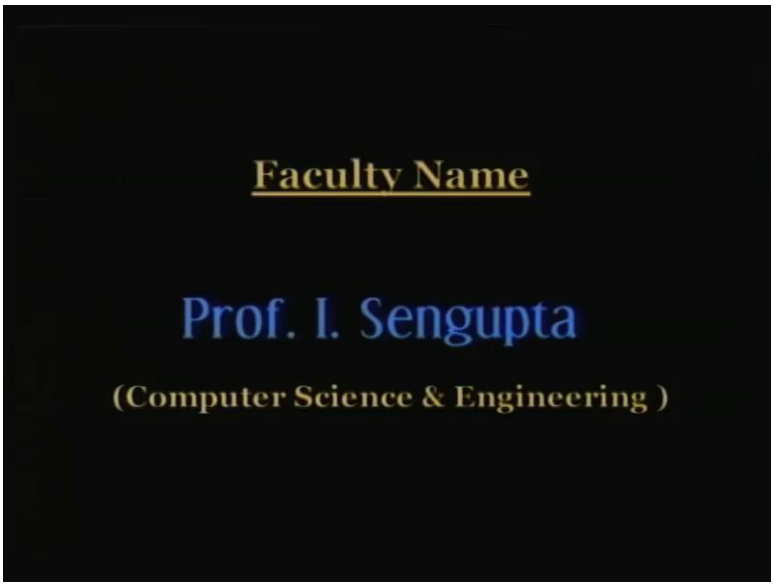
**Electronic Design Automation**
**Prof. Indranil Sengupta**
**Department of Computer Science and Engineering**
**Indian Institute of Technology, Kharagpur**

**Lecture No #17**
**Backend Design: Part III**

(Refer Slide Time: 0:40)

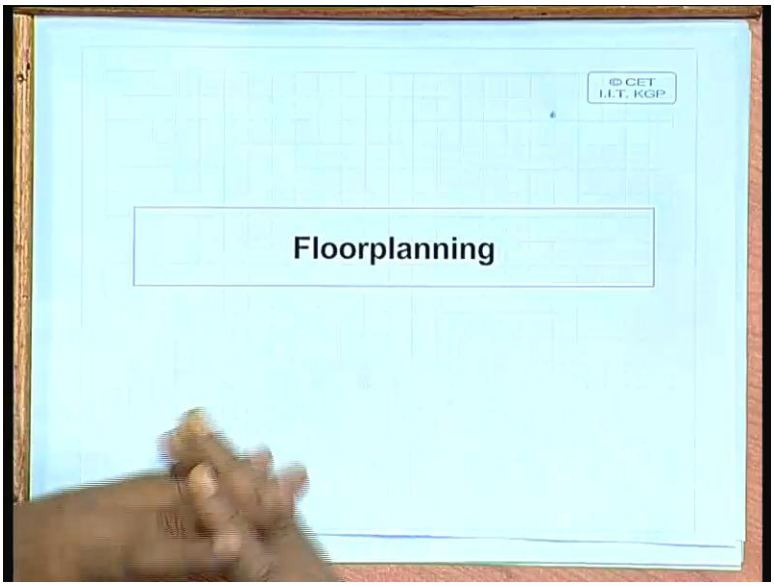
(Refer Slide Time: 0:51)




(Refer Slide Time: 0:59)

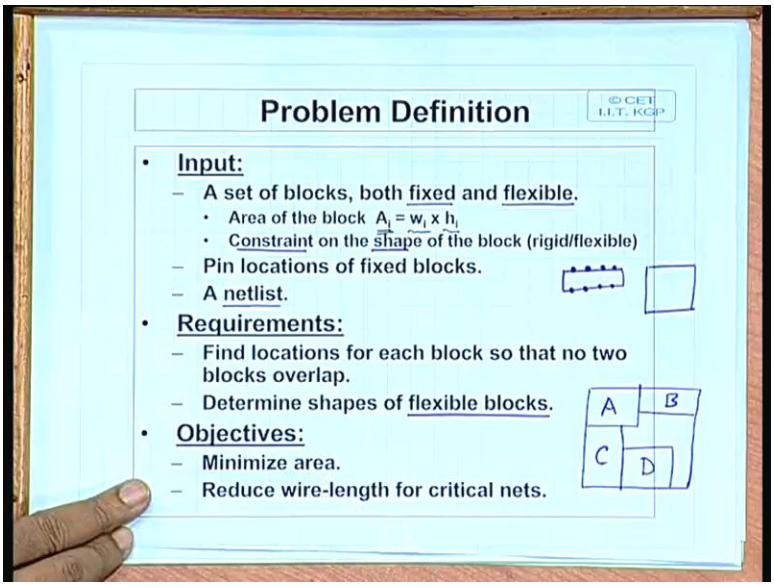

In your last lecture we were talking about partitioning.

(Refer Slide Time: 0:12)




Now as the next logical step in the backend design process, today we would be talking about a process called floor planning. So first let us try to understand what floor planning really means.

(Refer Slide Time: 01:28)



Talking about the problem definition, so what floor planning constitutes is that as an input it will take a set of blocks. Now the blocks may be fixed at this stage or it may also be flexible. So actually what do you mean by fixed and flexible are as follows. See some of the block may be pre designed. Means we exact we know the exact dimensions of the block in terms of the width and the height. But there may be some of the blocks we know how many components are inside that block. But we have not fixed up the shape of the block as yet. So such blocks we call as flexible blocks.

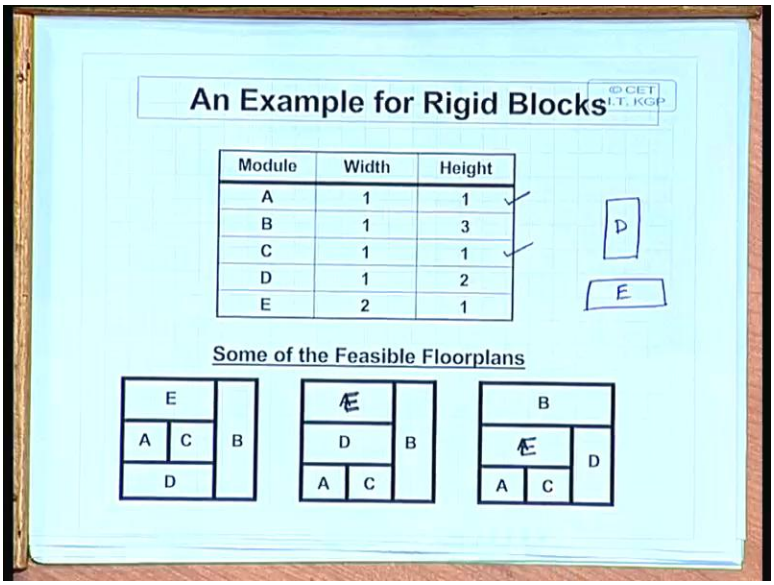
So as an input to the partitioning problem we have a set of blocks which can be both fixed and flexible. Now each block to start with we will be assigned a width and a height value which will give an area. Now in case of fixed blocks this $w_i$ and $h_i$ are statically fixed. But in case of flexible blocks we have to keep this area fixed you can adjust the values of $w_i$ and $h_i$ according to your requirement. So rigid and f0lexible whatever I am talking about these are the constraints on the shape of the block. So a block may be like this thin and long, it can be a square and so on.

Well we have specification regarding the blocks regarding the blocks. And for blocks which are of fixed shape, we also would be having some information about the exact location of the signal pins. But the blocks which are flexible means, the blocks have not yet been designed fully. So the positions of the pins are also flexible there. So if the blocks are fixed, we also have some pin locations. This may be required and of course a netlist which will indicate the interconnection among the blocks. So how they are interconnected? These are the inputs to the problem of floor planning.

So what we require here is that, we want to find a tentative location of each block on the floor plan; on the floor. Given a rectangle which indicates the area of the chip say on which we want to layout the blocks. So we want to find out the tentative location of the block. Suppose block a will go here, block b will go here, block c will go here and so on. These are the rough locations of the blocks. We are not showing any separation between the blocks through which the interconnections or interconnecting lines will go through. But we are roughly trying to get a relative ordering or the placement of the blocks.

