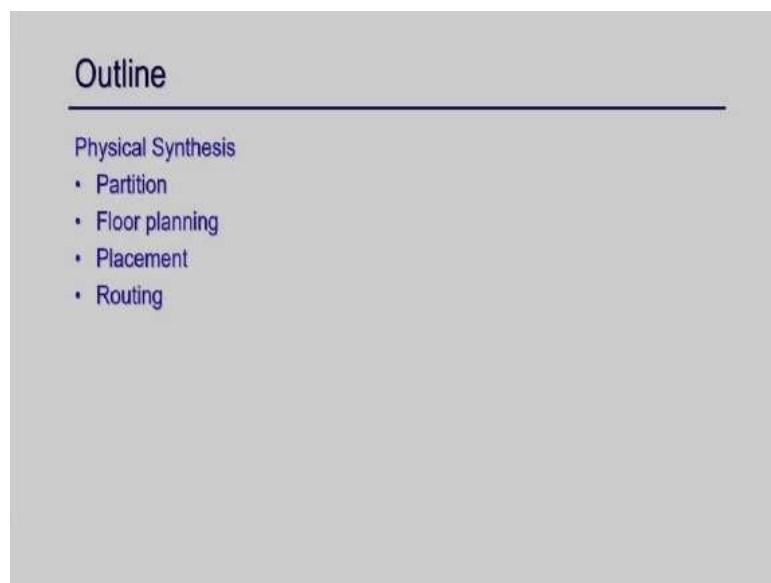


C-Based VLSI
Design Dr. Chandan
Karfa
Department of Computer Science and Engineering
Indian Institute of Technology, Guwahati

Module - 12
Recent Advances in C-based VLSI Design
Lecture - 41
Physical Synthesis: An Overview

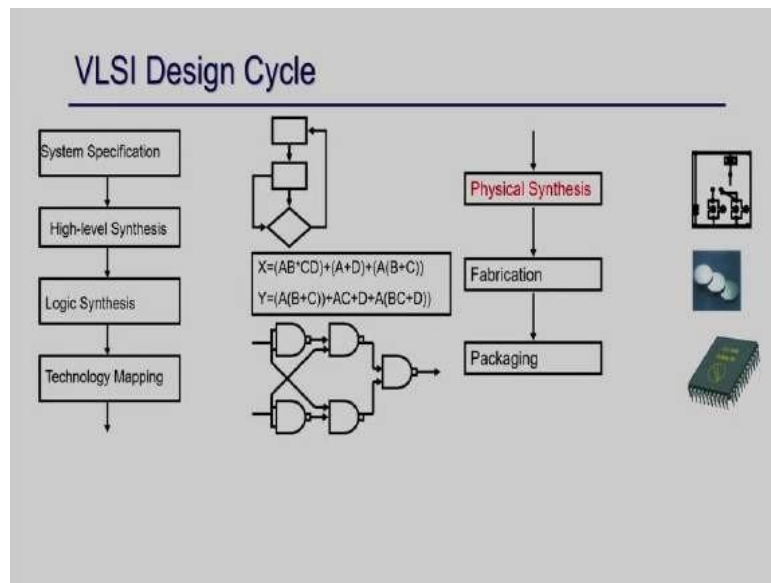
(Refer Slide Time: 00:54)



Welcome, everyone. So, today we are going to discuss Physical Synthesis. So, specifically, we are going to talk about all the sub-steps of physical synthesis; partitioning, floor planning, placement, and routing briefly. In physical synthesis, if you want to cover it might take maybe 20 lectures.

So, instead of going into detail or each technique in each techniques detail what I am going to talk about in this, I would like to try to give a brief overview of the whole physical synthesis process what is the objective of these sub-steps and what is the problem definitions, and what is the generic way to solve that. So, instead of going to solve them using exact algorithms.

(Refer Slide Time: 01:34)



So, let us start. So, if you look into VLSI design cycles, we have seen that we have already discussed that it starts from some specification, then goes through high-level synthesis, logic synthesis, technology mapping, and then physical synthesis, fabrication, and packaging right. And we already have discussed high-level synthesis, logic synthesis, and technology mapping. And we are going to talk about physical synthesis today.

So, if we just think about after the technology mapping step, what do we have? You have a transistor level or sorry gate-level circuit, which have I mean it has is a huge circuit and those are actually represented by the cell liabilities it means if it is FPGA, then it might be this LUTs RAM DSPTS or if it is A SIC then it is the library elements like NAND gate NOR gate a flip basic flip-flops ramps and alright. So, this is something that representations.

And now in physics physical synthesis what we are objective, we have to give a physical form of your chip right. Because so, far we have talked about a lot of optimizations, synthesis, and all those stuffs. Now you have to give a physical representation of your design right. So, which is important.

And now what do we do what do you mean by physical synthesis? Physical synthesis in the sense, that now you have to place all your gates or those transistor gates, which actually need to be converted to transistors in this physical synthesis. And then they

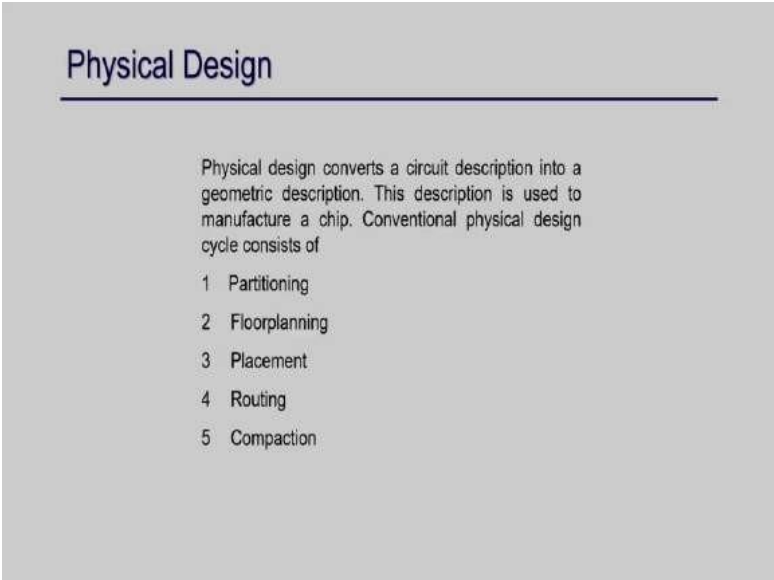
will be placed right in your actual chip area. And all the interconnection all the edges between the nodes that have to be connected physically in you within the chip layout area.

So, that is what is called physical synthesis. In the physical synthesis, you try to place those nodes or the basic units of your design into the physical area, and also, we try to connect them properly and your objective is always to try to find out the minimum area to pack all of your gates in some in an in-shear layout and you try to make the interconnection cost as low as possible right. So, this is the basic overview.

Once this is done, we will go for fabrication. And you also remember that at this point this gate-level design will be converted to the transistor-level circuit, which is a very obvious transformation because some just replace of equivalent circuit representation of each gate right.

So, that is not a big task of physical synthesis. So, we can easily convert those gates to the transistor level circuit, but what is important here is making a physical placement of those gates and making physical interpretations right.

(Refer Slide Time: 04:11)



Physical Design

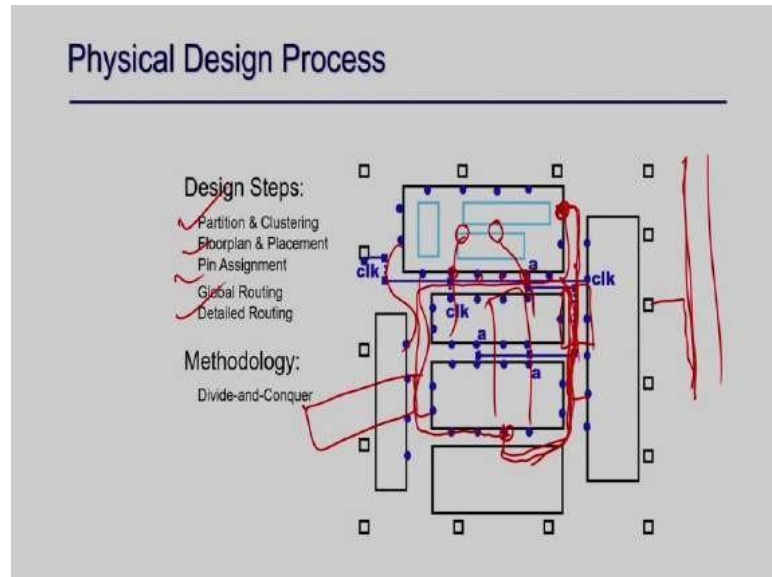
Physical design converts a circuit description into a geometric description. This description is used to manufacture a chip. Conventional physical design cycle consists of

- 1 Partitioning
- 2 Floorplanning
- 3 Placement
- 4 Routing
- 5 Compaction

So, let us go into the physical design steps. So, in physical design, the sub-steps are partitioning, floor planning, placement, routing, and final compaction. So, we are

going to talk about partitioning, floor planning, placement, and routing. What is then? Now, what is the objective of the sub-steps right?

(Refer Slide Time: 04:31)



So, if you look into this design so, when you have a big design. So, the first thing we want to do we want to our objective is to suppose, your design has said one million gates. So, we want to place them, but we try to place all these one million at a time is kind of (Refer Time: 04:49) job right, what is the basic approach we usually follow in every life so, whenever you have a big task, we try to make it we partition it, we have to make it sub task and we try to solve the sub task and then we try to combine the result.

So, this is the same approach is happening here, we want to place the whole thing together into that layout area. So, instead of doing that we try to partition the whole thing right, we make the small component, and we want to place each component at a time. And also, it is kind of a hierarchical or recursive approach.

So, suppose if you have the big circuit, we want to play around with its partitioning say 10 components. So, then I place these 10 components in the layout. Then each component we again break and we again try to place each sub-component of our component again right. So, it is kind of a recursive approach.

So, the first thing in this process is that you partition your big design into small components. So, that you can place each partition right. So, that is what is called the partitioning right, or clustering.

And then what is the floor planning and placement is that, when you have this each component that has to be also placed right. So, each component has to be placed in your area right. So, this is what is called placement floor plan and placement the floor plan and placement, I am going to talk about detail.

So, basically, you plan your floor first of all if you have to say 10 components how should place this can be placed this way or this can be placed this way also right, I mean you can think about either vertically or horizontally right. So, similarly, this can place this way the or the vertically as well.

So, you can have a lot of possibilities and maybe some other organizations give you a compact area instead of this organization. So, this floor planning and placement take care of that, once you have the small component the partition has to be placed properly so, that your total area is minimum.

And then this pin assignment is something. So, as we understand that from this component, there may be many interconnections going to some other places right it may go to this place this may go to this place, this may go here, this may go some other places right. So, there may be a lot of interconnections so, that is what is. So, basically, if you have a gate that will try to communicate right so, you have to assign some pin, because now you have a making interface or this module right.

So, whatever the interconnection from here is going out that has gone through some pins right so, these are the port kind of. So, we have to assign a pin for each component, which is called pin assignment and then you have to make the actual connections right. So, then once you have this then you have to make the global routing.

So, for example, you want to make the connection from this node to this node. So, you decided that this will go through this part or it may be in some scenario it might go through this also right it might go through this also, because of some reasons. So, that is what is called routing.

So, you know that I have to make a connection from this port to this port and you can understand for a bigger layout, there may be multiple such possibilities right and you have to make this connection exact right. So, which through which part of this channel it should go that you have to decide that is what is called global routing right.

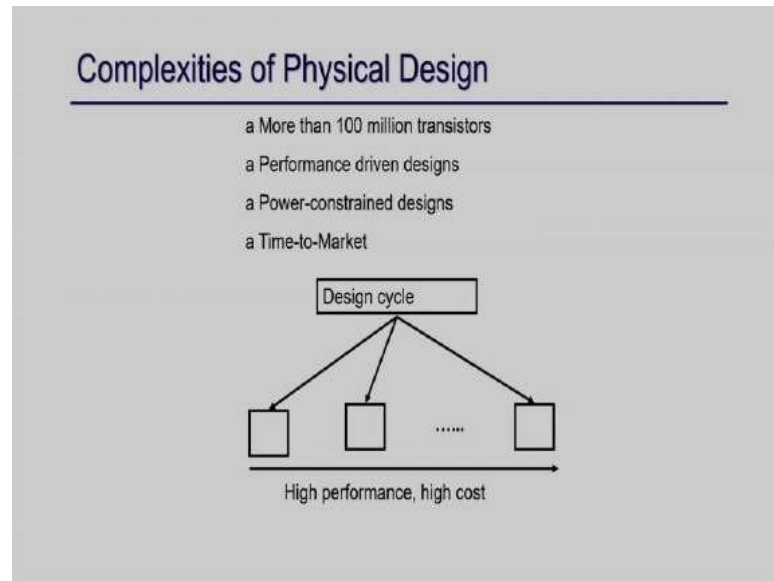
So, then once the global routing is done then you have to make the detail routing what is that so, detail routing is something once you have decided that there are four that will go through this portion of the design. Then you have to decide which there may be a track right.

So, this track this will go and the second track this will go there may be some track, which can be shared by 2 components we will discuss those detail so, which is called detail routing. So, that you do, you understand the basic essence of this whole process, that you break your big design into multiple components with this partitioning.

And then you decide for each component what is the exact area, and then how to place them optimally so, that your interconnection cost will be less and then you make the actual connection is ok, the previous that is pin assignment you make the point of your how many pins you require to make the interconnections for each component.

And then you go for global routing; global routing means you try to make the actual through, which region your particular where we will go that you will decide and then you make the actual detail routing exact track through which that particular connection will go so, that is what is called detail routing. And these are the overall steps of your physical design process.

(Refer Slide Time: 09:23)



So, what is the problem of the difficulty is in physical synthesis is basically, you can see that the number of transistors may be 100 billion or 10 million; it is a huge right, and handling, such scale design is not easy right. And also, sometimes you have that performance-driven design something like you want to achieve this timing like this your power dissipation should not go beyond this your area should not go beyond this particular (Refer Time: 09:48) right.

So, you have to always meet a target right so, this performance depends. And also, as I mentioned power constraints are also there. So, we cannot use arbitrary powers if your very long interconnection that will consume low power so, I mean a lot of power. So, you have to always also think about that your power constraint also meets and also the time to market so, you cannot just take infinite time to just do the physical to get a generate a physical chip which is optimal.

So, you always have time to market or constraints or the, and the pressure always from the marketing in order to finish your things within the time budget.

(Refer Slide Time: 10:32)

Why Physical Design still Relevant?

