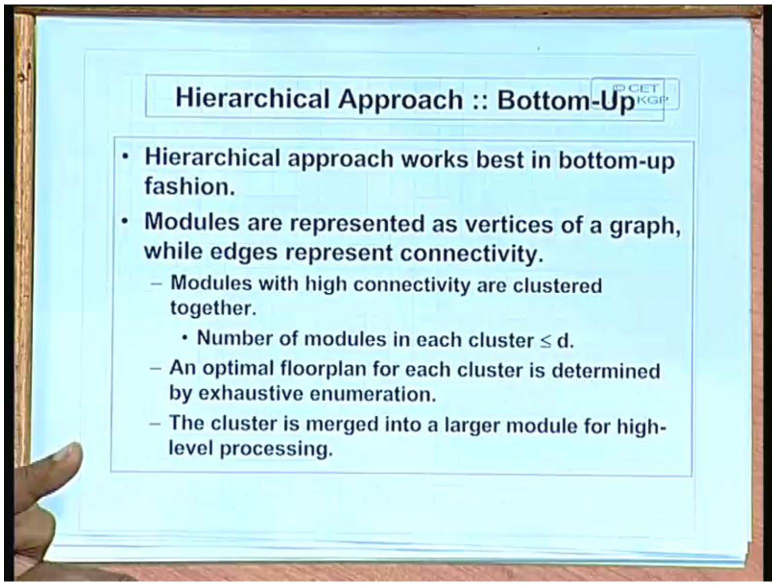
© CET
I.I.T. KGP

- The number of floorplans increases exponentially with the number of modules 'd' considered at each level.
  - 'd' is thus limited to a small number (typically d < 6).
- All possible floorplans for:

Now the problem which I had mentioned earlier that depending on the number of modules you consider at each level the number of possible floor plans increases very rapidly. Just I am showing couple of very simple examples. Well I am not starting with any graph. I am showing all possible floor plans of two blocks. If there are two blocks, you can have either this or this two possibilities. But for 3 you see that there are so many possibilities 1, 2, 3, 4, 5, 6. So for 4, it again goes up for 5, it is very large. So as the value of d is increasing, the number of possible orientation of the reality orientation of the blocks also goes on increasing.

So typically for this kind of a divide and conquer approach where we are exploring all the possibilities at each level and try to find out the best, limiting d to a low value is very important. Typically d equal to 5 is the maximum you can have. So as a rule of thumb we can say that d will always be less than 6. So at each level we would be exploring all the possibilities exhaustively. Because since the exhaustive search space is limited, we can do that and we will be finding out the best solution depending on some cost function at each stage. Now this hierarchical approach can operate bottom up or top down both.

(Refer Slide Time: 24:30)



Hierarchical Approach :: Bottom-Up

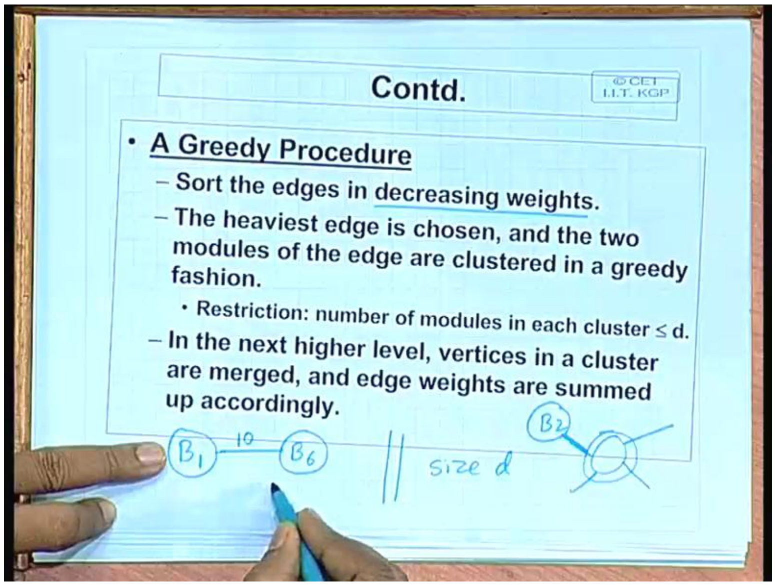
- Hierarchical approach works best in bottom-up fashion.
- Modules are represented as vertices of a graph, while edges represent connectivity.
  - Modules with high connectivity are clustered together.
    - Number of modules in each cluster ≤ d.
  - An optimal floorplan for each cluster is determined by exhaustive enumeration.
  - The cluster is merged into a larger module for high-level processing.

