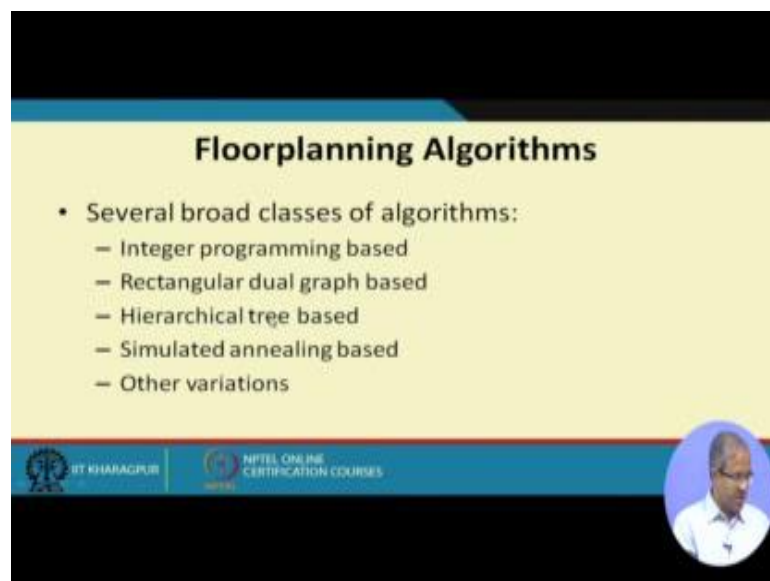


**VLSI Physical Design**  
**Prof. Indranil Sengupta**  
**Department of Computer Science and Engineering**  
**Indian Institute of Technology, Kharagpur**

**Lecture - 09**  
**Floor Planning Algorithms**

So, in this lecture we shall be talking about some of the algorithms for floor planning.  
So, floor planning algorithms.


(Refer Slide Time: 00:35)



**Floorplanning Algorithms**

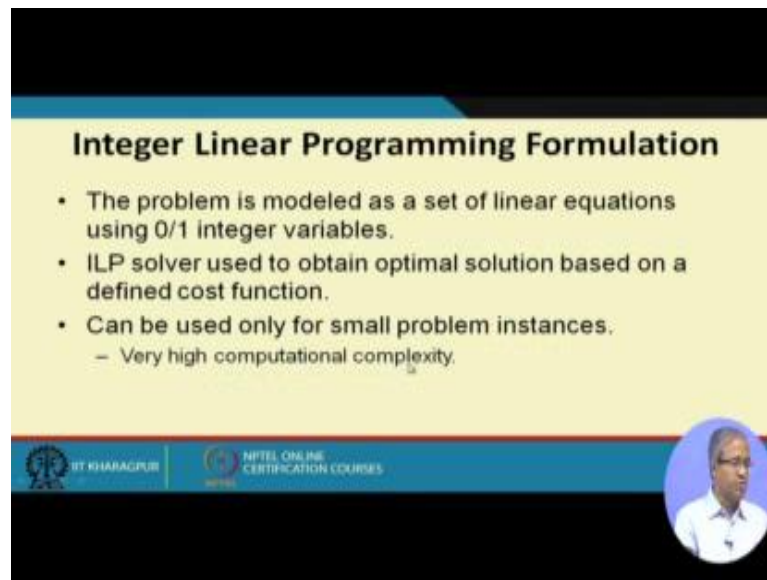
- Several broad classes of algorithms:
  - Integer programming based
  - Rectangular dual graph based
  - Hierarchical tree based
  - Simulated annealing based
  - Other variations

IIIT KHARAGPUR | NPTEL ONLINE CERTIFICATION COURSES



So, we shall be briefly looking at several broad classes of algorithms like we will see that some of them are integer, linear programming based. Some of them are based on some particular data structures like rectangular dual graph, then hierarchical tree. Simulated annealing is a very you can say common method which gives very good results and there are some other variations.


(Refer Slide Time: 01:06)



**Integer Linear Programming Formulation**

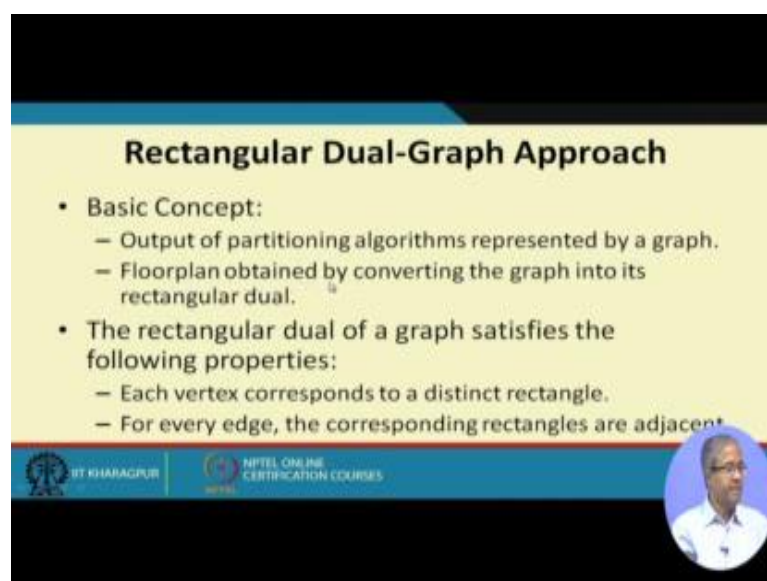
- The problem is modeled as a set of linear equations using 0/1 integer variables.
- ILP solver used to obtain optimal solution based on a defined cost function.
- Can be used only for small problem instances.
  - Very high computational complexity.

IT KHARAGPUR | NPTEL ONLINE CERTIFICATION COURSES



So, integer linear programming we shall not go into much detail; basic idea is that we model the problem as a set of linear equations, not only that we also specify some cost function some objective function that is to optimize, and we give it to an ILP solver. The ILP solver will provide us with the optimum solution, because ILP solver whatever solution it gives it is the best possible for the optimum solution. So, the optimality of the solution is guaranteed, but; however, the computational complexity is every high, and therefore, it can used only for very small problem instances; so for practical problems you cannot really use this.