And also as part of the flexible, as part of the floor planning if there are some flexible blocks, so at the end of the process we will also get the shapes of the flexible blocks. suppose d was flexible but when you place it finally we will say that well d will have to have a shape like this. And of course objective can be many. We will be yes look into objectives again. But typically the objectives are to minimize area; the total area of the floor plan. And of course to reduce the wire length for critical nets in order to minimize the delay. Okay. So let us take a simple example for fixed blocks and see that well means how a floor plan really looks like.

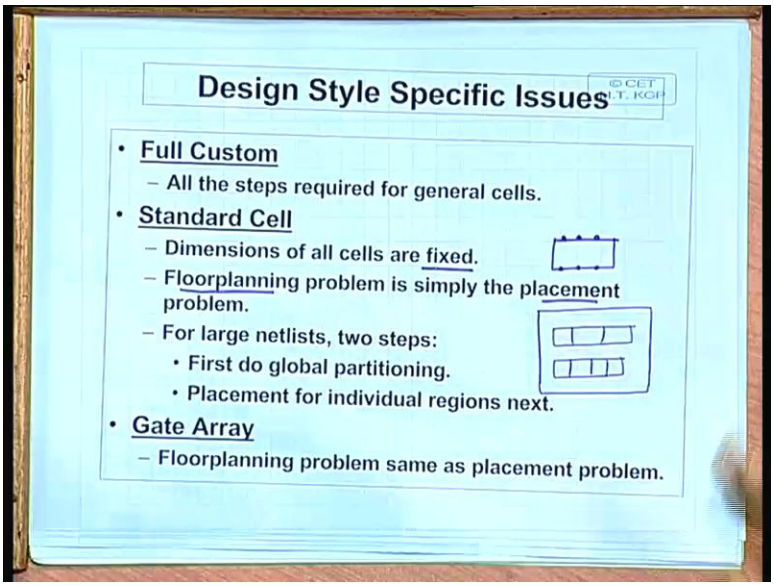
(Refer Slide Time: 05:25)




So here we have taken a very simple example with five modules a, b, c, d, e. And in terms of some unit the width and height of the blocks are also given. So as you can see that the blocks a and c are the smallest each of width 1 and height 1, d has width of 1 and height of 2, e has a width of 2 and height of 1. So d looks like this and e looks like this. This is your d, this is your e. But one thing you observe if you rotate the block d and e will look the same. So as part of floor planning sometimes we also have flexibility in orienting a blocks suitably by rotating it or even you can take mirror image of a block. Of course b is the biggest one; width 1 and height 3. So for this particular problem this diagram shows some of the feasible floor plan. There can be many others, but I am showing only 3.

So in the first one you see b has a width of 1, height of 3; b's position like that e has a width of 2 and height of 1, it has been positioned like that, d has been rotated width of 1 and height of 2; d has been rotated and a and c have been put side and side out here they are one by one. This is another alternative d has been put in the middle a, c. Here this is another alternative where b has been put in top in a rotated fashion, d comes here, e is rotated here and a and c out here. So as you can see for a given set of blocks there can be large number of such a you can say such feasible floor plans. So in order to find out which one is the best, we will have to evaluate some kind of a cost function on each of these and find out that which gives us the best solution. Okay. So here we would be looking at the way we can estimate the cost of a given floor plan.

(Refer Slide Time: 07:41)



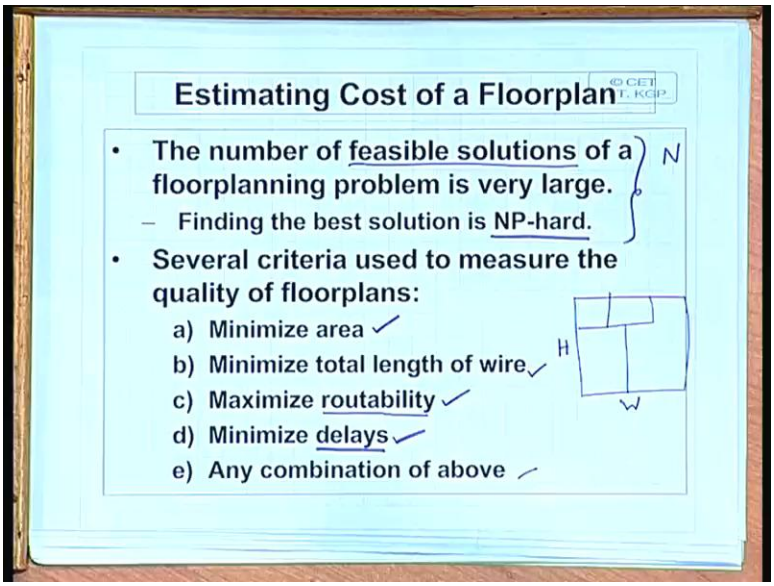

But before that let us look at some of the design style specific issues. Because this is important in understanding the impact of floor planning when we are trying to design something, it can be a full custom design, it can be a standard cell based or it can be gate array or FPGA based. Now the point to note is that for a full custom design, well we have complete flexibility in the placement and the ships of the block. So the floor planning problem as we have just stated, that applies completely to full custom designs. Blocks may be fixed blocks, may be flexible they may, be placed anywhere. So you have a general description of the blocks and we can place them

in any suitable way in order to optimize or minimize some cost function. But in case of standard cell you recall here cells are always picked up from a library. So it is not that we have flexible cells. So all cells are fixed or rigid which means the dimensions of the cells are fixed, so also are the locations of the pins.

So here we do not have any flexibility in that. And moreover if you recall the standard cell style of design, here these cells are placed along rows on the surface of the chip. So the floor planning problem and the placement problem becomes the same here. Because the placement problem means it is a refinement of floor planning to find out the exact location of the block on the chip along with its exact dimension. So actually in a placement you also keep some space for routing interconnection. So in a standard cell design floor planning and placement does not make any difference. Because when you say I am doing a floor planning, well these rows are already defined. You have to plan. That means where do you want to place a cell; means in which row in which position so that also gives the placement. Right? But if your netlist is large normally we follow two steps here.

First is that globally we do some partitioning so as to divide the big problem into several smaller sub problems and each of the sub problems we try to route or place in one or few neighboring rows. Okay. And the problem is quite similar for gate arrays or FPGA's. There also the locations of the devices are fixed on the chip. So basically we have to map some block of our design of our netlist into some particular cell of the array defined gate array. So there also floor planning and placement becomes the same. Right? So this is something you have to keep in mind, so that whatever algorithm we discuss here for floor planning, they primarily would apply to full custom designs. For standard cell and gate array we do not need the step of floor planning. Straight away we can go for the step of for the step of placement. Fine. Okay. Before talking about the ways floor plans are represented and the different algorithms for generating floor plan.

(Refer Slide Time: 11:18)



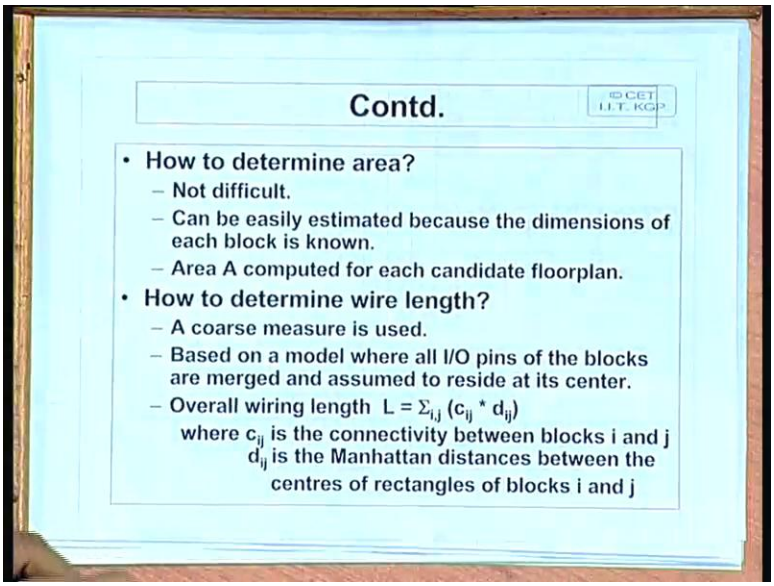

First let us look at again in slightly more detail how to estimate the cost. Now with regards to the cost, the first thing to notice that we had taken a very simple example as showed three possible solutions for it. But for a typical floor planning problem with typically large number of blocks, the number of feasible solutions can be very large. And in general finding the best solution out of that, you cannot have anything better than exhaustive search. We will have to evaluate all possible and find out. And means you will see later I will just show a few small representative examples that as the number of blocks n increases the number of feasible solution increases exponentially with that. So as a result the problem becomes an NP hard problem.

