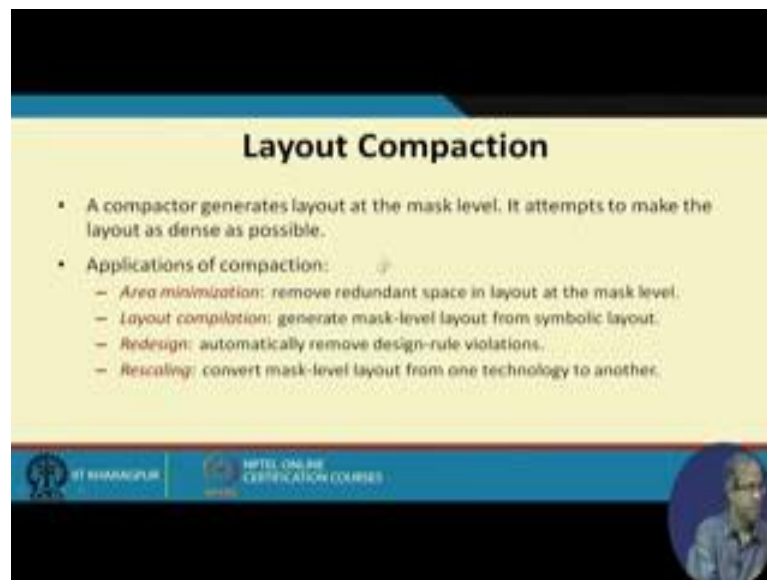


VLSI Physical Design
Prof. Indranil Sengupta
Department of Computer Science and Engineering
Indian Institute of Technology, Kharagpur

Lecture – 46
Layout Compaction (Part1)

So, we start with this lecture on Layout Compaction. Now the basic idea behind layout compaction is this, you see we have seen the various steps of physical design starting with some kind of specification and netlist we go through partitioning placement, floor planning routing then detail physical design. Finally, we create a layout which is nothing other than as we have seen already some collection of rectangular shapes, this rectangular shape may indicate short segments of connections on diffusion polysilicon or the various metal wires, they may represent transistors which is an intersection of a diffusion and a polysilicon rectangle or there can be various contact connections. So, here we talked about layout compaction.

(Refer Slide Time: 01:19)



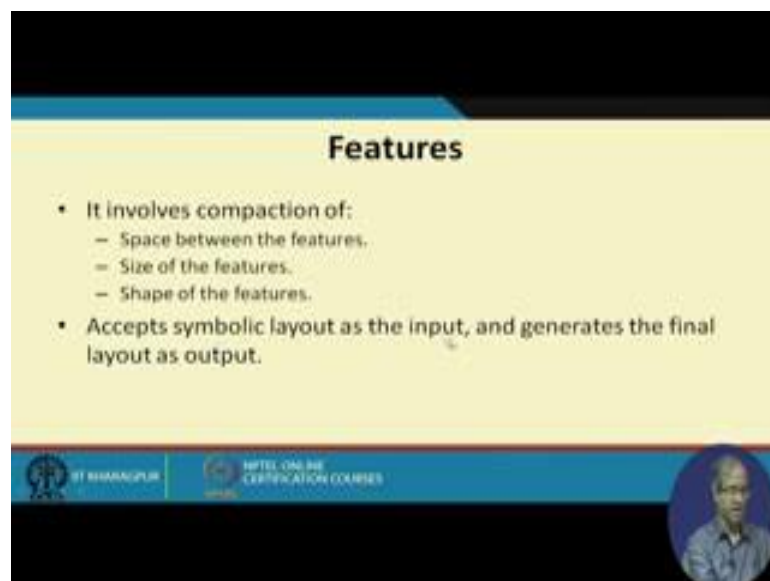
Now layout compactor what it does? It generates or it reduces the area of a layout at the mask level well. Compactor is general it can generate the layout from this stick diagram also like what I have said or given a layout it tries to reduce the area by bringing some rectangle closer to each other. See the idea is this you already have the design rules,

design rule specify that your wires has to be of minimum this width, they have to be of minimum this much separation.