(Refer Slide Time: 02:00)



**Rectangular Dual-Graph Approach**

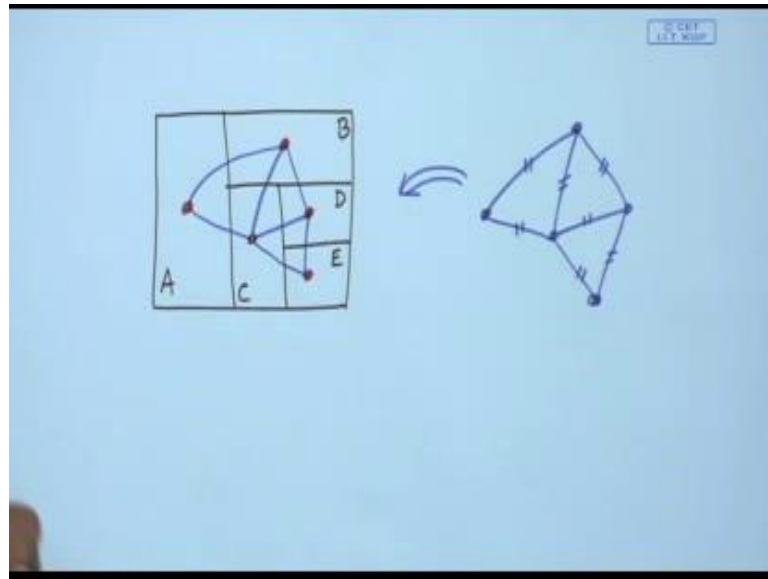
- Basic Concept:
  - Output of partitioning algorithms represented by a graph.
  - Floorplan obtained by converting the graph into its rectangular dual.
- The rectangular dual of a graph satisfies the following properties:
  - Each vertex corresponds to a distinct rectangle.
  - For every edge, the corresponding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

IT KHARAGPUR | NPTEL ONLINE CERTIFICATION COURSES



Rectangular dual graph approach this is very interesting approach, you can see here we utilize the linkage of floor planning with partitioning, typically when partitioning is carried out the output of the partitioning process is represented by graph. Floor plan is obtained by converting the graph into its rectangular dual. Now let us first very quickly see what this rectangular dual means.

(Refer Slide Time: 02:38)



Let us take an example let us start from the other way round let us start with the floor plan. Suppose I have floor plan like this and these are the different blocks A B C D and E. So, when we talk about rectangular duality, the idea is that every rectangular region in this diagram or in this figure will correspond to a vertex. So, which I am shown like this, there will be 5 vertices; and every pair of blocks which have a common boundary, like you see A and B is having a common boundary. So, there will be an edge connecting these two vertices. A and C is having a common boundary so there will be an edge connecting this. B and D is sharing a common boundary; D and C is sharing a boundary, D and E is also sharing and C and E is also sharing, not only that B and C is also sharing.

But here so we have drawn this graph from the floor plan; but actually what is done is the other way round. You will be given this graph, which will be the output of the partitioning problem that is what we just now said. So, the graph will be like this then from this graph you will have to translate it into a floor plan. Now it is not difficult to do it given a planar graph like this you can convert it with a rectangular dual replacing every

vertex by a rectangular region and you will get the floor plan like this. Now this is the output of the partitioning problem, where each vertex will indicate the blocks and the edges will indicate the number of connections between them; the weights of the edges show how strong they are connected, this is the idea behind rectangular duality.

So, just as you have seen the rectangular dual of a graph satisfies the properties, each vertex in the graph will correspond to a distinct rectangle in the floor plan, and for every edge the corresponding rectangles they are adjacent to will be sharing some horizontal or vertical segment.

(Refer Slide Time: 05:33)

**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)*.

The slide includes a diagram of a floor plan with five numbered rectangles and its corresponding dual graph. The floor plan has rectangles 1, 2, 3, 4, and 5. The dual graph has vertices 1, 2, 3, 4, and 5. Edges connect (1,2), (1,3), (1,4), (2,5), (3,4), (4,5), and (3,4).

NPTEL ONLINE CERTIFICATION COURSES


Now, this is another example you are shown, this a floor plan and this is the rectangular dual graph. Now one property you can see, this you can work out with other examples by hand yourself also that you take any such planer floor plan rectangular floor plan, you convert it to the dual graph, you will see that the dual graph will consist of a set of triangular regions this is called a planar triangular graph.

Planar means it can be laid out on a plane, without the edges crossing each other. So, planar means you can layout the graph on a plane surface, the edges will not be crossing each other; because the floor plan was planer, the graph you are embedding on it will also be a planer graph. So, it is automatic, and all the faces will be triangular in nature this is the property. So, this graph will essentially be a planer triangular graph, which you have to translate into a floor plan.


(Refer Slide Time: 06:49)

**Contd.**

- Every dual graph of a rectangular floorplan (without cross junction) is a PTG.



- However, not every PTG corresponds to a rectangular floorplan.



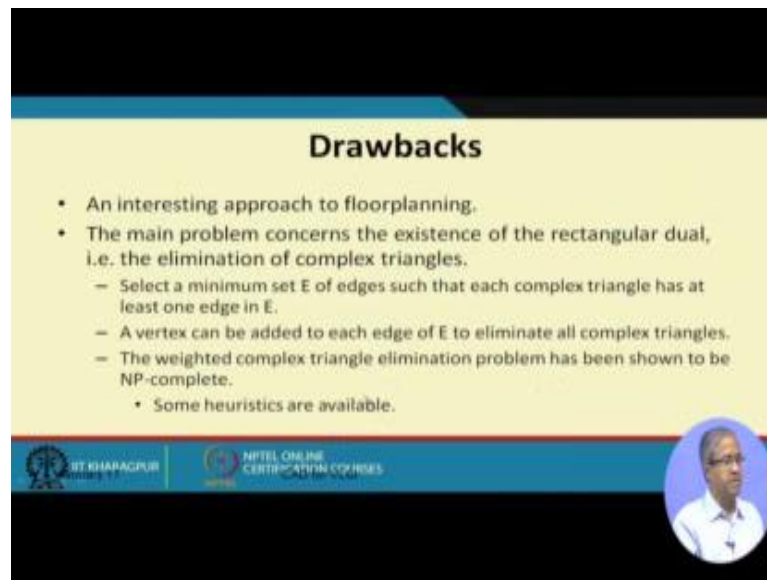
**Complex triangle**

IT KHARAGPUR | NPTEL ONLINE CERTIFICATION COURSES

Now, there are a few things; now if you have a floor plan with a cross junction see this will not correspond to a planer triangular graph, because 1 and 2 are adjacent, 1 and 4 are adjacent, 2 3 and 3 4. So, it will look like this, but 1 3 and 2 4 they do not share any horizontal and vertical line. So, if we have a situation like this, you do not get a PTG. But if you have a situation like this so what you can do if you slightly shift this edge 3 4 if you shift it slightly like this, then there will be a sharing between 2 and 4 there will be an edge between 2 and 4 coming in right.

