Advanced VLSI Design

Prof. Sachin Patkar
Department of Electrical Engineering
Indian Institute of Technology- Bombay

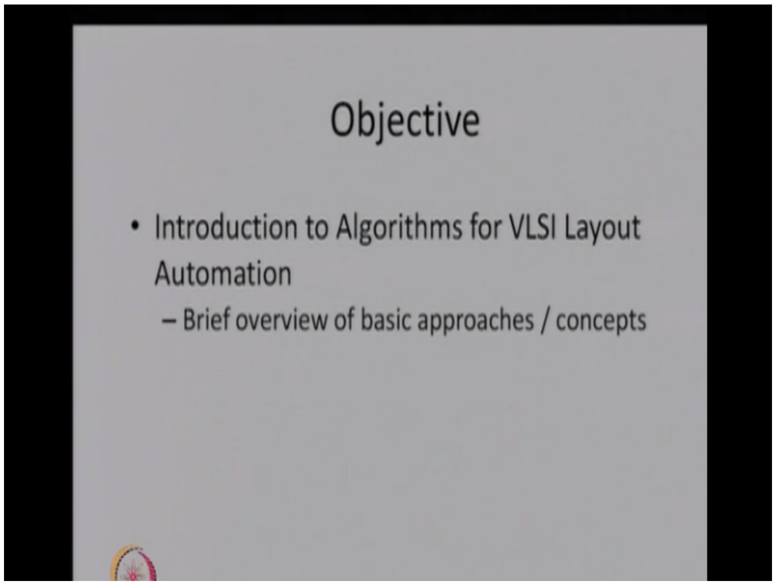
Lecture – 29
Brief Overview of Basic VLSI Design Automation Concepts

Welcome, today's lecture is a brief overview of some basic VLSI Physical Design Automation Concepts. I missed in the title it is Physical Design Automation that we are going to deal with, not general VLSI design which also includes Logical Design Automation. So, this and next couple of lectures, there will be some basic simplistic overview of fundamental concepts will be given without too much rigor and very formal treatment.

So, what VLSI Physical Design Automation is the stage in the design automation after the Logical Design Automation. The Logical Design Automation, the automation tools convert a behavioural specification given in some high-level language or some Hardware Description Language that is converted into a logic circuit, in terms of gates or may be in terms NMOS or PMOS switches.

So, this automated processes followed with the process of the flow of physical design, in which a logic net list is converted into a layout. Okay, so basically this is logic to layout automation. So, in this and the next couple of lectures, I will focus on some, you know, I will give you a brief overview of some simple fundamental basic concepts of a layout automation.

(Refer Slide Time: 02:04)

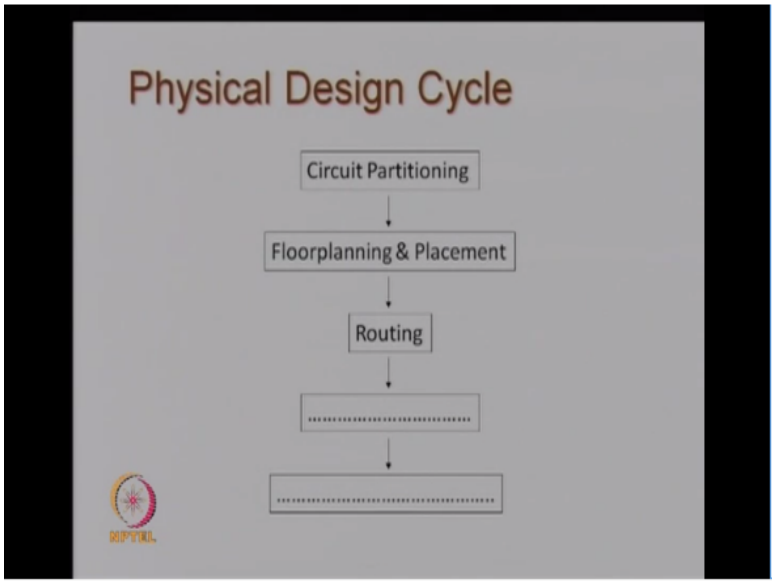


Objective

- Introduction to Algorithms for VLSI Layout Automation
  - Brief overview of basic approaches / concepts

I mean it is a subject by itself, so you can I mean look up the course material of a course on VLSI CAD and that will you more details of this alright.