Now, in a layout you may see, but well this separation between 2 features or 2 blocks or objects are greater than what the minimum design rule constraints specifies. So, you can possibly bring them to closer, basically this layout compaction means this you are try to bring things closer to each other without violating the rules or constraints such that you generate or you get a layout whose area is reduced as compared to what you had originally. So, compaction is a quite general tool. So, you can use it for a number of things of course, area minimization by removing redundant space what I was saying.

Second thing you can also use it for layout compilation means starting from a symbolic layout or stick diagram layout, you can directly generate the mask level layout the rectangles following the design rules. Redesign means you can have as I said earlier during our discussion on design rule check. So, you can have a phase where you check for design rule violations in your layout. So, one some violation is detected, this layout compactor tool can rework on that part where design rule violations are taken place and can create a correct design which you do not have any violation; and of course rescaling, when we go from one fabrication technology to another, you can use this kind of compaction tool to convert the layout from one to the other fine.

(Refer Slide Time: 04:08)



The slide is titled "Features" and contains the following text:

- It involves compaction of:
 - Space between the features.
 - Size of the features.
 - Shape of the features.
- Accepts symbolic layout as the input, and generates the final layout as output.

At the bottom of the slide, there are logos for "IIT BHARANGPUR" and "NPTEL ONLINE CERTIFICATION COURSES", along with a small circular portrait of a man in the bottom right corner.

So, when I talk about compaction there are a few things, space between the features, size of the features, and shape of the features. Normally shape of the feature is predefined in many cases, but in some cases there may be some constraint on the shape of the features as well. So, this can accept some kind of a layout as input, and generates the final layout as output. So, let us try to see the problem formulation.

(Refer Slide Time: 04:42)

Problem Formulation

- **Given:**
 - A set of geometric features $M = \{M_1, M_2, \dots, M_n\}$.
 - The minimum feature size, $s(M_i)$, for all i .
 - The minimum separation between features M_i and M_j , denoted as $d(M_i, M_j)$.
- **Objective:**
 - Minimize the layout such that

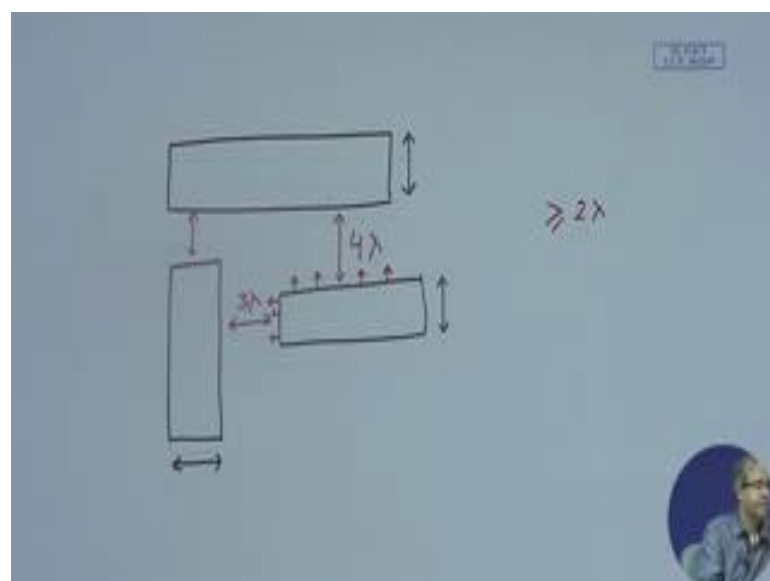
$$\text{size}(M_i) \geq s(M_i)$$

$$\text{dist}(M_i, M_j) \geq d(M_i, M_j)$$
 where $\text{size}(M_i)$ and $\text{dist}(M_i, M_j)$ are size of M_i and distance between M_i and M_j after the compaction, where $1 \leq i, j \leq n$.

IIT BOMBAY
 NPTEL ONLINE CERTIFICATION COURSES

So, we are given a set of geometric features M_1, M_2 to M_n .

(Refer Slide Time: 04:56)



So, the features when we talk about there can be any kind of features, I am just showing them as rectangular blocks without any colors, because I am not showing the layers in general. But what I am saying is that depending on the layers in which these blocks are, there can be several constraints like of course, design rule constraint says that you have to have some minimum width of this lines. So, we assume that those are satisfied you can also check it specifically, but what we are more interested is that, what are the separations between these 2 these blocks?