So, PTG is a property, where you which you must have an from PTG you can translate it into this graph. Now there is one problem though that every planer triangular graph cannot be translated into a floor plan, this is the one and only one problem case this is called a complex triangle. This actually indicates a complete graph of 4 vertices drawn like this, there are 3 triangular faces. Now you can try it out in any way you will see that you cannot draw a floor plan for which the corresponding dual graph will be this. So, this complex triangular is one difficult problem in this particular approach whenever you encounter it in a graph.

(Refer Slide Time: 08:32)



**Drawbacks**

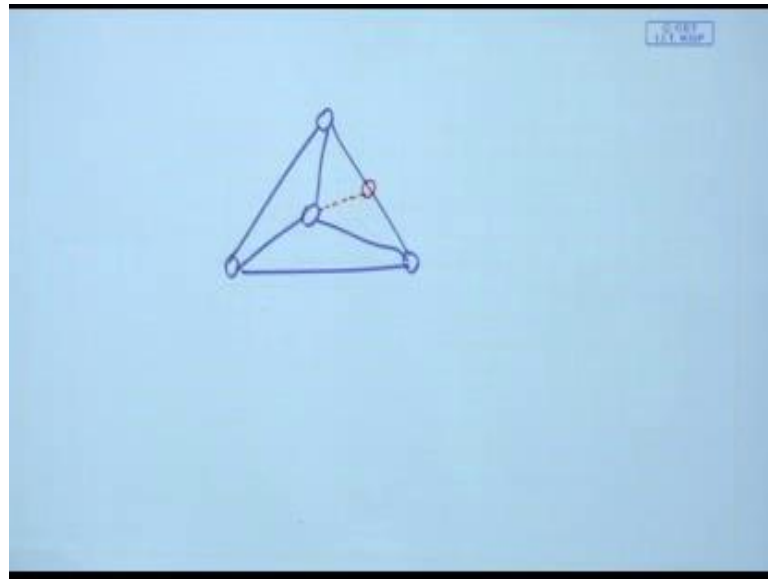
- An interesting approach to floorplanning.
- The main problem concerns the existence of the rectangular dual, i.e. the elimination of complex triangles.
  - Select a minimum set  $E$  of edges such that each complex triangle has at least one edge in  $E$ .
  - A vertex can be added to each edge of 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

IT BHARAGUR | NPTEL ONLINE CERTIFICATION COURSES

So, let me just tell you first. So, here what your saying is that we start with that graph which our partitioning algorithms is giving us some blocks, and their interconnections indicating their strength of connections. Now if we have the planner triangulations in that graph, you can map it to a layout or a floor plan. So, this you will always be having, but if you have a complex triangle there you cannot map that sub graph into a rectangular dual into a floor plan.

So, this is one of drawback, there is an interesting approach to floor planning where we start with a graph representing a partition, and from there we translate it into a floor plan; but the main issue is the issue of complex triangle. So, what we do one possible heuristic that has been proposed is something like this, you consider the complete partitioning graph, you identify all complex triangles that exists there you select one edge from each of those complex triangles, and in each of edges you add one edge, which means it is something like this. Let say your complex triangle look like this.

(Refer Slide Time: 10:00)



So, what we are saying is that in anyone of the edges you add a vertex and we add a dummy edge here. So, this is no longer a complex triangle. So, you that complex triangle problem solved if I add dummy vertices like this. So, the problem solution suggest is you add such dummy vertices, but again determining the minimum setup of weight this kind of edges, that can eliminate all complex triangle is also proved to be a difficult problem, but however, some heuristics are available and some tools I means have actually used this kind of heuristics to create a floor plan right.

(Refer Slide Time: 11:00)

### Hierarchical Approach

- 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.

BY KHARAGPUR

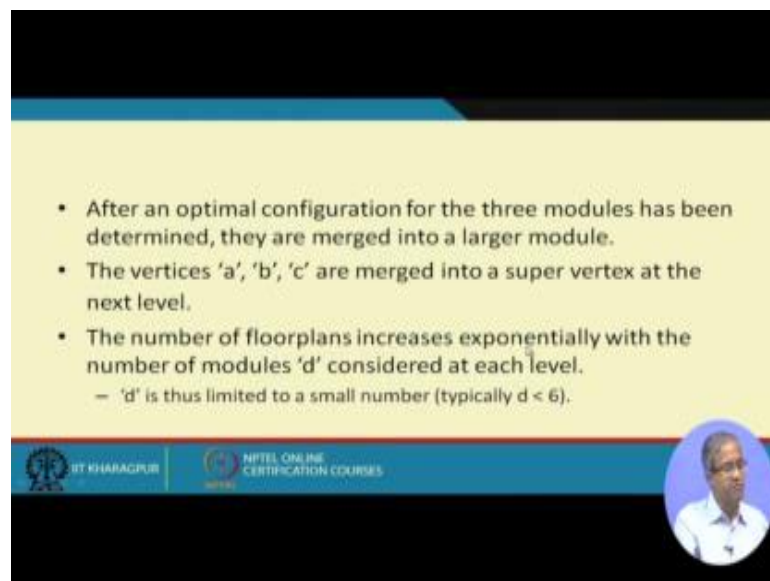
NPTEL ONLINE CERTIFICATION COURSES



This hierarchical approach is an interesting approach, which also is quite commonly used. So, here what we say is that, we use some kind of a divide and conquer approach; that means, we divide the problem into smaller sub problems, which are easily manageable like we consider a small graph with 3 vertices; that means, 3 blocks. So, these 3 blocks can be placed in 3 possible ways. So, what we saying is that, if we have a small set of blocks we have a graph like this, we have the properties of a b and c what are their areas and their properties how they are connected. So, you can explore the alternate floor plans and find out which one of these 3 is the best.

Supposes you find that the last one is the best. So, you freeze on this and you merge this a b c into a single micro node, which means you have already solved this sub problem; a b c can be best laid out in this fashion right the other two you ignore this is the basis idea.

(Refer Slide Time: 12:27)



- 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.
- 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  $d < 6$ ).