(Refer Slide Time: 02:21)



Physical Design Cycle

$$\begin{tikzcd}[row sep=small] \boxed{\text{Circuit Partitioning}} \arrow[d] \\ \boxed{\text{Floorplanning \& Placement}} \arrow[d] \\ \boxed{\text{Routing}} \arrow[d] \\ \boxed{\dots\dots\dots\dots} \arrow[d] \\ \boxed{\dots\dots\dots\dots\dots\dots} \end{tikzcd}$$

So, in the Physical Design Automation Cycle, there are several stages. The ones which I am going to concentrate on, rather overview in this two-three lectures are this first three stages where things begin with circuit partitioning. A very general kind of stage. A very important stage to manage the complexity of large designs. Circuit partitioning is followed by floor planning and placement. With the help of partitioning of a circuit.

A chip area is floor planned and different zones of the chip area are, a sort of, assigned to earmark for different sub-designs. Typically, this could be manual designed into very logical subsystems of the original of big complex design. So, floor planning one can think of it is as approximate, like you know, getting an approximate idea of how the layout would like which sub design would be on, in which portion of the chip.

How much area in which part of the chip would be reserved for which sub design. That is a general objective of floor plan. It is a bit course level process. At the final level, the placement process will find out detailed locations of the cells. The cells in the blocks or the cells in the sub designs are to be placed in appropriate locations of the portion of the area earmarked for that sub design. Okay, so that is a placement.

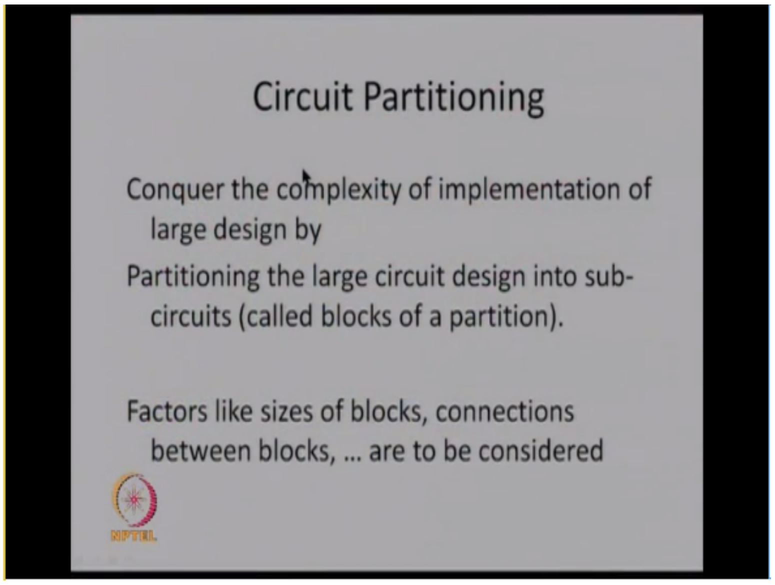
So, at the end of placement, we have these regions where the cells are going to be placed including the orientation, so we also know more or less precisely the location of the connection pins of those cells which have been placed and this information where the connection pins are at which coordinates of the chip, this information is fed to the routing stage. The routine stage will make use his of this placement information and connect up this.

Layout of single net or wires based on the locations of the terminal pins, so that is router's job, or we will stop more or less over here. Also included will be some overview of the core concepts of Static Timing Analysis and how they influence this physical design process floor. So, that will be subject of the next couple of pictures. So, in this lecture, I will just skim through some basic concepts and for routing.

I will give illustrate couple of basic algorithms through some example. So, I missed out something, I deliberately kept this blank. One can look at different levels of details. Within the routing, there will be some specific routing for clock tree as well as power grid, power and ground network and after routing stage, there will be typical stage of compaction of the layout and will be a stage of verification of the layout and so on so.