So we typically look at some kind of a heuristic in order to get a good solution instead of exploring all possibilities and trying to find out the best. Okay. And in fact we talked about area and interconnection. But there are several criteria which are used to evaluate the quality of a floor plan. Of course the most important are to minimize the total area. Say I have a rectangular region; I will be creating a floor plan. So I will have to find out what is the total width and what is the total height of this rectangular bounding box I require that is the estimate of the total area. Similarly I will have to estimate the total length of the wire and trying to minimize that. Because, yes. [Students noise not audible (Time: 13:09)]

See at in this step of floor planning the chip is little far of we are not thinking of a chip we are thinking of a rectangular area. So we are trying to think. That means what is the smallest such rectangular area in which you can map our entire design into. Okay. Because the exact chip layout will come little later during placement and minimize total length of wire is obvious because it is natural to put those two blocks closer together which are strongly interconnected. Then of course the critical nets should also be taken care of maximize routability is important for certain design styles. See for FPGA or gate array may be you will you will be able to find out a placement. Because there are available cells on your chip you can map one sub circuit into one cell like that.

But when it comes to routing, say for example in a FPGA, the total amount of routing resource is limited the number of routing rows and columns in the chip they are fixed. So the every possible placement may not be routable. You will find that there is a conflict here you are not able to finish the interconnection. So in that case you may have to may have to iterate up, you may have to change the placement of floor plan and again you have to try out the routing. So there one of the objective is to also look at routability. Of course this we will discuss when we discuss the problem of routing later. And talking about the critical paths because the critical paths will determine the delays. So if we know the critical paths, we may also try to minimize them. And of course we can have any combination of above area delay or area and length of wire taken together. Fine. Now this area and the total length of wire these are the two measures which are most popularly used.

(Refer Slide Time: 15:18)

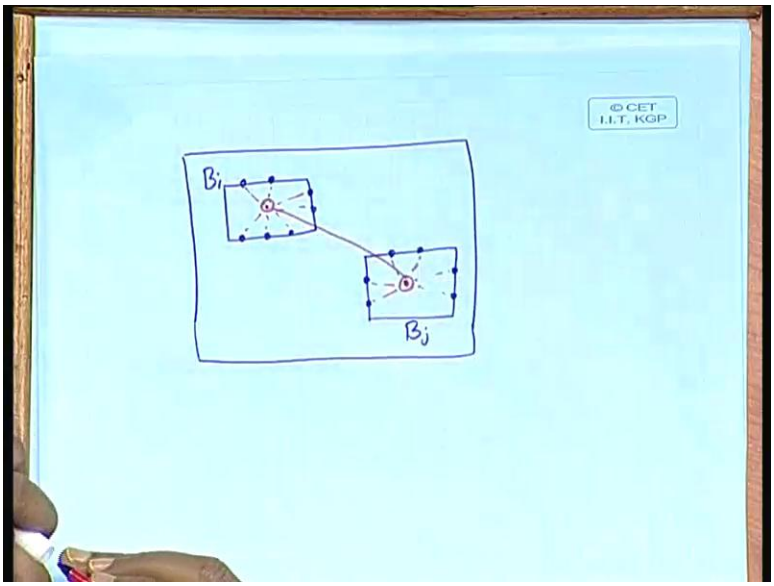



So the first question arises that how can we estimate the area? See area is not difficult to estimate because on the on the floor plan the size of each of the blocks we are placing are known in terms of its width and height or its total area say ai; ai is the area of this block. So since you have the area of each block known in terms of width and height and you are placing them in certain way. Suppose you placing them in this way ai, Bi and ci. So you know the total width of the floor plan. Similarly you can find out the total height of the floor plan. So multiplying them you can estimate the total area. So once we have a floor plan estimating area is easy. Because the width and height of each of the blocks as you place them on the floor they are fixed. Fine. Now the second question is that how to determine the wire length because here well wires, yes.

[Students noise not audible (Time: 16:17)] Well here when we estimate the area, we are well just like when we are building a house we just say that these are the four rooms and the adjacent say. You do not keep any space for corridors at this level. They will get refined in the later phase. So now we just say that this is the roughly the area of a block. This is roughly the area of another block. Since they are strongly connected, we place them side by side. But we do not explicitly keep space for routing here that will get refined in the next step in the placement phase. Yes,

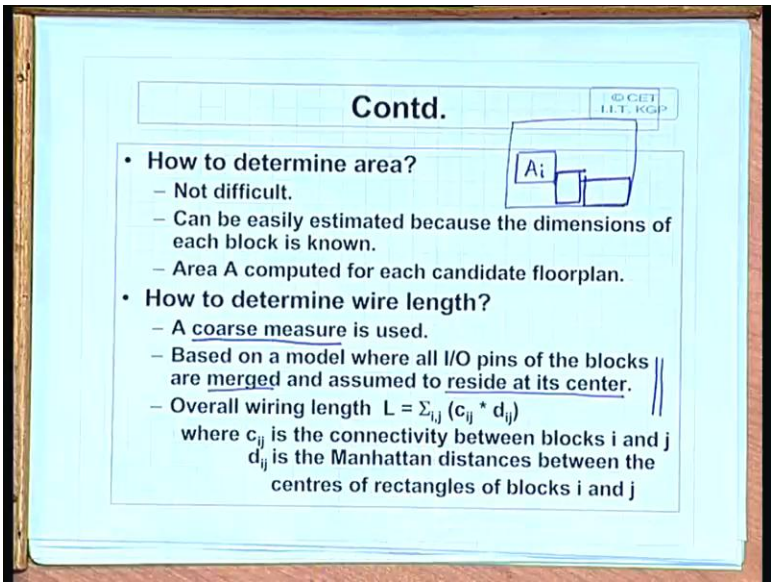
fine. And at this step, since we have a very approximate layout of the blocks. So while determining the wire length, we adapt a very coarse measure. Say the coarse measure is like this.