Bottom up is the most natural. So first let us look at the bottom up approach. Bottom up approach as I have just mentioned it is essentially a bottom up approach. The individual modules or blocks will be represented as the vertices of the graph while edges represent connectivity. Now you start by putting the modules together. You identify a set of modules. Suppose you have the parameter d, d equal to say 4 or 5, something you have chosen say d equal to 5. So your first task will be to choose five blocks which are most strongly connected among themselves.

This you can easily find out by analyzing the graph. So the modules with high connectivity you try to cluster together and of ofcourse restriction is that number of modules and clusters should not exceed d. Now just as I had mentioned that after doing this you exhaustively search all possible floor plan among those d blocks and try to find out the best the optimal floor plan. Then at the next higher level of processing the clusters are merged into a larger module. Moreover with respect to the floor plan, the floor plan you have obtained that rectangle is kept intact.

So that entire rectangle will be placed somewhere at the next higher level of processing. So from the leaf you slightly you just successively go up by putting the modules together, together and together and make it. So finally when you reach the root, you have a single node which is a union of all the blocks. Right. Then that node will represent your total floor plan. So in this approach as you move towards the you can say the individual blocks the leaf of the nodes towards the root your floor plan is getting constructed slowly step by step in hierarchical fashion. Now means one simple ways of implementing is to have a greedy procedure.

(Refer Slide Time: 26:46)



Contd.

© CET I.I.T. KGP

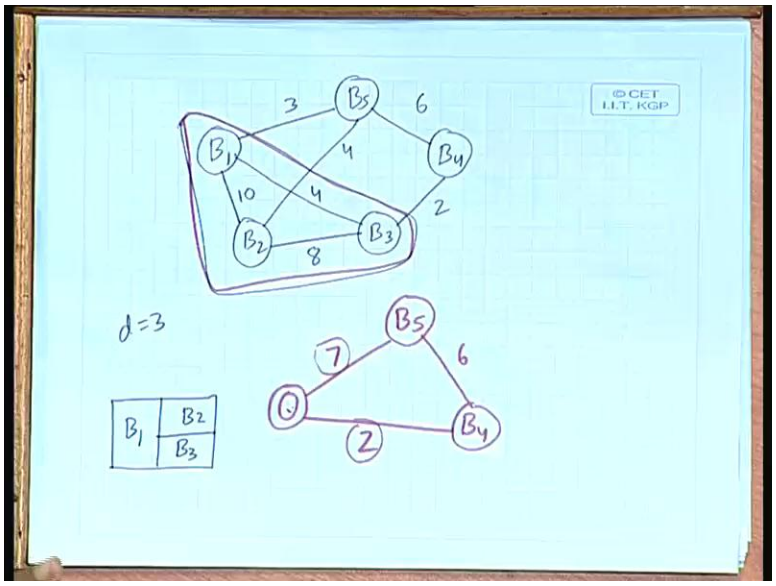
- A Greedy Procedure
  - Sort the edges in decreasing weights.
  - The heaviest edge is chosen, and the two modules of the edge are clustered in a greedy fashion.
    - Restriction: number of modules in each cluster ≤ d.
  - In the next higher level, vertices in a cluster are merged, and edge weights are summed up accordingly.

B1
10
B6
size d
B2

So you sort the edges instead of searching the graph every time and finding d blocks which are strongly connected that itself we will require some amount of searching. So you can have a simpler way of doing that like you sort the edges between nodes in the order of decreasing weights you choose the heavier stage the edges are in decreasing order. Suppose you find that b1 and b6, they are connected using the strongest edge with edge weight 10 say. So these are the two you put in the cluster. Then systematically you try to add more blocks to the cluster which are most strongly connected to this set b1, b6. So this you go on doing successively till you get a cluster of size d.

So you take the heaviest stage the two modules of the edges are clustered in a greedy fashion and in this way you just add some more to this which are strongly connected to this pair it becomes three. Then to that three you just add another block. It becomes 4. So the next high level this set of d blocks will become a super block and their weights to the other blocks will be the sum total of weights of the constituents. Like from here, you have a connection to b2. So you will the weight of this edge will be sum of the weight b1, to b2; b6 to b2 and all other nodes similarly. So the edge weights will be summed up accordingly. So even at the next level of iteration you do the same thing repeatedly. Right, okay. B2 we are not finding.