- Many existing solutions are still **very** suboptimal
 - E.g., placement
- Interconnect dominates
 - No physical layout, no accurate interconnect
- More new physical and manufacturing effects pop up
 - Crosstalk noise, ...
 - Manufacturability, reliability, **security**...
- More vertical integration needed
- Physical design is the **KEY** linking step between higher level optimization and lower level modeling

So, that is why this physical synthesis is very a complex process and I mean it has to be we will discuss all these sub-steps and there are a lot of methods are there. And also, one more important point has to be noticed here that all these sub-tasks that I talked about this placement, floor planning, global routing detail routing all are most more most of them are kind of NP-complete problem,

The NP-complete problems in the sense they have did not have a polynomial-time, they are not polynomial-time solvers they are kind of they if you are trying to find an optimal solution for them you need an exponential algorithm right. But physically you can understand with a hundred million transistors it just will not work right.

So, in most cases, you have a lot of heuristics. So, that is why basically this means physical synthesis find a lot of approaches there all are kind of heuristic base approach because all exact solution is not scalable you always have to think about some heuristic that is logical, which works for most of the cases right. So, that is why there are a lot of variations of the algorithm available for this sub-task and it might cover the whole semester just to discuss the physicals design right.

So, that is also another part of this the complexity of this physical design right. So, now, if you move on why and you can see that in this physical synthesis all the tools are available industry tools is available right. So, they are already mature they are working

fine, and then why need to think about more on this physical reason why this is still relevant.

So, the point here is that yes, the placement of all those things is available, but since you understand there are sub-optimal, they may not work always right, and most importantly the new whenever this the as the things as time move on the new complexities are coming up right.

For example, the reliability of your chip you try to interconnections are going very high nowadays and your and also this cross-talk you want to minimize. So, the new reliability of your design and security is also a big concern nowadays. So, all these new aspects are coming key aspects are coming new design you have to develop or you have to emerge with some new placement at algorithm or partitioning algorithm, that is would actually support this kind of new requirements of your design.

So, for example, the security is something is a very very important part in important issue nowadays you can think about the recent news on that processor the processor that almost all Intel processors or AMD processors are not they are vulnerable to attack right because of that branch prediction you might be aware of that, they might leak some information from that power which can be, which can be about the software that is running on that and mean some attack can, I mean attack can happen to your processor also.

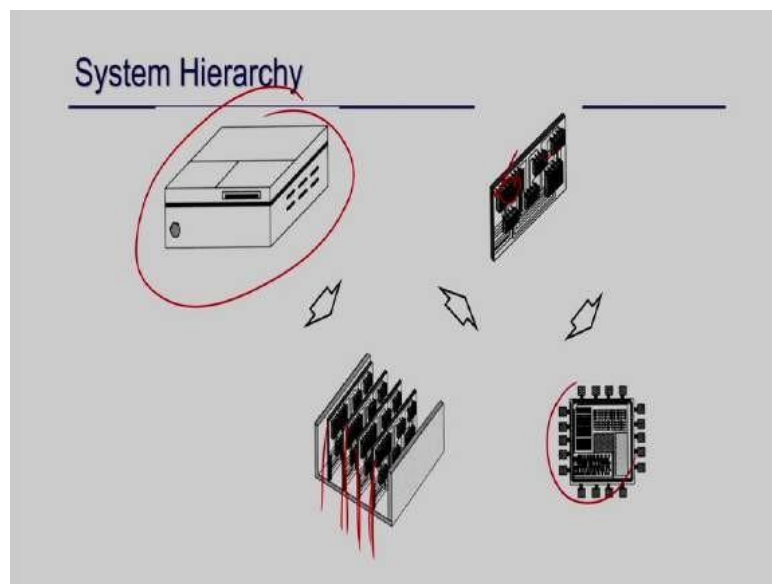
So, the kind of algorithm that is so, far developed is not concerned about that. So, just to this is just to give the highlight that you might have to think about new kind of placement or new kind of partitioning approach, there may be some new security constraint will come, because of the existing approach that is we have to always try to think of evolving new kind of technology or the algorithm or the technique that can support this new kind of constraints right.

So, this is also that is why still this physical design is relevant and we all we still talk about those synthesis steps right ok.

(Refer Slide Time: 14:04)



(Refer to Slide Time: 14:21)

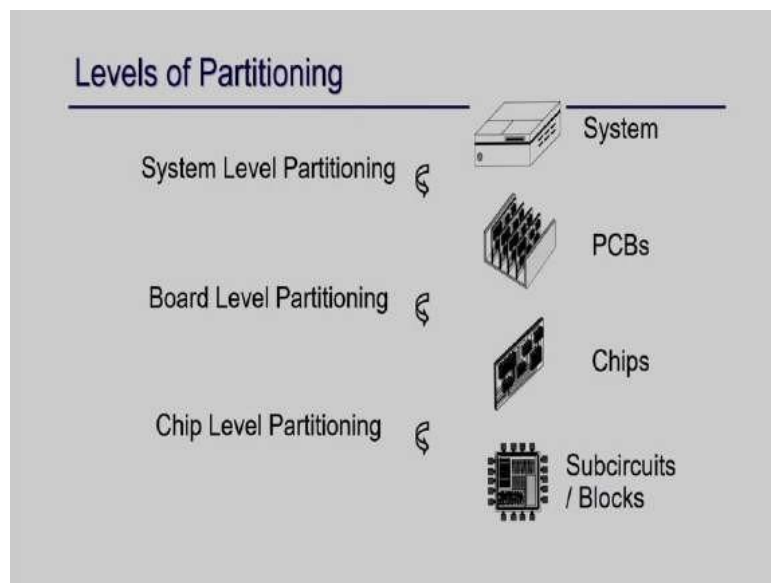


Now, I will move on to each sub-steps in each sub-step, what I am going to do is just going to define the problem and discuss the issues and then I will without going into each I mean all these existing algorithms on that ok. So, we will start with partitioning. So, partitioning is something very common in every approach right, if you just think about [FL] is just the A SIC prototyping what we do actually we have a big system right this might be half's board right. And so, where we just stack multiple FPGA, there are multiple stacks right.

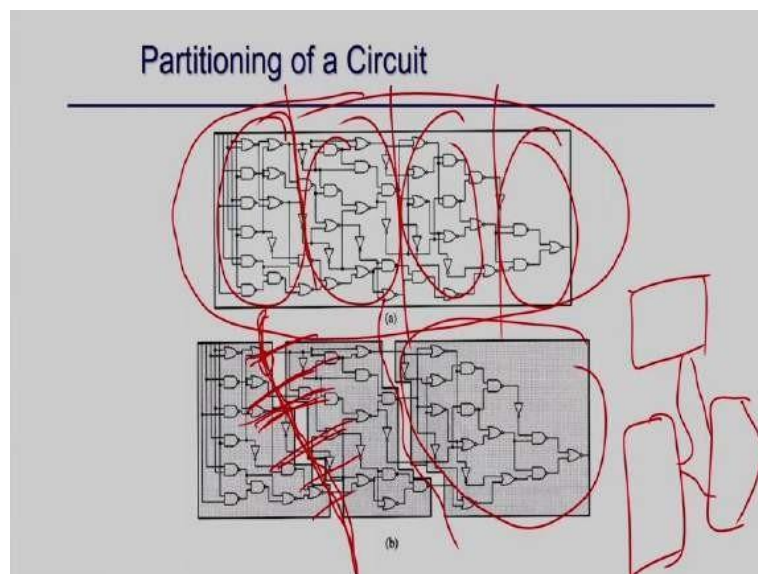
So, I want to map a big processor in a FPGA board and its not fit in 1 FPGA it might need 100 FPGA. So, that is how we stack right. So, we have some multiple layers and then each layer we have multiple FPGA, and then we probably and each of them is kind of FPGA right this is just an example.

So, similarly, whenever you have a big problem, we try to partition it and then we try to merge to get the solution and that is also the same approach you are going to solving this partitioning of a circuit right.

(Refer Slide Time: 15:01)



(Refer Slide Time: 15:03)



So, whenever you have a big circuit, our objective is finally, to place all these, for example, here is a big circuit is given our objective is to place all these gates in a chip area in the minimum possible area. So, one approach is that you take 1 by 1 and you place them it is not physically possible that I have already discussed.

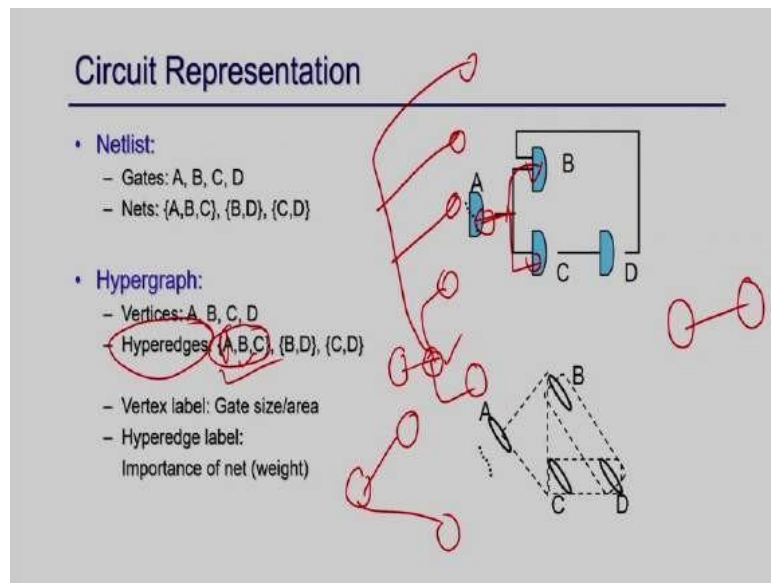
So, basic idea is that partition your design right. So, you partition your design into small components and you try to place this together you try to place these together something like this right. So, similar to this is whatever shown is there.

So, we have a partitioning here and we have these are the three partitions and I try to place this partition independently right. And once we have done, I can again recursively do the same partition within this component or see in this component the unless I have a module, which is easy to handle right. So, this is what partitioning of the circuit means we have to partition.

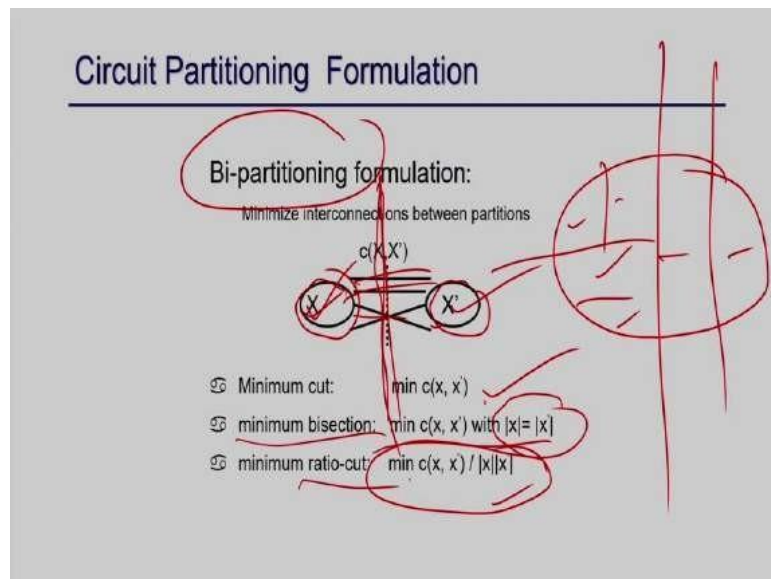
And whatever the things we have to take care of here are that this cut whatever the number of wire crossing this cut right. So, if you make a partition here you have to identify, how many wires are crossing rights. Because those are wire has to be routed later right. So, because you understand when you have this break these things and you have a say there are three components as I have shown here and now all the connection that was coming from this crossing this cut has to be routed here right so, they have to be routed here.

So; that means, your objective will be you are trying to find a cut, which has a minimum number of such routing logic right. So, that actually gives you the benefits that you have a lot of fewer interconnections and that is a good partitioning of your algorithm.

(Refer Slide Time: 16:46)



(Refer Slide Time: 16:49)



So, that is what is important. So, if you just formulate the circuit partitioning problem. So, our objective is something to try to bi-partitioning so, basically, if you think about the partition can be anywhere right it can be k-partitioning.

If I do part in k components; that means, it is a k partitioning if it is bi-partitioning you try to make it bi. So, you can do a recursive that I break the whole circuit into two then the (Refer Time: 17:13) will be again split into two and this is kind of a bi-partitioning right. So, this is the whole circuit I make it by two then this component again I make a by

two this one I want to make a by two then this component I make a by two this also I make a by two, and so on. So, this is what is called bi-partitioning.

Where I am going to partition the whole design recursively right. And so, if this is I say partition into two-component x and \bar{x} and this is my cut. So, what will be our objective, our objective is to min-cut the number of lines crossing here should be minimum, that may be your objective or it may be that you try to make a bi-partition into two partitions, where the number of nodes in this component and number of components is same, which is called a minimum bi section that is the bi-section, where the number of elements in both partitions is exactly same right.

Or you might have minimum cut stress you heard what it means it may be that the number of wire components crossing here divided by the number of nodes into several nodes is the minimum right. So, this your objective might be different, but apparently, the main objective is something you try to reduce the number of wire crossing that so, crossing in this cut, which is something will be finally, has to be taken care of whether routing algorithm. So, that will be our primary objective right.

So, and that is something this is how we can define the partitioning problem. So, one point here to be noted is that your graph, I mean usually we in a graph we consider the node between edges right so, what we have discussed that. But, whenever you have a multi-pin router, this AND gate output is going here and here.

So, then how do you represent this node right, this is something called hyperedges right there is one source and multiple-output. So, this kind of edge is called hyperedges, where you have to a represent the edge is between a set of nodes right, where the first start node is the source and the rest of the destination right. So, here A B C means A is the source node, and B and C is there right.

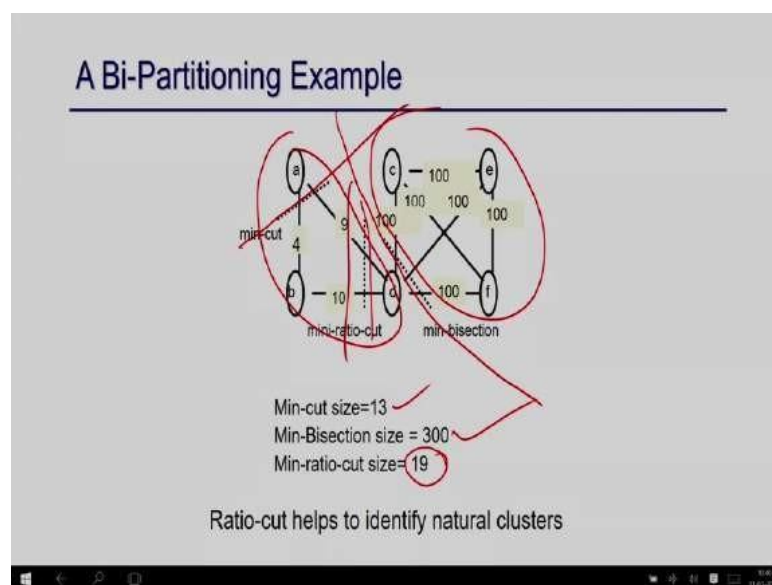
So, this is something like this. So, this is how we can represent the hyperedges. So, we can argue that I can put two edges here like this is also representing the same thing, but it will create the number of edges is more and it will create things more complicated right.