(Refer Slide Time: 17:18)



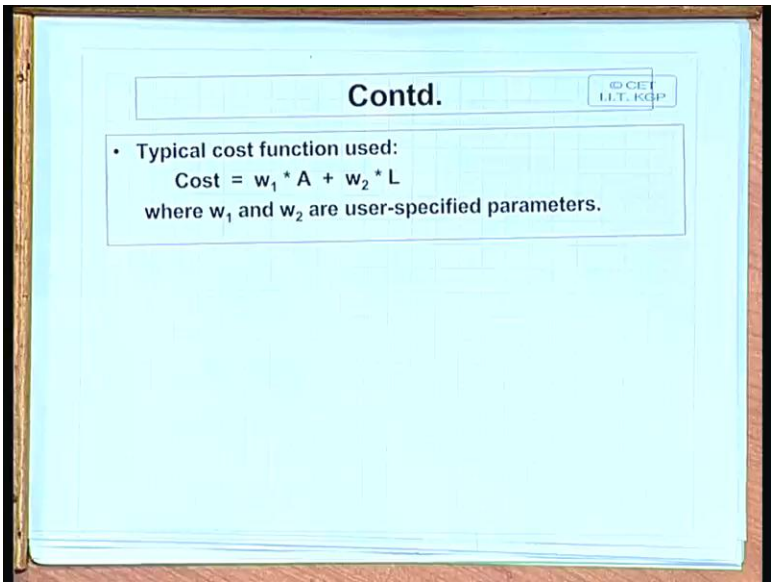

Suppose on the floor there are two blocks. Say this is block $B_i$ and this is block $B_j$. Now each of these blocks can be having several interconnecting pins or signal points. Okay. This may be also having several. Now the way distance is estimated is like this. That here for the sake of estimating the distance of interconnection with respect to floor planning, we assume that each rectangle well you can define the center of the rectangle. So we assume that all this pins are located in the center. Similarly here we define the center of the rectangle, so all this pins are assumed to be residing in the center of the respective block. And you compute the Manhattan distance between the two blocks, two centers of the blocks. Suppose a wire, a pin here was required to be connected to a pin here. But at this level the estimate would be just the distance between the centers. This is the estimate of the length of the wire. Because, right now we do not know anything about the routing regions or the channels. So we cannot exactly estimate the wire length. So we have a very coarse estimation. We just take the Manhattan distance and we estimate it.

(Refer Slide Time: 19:02)



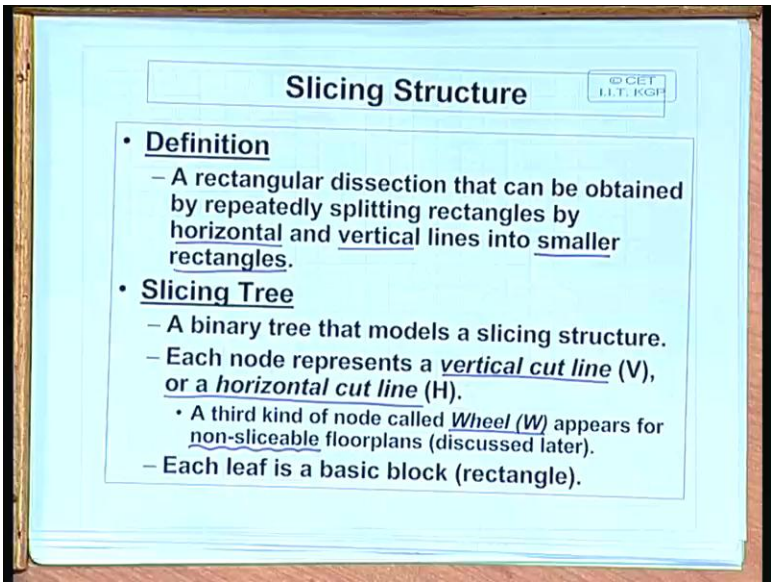

So as it is mentioned here, so all I/O pins are merged and assumed to reside at the center of the blocks. And the overall wiring length is the sum of well this factor for each net. $C_{ij}$ is the connectivity between the blocks i and j. That means number of blocks that connect i with j and the $d_{ij}$ is the Manhattan distance between block i and block j. So you find out the sum for all possible pairs of blocks i and j. And you get an estimate of the total wire length. So an estimate like this is normally used to find out the total wire length estimate, yes. [Students noise not audible (Time: 19:50)] Yeah. For all pairs of blocks we do this. Because i and j refer to two different blocks $B_i$ and $B_j$. So there if there are n number of blocks, there will be n into n minus one such pair n into n minus one by two in fact. So here we estimate this for every pair of pair of blocks and then you find the sum total. Okay. As I told you this area and the wire length are the most important measures to find out the area.

(Refer Slide Time: 19:02)




So to find out the cost, so typically we estimate the cost as some weighted average of this area and this length w1 and w2 can be user specified parameters means on to which you are getting more emphasis. So you can estimate it here. Of course in addition you can keep some other criteria like critical net. Because in critical nets and delays they are typically handled on a case by case basis. There is no general algorithm we can also take care of that and it can be means optimized them in the same framework. So you find a floor plan and then you evaluate the cost of the critical paths. How are they crossing? How far they are? And then you try to make some perturbation so that critical paths get improved. Okay. Fine. So before going into the algorithms we talk about some of the data structures that we typically used to represent a floor plan.

(Refer Slide Time: 21:27)

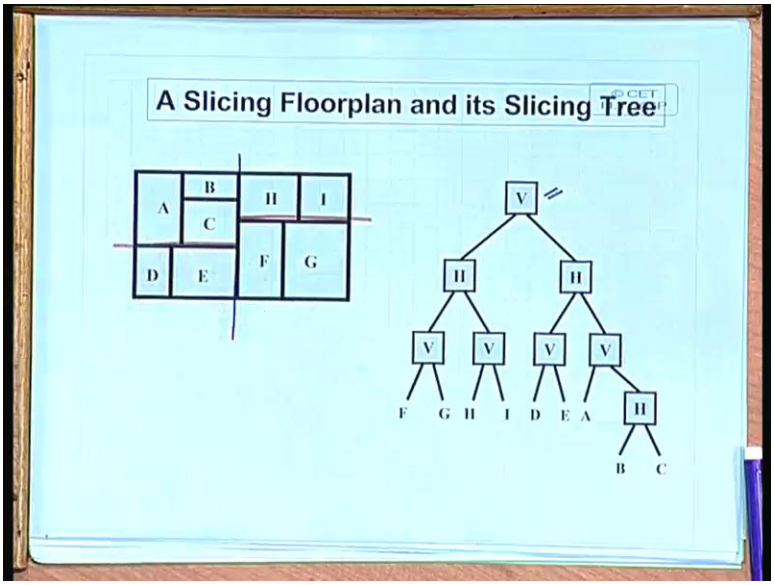



Now one such data structure is called a slicing tree which is based on the so called slicing structure. See slice as the name implies, well I have something I just cut it into two pieces, that is a slice and a slice I can make either horizontally or I can make vertically. So a slice divides an existing rectangle into two different rectangle. So a given big floor plan I can apply a number of such vertical and horizontal slices as I need and I can get several smaller rectangles. So the slicing structure can be defined like a rectangular dissection that can be obtained by repeatedly splitting rectangles using horizontal. And vertical cut liens using horizontal and vertical cut liens. And as a result we get smaller rectangles. Now the order in which we are applying the cuts that is captured or represented in a so called slicing tree. Now a slicing tree is a binary tree, this models the slicing structure means the order in which you are carrying out this slices.

So you start with the root of the tree, as you go down towards the leaves you get the details of the slices. Now in this tree other than the leaf node, the leaf nodes indicate blocks each leaf is a block. Block means the smaller rectangles which you get finally. But the each of the other nodes they represent a cut it is either a vertical cut line represent by v or a horizontal cut lien represented by h. So if it is a purely sliceable structure where you can get the each of the basic blocks at the low level by horizontal and vertical slices, then you can have a tree with v and h

nodes only. But we will see subsequently that some of the floor plans may not be sliceable using horizontal and vertical cuts alone. These are called non sliceable floor plans. And in those kind of floor plans, we may introduce another third kind of a node called a wheel. Now this wheel we would be discussing a little later. But for the time being we look at only v and h nodes, using vertical and horizontal cuts.