(Refer Slide Time: 28:58)

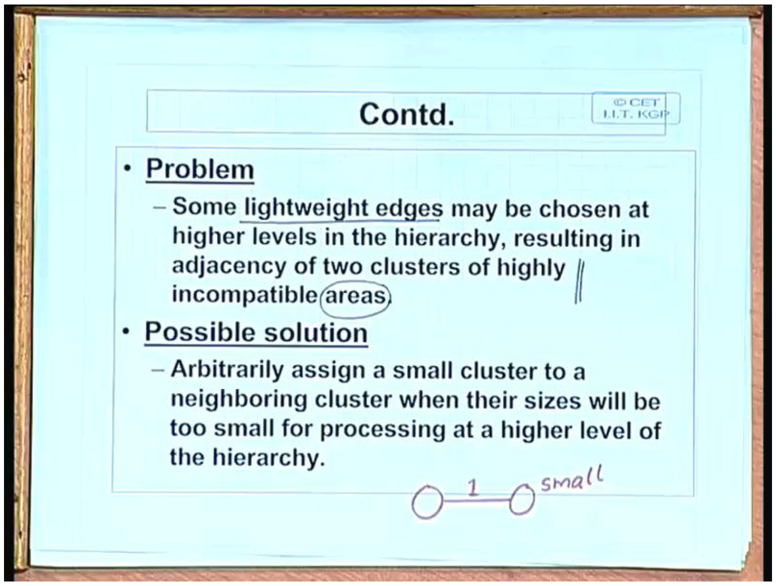


We are saying that what we are saying is that suppose you have a graph containing b1, b2, b3, b4 and b5. Say in the say our d is 3, d equal to 3. So you select b1, b2 and b3 in the first step. Because they were most strongly connected they this weight it was 10, this weight it was 8 and say this weights it was 4 say. Suppose the other weights were 3, 2, 6, 4, so this has now become a super vertex. So in the next level, this super vertex will be there in addition we will be having b5 and b4. B5 and b4 are already connected with an edge of weight 6. Now this will be connected to b5 with a weight which will be equal to 3 plus 4, 7 and this will be connected to b 4, with a weight well only 2, there is no edge b2, b4 or b1, b2.

So this new weights will have to be calculated as the sum total of the weights of the edges from b5 to 2. Means each of the nodes in this cluster. Right. So in this way you go on repeating [student noise Time: 30:44] information lost is [student noise Time: 30:49]. You see that information will be lost anyway. Because once we have chosen to merge these three into one. Which means we have already said that we have finalized a portion of the floor plan where b1 is here and b2 is here and b2 is here b3 is here. This we are not going to disturb anymore. So now if you say that you will be placing b5 somewhere here. [student noise Time: 31:20] Yeah I get your point. I get your point. I get your point. [student noise Time: 31:24] True. B5 may be placed adjacent to b3, may be yes true.

But at least they are placed means you can have somewhat close together may not be they are having a common boundary between them. But they are placed in closed proximity if you look at it, but mean you are right. So means, so means we are just we are treating this a black box and that means we are loosing some information regarding the details. Details mean this b5 is connected to precisely which block. Okay. Yes we are losing that. But our idea is that since we are taking the heavier edges before. So all the stronger things are together the weaker things may be as slightly far the apart. Yes. But this method suffers from some drawback. As we will show with an example.

(Refer Slide Time: 32:25)



Contd.