So, usually, we handle these hyperedges Steiner graph right. So, we think about this I can put a node here and this will become the same thing this is a Steiner node

and it will be a Steiner graph. And if it is only two values it might have it is 100. So, then it is this is something we can represent like this is a Steiner we will do a, discuss this more detail later ok.

So, here as I mentioned is formulation is that I want to make a find a cut such that between two partitions either my it will be minimum cut the number of crossings is less or the bisection have is exactly bisection of that, each component has an equal number of elements with a minimum number of cut here, or I can have a minimum cut ratio.

(Refer Slide Time: 20:17)



So, here is an example suppose this is my graph, where this 4 means the number of edges between these two this is 4, right there may be a greater number of edges, I just pick the count this is 100, means the number of interconnections between d and f is 100. So, min-cut is this right, if you just think about a cut here how many nodes are crossing 100 plus 100 and 300, if I cut here the number of nodes is crossing 4 plus 9 13. So, this is a minimum cut right.

And this is a minimum bisection because now if I cut here the number of nodes is there is 3 this is several nodes are 3 right and this is what is the minimum cut ratio because here the number of nodes here is. So, if you make a min-cut this part of the circuit has 5 nodes and this part of the circuit has only 1 node right. Whereas, here if you cut here, I have a number of nodes on this side is 4 which is side 4 right.

So, as I mentioned here is that min cuts why this is min-cut because the number of nodes is crossing here 2 you can think about this is also a min-cut number of nodes is crossing here is 2 number of edges is crossing is 2, but the weight is 19 here this is 13 so, this is the min-cut right.

And this is no other part where we have only 2 nodes passing right this is a min-cut ratio because in this case, the number of nodes is 19. So, this is 19 and this is the bisection where because this bi-side also has 3 nodes this side also has 3 nodes and the number of nodes is crossing 100 plus 100 plus 100 3 and 300 right. So, there are 300.

So, your partitioning is something like this you always try to partition it may be repetitive or it maybe you may in general, you may partition into k partition as an example here has given formulation here.

(Refer Slide Time: 22:12)

Circuit Partitioning Formulation (Cont'd)

General multi-way partitioning formulation:
 Partitioning a network N into N_1, N_2, \dots, N_k such that

- ⊕ Each partition has an area constraint

$$\sum_{v \in N_i} a(v) \leq A_i$$
- ⊕ each partition has an I/O constraint

$$c(N_i, N - N_i) \leq I_i$$

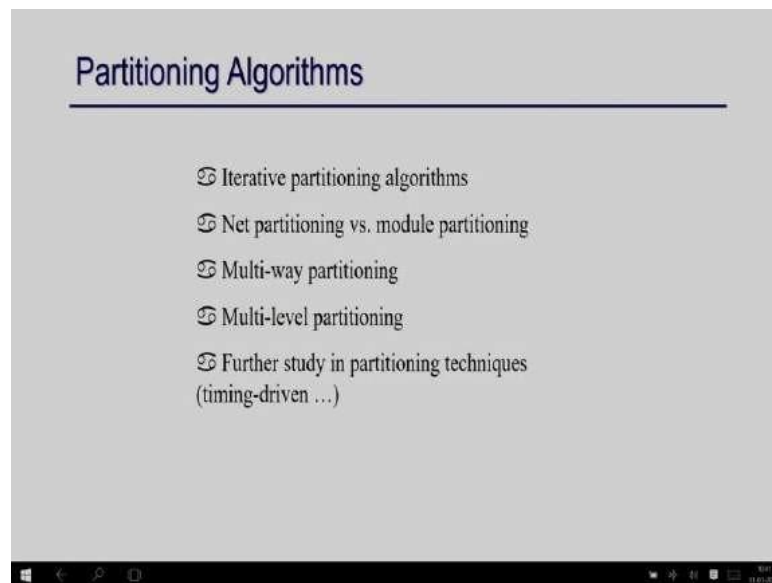
Minimize the total interconnection:

$$\sum_{N_i} c(N_i, N - N_i)$$

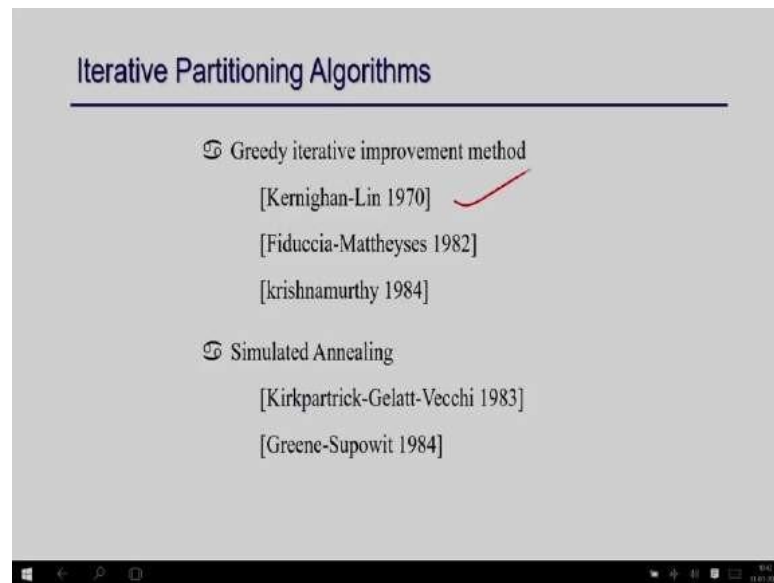
You may have to try to partition into at a, exactly say in the same time in k partition such that each component area there might be some area requirement that each component area is within the permutable size and the number of edge crossing there the cut size is always also the crossing that each cut should have at least this, I mean it should not go beyond this particular number of the cut size should not go beyond some I. So, that constraint has to be magnified, and also, we try to minimize the total cut size right.

So, it is something like this if you have a big circuit you try to break it into a k-partition like this. So, this is partition 1 this is partition 2 partition 3, and something like this. And each part of the partition should satisfy the area constraint each cut should satisfy the maximum possible cut size requirement and the total number of edges crossing each cut should be minimum right. So, this is what is the general circuit partitioning formulation right.

(Refer Slide Time: 23:17)



So, in partitioning, as I mentioned this is an NP-complete algorithm we can have various kinds of approaches it can be iterative that you try to do one partition at a time and then move on or it can be module partitioning versus net partitioning or it can be multi-way partitioning multi-level partitioning or it can have a timing driven partitioning and so on.



The slide is titled "Iterative Partitioning Algorithms" and is divided into two main sections. The first section, "Greedy iterative improvement method", lists three algorithms: [Kernighan-Lin 1970], [Fiduccia-Mattheyses 1982], and [krishnamurthy 1984]. The second section, "Simulated Annealing", lists two algorithms: [Kirkpartrick-Gelatt-Vecchi 1983] and [Greene-Supowit 1984]. A red checkmark is next to the [Kernighan-Lin 1970] reference. The slide has a dark blue header and footer with navigation icons.

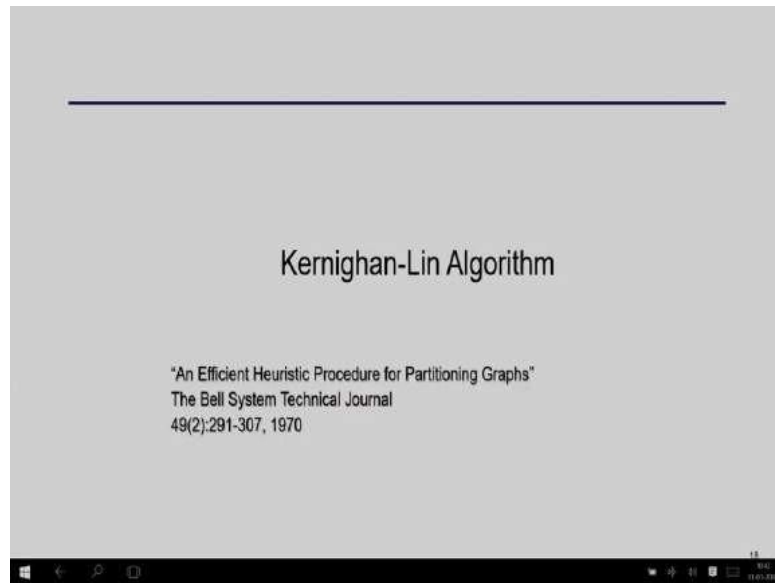
Iterative Partitioning Algorithms

- ☞ Greedy iterative improvement method
 - [Kernighan-Lin 1970]
 - [Fiduccia-Mattheyses 1982]
 - [krishnamurthy 1984]
- ☞ Simulated Annealing
 - [Kirkpartrick-Gelatt-Vecchi 1983]
 - [Greene-Supowit 1984]

So, there may be various kinds of techniques are there and also if you just go into this approach like iterative improvement algorithm like Kernighan-Lin algorithm Fiduccia-Mattheyses algorithm, Krishnamurthy's algorithm. Similarly, simulated and annealing approaches are there.

So, there are various kinds of approaches there it is not possible to discuss all of them here, I just give you a brief overview of the scaling algorithm. So, that you understand how this is practical and this is the simplest version of the algorithm brief compared to the other one right. So, that will give you a brief idea of how partitioning works right.

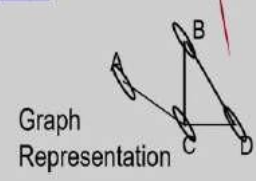
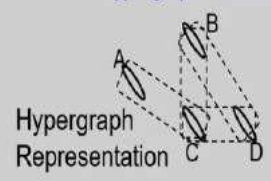
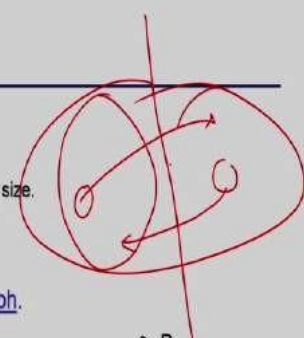
(Refer Slide Time: PAGE 23-59)



(Refer Slide Time: 24:12)

Restricted Partition Problem

- **Restrictions:**
 - For Bisectioning of circuit.
 - Assume all gates are of the same size.
 - Works only for 2-terminal nets.
- If all nets are 2-terminal, the Hypergraph is called a Graph.




Hypergraph Representation Graph Representation

So, if you just move on so, in this is actually do the bisection right. So, basically, you try to make the circuit into 2 right. And the idea is that you start with an initial any arbitrary partition you take a just any arbitrary partition you try to bi-partition the circuit and then in each iteration you try to take a note here and I try to ok node here and you move it here and you try to move it here. So, after every iteration, the number of nodes here and the number of nodes here remain the same right.

(Refer Slide Time: 24:43)

Problem Formulation

- Input: A graph with
 - Set vertices V . ($|V| = 2n$)
 - Set of edges E . ($|E| = m$)
 - Cost c_{AB} for each edge $\{A, B\}$ in E .
- Output: 2 partitions X & Y such that
 - Total cost of edges cut is minimized.
 - Each partition has n vertices.
- This problem is NP-Complete!!!!



So, here is so, basically, if you think about it initially my graph has 2 end nodes after the partitioning each component has each partition have, n nodes, and an E there are m number of edges. So, that there is it actually defined into 2 it partitions into this is X -component this is Y -component this is also N node this is also N nodes and so. And then you try to minimize the cut cost right. So, this is what is something the total edge cost is minimized right this is how the whole thing works.

(Refer Slide Time: 25:19)

A Trivial Approach

- Try all possible bisections. Find the best one.
- If there are $2n$ vertices,
of possibilities = $(2n)! / n!^2 = n^{O(n)}$
- For 4 vertices (A,B,C,D), 3 possibilities.
 1. $X=\{A,B\}$ & $Y=\{C,D\}$
 2. $X=\{A,C\}$ & $Y=\{B,D\}$
 3. $X=\{A,D\}$ & $Y=\{B,C\}$
- For 100 vertices, 5×10^{28} possibilities
- Need 1.59×10^{13} years if one can try 100M possibilities per second.

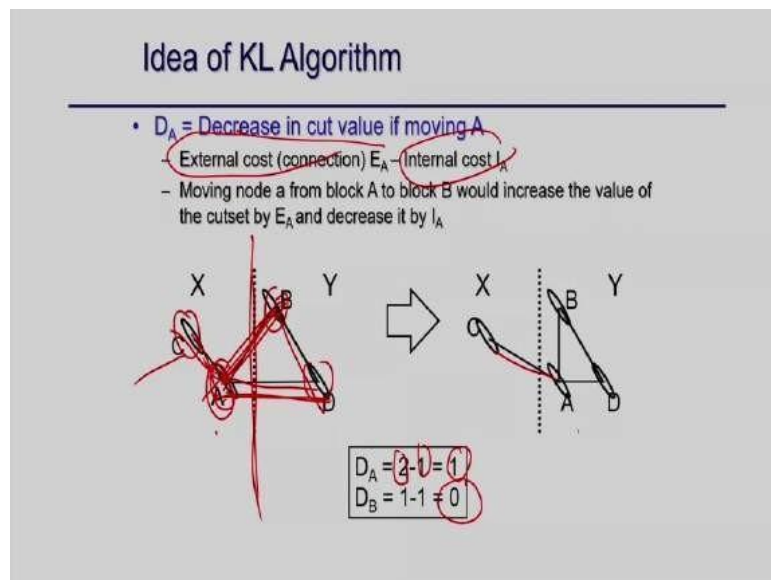
So, if you just think about how it should work you might try to all possible partitioning

right if we make a trivial approach, we just make all possible bisection of your circuit, and then you will find the best one right that will be a very trivial approach. Now, we think about there are $2n$ vertices. So, all possible combination is n to the power n right, which is huge n to the power n possible bisection is possible and then you just think about 4 vertices is fine, there are 3 possibilities.

But if there are if your node has 100 vertices only if the possibilities are 10 to the power 28 5 into 10 to the power 8 so, it is huge you can understand the element. So, even for only 100 vertices, it will take 10 to the 1 into 5.9 into 10 to the power 13 years. So, which is something infeasible right so, which is in infeasible right.

So, just to if you won't try to 100 million possibilities sorry. So, for 100 million possibilities you need this much time. So, these are just the same numbers to give you the value that you cannot just go for when you group for this approach right here. So, all possible finding all possible bisections and finding the optimal one would not work right.