(Refer Slide Time: 24:09)

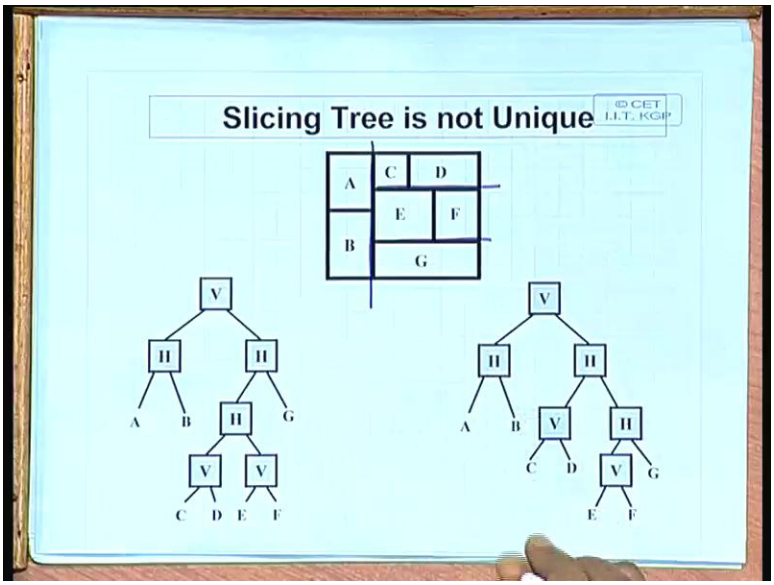



So we show here an example of a slicing floor plan and the corresponding tree; the slicing tree. Just have a look at this floor plan. And just have a look at this tree. Starting from the root, this v, this indicates a slice corresponding to this vertical cut. Because you see this v, partitions the tree into two parts one comprises of, f, g, h and i and other comprises of a, b, c, d, e, this part. Now you have divided it up into two parts. Now in each of the two parts you just make a horizontal slice each. So in this half this horizontal slice will divide it up into f, g and h, i. So it corresponds to this. Similarly in the other one it divides into d, e and the rest. So in the other part it corresponds to this. So this process will continue.

In the next level, there is another vertical cut f, g; this here h, I; this and you get finally the blocks f, g, h and i. And on the other side you have another vertical cut d, e; this you get d and e.

And you have another vertical cut here where on one side you get the block a and the other side you get another composite block where you need another horizontal cut like this, to get b and c. So here once you have this tree, just if you traverse the tree from the root towards the leaf or from the leaf towards the root. Towards the root means whichever way you traverse. You will get an idea regarding the order in which you are you are basically partitioning the total floor area into the floor plan into the placements of the blocks. Fine. Now one thing to note is that given a floor plan, the order you can apply this slices to arrive at this. This may not be unique.

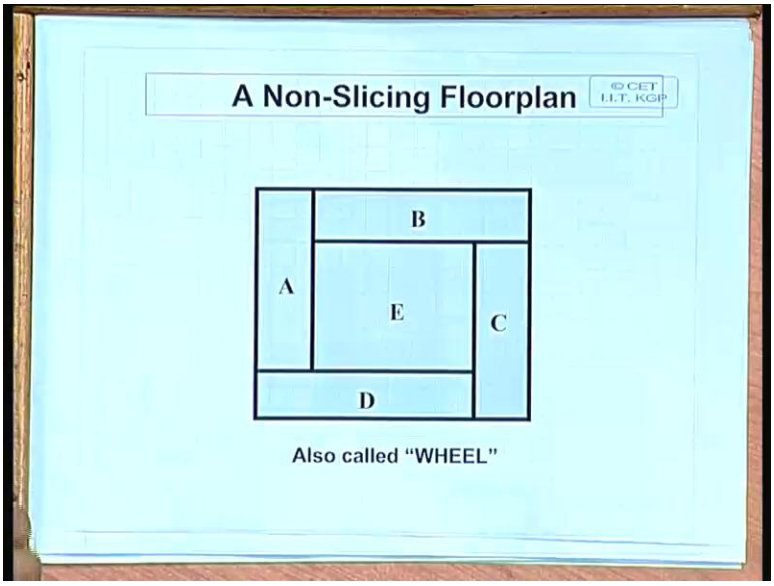
(Refer Slide Time: 26:23)




So I am showing here an example to illustrate the non uniqueness of the slicing tree. Take this example. So here so here one possible slicing tree is this. So the first vertical cut we apply like this in both these examples the first vertical cut is this. But here you start at the next level for this portion with a cut this. It divides it up into g and the rest here. But in this case you start with this cut first; c, d and the rest. So means in any sub block, if there are several horizontal or several horizontal vertical parts you can you can apply them in any order. So as you can see that there are two alternatives, one is that you first cut g, then you cut this in the middle then vertical cut c, d and e, f. Similarly here you first cut this then c, d you can do using a vertical cut, then we apply a horizontal cut here, then a vertical cut here in the middle. So for the same floor plan you can

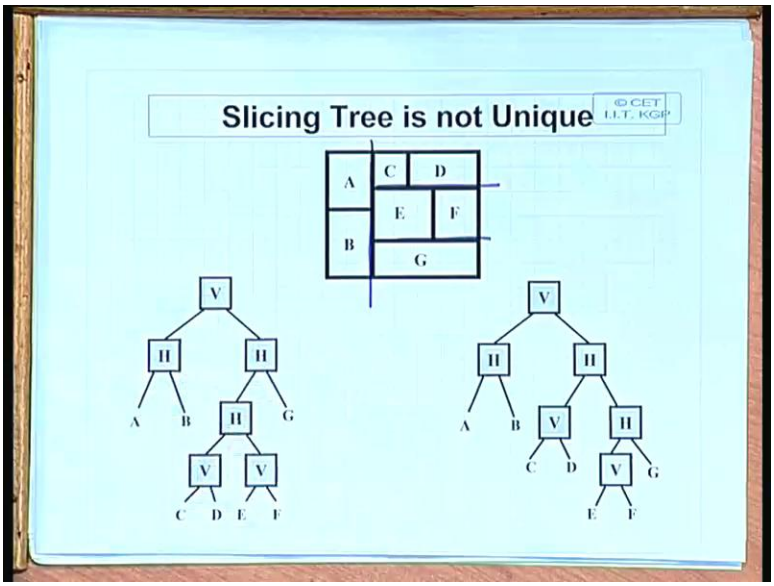
have more than one slice in trees. Okay. But we have said that all floor plans we cannot represent by simple vertical and horizontal cuts. Okay. There is only one primitive such floor plan layout which cannot be represented by horizontal and vertical cuts.

(Refer Slide Time: 28:05)



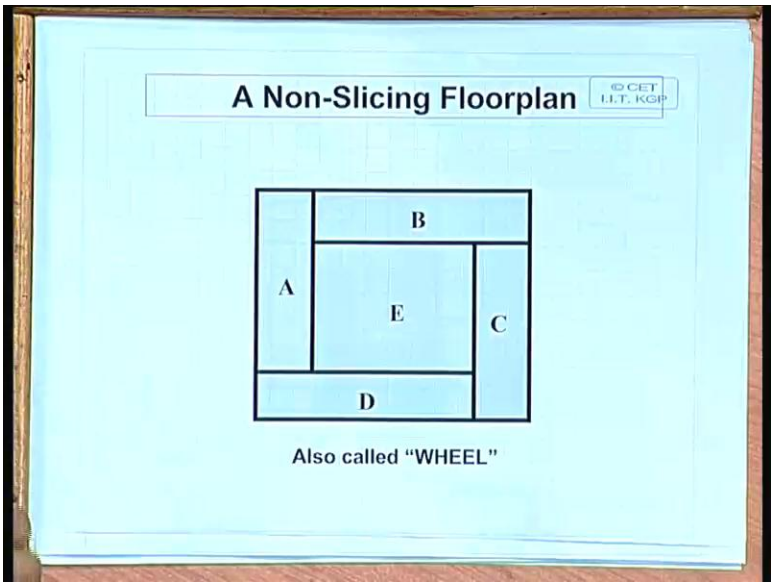

And that is called a wheel it looks like this. This is the only kind of a primitive floor plan structure which cannot be represented by simple slices. See if you have this floor plan you cannot apply simple horizontal and vertical slices cut them and get because of you want to cut this your block c also gets cut in the middle that we do not want. So this kind of a thing we called a wheel. And in the order we need we just name the blocks is that a, b, c, d and the middle one e. Now it is possible that you have a composite high higher level floor plan in which there may be portions which may look like a wheel. And inside a wheel this a, b, c, d, may be composite blocks themselves. So in that sense you can have some kind of hierarchical floor plan. Well whenever you define the slicing tree kind of a thing. For example if we again look back at the diagrams before.