For extraction, circuit simulation again, post layout simulation. So, let me begin with circuit partitioning, a few remarks on that.

(Refer Slide Time: 06:01)



Circuit Partitioning

Conquer the complexity of implementation of large design by

Partitioning the large circuit design into sub-circuits (called blocks of a partition).

Factors like sizes of blocks, connections between blocks, ... are to be considered

So, circuit partitioning is about conquering the complexity of implementation of a large design by partitioning the large circuit into sub circuits. Basically, the automation is required for very large designs. Complex digital systems where fairly complicated behavior requires a lot of, like you know, is realised with the help of tens of thousands of millions of gates. So, the design is extremely complex large scale.

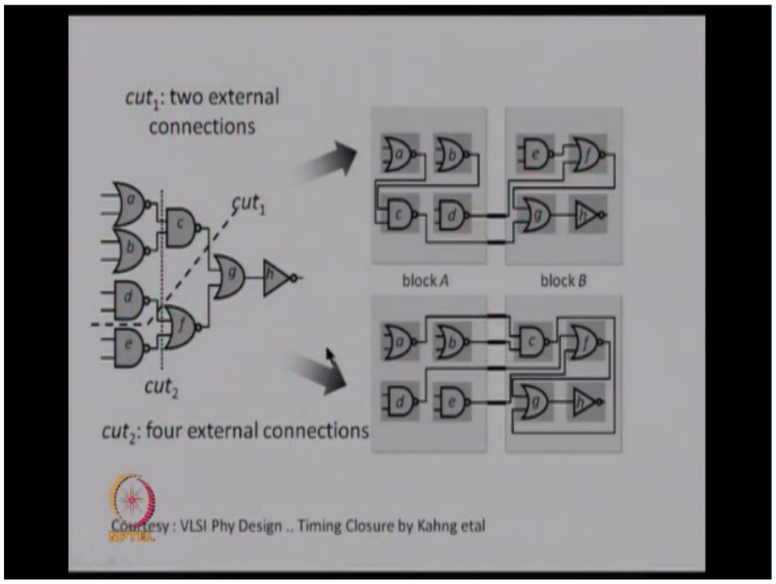
So, the algorithm even if there is efficient in terms of runtime, the size of the problem is so big that to really aim for efficient runtime, one has to divided and conquer approach wherein will divide the problem, solve sub problems more or less independently. Then conquer (()) (06:54) together solutions of the sub problems and get apparently good solution of the big problem. This technique does not necessarily always work perfectly.

But it is obvious natural paradigm for breaking down the complexity of problem by using this divide and conquer approach. So, we will partition the large design into sub circuits and these sub circuits I will call blocks of the partition. So, each block is a sub circuit and while partitioning the factors such as sizes of the block, how big the block is allowed to be, how many gates is it allowed to have.

These kinds of factor as well as the important factor of number of interconnections between the blocks. These factors are extremely important while deciding the partition. So, we would like to say keep the sizes of the blocks under control. They should not be too small or too large. Too small would be a waste of the resource that the block is going to be laid out on. Layout of the block is going to be better and then connections.

Also we would like them to be minimized for fairly intuitive two reasons as would be clear in the course of next lectures. So, partitioning is a very interesting large-scale optimization problem, very well researched and has applications in not just VLSI CAD, but a large set of applications in scientific computing and various other domains.

(Refer Slide Time: 08:41)


cut1: two external connections
cut1
cut2
cut2: four external connections
block A
block B
Courtesy : VLSI Phy Design .. Timing Closure by Kahng etal

So, here is a small toy illustration of circuit partitioning concept. So here is on the left, we have a circuit with this eight gates, A, B, C, D, E, F, G, H and the here there are two doted lines, dash lines marked here. One is labelled cutline which is separating gates A, B, C and D from the rest, that is E, F, G, H. So, cut one is separating the block of the circuit A, B, C, D from the other block E, F, G, H, whereas this vertical cutline.