(Refer to Slide Time: 26:30)



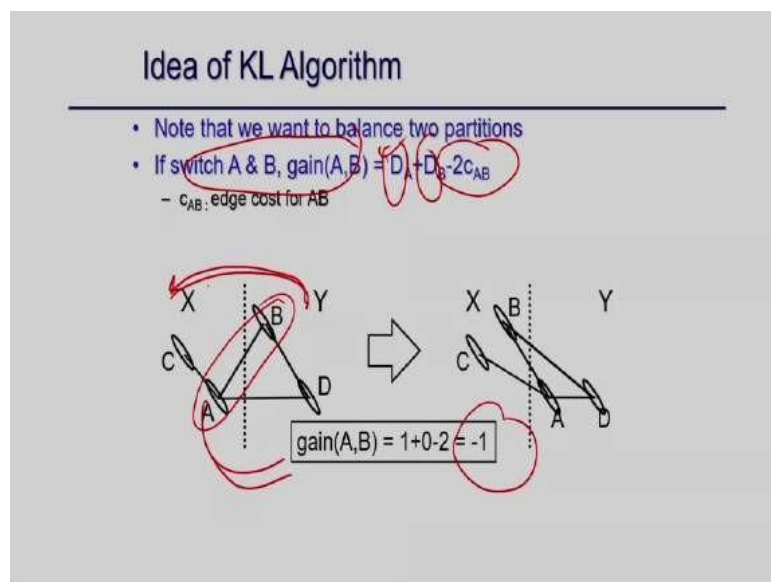
So, in this KL algorithm, the idea is that you ah the cost you have to understand that if you so, suppose there is an initial connection you try to move by a node from here to here what will happen. So, whatever the node from this node to the other side right so, that node will become they will not be the, they will not be in the cut right. So, I move A to here A. So, these 2 edges are coming here so, they are not part of the cut.

But whatever the connection from this node to this side those will be the part of the cut now right so, that is the cost. So, which is basically that the external cost that is this

So, if you just think about how it should work you might try to all possible partitioning whatever actually crossing the other part and minus then internal right. So, this is the gain if you move a node from this side to this side the gain will be this right.

So, for example, for A initially there are two connections to the other side so, 2 and there is only one connection to this same side minus 1. So, if we just move A to this side, you are cut side will be reduced by one your gain is a positive right. So, similarly, if you move this B from this side to this side you can (Refer Time: 27:30) Cs over B there is a connection here one and the connection. So, it does not have any gain so, if you move B from this side to this side effect, there is no reduction in the cut side.

(Refer Slide Time: 27:41)



Now, if you take 2 nodes. So, the gain of A is given by D_A , which is calculated by this equation this is the D_A there is cost and then this cost of D_B . So, that means we are moving A from this side B from this side cost of moving this plus this that will be your gain minus the interconnection between this two-node right there is the interconnection between this node.

Because they count it two times here right so, we have to just remove it. So, they will always remain the same. So, that is what is your actual switching between 2 nodes in the overall gain right. So, for example, if you just swap A and B overall gain is minus 1. So, which is negative, it is not a good choice to swap A and B; that means, moving A from the left side to the right side and moving B from the right side to the left side which is not a good move.

(Refer to Slide Time: 28:31)

Idea of KL Algorithm

- Start with any initial legal partitions X and Y.
- A pass (exchanging each vertex exactly once) is described below:
 1. For $i = 1$ to n do
 - From the unlocked (unexchanged) vertices, choose a pair (A,B) s.t. $\text{gain}(A,B)$ is largest.
 - Exchange A and B. Lock A and B.
 - Let $g_i = \text{gain}(A,B)$.
 2. Find the k s.t. $G = g_1 + \dots + g_k$ is maximized.
 3. Switch the first k pairs.
- Repeat the pass until there is no improvement ($G=0$).

Handwritten annotations on the slide include:

- A vertical line with a vertical arrow pointing down, labeled with $(n-1)$ at the bottom.
- A box containing $(1, 2, 3)$ and (a, b, c) .
- A box containing $(1, 2, 3)$ and (a, b, c) with arrows pointing to the elements.
- A box containing $(1, 2, 3)$ and (a, b, c) with arrows pointing to the elements.
- A box containing $(1, 2, 3)$ and (a, b, c) with arrows pointing to the elements.

So, the idea is something you start with an initial cut you take an arbitrary cut right of the circuit that may be high, then if there are n nodes here there are n nodes here and then you find out all partial partitioning. So, there are maybe $n \times n$ square appear right. So, for each, there is a node.

So, suppose there are 1, 2, and 3 here a, b, and c, there are 3 nodes. So, the partial partitioning is into this n into $n \times 3$ into 3×9 , right. So, we can take 1, a, 1, b, c. Similarly, 2, a, 2, b, c and 3, a, 3, b, c these are all possible possibilities of pair right.

Because our objective is now to swap two pairs. So, you find out all these $n \times n$ square possibilities and then you chase the best one, right. You take the best one which has the maximum gain. So, for each pair, you calculate the gain and you actually swap the, has the best gain right, and then you lock that two. So, that is again of that particular move.

So, then you just do this for n time so, you make n move. So, that at least some n nodes are from here it will move here and n nodes from here move there.

Some happening here and then maybe some of them have a negative gain right after some time your gain may be negative. So, you try to find out some k value. So, you make an n move, but you take the first k gain iteration which has a positive impact. So, you make those them fixed other we just other swapping that you done you just undo them right. So, that will not be done.

So, this is after this iteration you have another solution right and then you will keep doing this until I do not find any no improvement right so, this is one iteration. So, the overall algorithm is like I make an initially by partition then I do n n swapping right, based on the maximum gain to minimum gain and then I took only k gains, which k swapping which have a total summation is positive and then others, I will just discard

And after that I will get another solution right so, the cut side will be improved and then I am going to do the same process k times or sometime unless I do not find much improvement. So, this is just one idea of this scaling algorithm, which Kernighan- Lin algorithm does do bisection or bi-partitioning, whereby swapping this two-node this is just one approach there will be various approach and there is various kind of technique.

(Refer to Slide Time: 30:54)

Time Complexity of KL

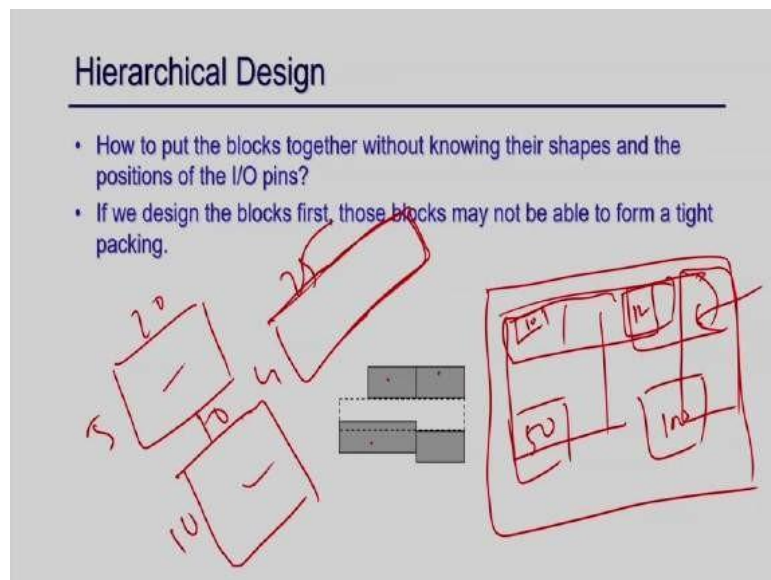
- For each pass,
 - $O(n^2)$ time to find the best pair to exchange.
 - n pairs exchanged.
 - Total time is $O(n^3)$ per pass.
- Better implementation can get $O(n^2 \log n)$ time per pass.
- Number of passes is usually small.

As I discuss here the complexity will be n^2 compared to the n to the power n . So, which is and for a better implementation we can have an $n^2 \log n$. So, this is also the one way of doing the partitioning bi-partitioning, and as I mentioned that there are various approaches., I mean I am not going to cover all of them.

(Refer Slide Time: 31:10)



(Refer Slide Time: 31:21)



Now, I will move on to floor planning. So, what you see after that so, after partitioning you have some components right, some partitioning component like this this this this. And now you have to think about planning your layout area right.

So, you have say 4 components now you have to physically place those components here right. So, your total area is minimum right. So, you have to place those components here and you understand their area is not always the same this may only say 10 units and this may be 100 units this may be 12 units this may be 50 units and so, on.

So, their area may be different right I am so; that means, their size is not exactly all square right. So, this may be a small thus 10 means this is small this is a small one and this is 100 means this is a big one so, their size ratio is also different.

And most importantly their aspect ratio is not fixed here. So, I just give you say 100; 100 you can construct obtain into 10 right this may say 5 into 20 or maybe exactly square where is 10 10 or maybe 4 into 95 it can have a different aspect ratio right. Because I just told you [FL] I have a block which has only 100 units. So, now, you have to fix this aspect ratio. So, for 100 should I choose 10 into 10 1 is better or 5 into 10 is better or 4 into 25 is better other options are also there right.


So, you have always had to decide the aspect ratio of each component based on their total area. So, that you can compact those all the component in a minimum possible area right because if you just think I mean if you take this kind of, which is rectangular one that will cover this portion of the said area and then you can probably you are not able to fix another component here.

Instead of if you take this say 10 crosses (Refer Time: 33:08) if you place this way probably another component can be placed here right if you do not place anything this portion will remain vacant right. So, you try to and that is called dead space. You try to reduce those dead space. So, in partition this floor planning what we do we try to finalize the aspect ratio of each component, and then it and our objective is to try to pack everything in a minimum possible area. So, your total area is minimum and total dead space is also minimum right.

Floorplanning

The floorplanning problem is to plan the positions and shapes of the modules at the beginning of the design cycle to optimize the circuit performance:

- chip area ✓
- total wirelength ✓
- delay of critical path ✓
- routability ✓
- others, e.g., noise, heat dissipation, etc. ✓



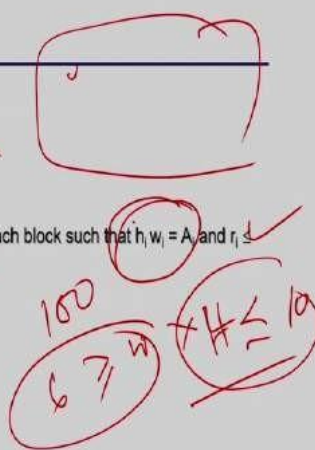
So, if we just move on so, floor planning is the problem of positions finding the planning the position, not exact plan fixing the block that is important because here I am just planning the positions right and then shape. The shape is finalized at the beginning of the design right, where your objective would be to optimize the area total wire length also because you try to because I know that now what is the connection between two-component base.

Because that is given by the cut size. So, if you have a say in this block here and there are a lot of connections to this block you should place this block as close as possible. So, this wire length will be less right. So, you try to place very connectedly two-component, which are very close by.

So, your total length is optimized or the delay of the critical path is minimized route ability is possible that physically all the connections can be made in the physical dime. So, that may be also your objective or another objective like you reduce the noise or the heat dissipation or crosstalk between the router wires. So, those can be also your objectives right. So, that is what is floor planning.

Floorplanning Problem

- Input:
 - n Blocks with areas A_1, \dots, A_n
 - Bounds r_i and s_i on the aspect ratio of block B_i
- Output:
 - Coordinates (x_i, y_i) , width w_i and height h_i for each block such that $h_i w_i = A_i$ and $r_i \leq h_i/w_i \leq s_i$
- Objective:
 - to optimize the circuit performance.



So, if we just define the problem so, you have n blocks of size area I know the only area and I also have some aspect ratio bound that if you have to say 100 10 is your length maybe minimum says 6 right. So, you have to find the W and L right width and height of W and H .

So, your width should be greater than 6 or say height less than 10 right, if you have given that kind of ratio constant on the aspect ratio you cannot just take a very long or very short, I mean short height or short width kind of things instead of some minimum constant on the ratio right this aspect ratio.

And then what is the output you try to find out some location of this particular block such that the two and you also find the width and height of the block is block such that total area is minimum. And also, all this height and width satisfy the aspect ratio constant given to you and your objective is something to optimize the circuit performance the performance may be one of them right.

So, this is what is the problem you have given A components, where the area is given to you and also the aspect ratio of height and width is not fixed. But some restrictions might be there your height and width should not go beyond these or should not be at least this kind of thing and you try to find out the try to place all the blocks in a physical area.

So, with and which coordinate of each block and you also find out the aspect ratio that mean width and height of each block we try to finalize. So, that total area fit into this area target and you have some optimization goal like optimizing the total area or optimizing the route ability interconnection cost and other others right. So, this is what is the floor planning problem.

(Refer Slide Time: 36:34)



So, as I mentioned this, I mean bound on aspect ratio is something that we can always pack everything tightly right because the problem I just discard about suppose this is your component blocks are like this right. So, suppose there is one block like this and one block like this ok, there are three such blocks ok. So, now, if you try to suppose this is your chip area. So, now if you just place this one says this will take this sorry if this will take tell you this is you try to place this here why it will happen here.

Now, suppose this will take in such a way that now this is cannot fit in here and this may not fit in here also, if you just place one of them here say suppose this I so, this is saying this block is something like this so, their length is more than this. So, they cannot be placed this way. So, suppose this is B, I want to place this way right.

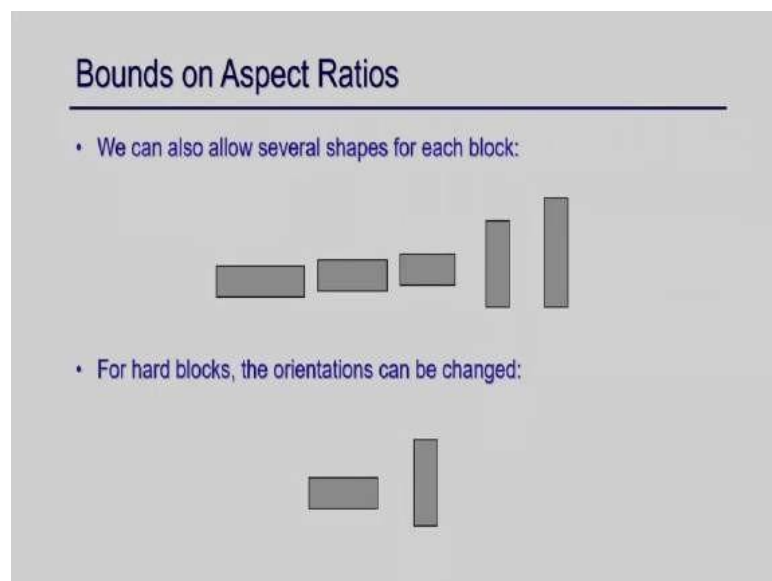
So, now, what happens? Now there is no way to place C because this is all fixed that there is no place then basically, you have to just increase the area of your design chip area just to place this C here right C here.