(Refer Slide Time: 29:08)



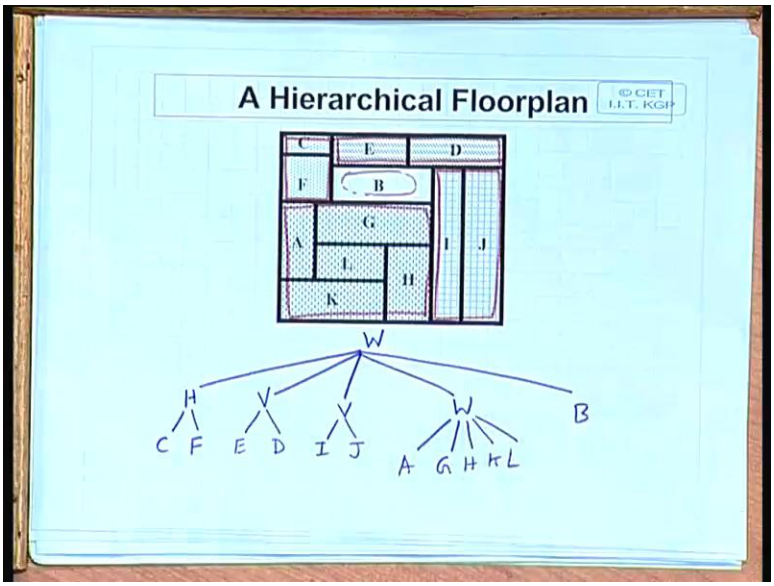

So here when you whenever you have a hierarchical structure like this or hierarchical set of slices you apply. Well you represent them as a tree. Now this tree in some sense represents a hierarchy like as if c, d and e, f can be combined together and whatever you get, you can combine there by horizontal line you can put them together. So again putting this and g horizontally you can make a bigger block putting this and this vertically you can put a bigger block. So this is some kind of a hierarchy you are specifying.

(Refer Slide Time: 29:43)




So now let us look at an example which also involves this wheel structure. Right.

(Refer Slide Time: 29:49)

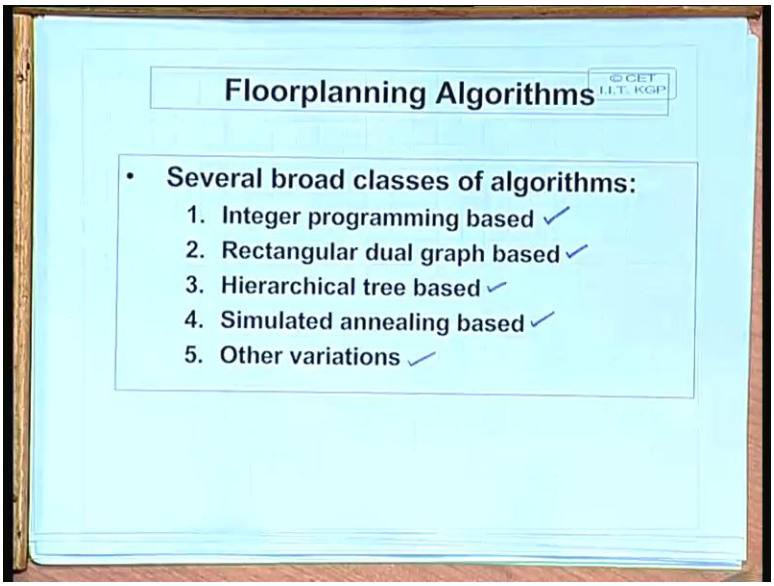



So let us take a slightly complex example like this. So this you can see that this block apparently looks quite complex, there are so many blocks. But if you look into it slightly in detail, you will

find that there are several regions in this and I am trying to highlight this region. This is one region, this is another region, this is another region and this is another region. Of course whatever remains in the middle, this is another region. Now just looking at the regions you can observe that this is nothing but a wheel. If you call this regions as a, b, c, d and e, this is nothing but a wheel. So at the top level this has a wheel structure. So now if we want to construct the so called tree that well you can say analog of that slicing tree. Because this is not sliceable, there will be some wheel nodes as well. So here the tree will look like this at the top level, you have a wheel we represent it by a node w.

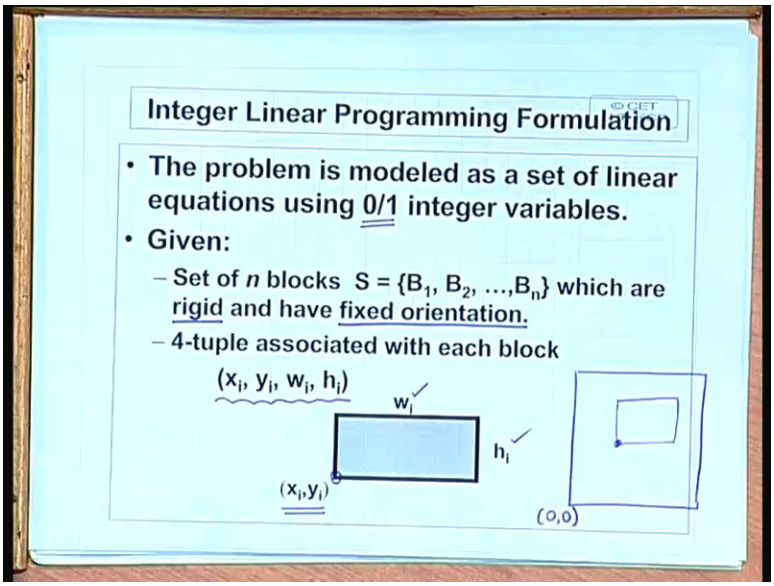
There will be five successors. The first successor this one we are calling a. Just if you recall the previous diagram, this we are calling a. So let us start with this c and f. So here they are divide by horizontal cut. So there will be a horizontal h node with c and f. Similarly next one this is the next one coming e and d with a vertical cut. So this a v node e and d the third one is this i and j with a vertical cut. This one is again a wheel. So here you will get another w node. So there will be five successors a, g, h, k, l and finally the middle one is b. So for this floor plan you can construct a tree like this where you can see instead of v and h, you have a third kind of a node w. So in general your floor plan can also contain wheel nodes. Right. Fine. So these are the ways you represent a floor plan. So now we look at some of the algorithms that people actually have talked about have used to generate a floor plan from a given problem description of the blocks.

(Refer Slide Time: 33:00)




So in fact there are a number of you can say classes of algorithms which have been reported. So some of them I am showing here. Well integer programming based approach is an interesting approach well which ofcourse is based on sound mathematical basis. Rectangular dual graph approach is another interesting approach which is based on some graph theoretic concepts. And hierarchical tree based and simulated annealing. These are practical methods which are found to work better. And you can have other variations like some combination of these or you can have genetic algorithm etcetera. So we will not be talking about other algorithms. So we would be just looking at the first four. So we start with integer programming based approach. Well here what is done is that the floor planning problem is framed as an integer linear programming problem. So just we just show you with a specific problem that how this problem formulation can be carried out. So let us ((Audio not clear Time: 34:12)).