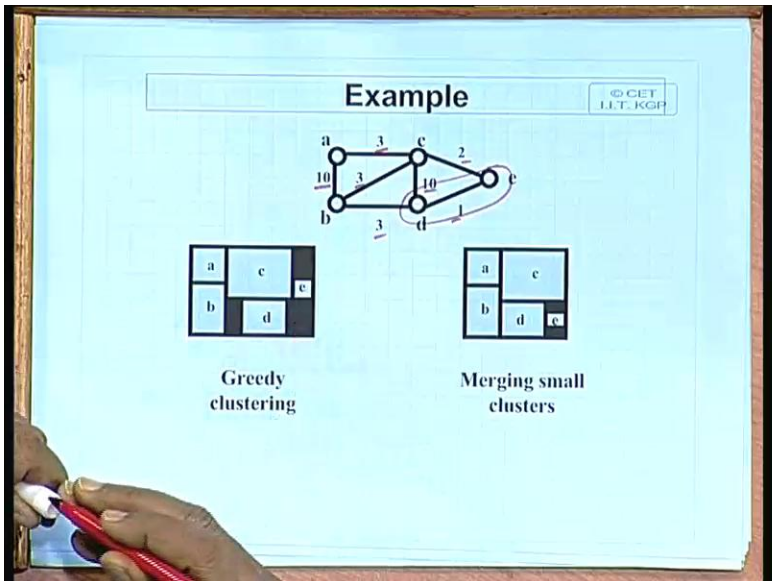
- **Problem**
  - Some lightweight edges may be chosen at higher levels in the hierarchy, resulting in adjacency of two clusters of highly incompatible areas
- **Possible solution**
  - Arbitrarily assign a small cluster to a neighboring cluster when their sizes will be too small for processing at a higher level of the hierarchy.

© CET I.I.T. KGP
1
small

See here we are following a greedy approach that all the stronger nodes or the heavy weight edges you are taking at the beginning towards the leaf. And as we move up the weights of the edges are decreasing. So there may be some so called light weight edges which we have chosen, we selecting at a much higher level. Some module which is very weakly connected to other that we are selecting at much later stage. Right. Now this may lead to adjacency of two clusters of highly incompatible areas leading to a gross wastage of total area that we will see with an example. I will show an example there can be wastage of area in terms of the floor plan. Now a possible solution suggested is just a heuristic.

So he says that if the weight of an edge is too small, then at the beginning itself you assign it to one of its neighbor. So that, whenever that neighbor is assigned, it will also take it with him. So it is not that these two nodes are strongly connected rather these two nodes are very weakly connected. But one of them is small this is the point to note one of them is a small cluster. It contains very few number of gates inside if one of them is a small cluster then allows that small cluster to piggy back on a bigger one which is adjacent to it may be with a very low weight edge. So arbitrarily assign a small cluster to a neighboring cluster when their sizes will be too small for processing. So let us take an example to illustrate. This what you mean by this.

(Refer Slide Time: 34:25)

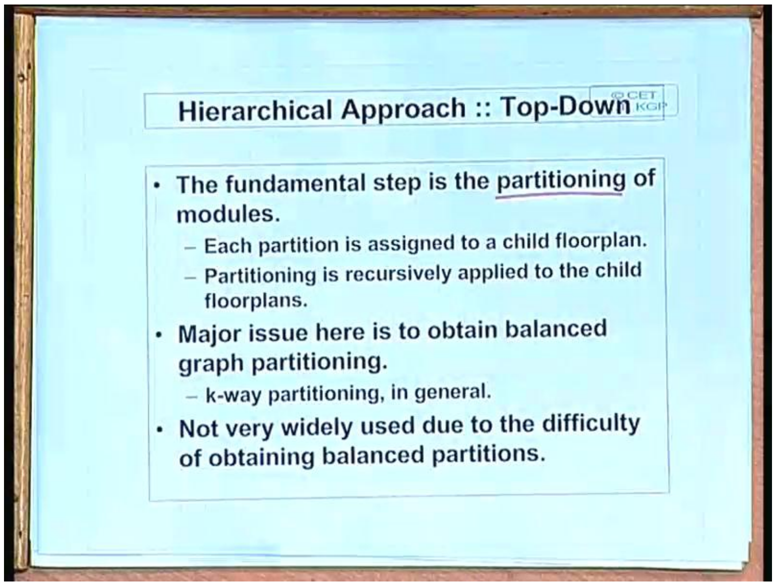


Example

Well, this is a given graph a, b, c, d, e, with the edge weights as 10, 3, 3, 3, 10, 2 and 1. So according to the greedy approach this, a and b will be merged first. So also will be the c and d. So a, b and c, d will go down together then 3, 3, 3, 9, they will be put together a b besides c d. Now since e has a weight of 2 and one to the others. So e will be taken last. But if e is very small cluster the placement will possibly or the floor plan will possible be something like this. So here you are wasting too much space. But in terms of that heuristics what it says, if it is a low weight node you try to assign it some other node which is adjacent to it, but may not be with a very high edge with a low in fact. Say means say means a low edge weight which will be possibly cause it to be handled at a much later stage under normal circumstances.