And many spaces are even vacant right if you just say instead of taking this if you just take another version of A, which is a little bit of this shape then probably what we can do we can probably feed the whole thing into the old areas. So, this will be your A this will be your B and this will be your C right.

So, A B C, this is what is the problem right. So, you might have a different aspect ratio for, which you can place them in a smaller area, but if you just estimate a different aspect ratio then you might need more area right, this perfectly resembles highlighting the problem.

So, one thing you always think [FL], I just make the width is 1 and the length is like this right so, then the problem is that your chip will be just a series of block right which is this length and which is not feasible right. This is not, this is not something, your because your layout cannot be a just a long strip which is not is allowed. So, that is why we have some bound on the aspect ratio ok.

(Refer Slide Time: 38:51)



So, as I mentioned that a given the same size can be like this right for a same-size your you can generate a different kind of block right and also an orientation maybe this way or that way. So, you have to consider all possible such orientations given on that aspect ratio of constraint and then you try to place all those blocks in the area. So, your total area of primary objective is to reduce the size of the area. So, you can place all of them in the smaller possible areas right. This is what floor planning is.

(Refer to Slide Time: 39:21)

Objective Function

A commonly used objective function is a weighted sum of area and wirelength:

$$\text{cost} = \alpha A + \beta L$$

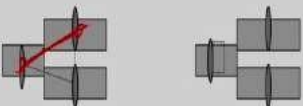
where A is the total area of the packing, L is the total wirelength, and α and β are constants.

And also, we so, in general, the objective function is the minimization area plus objective length A is the area and B is the L is the length and alpha-beta are some constant factors right. This is the overall objective of floor planning ok.

(Refer to Slide Time: 39:40)

Wirelength Estimation

- Exact wirelength of each net is not known until routing is done.
- In floorplanning, even pin positions are not known yet.
- Some possible wirelength estimations:
 - Center-to-center estimation
 - Half-perimeter estimation



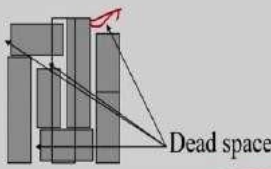
And this wire length estimation is something because now, I do not have the exact placement. So, how we can how can I estimate the wire length is just the distance between this these two. So, suppose there is a wire from this to this I can find

just the distance from the center points right. So, this might be your wire length because this is just kind of an estimation ok.

(Refer Slide Time: 40:01)

Dead space

- Dead space is the space that is wasted:

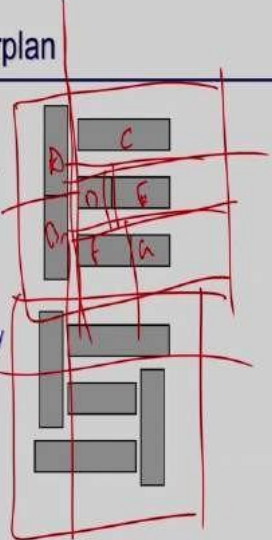


- Minimizing area is the same as minimizing deadspace.
- Dead space percentage is computed as $(A - \sum A_i) / A \times 100\%$

(Refer Slide Time: 40:17)

Slicing and Non-Slicing Floorplan

- Slicing Floorplan:
One that can be obtained by repetitively subdividing (slicing) rectangles horizontally or vertically.
- Non-Slicing Floorplan:
One that may not be obtained by repetitively subdividing alone.
- slicing floorplans are much easier to handle.



So, as I also mentioned that because of your different aspect rates you will create a lot of dead space here and there. So, your objective is trying to minimize the dead space as well right. So, this is also an objective during floor planning. So, and also there is another concept here is that sliceable and non-sliceable floor plan what is that, I mean

sliceable floor plan is something. So, you finalize your aspect ratio and you finalize your block height and width right and then you place them.

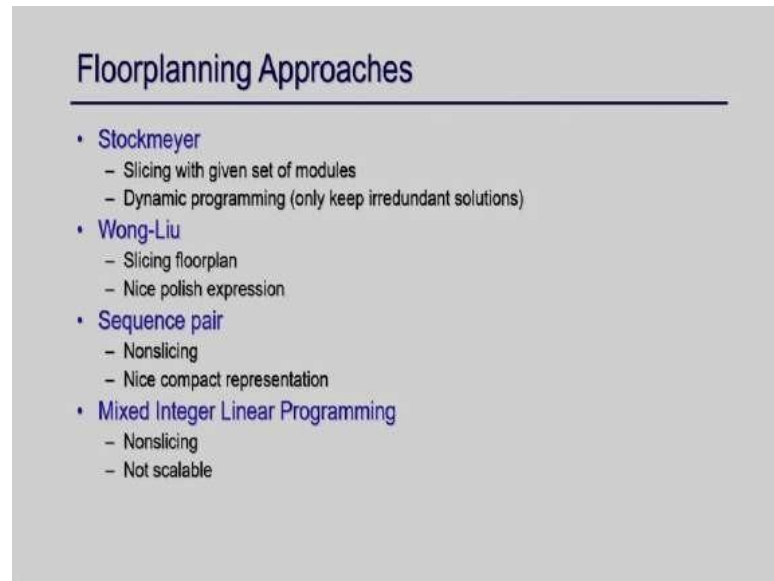
So, after placing one is sliceable means what you can obtain your floor plan can be obtained by repetitively just slicing the things horizontally or vertically suppose, this is how you are placing this entire slice this is A and this is B this is say C this is D this is E and this is F and this is G right. So, this is your placement.

So, then what is slicing? So, I make a slice here then I make a slice here then I make a slice here and then I make a slice here. So, I just bi-partition each component totally, and then this is sliceable. On the other hand, if we just look into this one. So, this is not a sliceable one, because you can see there, I cannot make this kind of partition if I try to make this will partition here. So, this is my component now this is also not a sliceable.

So, I cannot make this kind of bi-partition here right. So, here I just make a bi-partition like this, but here I cannot make such a thing. So, this is non-sliceable. So, it is generally good practice to make a sliceable floor plan. So, such way you make the connection so, that you can make a sliceable floor plan and we will see that is helpful in that we can we can see that this sliceable floor plan is easy to handle for place plan routing and all other component.

So, usually, the kind of floor plan we try to generate is sliceable right. So, what is sliceable I just repeat is something we can bi-partition each sub-component at a time to generate your layout plan right. So, this is how you can do say any order right horizontally or vertically ok. So, these are some timelines that we should be aware of during floor planning.

(Refer Slide Time: 42:13)



Floorplanning Approaches

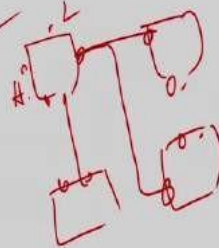
- Stockmeyer
 - Slicing with given set of modules
 - Dynamic programming (only keep irredundant solutions)
- Wong-Liu
 - Slicing floorplan
 - Nice polish expression
- Sequence pair
 - Nonslicing
 - Nice compact representation
- Mixed Integer Linear Programming
 - Nonslicing
 - Not scalable

So, again this floor planning you understand, there are a lot of such approaches various approaches are there, and when because this is an NP-complete problem and a mean optimal solution cannot be obtained by some optimal algorithms.

So, we have various kinds of heuristic approaches is also available for this step as well ok. So, I will move on to the next step, which is placement. So, once you have so, basically what is the placement problem now actually physically going to place those blocks in your layout area.

Problem formulation

- Input:
 - Blocks (standard cells and macros) B_1, \dots, B_n
 - ~~Shapes and Pin Positions~~ for each block B_i
 - Nets N_1, \dots, N_m
- Output:
 - Coordinates (x_i, y_i) for block B_i
 - No overlaps between blocks
 - The total wire length is minimized
 - The area of the resulting block is minimized or given a fixed die
- Other consideration: timing, routability, clock, buffering and interaction with physical synthesis



The diagram shows a schematic of a circuit layout. It features several rectangular blocks of varying sizes and orientations. These blocks are interconnected by a network of lines representing nets. Some blocks have small circles on their edges, representing pins. The layout is drawn in red ink on a light background.

So, the problem inputs of this block are that there are say N blocks and I know because now the shape is fixed by the floor planning pin positions also fixed by the floor planning because the pins exactly where the pin will be there that is also fixed by then.

And the nets are also given because you have now let a block. So, each block where exactly the pin pins are there that is finalized and for each block is given and exactly the connections among them right from pin to pin the connections are also given to you right. So, this all is given to you now.

So, these are all the inputs of all blocks their pin configuration their exact shape height, and width are also given for each block because now the fixed size is already fixed by the floor planning and all the nets exact net, where exactly the connections are there given right.

So, now you try to find out the exact position of this each block coordinate of each block. So, that no block overlaps total wire length is minimized and the area of the resulting block is also minimized for a given dime right so, this is something is the placement problem. And you can see you can have other objectives like this conditionalized, timing, rout ability clock similarly like the floor planning also this can be your optimization criteria ok.

(Refer Slide Time: 44:07)

Floorplanning v.s. Placement

- Both determines block positions to optimize the circuit performance.
- Floorplanning:
 - Details like shapes of blocks, I/O pin positions, etc. are not yet fixed (blocks with flexible shape are called soft blocks).
- Placement:
 - Details like module shapes and I/O pin positions are fixed (blocks with no flexibility in shape are called hard blocks).

So, you can see that floor planning and placement are very closer because in-floor panning also we try to find the position of each block, and here also try to find out so what is the difference right. So, the basic difference is that in during floor planning the shape is not given only the number of the block given IO pin positions is also not given right. So, you are kind of the blocks are soft blocks.

Whereas in the placement I know there are some positions given by the floor pan, but that is not the fixed position right. So, the I but their shape is fixed IO positions are fixed right. So, this is kind of a hard block. So, in the soft block, I have other options like I can take a different aspect ratio you do not try to fit in, but in placement time all the blocks are fixed, were height and width.

So, I can it is a kind of hard block I cannot make any change in the aspect ratio. So, that is the difference bit. So, both are trying to find the position of the circuit. So, it may be that whatever the floor planning given generated the position of the block that may be the final one, but placement you try to see if there can be some other movement that can be done without changing the aspect ratio.

And finally, it may accept the floor planning positions or it may try to find out the better placement of these blocks right. So, floor planning as also I mentioned these are also n block and you try to find out the coordinate. So, you can see the input is almost the same

for both right the set of block output is coordinates, but the only difference is that I O pins aspect ratio those things are not mentioned here.

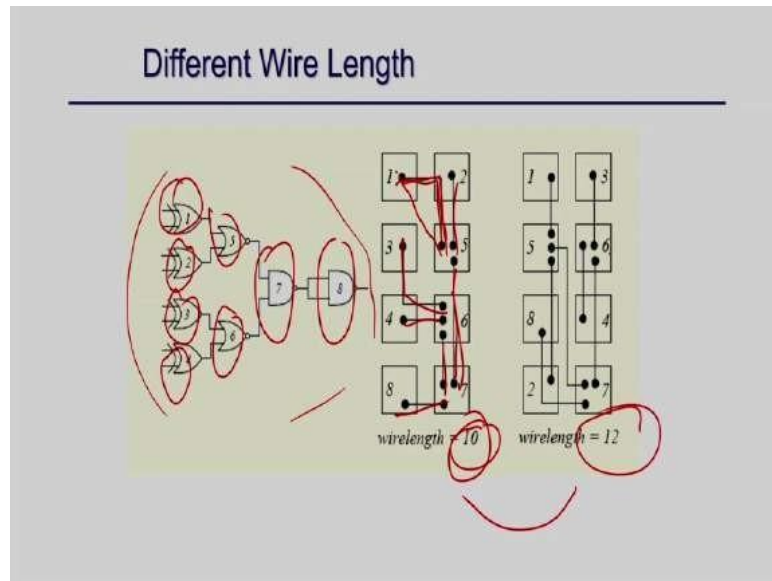
(Refer to Slide Time: 45:40)

Floorplanning Problem

- **Input:**
 - n Blocks with areas A_1, \dots, A_n
 - Bounds r_i and s_i on the aspect ratio of block B_i
- **Output:**
 - Coordinates (x_i, y_i) , width w_i and height h_i for each block such that $h_i w_i = A_i$ and $r_i \leq h_i/w_i \leq s_i$
- **Objective:**
 - To optimize the circuit performance.

And whatever the floor plan positions? Find out for each block that may not be the final one. Placement tries to re-loop to that position and try to optimize the area and there may be some little bit of movement of the blocks can be done so, that your total wire length is minimum or you can x you can actually. So, remove your total area, but after placement, all the position of each block is finalized then you cannot have any other change then the only thing to be done here is the actual making of the physical interconnections right.

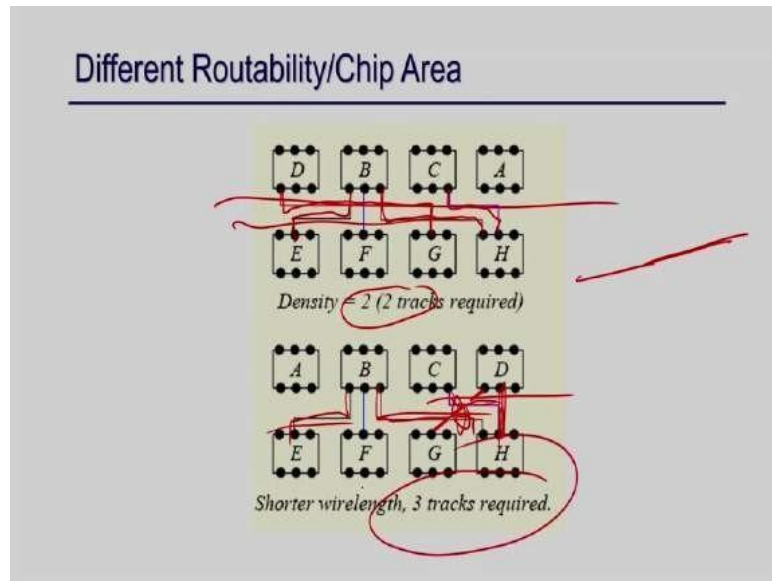
So, after floor panning there is a possibility just to exchange some block position or moving to the other places is possible without the die area intending to reduce the number of interconnections total area, but after placement, you cannot have any option to move your, positions right this is what is after placement.



So, you just think about I give an example here. Suppose, this is your circuit, and suppose, each component is a block ok just for a simplicity. And say floor planning decides this particular size and the total wire length is. So, if we just place them here like this way so, the total wire length is like this right. So, if you make this connection here. So, if you have to make a connection from 1 to 5, you cannot make a connection like this right because it has to go either vertically and then horizontally.