So, after the optimal configuration for the 3 modules are determined as I said you merge them into a single module. So, the next higher level that single so called super vertex will be there, and you go on repeating this. But the problem here is that as I had said you are considering small number of vertices at each steps, but how small? 3 4 5 6 well it is seen that the number of such possibilities they increase very rapidly with the number of vertices.



So, typically we limit the number of nodes to 5 not more than that, the number of possible floor plans increases exponentially with  $d$ ;  $d$  is the number of vertices in the graph.

(Refer Slide Time: 13:17)

• All possible floorplans for:

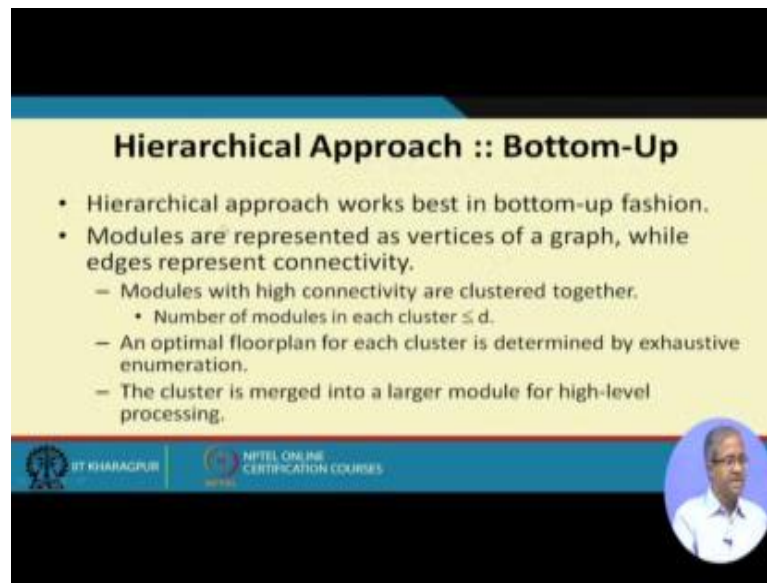
$d = 2$

$d = 3$

The slide displays two rows of floorplan diagrams. The first row, labeled  $d = 2$ , shows two diagrams: a single horizontal rectangle and a rectangle divided into two vertical sections. The second row, labeled  $d = 3$ , shows six diagrams: a single horizontal rectangle, a rectangle divided into three vertical sections, a rectangle divided into two horizontal sections, a rectangle divided into two vertical sections with the top-left section further divided horizontally, a rectangle divided into two horizontal sections with the top-left section further divided vertically, and a rectangle divided into two vertical sections with the top-right section further divided horizontally. The slide also features a logo for BIT KHARAGPUR and NPTEL ONLINE CERTIFICATION COURSES, and a small circular inset image of a man in a white shirt.

So, just an example for  $d$  equal to 2 these are the 2 possibilities, for  $d$  equal to 3 you can have 6 possibilities, in this way it goes very rapidly as we move to  $d = 4$  or  $5$ . So, the idea is that if you have a graph of that size, you try to find out explore all possibilities and find out the optimum configuration of the floor plan. Take it; freeze it and merge all the vertices into a single super vertex for the next level right.


(Refer Slide Time: 13:52)



**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  $\leq d$ .
  - An optimal floorplan for each cluster is determined by exhaustive enumeration.
  - The cluster is merged into a larger module for high-level processing.

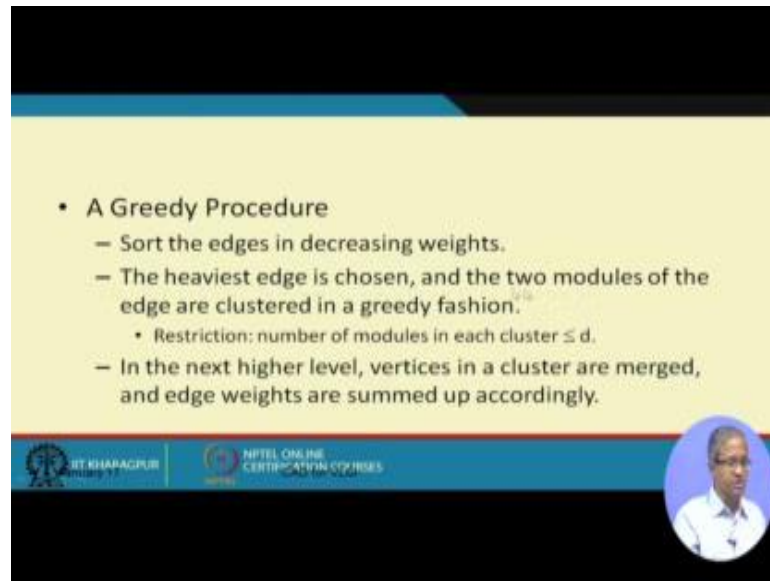
IT KHARAGPUR | NPTEL ONLINE CERTIFICATION COURSES



So, bottom up is the natural approach in this hierarchical method. So, modules we represent as vertices in the graph, while the edges will indicate connectivity the edge weights. So, the modules which have higher weights the pairs of modules, they will be clustered together; with the restriction that number of modules in each cluster will be limited to  $d$  this  $d$  is our limit, like the idea is this we have a graph each vertex indicates our blocks, the edges the weight of the edges indicates how many connections are there between the blocks.

Suppose I select a set of  $d$  such nodes which are strongly connected, you take that set of  $d$  nodes, place them optimally because you know all possibilities, explore all the possibilities, find out which floor plan configuration gives you the minimum cost take that, and replace those five nodes by a single micro node, but limit that number to  $d$  equal to 5. So, exhaustive enumeration is done at every step, this cluster is merged into a larger module for higher level processing repeat this process.