But what I do is that let e and d come together at beginning itself. So here whenever you are assigning c and d together you are assigning actually c and d e together. Since d is smaller than c, possibly e can sit side by side with d. So the total floor area will be less. So otherwise if you follow a purely greedy approach you may land up wit the floor plan like this. Now at the area will be more right. [student noise Time: 36:16]. That information we given now be because when you are doing a floor planning. I am saying that you are given the blocks b_i b_j. So each block will be having some width or height or in terms of gates number of gates some information is given. Because here bottom up is a natural approach. But you can also think of a top down approach for floor planning.

(Refer Slide Time: 36:41)



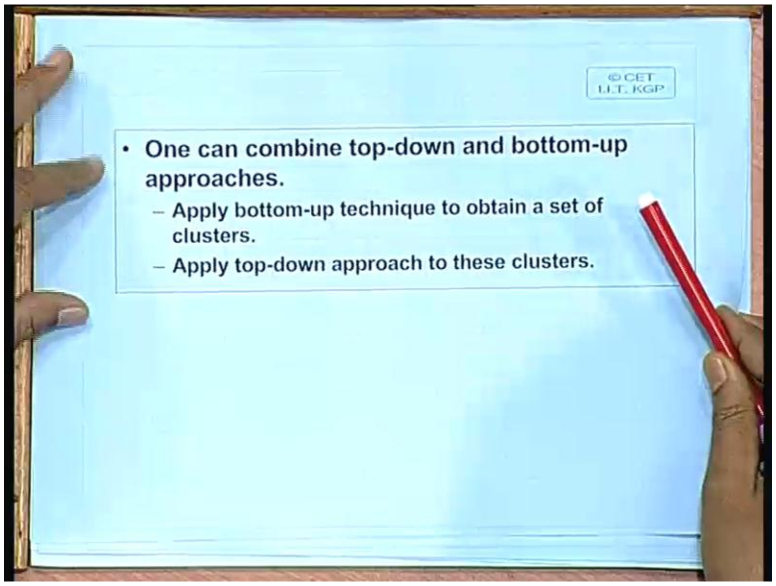
Hierarchical Approach :: Top-Down

- The fundamental step is the partitioning of modules.
  - Each partition is assigned to a child floorplan.
  - Partitioning is recursively applied to the child floorplans.
- Major issue here is to obtain balanced graph partitioning.
  - k-way partitioning, in general.
- Not very widely used due to the difficulty of obtaining balanced partitions.

See here from the top down approach, well you are actually doing partitioning and floor planning together in this case. Given the overall netlist you are carrying out partitioning that is from the top you are doing. From the total netlist you are doing partitioning to smaller netlist again doing partitioning to smaller netlist. So if we create the floor plan from that order itself, so as you are doing partitioning you are also planning that means on floor where we will be placing them this is so called top down approach. So each partition you are making through partitioning is assigned to a child floor plan and you are you are doing this partitioning recursively to each of these child floor plans so as it becomes smaller and smaller. But the problem here is that well there are good algorithms for partitioning a given.

You can say graph into two parts like you can use Kernighan-Lin algorithm. But best result will be obtained if we can do a k-way partition, in general for most floor plans if we instead of 2, if you can divide into 4, 5 or 6 and accordingly place it at high level then break each of them up. Because if you break it up into two, most of the time you will not get a very efficient floor plan. But the problem is that k way partitioning is a difficult sub problem. In general you do not have very good algorithms for doing very good balanced k way partitioning balanced partitioning is not available. So actually in practice we never do top down approach to floor planning like this, rather we can do some kind of a combination.

(Refer Slide Time: 38:38)