Suppose you may find that for this one, the actual separation is 4 lambda, but our design rule says that it should be greater than equal to twice lambda. So, in that case we can possibly push this rectangle up like 2 lambda. Similarly let us say this is we say this is 3 lambda, but again our design will say it has to be minimum 2 lambda. So, we can again push this to the left by 1 lambda. So, in this way we can create a layout which will be smaller in terms of the total area. So, there are 2 kinds of constraints which are specified one is the minimum features size.

So, for the each of the geometric features this s_{M_i} will indicate typically the minimum width, and in some cases also height and width, and the minimum separation between every pair of M_i and M_j , this is denoted as $d_{M_i M_j}$. The objective will be given this layout we try to minimize the layout such that size of M_i where size of M_i is the size of this after compaction, and $dist_{M_i M_j}$ is the distance after compaction, they should be greater than equal to the minimum value that is specified for every pair of i and j this is; obviously, the problem formulation this is what you are trying to do.

(Refer Slide Time: 07:33)



Design Style Specific Issues

- Full-custom Design Style
 - Compaction is very critical.
 - After placement and routing, a large amount of space is left vacant.
- Standard Cell Design Style
 - Since the heights of the cells are fixed, the height of the layout can be minimized by minimizing channel height.
 - A restricted type of compaction, called channel compaction, is used.
- Gate Array Design Style
 - Compaction is not applicable as the position of the gates is fixed.

IT MANAGER | INTEL ONLINE CERTIFICATION COURSES

Now, some of the design style specific issues like when we talk about the full custom design style, it is here where compaction becomes very critical. Because the blocks can be located almost anywhere in the chip, so there is no regularity in the layout unlike the standard cell where you put everything on the cells, and they are already predesigned. You see for standard cells each of the cells are picked from a library, they are already predesigned and optimized there is no scope for any further compression or compaction. In the standard cell the only thing you can compact is rows may be the amount of space you had kept between the rows, you did not use that whole space for routing, you can move 1 row above a little bit.


But for full custom it is very general, you can have the possibilities of compaction in almost all ways, typically after placement and routing lot of space are left vacant you have to compact them. So, for standard cell design style as I have said the heights of the cells are already fixed so we cannot do anything there. So, the height of the layout can be minimized only by minimizing the channel height. So, this is something called channel compaction. For gate arrays again because the locations of the gates are already there is no scope for compaction here.

(Refer Slide Time: 09:15)

Compaction Algorithms

- Based on minimum distance between features
 - a) Constraint graph based
 - b) Virtual grid based
- Based on direction of movement of features
 - a) 1-D compaction
 - b) 1½-D compaction
 - c) 2-D compaction

ST BHARAGURU | NPTEL ONLINE CERTIFICATION COURSES



So, when I say compaction, generally we do it for full custom design. So, broadly speaking compaction algorithms can be classified into 2 types first is graph theoretic approach which are based on minimum distance between features, there is something called constraint graph which can be used or the concept of virtual grid. Second broad class is more direct, this along some directions 1 dimensional, 2 dimension or something in between it is called 1 and half dimension, you can move the features and try to carry out compaction.

So, let us look into this one by one. So, the constraint graph based compaction it defines something called a constraint graph.

(Refer Slide Time: 10:01)

Constraint Graph Based Compaction

- Constraint graph $G = (V, E)$
 - Each vertex $v \in V$ represents a component.
 - The set of edges E represents constraints.

Constraint Types

- Connectivity constraints
- Separation constraints

NPTEL ONLINE CERTIFICATION COURSES

Now, in a constraint graph every vertex represents a rectangle or a component, and the edges they represent various kinds of constraints. The constraints types can be 2 types, one is they may say something related to connectivity with 2 wires must be connected to each other, or something related to separation.

(Refer Slide Time: 10:40)

(a) Connectivity Constraints

- If two features X and Y are required to be within a distance s of each other.
 - A physical connection can be represented in the graph as a pair of edges between X and Y , each with weight $-s$.

Diagram illustrating connectivity constraints: A physical connection between features X and Y is shown as a pair of edges between X and Y , each with weight $-s$.