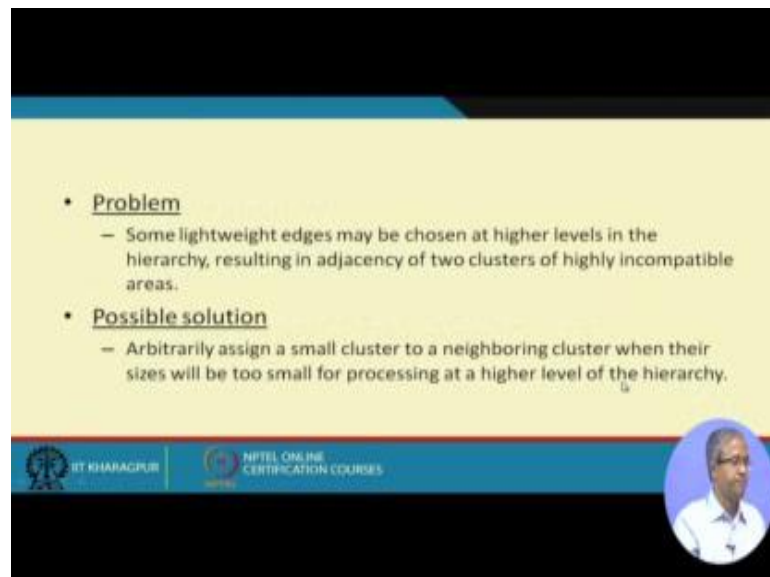
(Refer Slide Time: 15:17)



- **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  $\leq d$ .
  - In the next higher level, vertices in a cluster are merged, and edge weights are summed up accordingly.

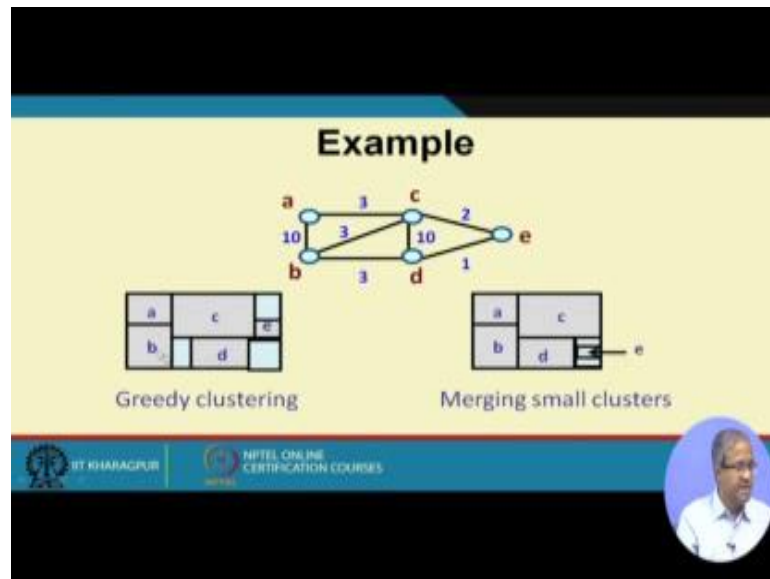
Now, this is of course a greedy procedure, where you sort the edges in decreasing weights typically the heaviest weight indicating the strongest connectivity, will typically be chosen first and the modules are clustered in a greedy fashion, and this is repeated I shall be showing this with an example that how this is done. So, let us take an example

(Refer Slide Time: 15:40)



- **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.

(Refer Slide Time: 15:43)



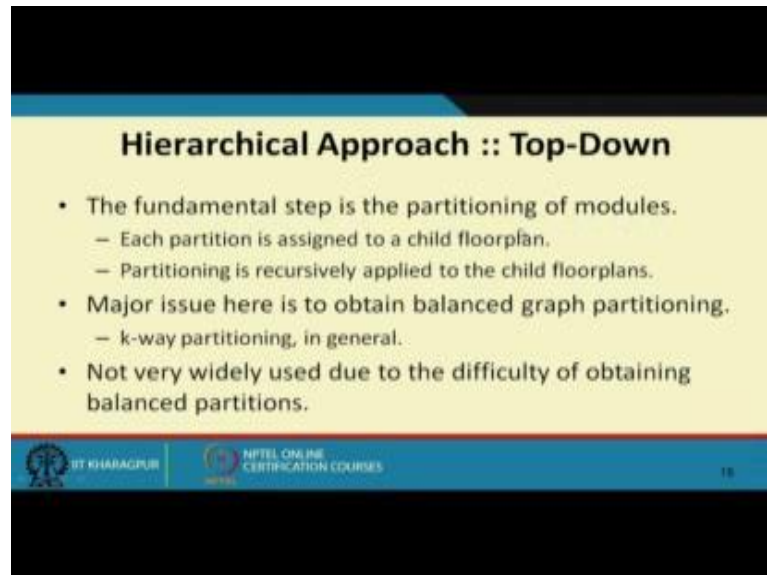
I just come back to this let us take this example first this is a graph with 5 vertices the weight of the edges are shown. So, you can see that in a greedy clustering approach the vertices which are strong most strongly connected like a and b, c and d you merge them together, you put a and b together, you put c and d together. So, once you do it you only are left with e. So, which is relatively mean weakly connected 2 and 1 with this clusters. Now what happens is that this a b gets connected together, c d gets connected together, but you do not have any where to place e. So, in this greedy clustering possibly you will be placed on the right. So, you will be getting a lot of empty spaces on top and bottom ok.

So, one heuristic that has been suggested is that this is a problem; some lightweight edges may be chosen at higher levels in the hierarchy resulting in adjacency of two clusters of incompatible areas that may lead to an increase in the layout area, like you see here. This e, this a b and c d are nicely compatible in terms of the height, so e you cannot place anywhere down. So, you have to place it on the right. So, this happens. Now this heuristic says; so if you see that a cluster is connected to a neighboring cluster with a very small weight, you club them together initially this is just a heuristic. The idea is that you see d and e are very weakly connected with a weight of one, you club d and e together at the beginning itself. So, d and e will be side by side ok.

Now, you use the rest a and b together, c and d together. So, here you will see that the solution we get will be more compact, because d and e because they are together you can place it just below c. But here because c and d are merging, d will be placed just to the

center of c below it. So, you do not have enough space to keep e either to the right or left, but here it happens; so this just a heuristic which give reasonably good solution.