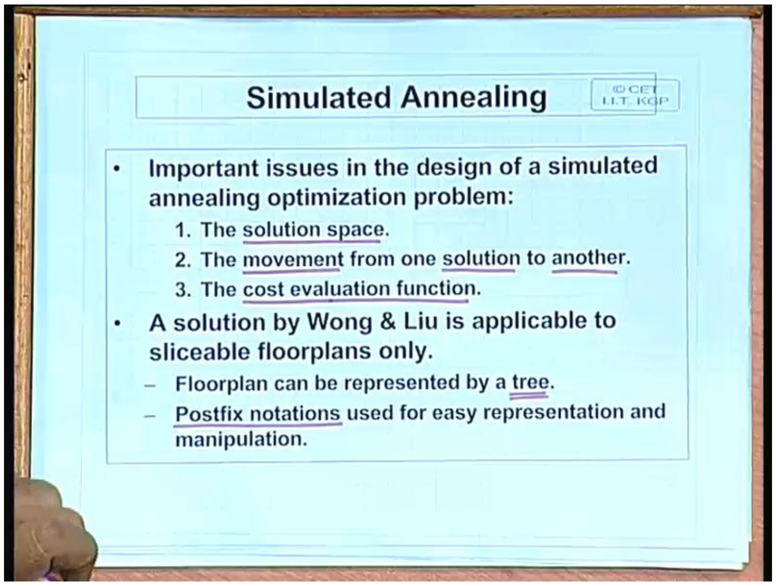

© CET
I.I.T. KGP

- One can combine top-down and bottom-up approaches.
  - Apply bottom-up technique to obtain a set of clusters.
  - Apply top-down approach to these clusters.

What kind of a combination? See, you start with the bottom up approach. From the given blocks b1, b2, b3 at the lowest level, you go bottom up and beyond a point you will get a certain bigger clusters. Okay. Now these clusters will be a will be just union of certain blocks. Now you take these clusters one at a time and again do a top down approach. Well there is no guarantee that this will always give a good approach because the bottoms up you are doing using a greedy thing. So using a greedy thing you are merging 10 blocks to get a cluster. Now again you take that cluster using some good partitioning algorithm try to see if you can do something better; from top again you move towards the bottom. But this is rarely used.

People typically use a pure bottom up approach with some heuristic to improve upon the solution or some kind of simulated annealing or genetic algorithm as I had said. So this particular method of hierarchical approach to floor planning, as we have seen that, so essentially you combine the blocks systematically and at each step you also keep track of some information regarding the floor plan you have generated so far. So you go on merging the floor plan to higher and higher levels. Now next we just finally look at the method of simulated annealing. So simulated annealing. Well I am repeating this approach is used in many steps of physical design automation.

(Refer Slide Time: 40:41)



Simulated Annealing

© CET I.I.T. KGP

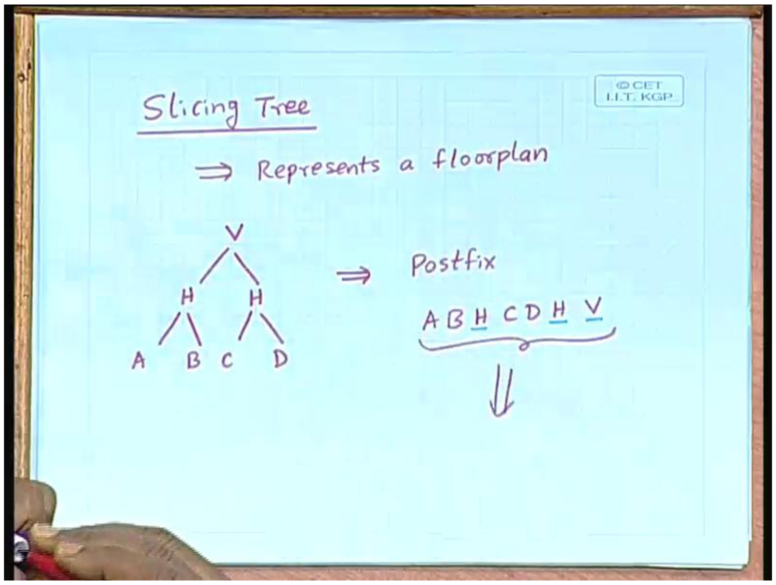
- Important issues in the design of a simulated annealing optimization problem:
  1. The solution space.
  2. The movement from one solution to another.
  3. The cost evaluation function.
- A solution by Wong & Liu is applicable to sliceable floorplans only.
  - Floorplan can be represented by a tree.
  - Postfix notations used for easy representation and manipulation.

Because of the large search space involved and the way well this problems have been explored using simulated annealing and it is found to work very well to give very good results. Now in the floor planning problem also simulated annealing have been tried out by many researcher by many people. There are been several different ways of approaching or means of some modeling the problem defining the moves evaluating the cost function they have been evaluated. But what we would be discussing is a very interesting approach which was proposed by some gentleman Wong and Liu.