NPTEL ONLINE CERTIFICATION COURSES

Now, connectivity constraints like say here I am give an example, suppose there are 2 features X and Y , and the feature the location is identified by the middle points say along X axis this is the middle of X , this is the middle of y .

The connectivity constraint says that X and Y must be within distance S of each other, this S can be 0 also. So, if there has to be a direct connection this S can be 0 also. So, if there is a constraint like this, then in the graph we add a pair of edges one from X to Y , other from Y to X with negative weights. The weight of the edges will be minus of S , minus of S indicates that it is that this X and Y must be within this range and they cannot be moved away, because if you move it away the value of S will increase and the weight minus S says it will make the cost function worse.

So, the cost function is defined accordingly, when you make such moves. So, the other kind of constraint is separation constraint it says, that 2 blocks must be at least d distance away from each other.

(Refer Slide Time: 12:02)

(b) Separation Constraints

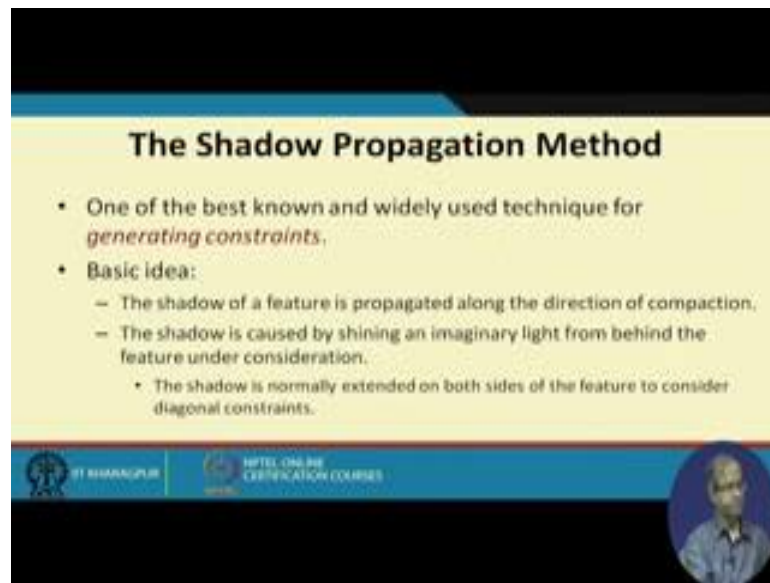
- Two features X and Y are required to be at least d units apart from each other.
 - Represented as an edge from X to Y of weight d .

The slide contains two diagrams illustrating the separation constraint. The left diagram shows two rectangular boxes labeled 'X' and 'Y' with a double-headed arrow between them labeled 'd', indicating a minimum distance. The right diagram shows two circular nodes labeled 'X' and 'Y' with a directed arrow from 'X' to 'Y' labeled 'd', representing the constraint as a directed edge in a graph.

At the bottom of the slide, there are logos for IIT BHARANPUR and NPTEL ONLINE CERTIFICATION COURSES, and a small circular inset image of a person speaking.

So, here you add an edge with a positive weight d . So, in general they will be having a large graph with both positive and negative edges that will be your so called constraint graph. Now the question is how do we generate the constraint graph?

(Refer Slide Time: 12:28)