So, if you just think about the length of this is 2 1 and 2, if you count here 1 2 3 4 4 and 5 6 7 8 9 10 so, the total wire length is 10. Now, I just have another placement because all of the same size here. So, I have all the sizes I make a connection like this 1 3 here instead of we are just swapping 2 and 3 and this 5 and 6 5 and 6. So, this is 6 here and 5 here. So, I make some interconnection changes here and there, and now the total length these 12.

So, we can see that even of the same block size and same area, I can make some swapping some exchanges here and there. So, your total area remains the same, but the total wire length may reduce right from this to this is what the placement tries to do right it the same size block can be swapped or can be moved to some other places, where some dead spaces are there. So, those kinds of things can be during placement.

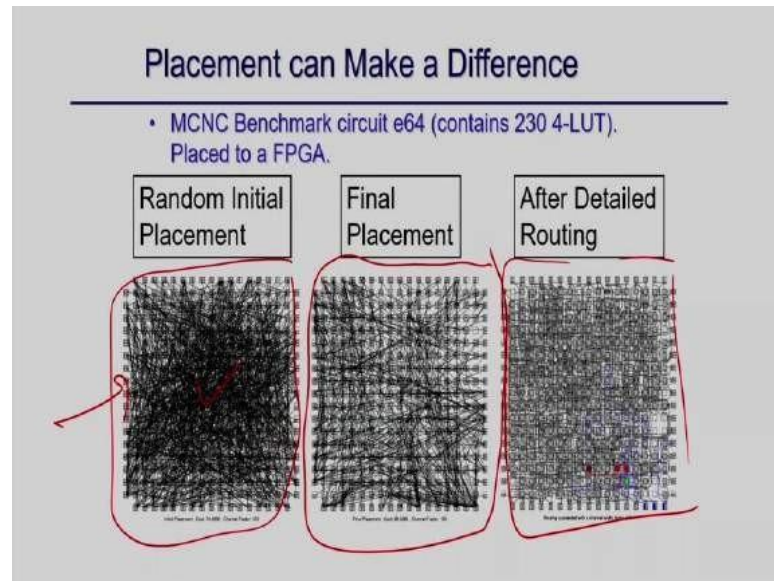


And also, sometimes if you just try to make the connection you try to see that you cannot have the two interconnections two connections can cross in the same line. So, if you just plus this block like B C C A and A FG H, then you to make the all the connection you need only 2 tracks right you can make a connection like this. And you have a connection here, they are not overlapping. So, they can be put in the same track and this connection and this connection can be put in another track. So, you need 2 tracks right.

Whereas, if you have some different placement A B C D E F G H, now this connection and this connection cannot be placed in the same because, there is an overlapping portion. So, they have to be in 2 tracks, similarly, this has to be made here. So, I need 1 track 2 tracks, and then just to B B to E. Now, I cannot make B to B to E is there B to F is there D to H D to H ok D to H, I cannot make the connection now I need another kind of layer right.

So, this is something I need 3 tracks just to give an example here is something that even after floor planning there are a lot of opportunities to just do some reorganization of your block so, that you can this is the total wire length as well as the total number of track requirement right. So, this is what is the objective of the placement step.

(Refer Slide Time: 49:19)



So, you can see that there is some random initial placement, what is generated by floor plan effectively the placement looks like this right this is just the snapshot of their layout, but you can make some modification you see the number of wire length become very less here right so, you have the this is very parse compared to this dense placement right, and this is the after final routing.

So, you can see that you can actually make a lot of interconnections less you just moving your blocks right during placement. So, the placement has a huge impact on generating the final route interconnection cost right.

(Refer Slide Time: 49:57)

Importance of Placement

- Placement is a fundamental problem for physical design
- Glue of the physical synthesis
- Becomes very active again in recent years:
 - Many new academic placers for WL min since 2000
 - Many other publications to handle timing, routability, etc.
- Reasons:
 - Serious interconnect issues (delay, routability, noise) in deep-submicron design
 - Placement determines interconnect to the first order
 - Need placement information even in early design stages (e.g., logic synthesis)
 - Placement problem becomes significantly larger
 - Cong et al. [ASPDAC-03, ISPD-03, ICCAD-03] point out that existing placers are far from optimal, not scalable, and not stable

So, placement as I mention it is having even reason so, basically as I mentioned that route interconnection cost is going more day by day right. So, the placement becomes important just to reduce the total interconnection cost right.

(Refer Slide Time: 50:16)

Requirements for Placers

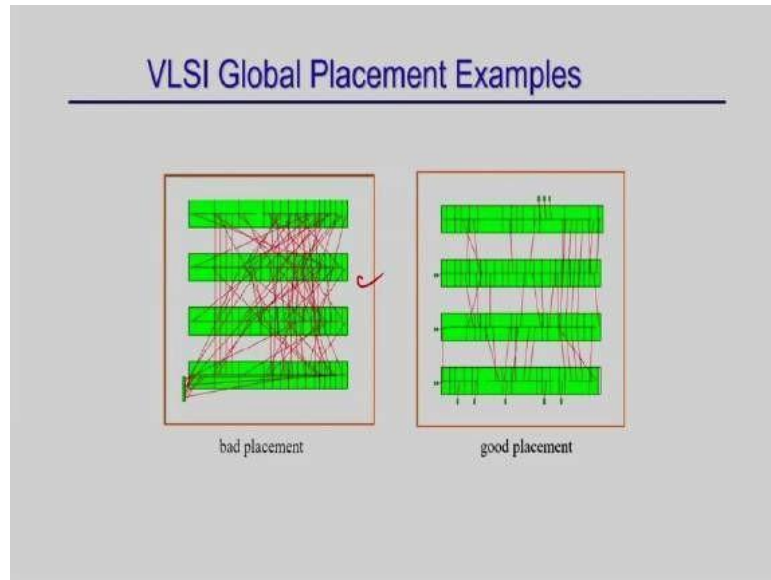
- Must handle 4-10M cells, 1000s macros
 - 64 bits + near-linear asymptotic complexity
 - Scalable/compact design database (OpenAccess)
- Accept fixed ports/pads/pins + fixed cells
- Place macros, esp. with var. aspect ratios
 - Non-trivial heights and widths (e.g., height=2rows)
- Honor targets and limits for net length
- Respect floorplan constraints
- Handle a wide range of placement densities (from <25% to 100% occupied), ICCAD '02

So, this is something that is important. And the problem with the pressure is that now with the number of 100 million cells right we have maybe 10 million cycles and we have to also place this macro, like this d s ps and all those. And you also have you have to place the ports pads everything you have to find you have to also find out the limit and the

honor the target and limit of each length net length respect the floor plan constraint and

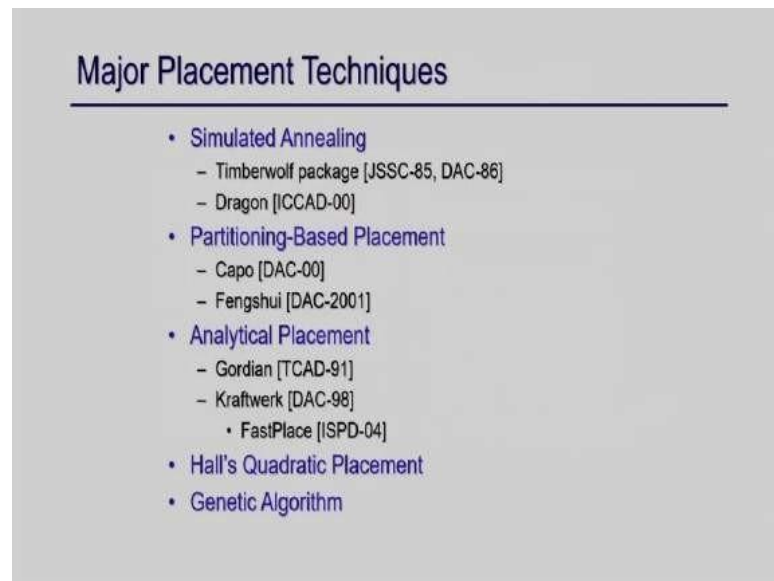
all those things. So, these have to that is placement has to satisfy all these kinds of constraints ok

(Refer Slide Time: 50:51)



So, here is also another example that bad placement. So, these are all components are placed and this is the interconnection you just see there are a lot of crossings right and here if we just reorganize them, you can have a good placement the number of wire length is less so, the placement does this. So, make a good placement. So, your total wire length is reduced.

(Refer Slide Time: 51:10)

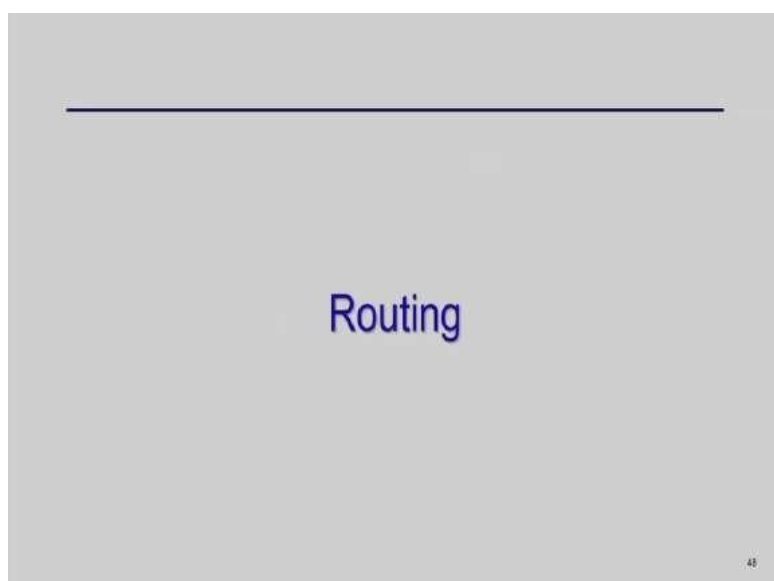


Major Placement Techniques

- Simulated Annealing
 - Timberwolf package [JSSC-85, DAC-86]
 - Dragon [ICCAD-00]
- Partitioning-Based Placement
 - Capo [DAC-00]
 - Fengshui [DAC-2001]
- Analytical Placement
 - Gordian [TCAD-91]
 - Kraftwerk [DAC-98]
 - FastPlace [ISPD-04]
- Hall's Quadratic Placement
- Genetic Algorithm

So, again in placement also since it is also another NP-complete problem you have various approaches like simulated annealing partition-based approach analytical approach and so, on. And again, I am not going to detail or discussion all of them, but the basic objective is understood here just to reorganize your placement I mean repositioning our block so, that you can reduce your total area and primarily total wire length ok.

(Refer Slide Time: 51:37)

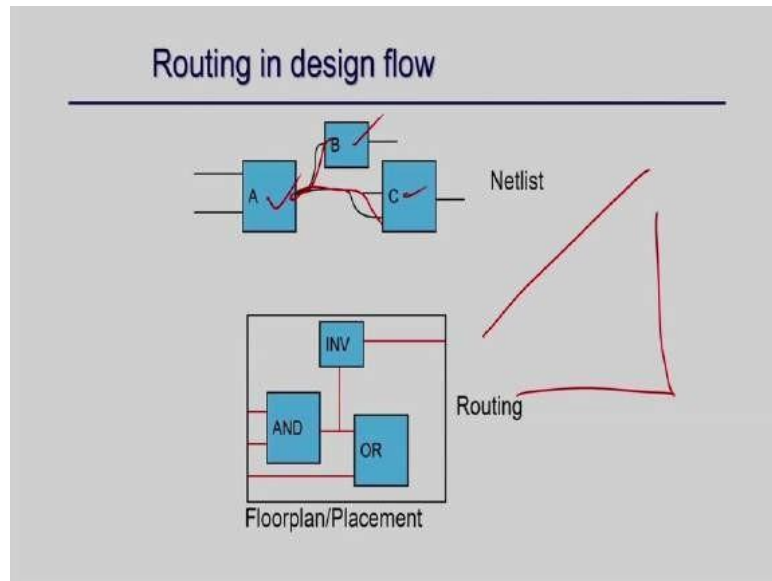


Routing

46

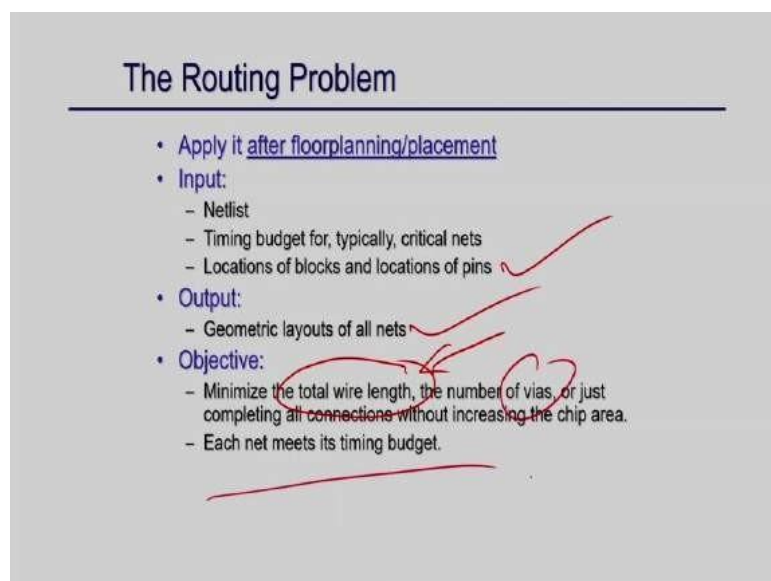
So, I will move onto routing now. So, I know the positions of each block are fixed they are final I cannot change their position now, but I have to make the physical connection.

(Refer Slide Time: 51:45)



So, I know these blocks are placed here this block is placed here and I have to make these connections now right this is what is routing. And you can see here, I make this connection like this. So, to remember that finally, this connection cannot go like this right it has to go vertically or horizontally you have track vertical and horizontal track you have to make the connection through this right.

(Refer Slide Time: 52:07)



So, the routing problem is that is after floor planning or placement. So, you have a net list; that means, its component positions and the timing budget that is this for each net it

should meet this much of timing and the location of block a location of pins and output is

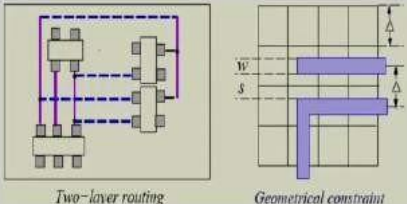
that actual geometrically layout of the all nets ok.