(Refer Slide Time: 18:22)



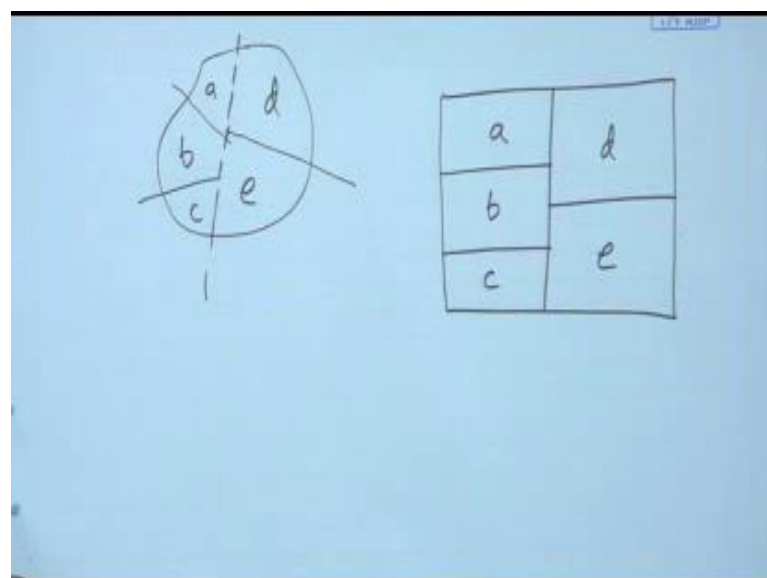
**Hierarchical Approach :: Top-Down**

- 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.

BIT KHARAGPUR | NPTEL ONLINE CERTIFICATION COURSES | 18

Now, top down is the other alternative. So, here you start from the partitioning process. So, you start from the whole netlist. So, the idea is this.

(Refer Slide Time: 18:38)

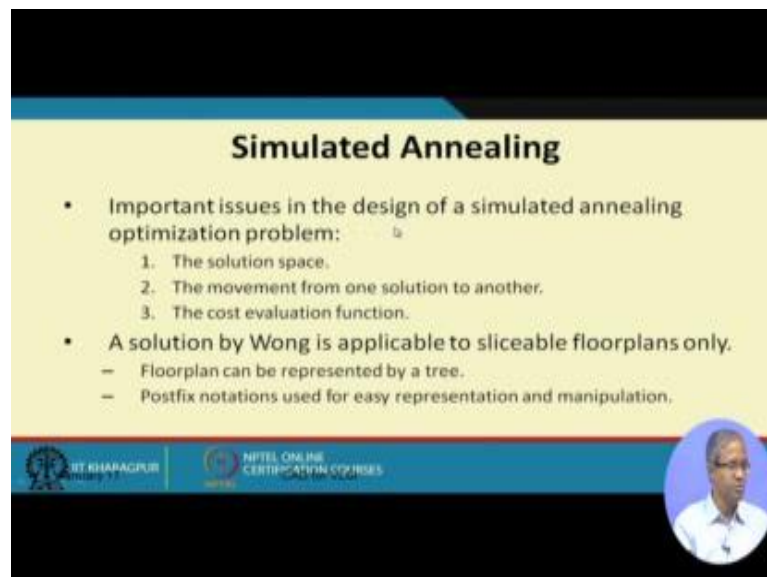


So, you have the entire netlist that corresponds to your complete floor plan, you partition this netlist into two parts, in the floor plan this corresponds to two parts.

You partition this into 3 parts let say, here itself you partition the floor plan into 3 parts, here you partition it into 2 parts, here you partition in 2 parts. So, there is a one to one correspondence, top down you are partitioning the netlist and at the same time in a corresponding fashion you are also partitioning the floor plan. So, you will be getting let say these are a b c d and e. So, you will be getting exactly where in the floor plan you will be placing them right. So, this is the top down approach.

But the major issue is that we have seen the Kanninen Leigh partitioning algorithm, which divides a graph into two parts, but here in general we need a method which can partition a graph into more than two parts; but such good algorithms are not very commonly available we have to use heuristics and approximation methods. So, k-way partitioning it is called. So, there are algorithms for k-way partitioning, which can be used here, which divides a netlist into k difference parts; but again the computation complexity to obtain good solution there is quite high; so that is one problem, because of that this is not very widely used balanced partitions is difficult to obtain, balanced means the partitions of equal sizes so it is difficult to get right. Simulated annealing is an interesting method.

(Refer Slide Time: 20:35)



**Simulated Annealing**

- 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 is applicable to sliceable floorplans only.
  - Floorplan can be represented by a tree.
  - Postfix notations used for easy representation and manipulation.

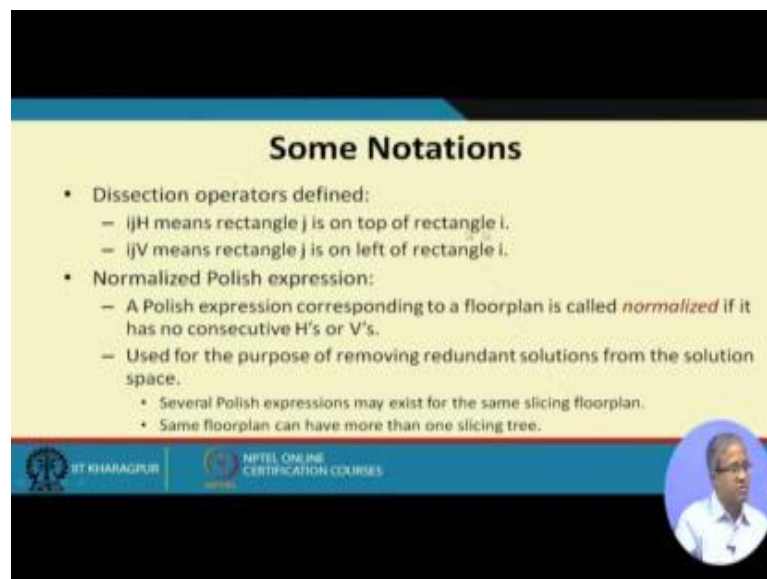
Logo of IIT BHARAGUR and NPTEL ONLINE CERTIFICATION COURSES are visible at the bottom of the slide. A small circular inset image of a man is also present in the bottom right corner of the slide.

So, simulated annealing is a very commonly used method which is used in VLSI care in other applications also, this modules the annealing process of metals. So, this is basically an optimization problem a solution to an optimization problem, where do you have a

solution we start with a solution, we have a set of moves that can take us from one solution to the other, and we have a cost function in which you can evaluate. This means for a given problem I have my solution, their said means that we have a solution representation. It is an iterative process; at every step we make some small changes in the solution these are called moves. After making the change we check whether we are getting a better solution or not, if we get a better solution there is a cost function we can check using that if we get a better solution we always accept it, but if you get a worst solution then also we may expect it, but that probability will decrease with time, this is called the annealing process or the cooling process in metals whatever it call. So, this is the basis idea behind the simulation simulated annealing algorithm.

So, in this method for floor planning the floor plan is represented by a tree, and the polish notation that I talked about earlier that postfix notation, that same kind of postfix notation is used for representation and also for manipulation of the moves. So, let see how.