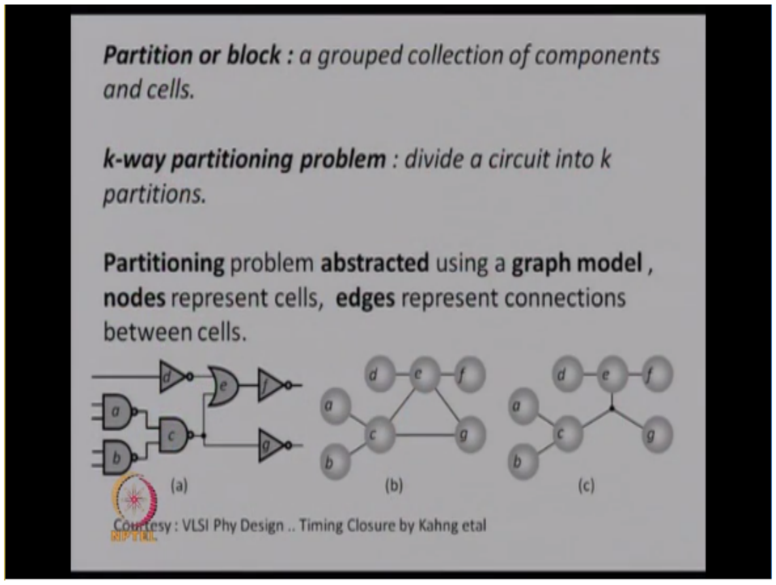
I mean this particular dotted cutline which are labelled cut 2 to is separating the block containing A, B, D and E, these gates from the block containing the gates C, F, G, H. So, these two cuts are describing two different partitions, each partition having two blocks. Okay, but the quality of these different partitions will have different implication as far as this layout is concerned and that is one of the most obvious thing that you see over here is that.

For the first partition which we have obtained using cut one which separates A, B, C, D from E, F, G, H, you see that there are only two connections going from one block to the other block, that is the layout of one of the blocks. Between the layout of these two blocks, there are only two connections crossing. So, it indicates like long-running wires will be relatively few. On the other hand, for the other cut which separates A, B, D, E from C, F, G, H.

You see that there as many as four signal nets being cut between the layouts of this blocks, that means there is more chance of very long-running wires and that clearly like from the intuitively from the timing point of view, it could be a bad layout. By the way, this example and these figures are taken from the book VLSI Physical Design, title is VLSI Physical Design from graph partitioning to timing closure. The authors are Kahng, Lienig, Markov and Hu. It is published by Springer.

It is a good compact book with several topics of physical design explained very well. Algorithm details are given plus a lot of illustrations are there and a good book for both students as well as practitioners. So, this as well as few other examples will be borrowed from this book. So, some concepts in partitioning like terminology.

(Refer Slide Time: 11:39)



Partition or block : *a grouped collection of components and cells.*

k-way partitioning problem *: divide a circuit into k partitions.*

Partitioning problem **abstracted** using a **graph model** , **nodes** represent cells, **edges** represent connections between cells.

d
e
f
a
c
b
g
(a)
(b)
(c)

Courtesy : VLSI Phy Design .. Timing Closure by Kahng etal

Partition of block is a group collection of components of cells. In general, very typically one looks for a two-way partitioning. Partition into two blocks but one can in general talk about K-way partitioning where the task is to divide a circuit into K partitions. So, very often or like very typically partitioning problem is studied with abstract graph models where nodes represent cells and edges represent connection between the cells.

Sometimes, in the context of VLSI CAD, the extension of graph, this hyper-graph concept is more often used. So, here is a net list and the cells of that net list, A, B, C, D up to G are represented by the nodes of this graph, the way these cells are connected to each other in some sense, maybe sometimes with the direction information, sometimes without direction information. Here in this case, there is no direction information captured here.

So, not everything that we know about the circuit is required for the sake of partitioning, like only the abstract essence is required and that is the graph model and that graft is subject to partitioning algorithms of like various complexity of cleverness, degrees of cleverness, and we get useful solutions of information out of it. More generally so-called hyper-graph model is used which I am not going to go into but very popular standard model.