(Refer Slide Time: 34:17)

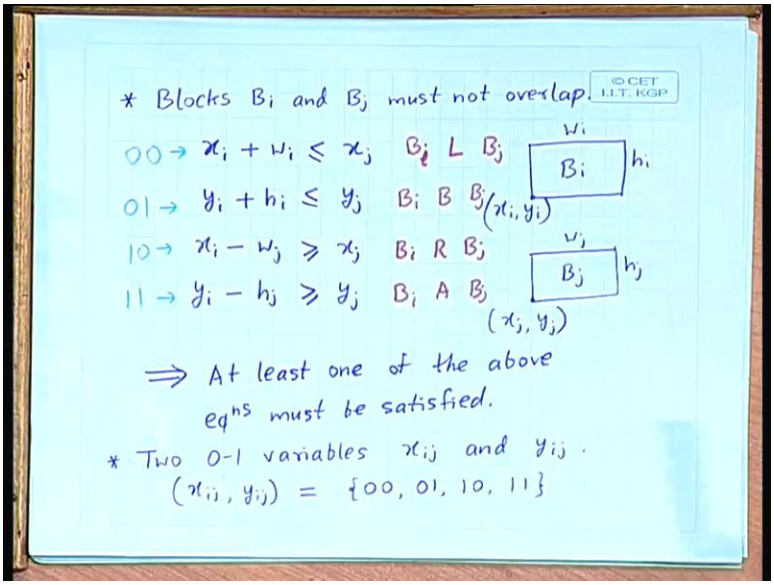



So the integer linear programming formulation here so essentially what we do given the details of the blocks we try to model the problem as a set of linear equations which uses variables which can take on the values of 0 and 1. So this is a 0, 1 integer linear programming problem. Of course there are some variables which are not 0, 1 which can be which can be arbitrary integers we will see. But there are some variables which can take on only 0, 1 binary variables. So as the input we have description of a set of n blocks which we need to place on the floor plan on the floor. This blocks we call $B_1$, $B_2$ to $B_n$.

And in order to illustrate the formulation would be showing we are assuming that these blocks are rigid. Of course there exists formulation where the blocks can also be flexible. But here we are assuming the blocks are rigid and also have fixed orientation. That means we cannot rotate the block. Well again in some other formulation the rotation is also allowed. But here we assume that the blocks are rigid and you cannot rotate the blocks and with each block we associate some information. Well suppose this is the floor the overall rectangle. So the lower left corner we assume is having a coordinate (0, 0).

This is the origin, so with respect to the origin whenever you place a block somewhere, this lower left coordinate is assume to have coordinate of $x_i$ and $y_i$. So $x_i$, $y_i$ is the coordinate of the lower left coordinate, $w_i$ is the width and $h_i$ is the height. Now since the blocks are flexible what we assume is that $w_i$ and $h_i$ are constants $x_i$ and $y_i$ are the values we have to find out. Right. Okay. So now let us see that how we can formulate this problem as a set of linear equations. So just remember for each block we have $x_i$, $y_i$ the lower left coordinate width and the height. Fine.

(Refer Slide Time: 36:44)




So what we require is that, the main criteria is that, given two blocks say $B_i$ and $B_j$, blocks $B_i$ and $B_j$ must not overlap. This is the main condition to be satisfied whatever floor plan we get, if we can ensure that no two block overlap, then we have feasible floor plan. Okay. So in order to have this, let us see that what are the inequalities that need to be satisfied. Say in terms of the sizes of the blocks, we have the sizes i and j and also they coordinates $x_i$, $y_i$ and $x_j$, $y_j$. So the equations will be like this, $x_i$ plus $w_i$ less than equal to $x_j$. I am writing four equation, then I am explaining which represents what; $y_i$ plus $h_i$ less than equal to $y_j$; $x_i$ minus $w_j$ greater than equal to $x_j$; $y_i$ minus $h_j$ greater than equal to $y_j$. Okay. So I have two blocks $B_i$, $B_j$ but the lower left coordinates are $x_i$ $y_i$ and $x_j$ $y_j$; $w_i$ $h_i$ are the widths and heights of course. This is $w_i$, this is $h_i$, this is $w_j$ and this is $h_j$. Okay. Now let us try to interpret these four equations.

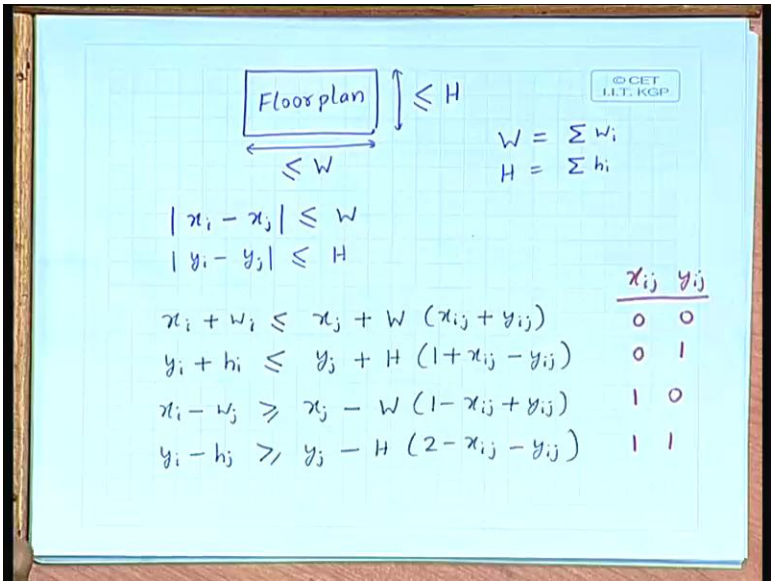
The first one says $x_i$ plus $w_i$ is less than equal to $x_j$ which means block j is starting somewhere to the right of i. So this essentially means that $B_j$ is on or you can say $B_i$ is on the left of $B_j$. because this $x_i$ coordinate plus its width this is less than equal to the starting point of $B_j$. So $B_j$ is somewhere on the right. So $B_i$ is lying to the left of $B_j$. Similarly in terms of y, if $y_i$ plus $h_i$ less than equal to $y_j$ that means $B_j$ is on above $B_i$. So $B_i$ is you can say below $B_j$. Similarly this means if you take $w_j$ on the right hand side it means $x_j$ plus $w_j$ is less than equal to $x_i$. That means you interchange i and j it happens. So this means i is on the right of j. So $B_i$ is on the right of $B_j$. Similarly this i is below j or sorry $B_i$ is above $B_j$. $B_i$ is above $B_j$ because $h_j$ plus $y_j$ is less than equal to $y_i$. So you see just in terms of the relative or positioning of these four blocks, in order to avoid them from overlapping, we have written four equations.

And in order to ensure that the blocks do not overlap, we need not satisfy all these equations. If we look at them a little carefully, it is sufficient if atleast one of this equations are satisfied. Okay. So what you want is that atleast one of the above equations must be satisfied. So the way we express or we write down the equations should also capture this that atleast one of this must be done, must be satisfied. So in order to do this, what we do is that, we introduce for block pairs i and j. We introduce two 0, 1 integer variables. We call them $x_{ij}$ and $y_{ij}$. Now the interpretations of the values of $x_{ij}$ and $y_{ij}$ are very simple. See $x_{ij}$ and $y_{ij}$ since they are 0, 1 variables, they can be either 0, 0 or 0, 1 or 1, 0 or 1, 1. Now these are the four equations which are given. Okay. The first equation correspond to this 0, 0 combination. Second equation corresponds to 0, 1. This is 1, 0, this is 1, 1. So the combinations of $x_i$ $y_j$ values indicate that which of the four equations we are referring to.

And depending upon on which equation we are trying to satisfy, the value of $x_{ij}$ and $y_{ij}$ will be that. The idea is that some how we will show how we would be writing down some equations which will be combining $x_{ij}$ $y_{ij}$ with these. They will be composite equations and we will also try to find out some value of $x_{ij}$ $y_{ij}$ which can be one of these four such that this equation is satisfied. We will see suppose that for certain particular value of $x_{ij}$ $y_{ij}$, suppose it is 0, 1. 0, 1 means we are just referring to the second equation. So the way we would be modifying the equation is that we would see that the other three equations will be trivially satisfied. Only this equation will act as the constraint which needs to be satisfied. So we will have to be suitably chose the value of $y_i$

and $y_j$ for example. But others are trivially satisfied so there is no question of determining values of $x_{ij}$ or $y_{ij}$, $x_i$ or $y_i$. So we will see this. So we combine this with these two new variables to get some set of new equations.

(Refer Slide Time: 44:09)




But before that we also assume one thing. Suppose this is your floor plan you want to get and there are some restrictions you are also imposing what kind of restrictions you are saying that the total width of the floor plan must be less than or equal to capital W and total height must be less than equal to capital H. Now W and H may be specified by the user or you can have a rough estimate as follows. W will be the sum of all $w_i$'s and H will be the sum of all $h_i$'s. This will give you the trivial maximum values of h and y all blocks placed one after the other. Okay. So there are some other obvious constraints which come up these are for any two blocks $x_i$ and $x_j$, so the difference between their x coordinates will never exceed w because this is the maximum. Similarly the difference between their y coordinates will never exceed H. Okay, fine.