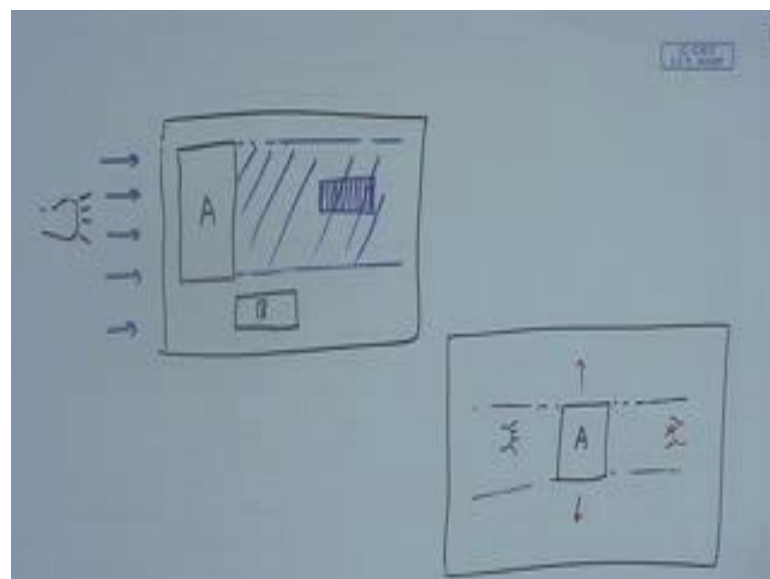
The Shadow Propagation Method

- One of the best known and widely used technique for *generating constraints*.
- Basic idea:
 - The shadow of a feature is propagated along the direction of compaction.
 - The shadow is caused by shining an imaginary light from behind the feature under consideration.
 - The shadow is normally extended on both sides of the feature to consider diagonal constraints.

IT BHARANGPUR | NPTEL ONLINE CERTIFICATION COURSES

Constraint graph means given a number of blocks, blocks means rectangles I am talking about here in this context. So, what are the constraints separation their connectivity and so on. So, one of the very interesting methods of generating this kind of constraint graph is something called shadow propagation. Shadow propagation means the idea just follows suppose I have the layout with me. So, I am using an imaginary source of light.

(Refer Slide Time: 13:26)



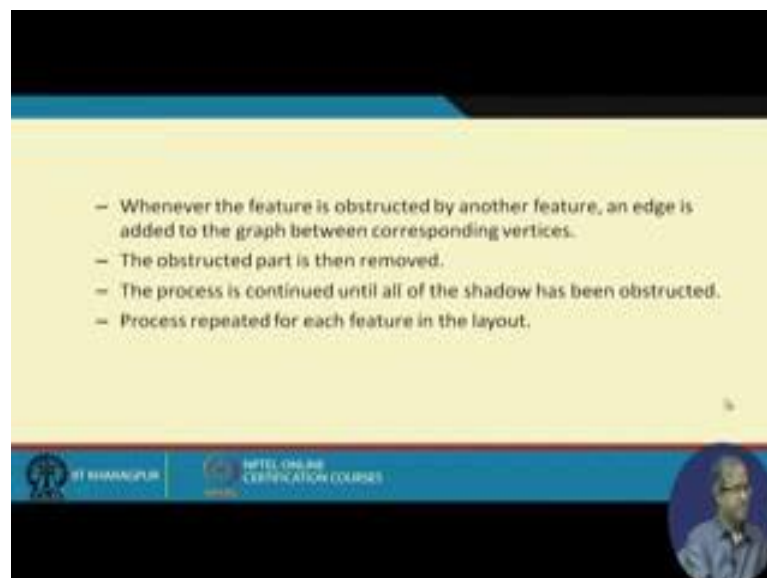
So, I am illuminating my layout from one side, some of the blocks will be generating some shadows like what I mean is that something like this, like I have my layout here let

us say there is a block, block A sitting here. So, I am using an imaginary source of light from this side. So, what will happen if the light falls as parallel rays on this edge, this block A which is opaque will be generating a shadow in this region; so there can be some blocks in this region also let say there was a block here. So, this shadow will totally cover this block such things can happen.

But there is a, but if there is some block out here let us say a block B here, this B will also get the light fine. So, this shadow of the feature is propagated along the direction of compaction, here I am saying from left to right. The shadow which is generating is slowly propagated from left to right, and as I said this shadow is caused by shining an imaginary light from behind the feature and the consideration, and the shadow is extended to both sides of the feature usually not on the right like here what I am saying is that in general for a complete layout so I may consider a feature which is inside in the middle let us say A.