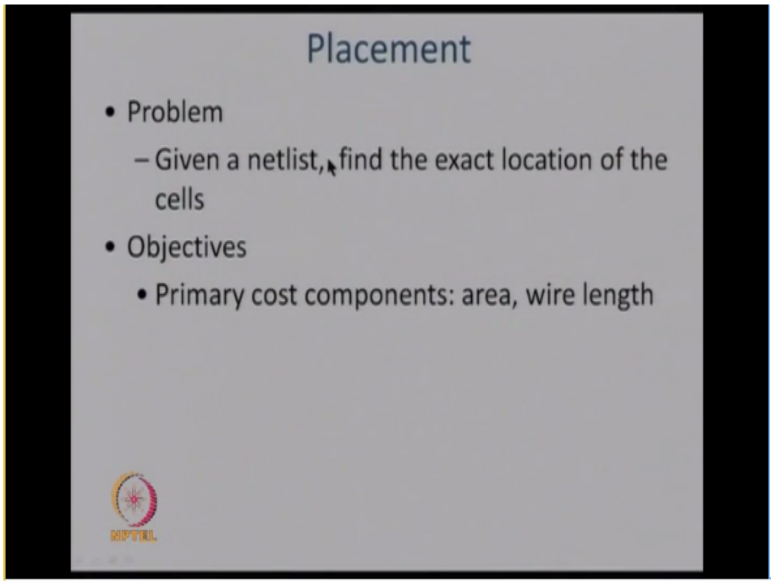
You look up in literature more and you will encounter it very easily and like there will be very lucid descriptions or explanations of that concept. So, I think that was just few terminology, few terms that one encounters in the context of partitioning. I will give one special lecture on partitioning alone which will give us illustration of one of the central pioneering algorithm heuristics for circuit partitioning.

Of course, it will not be a very clever kind of algorithm with lot of sophistication but it turned out to be an algorithm which solved a problem decently and which gave rise to lot of very interesting variations and extensions. So, that is later, so partitioning I will skip over to the next. In fact, after partitioning, I should say partitioning group would have been used for deciding the floor plan of the chip.

Like if some good partition is known which divides a big circuit into some very modular logical sub circuits or some algorithm finds good partition, good blocks, then correspondingly the chip area is divided into appropriately large or appropriately shaped rectangles or regions and different regions are earmarked for different sub designs which have been obtained in the process of partitioning.

After floor planning, one gets a rough idea about where the layout of each one of the sub designs is going to be and then the placement problem is going to detail out the exact locations of the cells within each block of circuit and within the corresponding area of the chip.

(Refer Slide Time: 15:31)



Placement

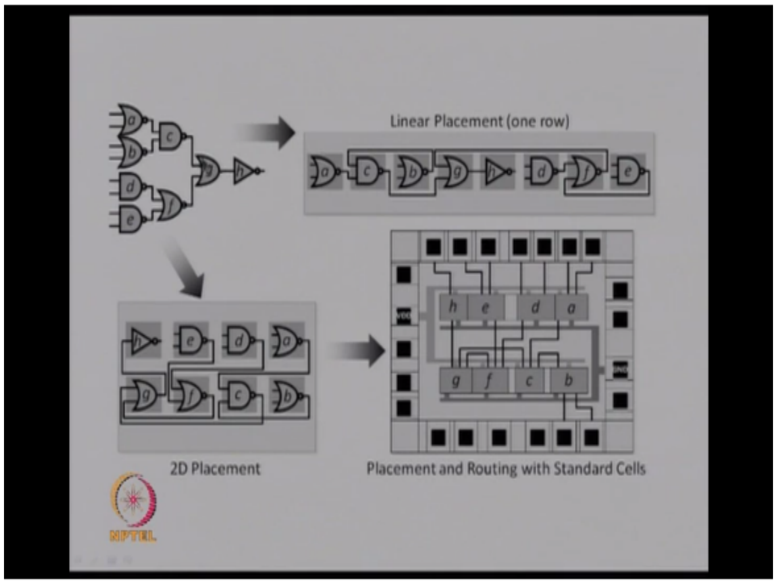
- Problem
  - Given a netlist, find the exact location of the cells
- Objectives
  - Primary cost components: area, wire length

So, placement is the problem where we are given a netlist, like logic netlist of circuit. We have to find the exact locations of the cells. The primary objectives are area and wire length. We like to, for natural reasons, intuitive reasons, we would like to minimize the area of the layout because one typical wants as small as possible VLSI chips and we would also like to minimize the wire lengths, either the total wire length or the maximum length of any particular wire.