(Refer Slide Time: 22:35)



**Some Notations**

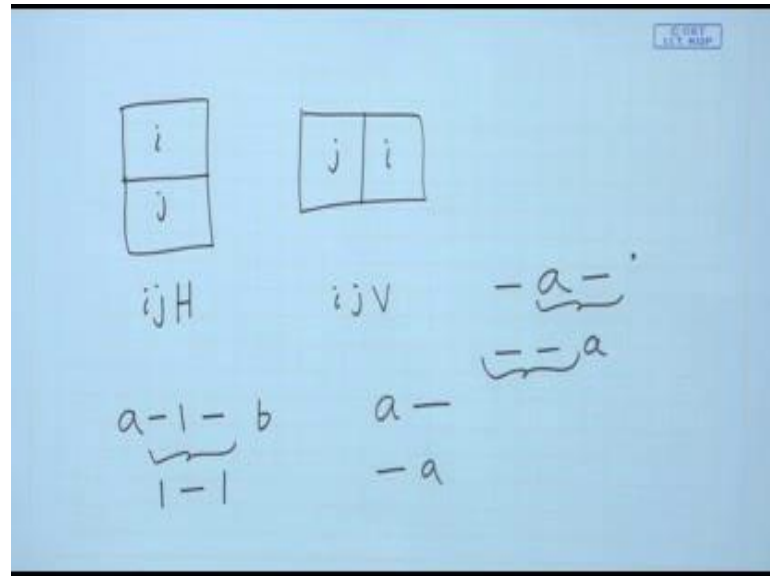
- Dissection operators defined:
  - ijH means rectangle j is on top of rectangle i.
  - iV means rectangle j is on left of rectangle i.
- Normalized Polish expression:
  - A Polish expression corresponding to a floorplan is called *normalized* if it has no consecutive H's or V's.
  - Used for the purpose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

BT KHARAGPUR | NPTEL ONLINE CERTIFICATION COURSES

So, this notation I am just repeating for your convenience, this i and j indicates two blocks, ijH means something like this.



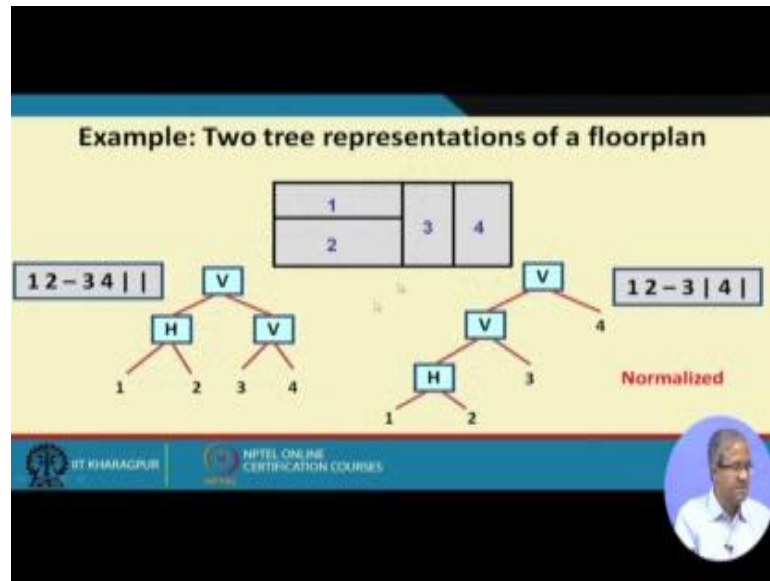
(Refer Slide Time: 22:51)



This  $i$  and  $j$  are connected by a horizontal line  $i$  and  $j$ , this will denote by  $ijH$ . Similarly  $ijV$  this means rectangle  $j$  is on the left of rectangle  $i$ . If I write  $ijV$  this will correspond to  $j$  is on the left of  $i$  separated by vertical line. So, every such slice is representing the polish or the postfix notation like this.

Now, define a normalized polish expression, which says that a polish expression is called normalized if it does not contain consecutive  $H$  or  $V$  symbols; this is used just for reducing some amount or redundancy because as you have seen earlier the same floor plan can have multiple slicing trees, or they can have multiple polish notations. So, this is one way to normalize it fine.

(Refer Slide Time: 24:03)



Let us take an example this is a floor plan, there are two alternate slicing trees you can see, you can either merge 1 and 2 first, 3 and 4 first and then combine them or you can merge 1 and 2 first, you combine 3 with it than combine 4 with it. So, the corresponding polish notations will look like this 1 2 H; that means, dash 3 4 V; that means, bar this this bar this this bar, this and similar for this 1 2 dash, 3 bar 4 bar, you can see here there are two vertical notations consecutive, but here there is no such this one will be considered to be normalized.

(Refer Slide Time: 25:03)

**Parameters of the Algorithm**

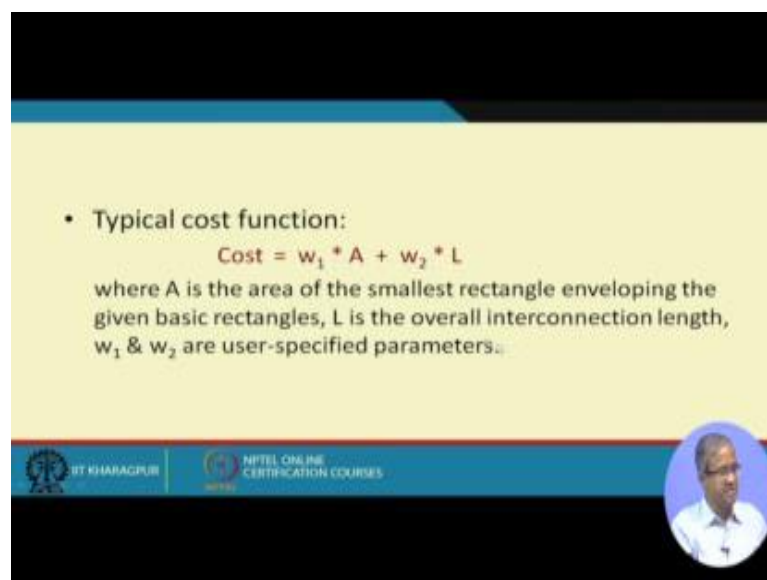
- Solution Perturbations (Move)
  - a) Swap two adjacent operands
  - b) Complement a series of operators between two operands (called a *chain*).  $\rightarrow V' = H$  and  $H' = V$
  - c) Swap two adjacent operand and operator.
- We accept a move only if it results in a normalized expression.
  - Only move "c" may result in a non-normalized solution.
  - We need to check only when move "c" is applied.