And your objective is to actually this is how when where the actual interconnection happening. So, your actual objective is to minimize the total wire length right and also the number of vias, I will just discuss now and you are also each net should meet the timing budget so, that is what is the routing problem.

(Refer Slide Time: 52:45)

The Routing Constraints

- **Examples:**
 - Placement constraint
 - Number of routing layers
 - Delay constraint
 - Meet all geometrical constraints (design rules)
 - Physical/Electrical/Manufacturing constraints:
 - Crosstalk
 - Process variations, yield, or lithography issues?

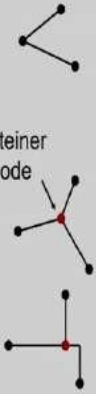


Two-layer routing *Geometrical constraint*

(Refer Slide Time: 52:47)

Steiner Tree

- For a multi-terminal net, we can construct a spanning tree to connect all the terminals together.
- But the wire length will be large.
- Better use **Steiner Tree**:
A tree connecting all terminals and some additional nodes (Steiner nodes).
- **Rectilinear Steiner Tree**:
Steiner tree in which all the edges run horizontally and vertically.



Steiner Node

(Refer Slide Time: 52:48)
PAGE 52:48)

Routing Problem is Very Hard

- Minimum Steiner Tree Problem:
 - Given a net, find the Steiner tree with the minimum length.
 - This problem is NP-Complete!
- May need to route tens of thousands of nets simultaneously without overlapping.
- Obstacles may exist in the routing region.

(Refer Slide Time: 52:53)

Kinds of Routing

- Global Routing ✓
- Detailed Routing ✓
 - Channel
 - Switchbox
- Others:
 - Maze routing
 - Over the cell routing
 - Clock routing

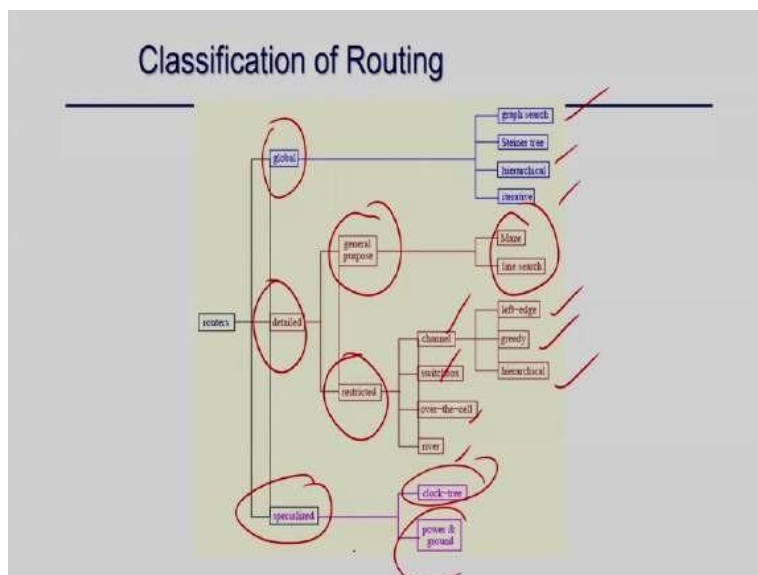
So, routing this routing problem has to be handled here. So, the routing has two steps first is global routing, the second is detailed routing and the other kind of techniques are there like clock routing, call routing, and maze routing general cover here.

Approaches for Routing

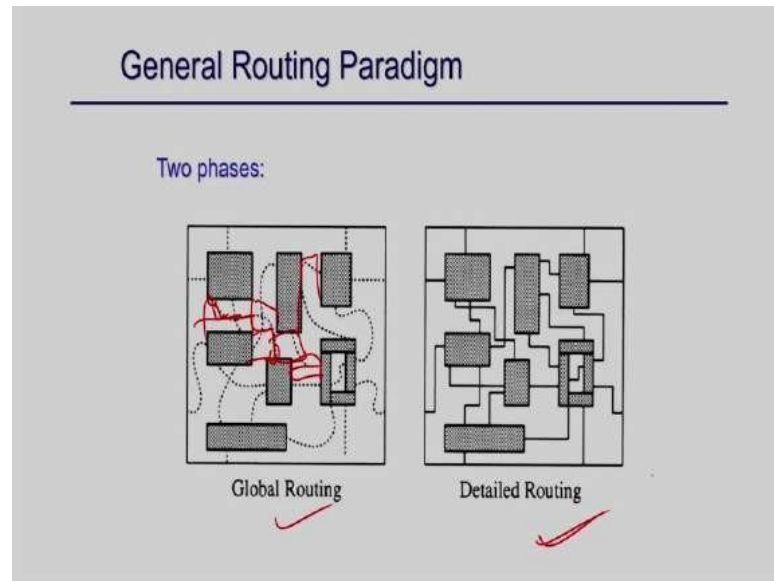
- **Sequential Approach:**
 - Route nets one at a time.
 - Order depends on factors like criticality, estimated wire length, and number of terminals.
 - When further routing of nets is not possible because some nets are blocked by nets routed earlier, apply 'Rip-up and Reroute' technique (or 'Shove-aside' technique).
- **Concurrent Approach:**
 - Consider all nets simultaneously, i.e., no ordering.
 - Can be formulated as integer programming.

(Refer Slide Time: 53:07)

Classification of Routing



(Refer Slide Time: 55:09)



So, primarily so, initially what is happening here. So, here are these things is there the routing has two-phase first is global routing, the second is detailed routing so; that means, what I just decide [FL] from this connection to this connection this will go through this part of the circuit and this part of the circuit and this part of the circuit. This is what I decide, not exactly the track through exact which track will go which is called global routing.

So, in global routing, I just decide through which part of the circuit or which are the channels through which it will go that I decide, for example, just the, for example, say for this I just decide it will go through this track plus this track something like this. So, that is what I, but I do not finalize the exact track number here and what is done in the detailed routing. So, it has two phases global routing and detailed routing.

(Refer Slide Time: 55:56)

Approaches for Routing

- **Sequential Approach:**
 - Route nets one at a time.
 - Order depends on factors like criticality, estimated wire length, and number of terminals.
 - When further routing of nets is not possible because some nets are blocked by nets routed earlier, apply 'Rip-up and Reroute' technique (or 'Shove-aside' technique).
- **Concurrent Approach:**
 - Consider all nets simultaneously, i.e., no ordering.
 - Can be formulated as integer programming.

So, again this routing can have two approaches in the sequential approach you take one net and you try to route that one at a time, and maybe after some time you end up in a situation, where the router routing is not possible maybe one of the channels is over congested, you cannot put another signal through that.

So, you can actually when ripping up and reroute; that means, ignore some of the things you draw move back, and again restart placing to somewhere else. So, this is the sequential approach or in concurrent approach also you can take all the nets simultaneously, I will try to do them at the same time ok.

In the routing, as I mentioned the primary is global routing and then detailed routing global routing has a different kind of approach, in detailed routing also general-purpose in the smaller case we usually have a major routing or line search, in restricted one I can have a channel routing switch box over the cell and river and so on.

And for channel routing we usually apply the Left edge algorithm, Greedy algorithm, or Hierarchical algorithm, I am not going to detail them, and the specialized ones like routing the clock or the power of your circuit right.

(Refer Slide Time: 55:05)

Global Routing

Global routing is divided into 3 phases:

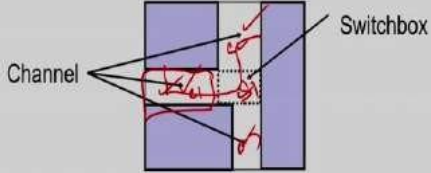
1. Region definition
2. Region assignment
3. Pin assignment to routing regions

So, if you just go into the, I will just define the global routing problem now basically the global routing has three-phase that the region definition, region assignment, and pin assignment are ok.

(Refer Slide Time: 55:16)

Region Definition

Divide the routing area into routing regions of simple shape (rectangular):



- Channel: Pins on 2 opposite sides.
- 2-D Switchbox: Pins on 4 sides.
- 3-D Switchbox: Pins on all 6 sides.

So, what is the region definition? So, when you have a channel between so, that is the part of the circuit, which is basically between two blocks, which is called a channel. So, this is also a channel and you have this is also a channel and this is a switch box,

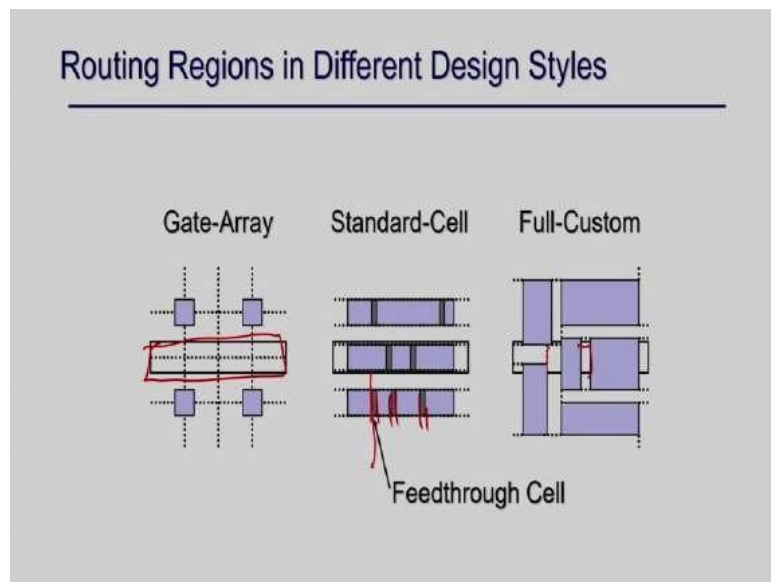
because this has multiple (Refer Time: 55:33) the data can come from this side this side

this site as well as this side.

So, this has more than 4 inputs. So, it has only data that can come only from this port will have only 2 sides that are the channel and that is the switch box. So, based on the routing area you decide the channel number you have to define the region that I have in this graph, I have 3 channels I have 3 channels and 1 right because after this detailed routing I have to set this particular connection go through this is say channel 1 this is channel 2 this is channel 3 and this is a switch box 1.

I have to say that I have this connection that goes through this to this so, that means it covered to C 1 switch box and C 2, then whenever I am going to detail routing of the channel C 1, I have to take care of this node right this particular connection. So, this is what is something. So, I just assign the first I have to design how many regions are there how many channels are there how many switch boxes are there that I am going to decide first, which is region definition.

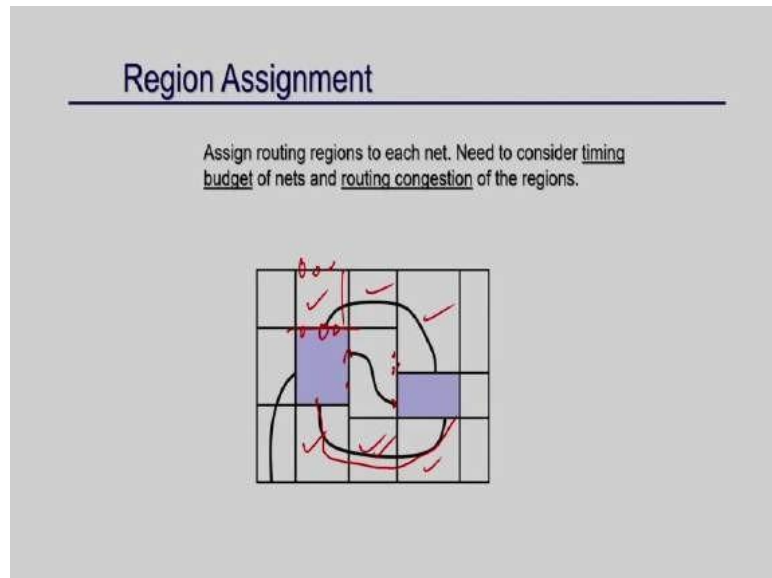
(Refer Slide Time: 56:32)



And then this is the and so far, this is just to say is that for gate array that is FPGA this region is something like this right. So, this is what is the regions. Because there are the, connect interconnection area is a prefixed standard cell, I have to make them, I have to make a connection from this to this feedthrough. So, these are the channels right. And for

full custom where you have all the option you can have a switch box and channels and all the possibilities will come, which is the A SIC design ok.

(Refer to Slide Time: 57:01)



So, after that, as I mentioned the first in the region you define how many channels are there and how many switch boxes are there. And then you make the actual connection and you assign this region assignment this particular net goes through this switch box, this switch box or this switch box at these regions right. This goes through this channel, this channel or, this switch box something like this.

So, you have to assign the so, your channel and region or switch box to each net which is called region assignment.

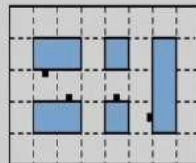
(Refer Slide Time: 57:27)

Graph Modeling of Routing Regions

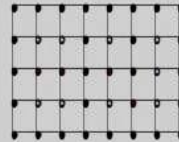
- Grid Graph Modeling
- Checker Board Graph Modeling
- Channel Intersection Graph Modeling

(Refer to Slide Time: 57:30)

Grid Graph



▪ A terminal

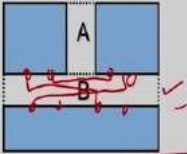


• A node with terminals

(Refer Slide Time: 57:30)

After Global Routing: Detailed Routing

The routing regions are divided into channels and switchboxes.



So only need to consider the channel routing problem and the switchbox routing problem.

And then this let us forget about and then the final after this is the pin assignment. Pin assignment is something you this then final as I mentioned here after that you make the pin assignment, because what is happening here once you finalize one particular channel you know how many pins are there right.

Similarly, for this you know how many inputs are coming and how many pins are at this side so, that is what is called pin assignment. So, after this, we decide the pin of this pin assignment for each channel or the switch box. And once this is done, I am going to go for this detailed routing. So, you can see that suppose this is one of my channels I know for this particular channel four-pin here right.

So, these are the pin positions these are the pin position and I know that I have to make a connection from this to this pin or maybe this pin to this pin and so, on right. So, I know a pin configuration and all those, then I have to make the actual connection in this channel at a time. So, I can reduce the number of tracks here right. So, that is what is called the channel routing problem. And whenever your switch box has a connection from all four sides that is your switch box problem ok.

(Refer Slide Time: 58:44)

Channel Ordering

The width of A is not known until A is routed, we must route A first.

What should be the routing order for this example?

So, as I mentioned then again one important concept that is channel routing so channel ordering. So, suppose so, I have to decide the width of this. So, now if I do not know the width of A I cannot decide the total width of this B right so, I have to. So, I have to decide A first and then B right so, but in some scenarios, there may be a circular dependency if you try to do B then you have to do C if you have to do the C A, and something like this.