So, all kind of combination of such objectives will be used for optimization. Wire length is the directly related to complexity of routing as well as the delay, the timing performance critic. So, placement is a complex problem because it is going to deal with large problem and it has both combinatorial as well as geometric character and lot of research has happened in this area and there is a variety of approaches fairly interesting subject by itself.

So, one specific lecture I will illustrate, one interesting example of a placement algorithm, just for the sake of flavor, so that will be later. So, I will just give a very brief introduction to the notions involved in placement or example, not to algorithm itself in this lecture. In later lecture, there will be illustration of a specific algorithm which I just mentioned which one.

(Refer Slide Time: 17:22)


Linear Placement (one row)
2D Placement
Placement and Routing with Standard Cells

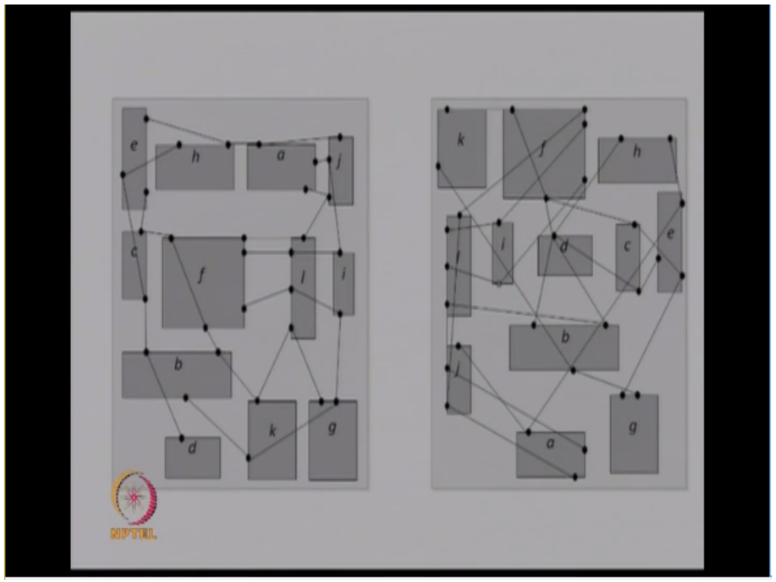
So, placement as shown again in this figure from the same book that I mentioned earlier. There is a netlist with these eight gates and if the target of the placement is a one-dimensional chip wherein we can accommodate a row of standard cells for example, then the placement problem is really about like ordering each one of the cells which would be implemented as a standard cell. The placement is essentially deciding on a linear ordering, sequential ordering of this cells.

Depending on in which order we layout the cells, the length of the wires would be optimized or maximum length of any wire would be optimized and that would give rise to different degrees of timing performance. If the target chip is two-dimensional, the placement might look like this in two different rows in 2D fashion and depending on the placement of cells in different positions, the wires might look congested or might become bit long or total wire length might be large.

And router might have more difficulty and the timing might be better or worse depending on the quality of the placement. After placement and routing, one would typically have a picture like this where we have this gate placement as standard cells, like in two different rows with sometimes some gas between them through which routing can be done, but there will be optimization of this placement done by the placement algorithm and.

The router will take care of finding the best possible route so that the amount of routing region is as minimum as possible, so that the whole area is as small as possible. At the same time, the locations of the terminals, IO pads would also be taken into account while doing this placement and routing. So, in the end, we would have layout of this kind and for last designs it will have to be necessarily automated.