This is some kind of an indirect approach in a sense. Well you recall that for the simulated annealing method you require. Well of course this solutions space to be explored some way of representing the solution. You have to define some cost functions and some moves the moves will allow you to go from one solution to another and the cost function will allow you to evaluate the solutions. So I am repeating, so if it is a better solution you are moving to you always accept if it is a worst solution you accept with a certain probability. Now here we have seen earlier that for a floor plan we can have a slicing tree structure to represent the floor plan.

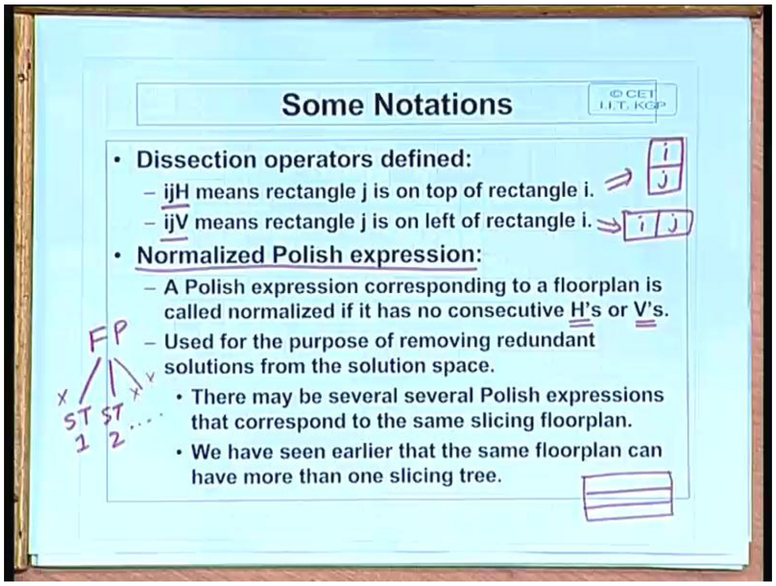
While the method we are discussing right now is applicable only to sliceable floor plan. But it can be extended to include also the wheel structure. Right. Now we are assuming that it is a purely sliceable floor plan. So the floor plan can be split only using only using horizontal and vertical slices and you have a corresponding slicing tree. Now this particular approach what it says is that for every such slicing tree, well you can represent the tree in certain ways here. They use the post fix notation of representing the tree. And all the manipulation regarding movements they had applied on the post fix notation.

(Refer Slide Time: 43:20)



So actually this slicing tree whatever I am talking about this slicing tree is a data structure which actually represents a floor plan. Okay. Now say suppose you have a slicing tree like this. Let us take an example vertical, horizontal, horizontal, A, B, C and D. So what I am saying is that there is a tree this can be represented in some standard notation. Here we are using the polish postfix notation. This tree in polish postfix notation will be A, B, H, C, D, H, V, where H and V are the operators A, B, C, D are the operators. So what I am doing is that in order to move from one floor plan to the other, we define moves which are actually some perturbations or modifications on this postfix structure itself. So we modify this postfix structure in such a way that we get another postfix structure which represents a move, which is actually that will be corresponding to some other slicing tree which in turn will be corresponding to some other floor plan. Right. So just let us introduce some notations first before going into that method. So as I have shown in this illustration H and V are the operators.

(Refer Slide Time: 45:17)



Some Notations

© CET I.I.T. KGP

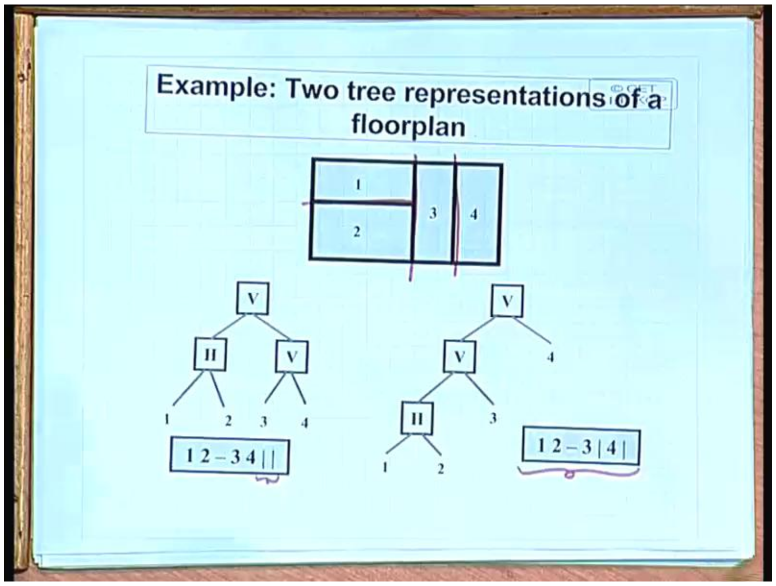
- Dissection operators defined:
  - ijH means rectangle j is on top of rectangle i.
  - ijV means rectangle j is on left of rectangle i.
- Normalized Polish expression:
  - A Polish expression corresponding to a floorplan is called normalized if it has no consecutive H's or V's.
  - Used for the purpose of removing redundant solutions from the solution space.
    - There may be several several Polish expressions that correspond to the same slicing floorplan.
    - We have seen earlier that the same floorplan can have more than one slicing tree.