BIT KHARAGPUR | NPTEL ONLINE CERTIFICATION COURSES

So, for representation will be using only normalized expression; this is what we do then some of the parameter that the moves I talked about. So, 3 moves are defined here, the first move says you swap two adjacent operands, move two says you complement a series of operators between two operands this is called a chain. Complement means if it is a V you make it H, if it H you make it V. So, complement a series of operators between two operands means, suppose I have something like I have a block a, I have a block b between that say I have dash bar dash let say. So, what it says these you replace by bar dash bar complement each of them, this is called a chain; and the third move says swap two adjacent operators and operands.

So, if you have a dash you replace it by dash a something like this swap them; and the first one is of course, swap to adjacent operand; that means, if it is ab make it ba, and the point you notice that after making a move we will consider it only if it results in normalized expression, because you can check moves a and b will leave a normalized expression to a normalized one only. Only move c can result in a case where a normalized expression can become unnormalized like you might be having a situation it was a dash, let say dash. So, if you swap these two; it may become dash dash a. So, you may get two consecutive dashes it may become unnormalized fine.

(Refer Slide Time: 27:14)




• Typical cost function:

$$\text{Cost} = w_1 * A + w_2 * L$$

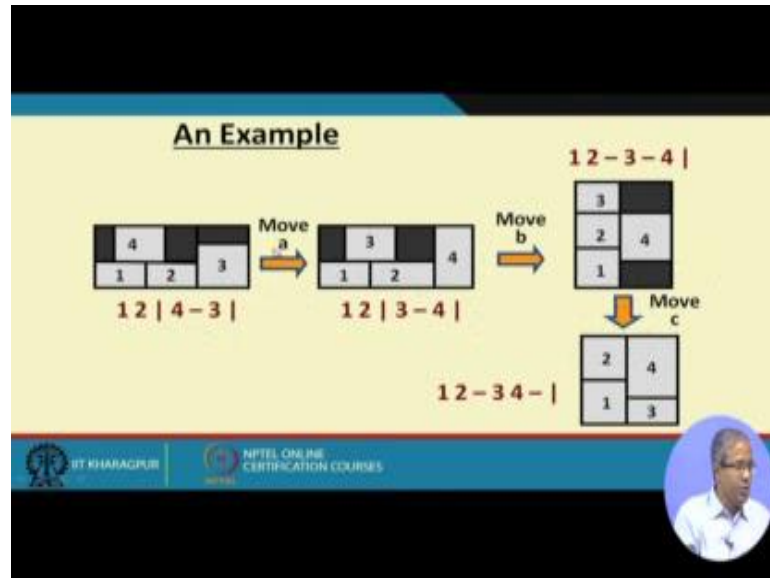
where A is the area of the smallest rectangle enveloping the given basic rectangles, L is the overall interconnection length,  $w_1$  &  $w_2$  are user-specified parameters.

BT KHARAGPUR | NPTEL ONLINE CERTIFICATION COURSES



So, the cost function again will be very similar, you have area you have the wire length then some weighted function this I have already discussed earlier, same kind of cost function you can use.

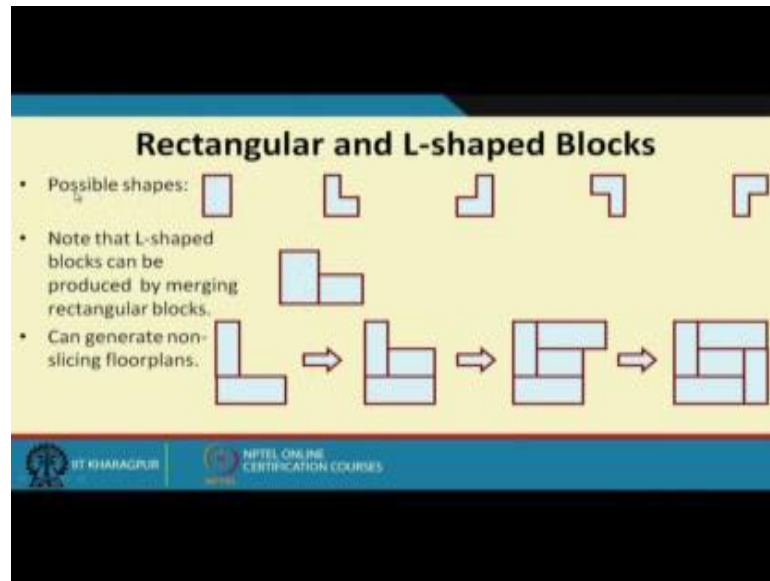
(Refer Slide Time: 27:31)



And this is just an example. Let say this was an initial solution from where you are starting; this is the corresponding polish notation. So, I am just illustrating one possible sequence of moves, you make moves a which means you are swapping two operands, this four and 3 you make it 3 and 4. So, what will happen? 3 will come here four will come there, so we get a solution like this. Move b; so between let say 2 and 3 you complement everything, there is only one you complement it to dash.

So, new layout looks like this, move c; move c what I do? This 4 and this dash you swap them this become this. See this example I have manufactured deliberately just to show you that some sequence of move can result in a very nice final solution, but in the practical (Refer Time: 28:39) case. You will have to do a lot of such iterations, lot of trails and errors and you will possibly arrive at a good solution at the end, but this example actually shows you that it is possible to find a sequence of moves through which from a bad solution you can go to a good solution.

(Refer Slide Time: 29:00)



And lastly let us just talk about rectangular and L-shaped blocks, you see the block shapes can be anything not only rectangle they can be L-shaped like this. But even if you restrict yourself only to rectangular blocks, whenever you put blocks side by side it can take the shape of L like this and also whenever you put the blocks in sequence one after the other, you can lead up to a situation as I it is shown here where you get a wheel kind of a structure non sliceable.

So, although each blocks were rectangle as we move one after the other you go on connecting, at the end you might get a wheel kind of a structure. So, these are some complexities you need to consider while generating constructive rectangular floor plans. So, I just tried to give an overview regarding the possible floor planning algorithms without trying to go into the very details of them. So, I think you will have a fair idea regarding the approaches that are followed.

Thank you.