(Refer Slide Time: 19:50)



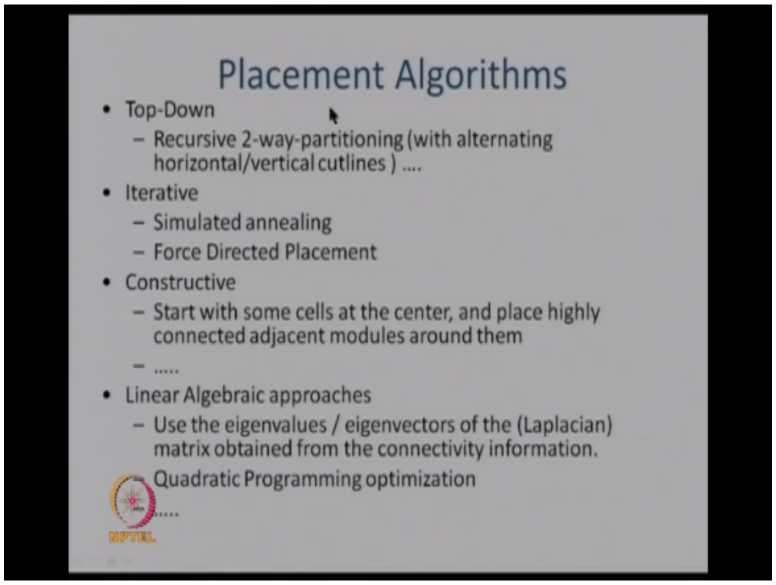
Here is a picture of how placement can influence the quality of routing. So, both these figures show some different placements of a set of cells from A to whatever G or A to K. Both this placement roughly occupy the same area, so that means the same amount of routing area, but the placement on the left hand side is clearly like you know something that favours a better routing, so as seen, like although this is not the routing.

But we know that the wires are going to go from the terminals of the cells to the terminals and so on so forth. This information is known because the placement process has computed the locations of the pins and the circuit information has the connections how the pins are connected to each other. So, that gives us some approximate picture of how the routing will look like.

Whereas in this the routing appears to be much worse.

So, the placement influences the quality of routing in a significant fashion. So, placement is a very important part of the process.

(Refer Slide Time: 21:12)



Placement Algorithms

- Top-Down
  - Recursive 2-way-partitioning (with alternating horizontal/vertical cutlines) ....
- Iterative
  - Simulated annealing
  - Force Directed Placement
- Constructive
  - Start with some cells at the center, and place highly connected adjacent modules around them
  - .....
- Linear Algebraic approaches
  - Use the eigenvalues / eigenvectors of the (Laplacian) matrix obtained from the connectivity information.
  - Quadratic Programming optimization
  - .....

In placement algorithm, there is a rich set of placement algorithm broadly classified into this paradigms. First one is stop down, not in specific order, but first popular paradigms is top-down which is a recursive paradigm. Then, there is iterative method for placement VLSI layout circuit placement. Then this constructive or incremental process of generating a layout or placement.

And there quite interesting mathematical approaches based on linear algebra, matrix computations or mathematical programming, in particular say quadratic programming. Many of them use eigenspectrums, eigenvalues and eigenvectors of matrix called Laplacian which is matrix obtained from the connectivity information. This top-down approach is a recursive approach which typically uses two-way or min-cut partitioning which partitions the circuit into two parts.

And these two different parts are to be laid out on two roughly equal portions of the chip, and this two portions of the chip would have been found by like choosing a horizontal or vertical cutline dividing the chip area into two parts. So, the circuit is partitioned into two blocks and the chip is partitioned into two parts and then we have two separate sub problems which should be solved separately.

And then maybe with some awareness of each other and the solution should be stitched together. If each one of the sub problems itself is not small enough, then we have to do this. We can go ahead and do the same process recursively. Partition each one of the blocks into two parts, but now like you know use the alternative, like other kind of cutline. If we had earlier chosen to divide the chip into two parts, left and right.

Now for the next level we can divide the chip into each portion of the chip, left and right portion into top end bottom, left top and left bottom, similarly, right top and right bottom. So, we can have alternating division or space division of the chip and correspondingly we will be recursively subdividing the given netlist also. So, appropriate circuit partitioning algorithm would be used depending on the efficiency or quality.