(Refer to Slide Time: 59:10)

Channel Ordering

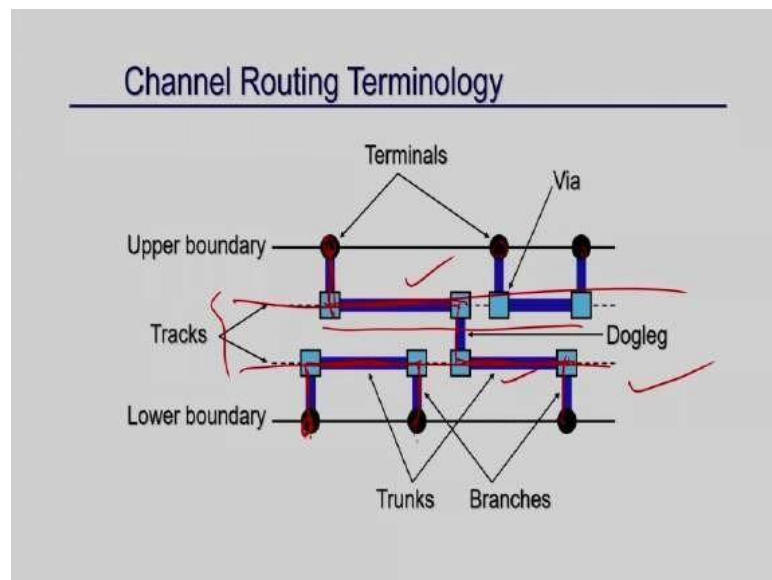
No feasible channel order!

Need to use switchbox

1. Fix the terminals between A & B
2. Route B, C, then D (channel)
3. Route A (switchbox)

So, then it will be a circular dependency and there may not be a. So, you cannot have a specific order rights. So, then in that sometimes you need a switch box. So, why do you need a switch box? You can define some portion of the circuit, which is the switch box and the arch. So, you can reduce the dependency and we can make a channel order ok.

(Refer Slide Time: 59:34)

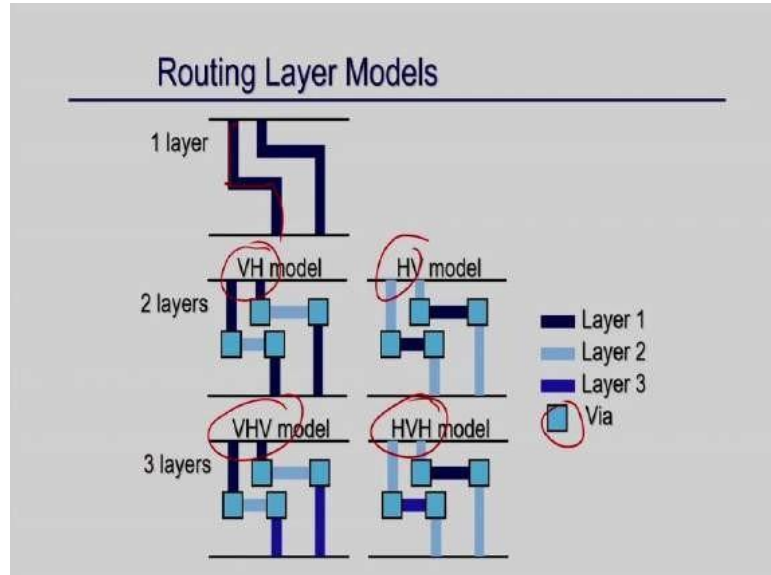


So, once this channel so, there is some other terminology right. So, now, I know the terminals on each side and I have to make the connection. So, you can make this connection so, this is tracks. So, your objective is to reduce the number of tracks. And so, suppose, for example, you can make the connection only through one track or maybe for one connection you may have to you needed two tracks this is one track. So, that is what is called dogleg that this because you cannot go through this, because there may be overlapping here in the track, which is not allowed.

So, you probably have to break this whole interconnection into two parts, this is one part and this is one part through, which you make this dogleg connection you break the whole connection into two tracks ok. So, these are the kind of terminology you should aware of that in the channel the number of pin configurations is given, and what are connections, I have to make are also given and you try to put all these connections in the minimum number of tracks ok.

And this is something and sometimes you make the connection the whole connection in one track or in order to make the split it and, which is called dogleg ok.

(Refer Slide Time: 60:34)



And your layer maybe there may be 2 layers or maybe you have to make all the connections into 1 layer, what you may have the horizontal is 1 layer. So, you have 1 horizontal layer followed by 1 vertical layer. So, whenever you are going from the horizontal to vertical you go through some via right.

You can understand that there are these horizontal layers, we make the horizontal connection here. And then suppose I have to make the vertical connection in this layer. So, I have to make a connection through this, which is through via to this right. So, which is called VH or HV may be horizontal up, then vertical up then horizontal or horizontal and vertical like this, or maybe you have a 3-layer 2 vertical layer 1 horizontal layer or 2 horizontal 1 vertical layer.

So, these all things are possible and again, I am not going to go into detail about this, but when you are working on the challenge routing problem these are the terminology that should come to you right.

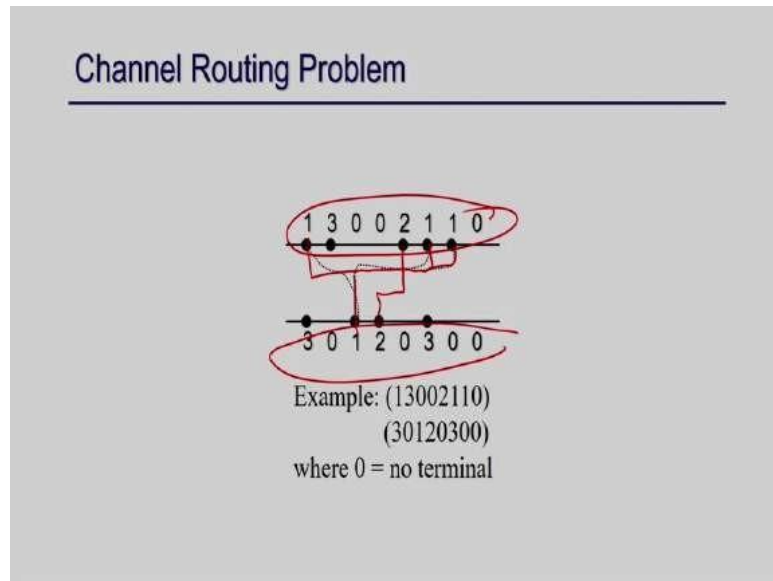
Channel Routing Problem

- **Input:**
 - Two vectors of the same length to represent the pins on two sides of the channel.
 - Number of layers and layer model used.
- **Output:**
 - Connect pins of the same net together
 - Minimize the channel width.
 - Minimize the number of vias.

And your channel routing problem is as I mentioned there are two vectors of the same length pins are there the pins are given on two sides of the channel and there the number of layers is given whether it is a VH model HV H model and both that is also given and you actually and the pin number.

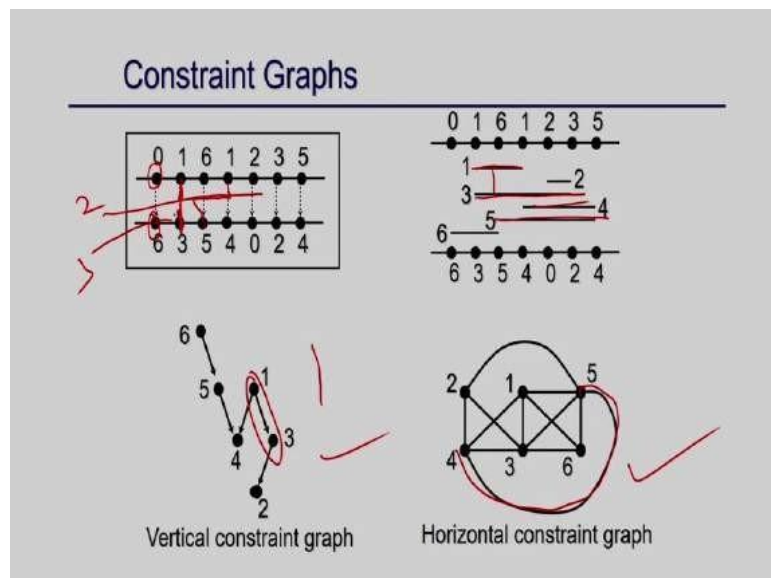
So, this is 1 and this is 1; that means, you have to make the connection like this right. So, whatever the pin number will you have to connect those. So, if this is 2 this is 2; that means, you have to make a connection through this right so, this is also so, given to you.

And you want to make the connection pin together and minimize the total channel width number of vias width means, the number of tracks right the number of channel width means the number of tracks and also you try to make a lot of vertical connection also this vias you have to make a vertical connection which also we try to reduce ok.



As I mentioned here an example. So, this is 1 and this is so, I have to make a connection like this right. Similarly, this is 2 means, I have to make a connection like this. So, this is what is how the problem is given you just given this set and this set right this is and you have given whether it is a VH HV and this and you have to solve this right.

(Refer Slide Time: 62:41)



So, you can see here there can be various constraints right. So, if you have a this and this that means they cannot be put on the same track because they are overlapping. So, this is

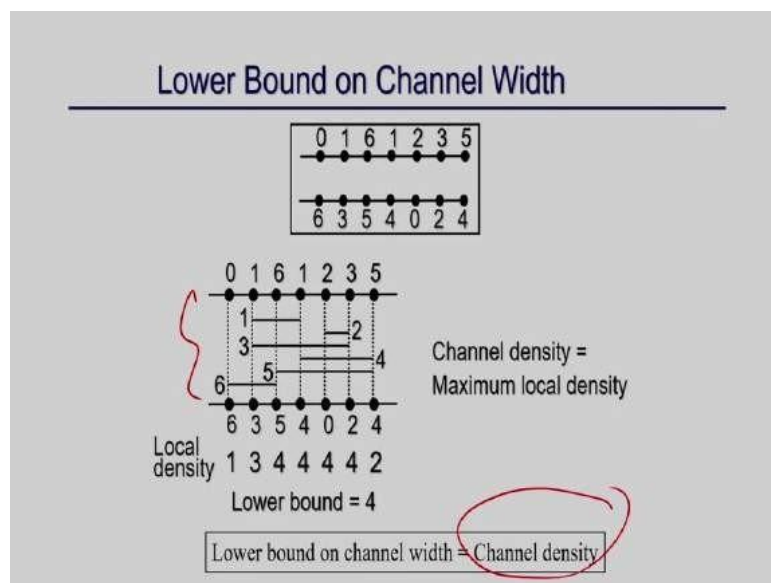
horizontal constraint you can construct a constraint graph like this for example, this and

These 5 and 4 have overlapped.

So, 5 to 4 have an edge right. Similarly, so, this is the horizontal constraint. Similarly, if 0 of there. So, you cannot they are do not cross in horizontal line right for a whatever I am trying to say is that if 6 is there so, basically say 1 to 3 right 1 to 3 there is a connection so; that means, whatever the connection is happening. So, 1 to 1 this 3 cannot go here, right three cannot go across this.

So, if this is the track say 2, the connection from three has to happen below this at least from 3, it cannot go crossing this right so, that is what is called a vertical constraint. So, you should have those you can extract out those vertical constraints or horizontal constraints, and using that you can solve your problem.

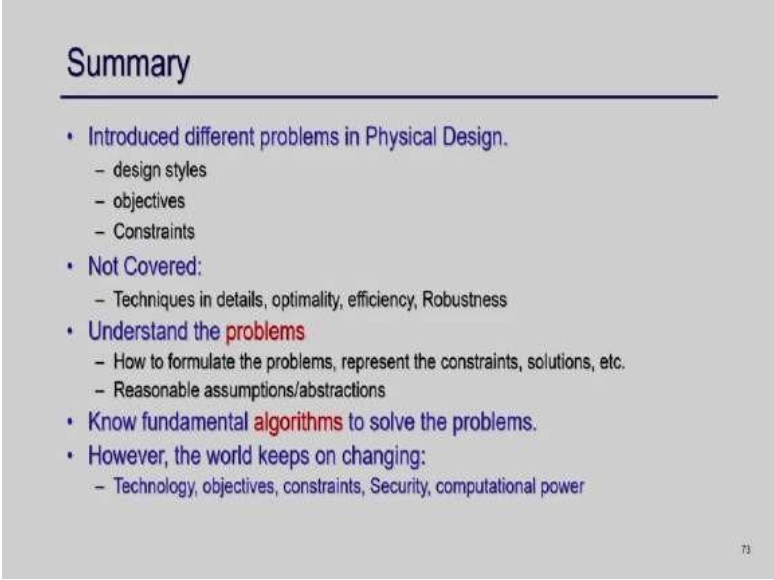
(Refer Slide Time: 63:41)



And you can identify the number of minimum tracks required using some logic also it is the channel density. So, those kinds of approaches and there are lots of algorithms is also their left edge algorithm and other kinds of solving algorithms, which actually can handle dogleg connections or say VHV model and all. Again, I am not going to detail those algorithms, but just to give you the problem flavor what is the channel routing problem and what is happening here ok.

Similarly, we have a switch box routing I just talked about the switch box I mean the channel because it has only two kinds of sides upside and downside, but in the switch box, I have a connection involved from whole all 4 sides so, it is a more complex problem. So, that is also there.

(Refer Slide Time: 64:29)



Summary

- Introduced different problems in Physical Design.
 - design styles
 - objectives
 - Constraints
- Not Covered:
 - Techniques in details, optimality, efficiency, Robustness
- Understand the **problems**
 - How to formulate the problems, represent the constraints, solutions, etc.
 - Reasonable assumptions/abstractions
- Know fundamental **algorithms** to solve the problems.
- However, the world keeps on changing:
 - Technology, objectives, constraints, Security, computational power

73

So, in summary, I have discussed this physical synthesis floor, I give a very brief overview of this physical synthesis what is placement, what is floor planning, what is partitioning initially and then what is routing and how what is the problem definition there, what is I have not covered is that the techniques the detailed technique in detail, their efficiency, their robustness those part I have I do not cover here.

So, the objective of this particular lecture is something just to give you an understanding of the problem. How to formulate that problem represent that constraint and how to represent the solution etcetera and you just think about what kind of assumption is there what are the constraints are there and also how we can fundamental algorithm to solve this problem like this heuristic base approach or sometimes you need left edge algorithm or those things or maybe sometime you will go for genetic algorithm or sometimes simulated when it annealing algorithm.

But so, but without going into detail about the solution we just talk from a very overview level, and also this physical synthesis is still relevant in the sense that the world is changing you the new kind of constraints are coming, like this, you have security constraints

nowadays, you can have the reliability you can have other new constraints like the criticality of your power of those things.

So, there is always a scope to evolve some new kind of placement floor planning or routing algorithms which can satisfy these kind of new in new strategies ok. So, with this, I conclude this particular module.

Thank you