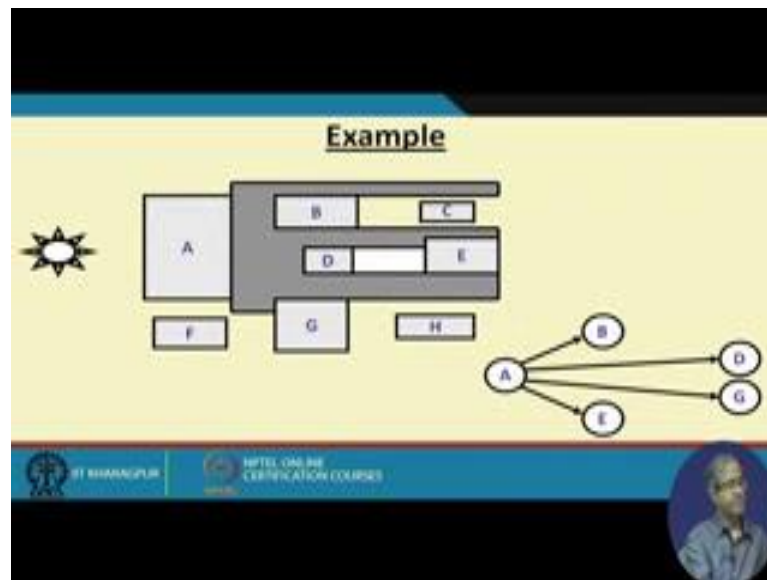
So, for this sometimes I imagine my light to be here so that shadow will be propagated along this direction, but may be after that I will be imagining my light to be here so the shadow will be propagating in this direction.

(Refer Slide Time: 15:32)



So, the constraint on all the sides similarly up similarly down, they will be considered in all directions. So, whenever the feature is obstructed by another feature we add an edge this is the idea, and the obstructed part is removed.

(Refer Slide Time: 15:56)

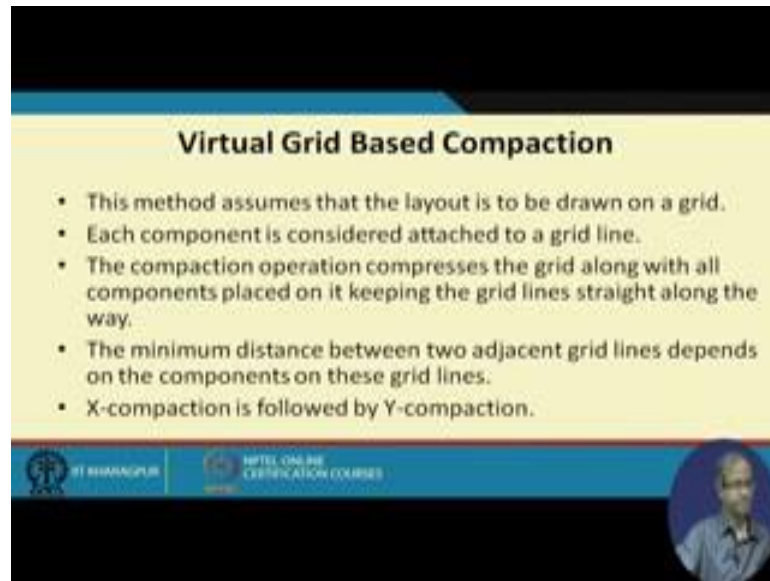


This process is continued until all of the shadows has been obstructed and is repeated by every block or feature in the layout. So, I am illustrating with the small example let us take this. Suppose we are illuminating this light with respect to a block A. So, we are trying to look at the dependency of the block A. So, we are illuminating a light from this side, this light reaches A this light also reaches F, but you see because of the geometry it does not reach see for this there will be a shadow. So, I am extending this a little bit. So, this is the shadow I am showing, the shadow is being extended to B, but the shadow does not touch C because C is hidden by B right.

Similarly, here the shadow reaches D it also reaches part of E. Shadow reaches G, but does not reach H, because H is hidden behind G, but light directly reaches F. So, with respect to the shadow see as shadow reaches where reaches B, reaches D, reaches G and reaches E. So, there will be one edge in the constraint graph corresponding to all these. So, and you when you add these edges depending on the design rule you can also add the weights. So, A and B minimum how much separation it should be?

Suppose this is polysilicon this is also polysilicon this should be 2λ , similarly A and D how much minimum separation A and D, how much separation like this. So, this you repeat for all the blocks and we will getting all the edges in the graph right. So, this is the basic idea behind generating the constraint graph.