So in the polish notation the notation used are i, j, h which means rectangles i and j are sliced using a horizontal line which means i's on top of j's, this means i is on top of j something like this. Similarly i, j, v means i is on the left of j, there is a vertical slice right. And due to some reason which we would be understanding thoroughly that we only consider normalized polish expression in this problem. Normalized polish expression means a polish expression which does not have consecutive h or consecutive v. This example which you have taken this is a normalized expression because h and v's are nonadjacent 2h. There is no 2h or no 2v together.

So a polish expression will be called a normalized if there are no consecutive h's or v's. Roughly speaking, this is used for the purpose of removing some kind of redundancy in terms of the solutions. See ultimately we want to explore the solutions space and we will see that corresponding to the same floor plan, there can be several different slicing trees slicing tree1, slicing tree 2 and so on. Some of the slicing trees will be corresponding to normalized polish expressions. Some of these will be corresponding to non-normalized polish expressions. For a vast majority of floor plans, there will exist normalized polish expressions.

Well, with the exception of trivial floor plans like this that two horizontal slice for one after the other. So for this you will not get a normalized expression. Okay. But for others you will get some slicing tree which corresponds to a normalized expression just in order to reject the other ones to reduce the search space, we decide to concentrate on normalized expressions only. Because for vast majority of floor plans there will exists atleast one solution which will have a normalized expression. Now by ruling out the non-normalized one un-normalized one's we are limiting our search space, that's all okay. The same floor plan can have more than one slicing tree; let us take an example.

(Refer Slide Time: 48:16)



Example: Two tree representations of a floorplan

Here so this is a simple example with three sub blocks separated by two vertical slices but one of them also separated by horizontal slice. Now you see that this particular floor plan can be represented by two alternate sequence of slices. Well in the first one the first slice is this vertical slice in the middle. Then in one side you make a horizontal slice and the other side you make a vertical slice. So the polish expression will be 1, 2, H. Well some times while representing h is represent by dash v is also represented by a bar which means horizontal cut and a vertical cut. So 1, 2, H, 3, 4, V, V. So there are two vertical operators which are consecutive. So this is not a normalized representation.

But in the other one you start with this slice first. First cut this, then cut this, then cut this. So here you will be having one two horizontal, three vertical, four vertical. So you look at this expression you will find that there are no consecutive v or h operators. So here the idea is that v well for most of the floor plans we will have solutions like this. So in the algorithm that we will be discussing we will be explaining it in the next class. So there we will be seeing that we would be looking at only normalized expressions and we would be making some moves. And another thing while we are defining the moves we will see that there will be not a single way of defining move there will be three alternative moves.

You will be defining you may be selecting one of them at random. Now out of those three two of them will automatically leave this expression into a normalized one. But only the third one can make it un-normalized. So we will have to make some special check for the third move if you choose to apply that. So for the third move if you choose to apply that we will just we have to make it well if you see that the move if we apply makes the equation non normal un-normalized. We will not accept that move we will select some other move at random. So in our next class the first thing is that we would be discussing some more details of this algorithm regarding the moves we will also illustrate the algorithm with an example.

Then we will be moving on to a related you can say related sub problem. Well we had looked at floor planning. Next we would be looking at a related sub problem that is called a pin assignment. Let us say for some of the blocks which are already designed the positions and the definitions of the pins are already fixed. But for the blocks which are flexible there is some flexibility which we have I can put pin number say a and b like this. Or I can even reverse them. So I will try to optimally place the pins on the boundary of the block so that the routability and the routing cost get minimized. Right. So this all this things we will be discussing in our next class. Thank you.