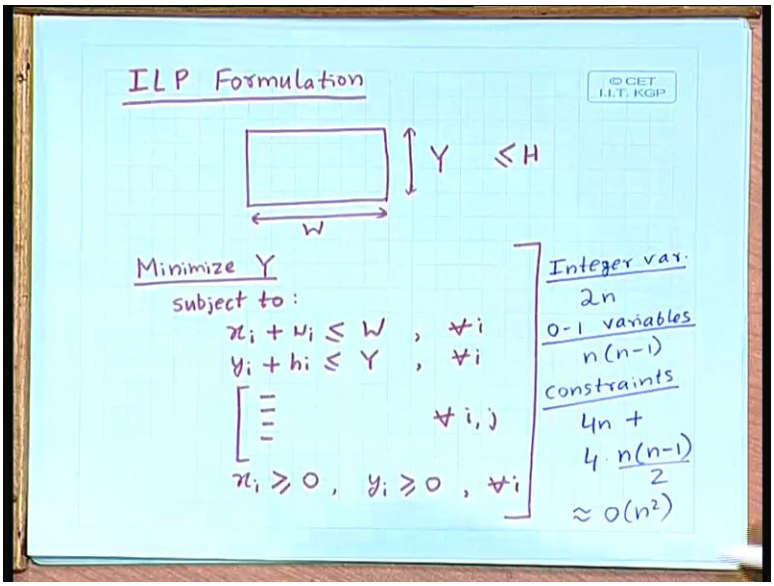
So now let us try to modify the earlier four equations, these four, taking into account $x_{ij}$ and $y_{ij}$. So I am showing you the modified equations. $x_i$ the first equation becomes so earlier. It was only $x_j$. Right. But now we are adding something to it plus capital w, x, $y_{ij}$. So I am explaining what

this means. Second equation was $y_i$ plus $h_i$ less than equal to $y_j$ only. But now we are adding something more H into 1 plus $x_{ij}$ minus $y_{ij}$. The third one $x_i$ minus $w_j$ greater than equal to $x_j$ minus w, 1 minus $x_{ij}$ plus $y_{ij}$. Similarly last one $y_i$ minus $h_j$ greater than equal to $y_j$ minus h 2 minus $x_{ij}$ minus $y_{ij}$. Well now let us look at this equation a little more carefully and find out what this means. So I just told you that the $x_{ij}$ $y_{ij}$ combination actually tells us that which of these four constraints are in force. You can say which is the one you checking. And the combinations were (0, 0), (0, 1), (1, 0), (1, 1).

Suppose the values are (0, 0), then you see the first equation is same as the original unconstrained equation $x_i$ plus $w_i$ less than equal to $x_j$ this part is 0. But the other equations you note for the second one $y_i$ plus $h_i$ less than equal to $y_j$ plus H. This will always be true because H is the maximum height. Okay. This this will be trivially true similarly $x_i$ minus $w_j$ greater than equal to $x_j$ minus w, total width. Or put it other way round $x_j$ plus $y_j$ plus $w_j$ less than equal to $x_i$ minus w, $x_i$ plus w, this will also be trivially true. Similarly this one $y_i$ minus $h_j$ greater than equal to $y_j$ minus 2 h. So other three equations are getting trivially satisfied. So only one of these equations is having the hard constraint that has to be satisfied by suitably selecting the values of $x_i$ and $y_i$, $x_j$ and $x_j$. Similarly let us see another combination. Suppose the values are 1 and 0. So for 1 and 0 only, for the third equation this portion is becoming 0, this part is disappearing.

But for the other equations you see they are getting trivially satisfied. The first equation is becoming $x_j$ plus w. This is becoming $y_j$ plus 2H this is becoming $y_j$ minus H they are getting trivially satisfied. So only one of this equation is having this new term as 0 and the other terms as either w or 2H or H or 2H. So there will be trivially satisfied then. So this is the way in which the constraints are modified in order to take care of well the fact that one of them need to be satisfied. So now if you pose it as a problem with so many variables, so now you are required to find out $x_i$, $x_j$, $y_i$, $y_j$, $x_{ij}$ and $y_{ij}$. So you will come up with a combination which will also give you some values of $x_{ij}$ and $y_{ij}$. So actually which will tell you some orientation $x_i$ $B_i$ is on the right of $B_j$ or $B_i$ is above $B_j$. So for that what are the values of the other things so that all of them get satisfied? So now with this we can formulate the linear programming formulation now.

(Refer Slide Time: 50:10)




So it is an integer linear program ILP since all the variables have integer values. See in the ILP formulation the way we are presenting here is that well instead of assuming variability in both direction we are assuming that width is w. But height we are trying to optimize minimize. So let us call it y. Well of course height is less than equal to h. This h remains but the actual value of y or of the height is y. So now our formulation becomes like this. Minimize y, this is going to do. Subject to there are several constraints. The first ones are the obvious constraints $x_i$ plus $w_i$ less than equal to capital W. $y_i$ plus $h_i$ less than equal to capital y, then there are the four constraints. We had just shown these four equations. These four equations. So these four equations will come in here, then some trivial constraint that $x_i$ and $y_i$ must not be negative $x_i$ greater than equal to 0, $y_i$ greater than equal to 0. So this is the integer linear programming formulation.

So of course the first equation is for all i, second is also for all i, the last ones are also for all i. But the middle ones are for all pairs of values of i and j. So if you roughly estimate, that means how many equations and variables you are trying to solve, well in terms of well the general integer variables they are $x_i$'s and $y_i$'s there are 2n such variables. So integer variables are 2n. And 0-1 variables how many are there? Those are the $x_{ij}$ and $y_{ij}$. So $x_{ij}$ for every pair you will be having, so n into n minus 1; n into n minus one. And how many such equations or constraints

you are having? Total number of constraints, this also you can estimate for the first one corresponds to n for all i. This is n this is n and this is again n 4n. 4 n plus here there are four equations you again have a look at those equations there are four equations with combination of values of i and j. So there are four equations with n into n minus 1. So this is roughly the number of constraints which are there.

But actually we need not have to take care of $B_i$, $B_j$ or $B_j$, Bi; so you can have it divide by two also. That way you get this. So in that way it becomes half. But still you can say there is number of constraints is of the order of n square. So the problem is that if the numbers of blocks are large, then this method is not feasible. Well this method is mathematically very sound. If you can formulate it, you can feed to an ILP program. It will find out a solution. But the problem is that it will take enormous amount of time for big problems. So actually what people normally do? So instead of feeding the original floor planning problem to ILP and trying to solve it first, do some kind of a divide and conquer. It partitions into sub problems and then feed each of sub problems to the ILP solver. Okay, Fine. And as I said earlier that ILP formulations for the other variations also exist, but I am not going into details. Number one thing is that, well you can also take care of orientations of the blocks that whether means if you are rotating a block by 90 degree or not.

See rotating is not very difficult to capture for each block. You can add another variable say $r_i$ which indicates whether it is rotated or not rotated to be 0-1 variable. And if it rotated then $x_i$ and $y_i$ or means $h_i$ and $w_i$ gets interchanged. So the equations you can accordingly modify. Now another modification was suggested that was for the flexible blocks. Flexible blocks say that you have $w_i$ and $h_i$ for each block which gives you the area. Now the keeping the area fixed, you try to find out the actual values of $w_i$ and $h_i$. But the problem here is that if you put the constraint as it is, like this $a_i$ equal to $w_i$ $h_i$ then it becomes a non-linear problem. But there are ways to convert this nonlinear problem into a linear problem, then use it for ILP solution. So the all these things are there. Now in the next class we would be looking at some other methods using which this problem of floor planning have been tackled by different researchers. So thank you.