(Refer Slide Time: 17:45)



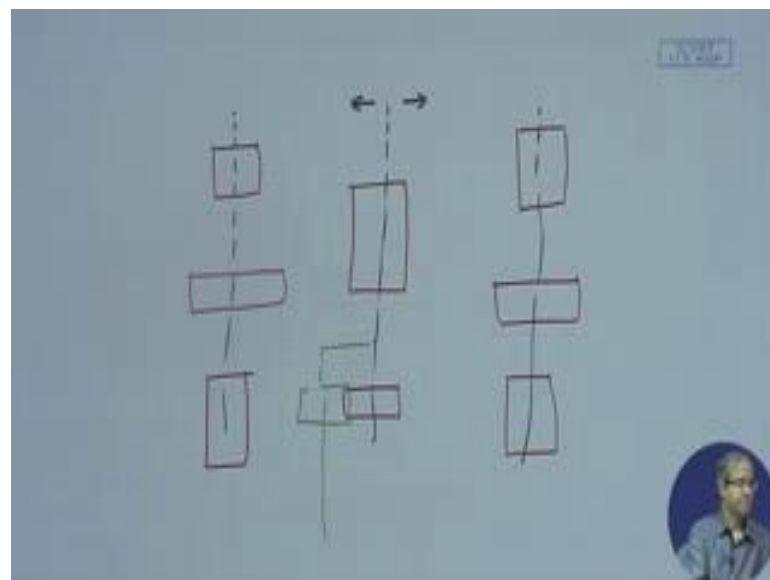
Virtual Grid Based Compaction

- This method assumes that the layout is to be drawn on a grid.
- Each component is considered attached to a grid line.
- The compaction operation compresses the grid along with all components placed on it keeping the grid lines straight along the way.
- The minimum distance between two adjacent grid lines depends on the components on these grid lines.
- X-compaction is followed by Y-compaction.

IT BHARANGPUR | NPTEL ONLINE CERTIFICATION COURSES

Then we come to something called virtual grid based compaction. So, here we are imagining a grid on the layout surface. So, where assuming that the components are all attached to a grid, it means it is something like this.

(Refer Slide Time: 18:09)



Let us say we imagine some grid lines, let us say in one direction only I am showing and the components on the rectangles they are all tagged with the grid, one is like this, one is tagged like this other is may be like this.

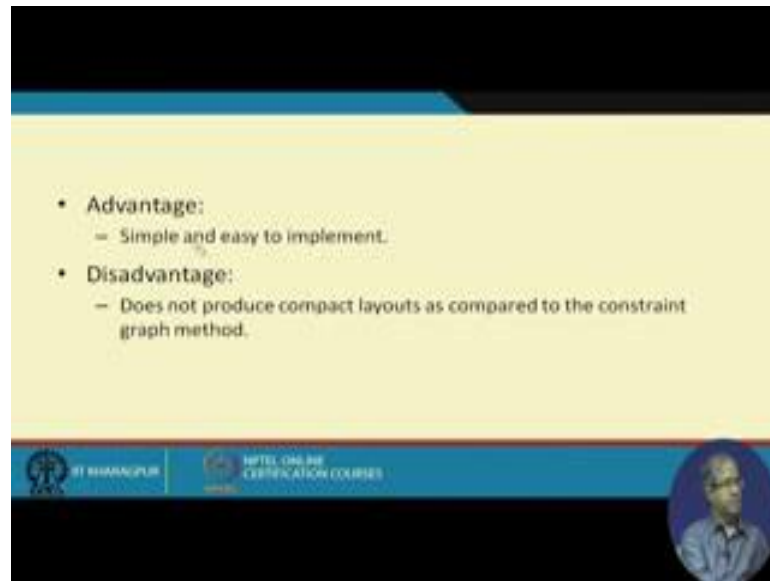
Similarly, here there can be one object like this, one object like this. So, here what I am saying is that here we are allowed to move this grid lines to the left or to the right whatever is applicable, but when you move it here all the objects which are attached to the grid, they will all move simultaneously. This grid line if you move to the left all this three blocks will also move. So, you look at the separation minimum separation, you see that whether the separation is greater than the minimum permissible, if not you move it as long as there is no design rule violation.

As soon as there is there is a violation you stop there. So, here each component is considered attached to a grid line, the compaction process compresses the grid along with all components placed on it. So, they all move together, keeping the grid lines straight like we do not make the grid line crooked we will see a version later, but you can do it. Grid line crooked means let us say for this grid. So, I do not move this, but I move this one a little left let say here. So, now, my grid line becomes like this, it becomes zigzag this is not allowed in this method.

So, the minimum distance between 2 adjacent grids depends on the components on these grid lines. Component means let say this can be metal this can be diffusion, this can metal this can be polysilicon, this can be metal. So, depending on them the separation will be defined. So, you can do compaction along the X direction, you can also do a compaction along the Y direction. So, you can define the wires vertically which means you are doing X direction compaction or you can define this grid lines I mean you can define that horizontally, means you will be doing compaction in the vertical direction ok.

So, you can do X compaction followed by Y compaction or vice versa both you can do ok.

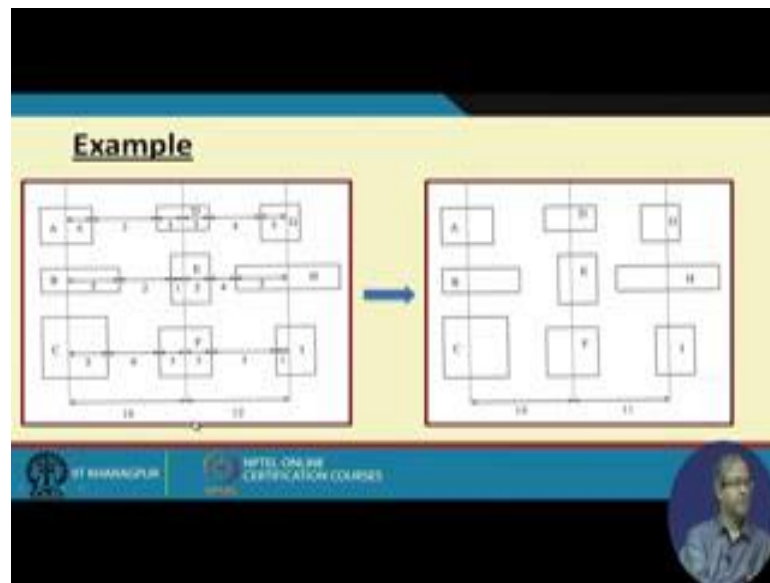
(Refer Slide Time: 21:05)



- Advantage:
 - Simple and easy to implement.
- Disadvantage:
 - Does not produce compact layouts as compared to the constraint graph method.

So, the advantage is that it is fairly simple and easy to implement, you have to layout with you containing all the rectangles, you just assume the grids and you move the blocks around to align them with the grids, then let the grid lines move as much as possible you can compact. But the drawback is that it does not produce very compact layout as compared to constraint graph method for example, because you are constraint. So, when you are defining a grid you are saying that there are 10 blocks connected to the grid. So, either they all move together or none of them moves, but it may be possible that out of this 10, 4 of them you could have moved, but the other 6 you are not able to move, that you are not allowing here right.

(Refer Slide Time: 22:00)



So, a very simple example let us consider situation here where there are 9 such blocks as objects/features. So, the separations between them are shown as 3, 2, 6, 4, 4, 3 and the distances between these 2 grid lines are 16, and 7, and the third grid line is 12. Now you may see it will actually do not show, but these may be some features are set that you can bring them closer together. So, by doing that this example shows that you have been able to bring DF closer together so now the distance becomes 14; similarly GHI also you move closer together this distance becomes 11 something like this. Similarly when you move in the Y direction you will be imagining a grid like this, moving in the other direction right.

So, this is exactly what we do here in case of this kind of virtual grid-based compaction, where given a complete layout which I am repeating; given the complete layout you imagine grid lines, and as when I am required to align these small rectangles to the nearest grid, then let the grids move with all the attached blocks or objects with it. So, this is how you are doing the compaction. So, here the checking for the design rule is also simple and of course, this algorithm will be linear in the number of blocks, so the time complexity will not also be very high all right. So, this is the method. So, we shall later see means another variation where you do not restrict ourselves to a, you can say vertical grid. So, grid can be broken and made zigzag if required this can be one variation.

With this we come to the end of this particular lecture, in the next lecture we shall be looking at some more methods of layout compaction, in particular the more direct methods those 1 D, 2 D and 1 and half D.

Thank you.