In fact, many different heuristics can be tried. One does not always put full faith in just one algorithm. I mean most of this problems are hard and so we little of guarantees are given about the qualities but roughly the benchmarking would indicate that many of the algorithm are good. They perform quite well, still one would like to try out different algorithms and choose the best result among them.

Then, about iterative paradigm, iterative algorithm is one of the most well-known algorithm like based on one more celebrated well-known paradigm called simulated annealing which is like an extension of Metropolis Algorithm regarded as one of the top 10 algorithm of the century. Very popular paradigms which is based on interesting theory of Markov Chain, so it is not extremely, but it does tend to give like good solutions if you run it long enough.

And it is always available as one of the options in the kit for the placement algorithms. Another important iterative method that is again based on some interesting concept from physics is the so-called Force Directed Placement. The Force Directed Placement is based on like it uses this mass spring model, so the cells which are regarded as masses and connections between the cells depending on the degree of connection, the amount of connection.

Spring is strong or a weak spring is attached and based on the forces exerted, the masses will settle down in some position and that position is used as an estimate or as an approximation to the placement detail of locations. Constructive algorithms are incremental, they start with some cells at the centre and then place highly connected adjacent models around them. So, it is incremental.

You make some clever decision about what to place first, which seems to be at the centre of the things and then based on those location and the connectivity of some of the models which are highly connected to this setup already place cells. We start placing those highly connected adjacent models around them and so on so forth. Then, less thickly connected models will be in the outer zone and so on.

So, this incremental thing might look nice, but again the choice of which cells to choose, how to estimate, how to efficiently compute the good neighbourhood or good cells for the next placement and all that. There is lot of cleverness possible here. Various heuristics would have been designed for these paradigms. As I already mentioned about Linear Algebraic approach, the use, the eigen spectrum of Laplacian and to embed the prop.

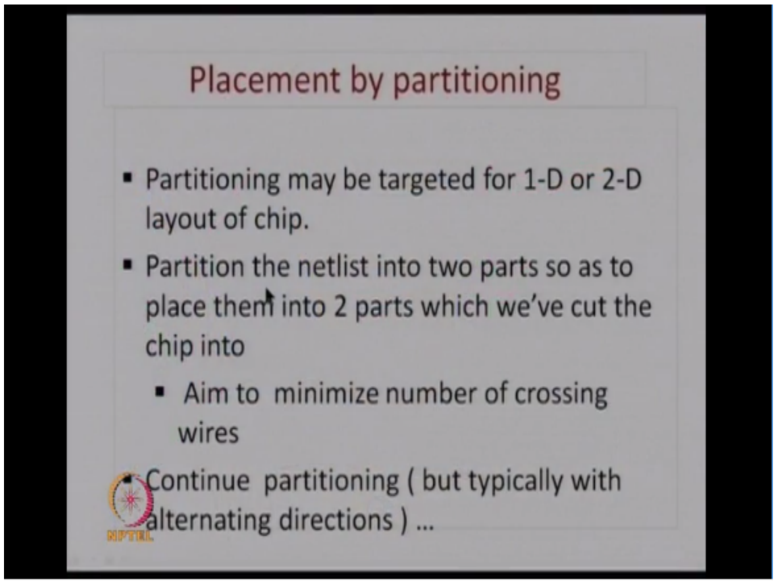
Convert the more or less combinatorial problem of finding the locations or the positions at which the cells should be placed. This problem is converted into geometric problem where the geometric information is obtained with the help of several eigenvectors for example. Which eigenvectors, that would depend on the smallest eigenvalues or set of certain smallest number of smallest eigenvalues.

So, it is backed by interesting theory from like Linear Algebra and Spectrographs. One of the main theorems, easy but important theorems whose variations are used here is the so-called Hall’s Theorem. Very similar to these linear algebraic eigenvectors based approaches, there is a mathematical programming approach based on quadratic programming where we use sum of the squares of like the kind of square of Euclidean distance, length of the nets.

So, it will have some of the squares kind of form, so it will have the quadratic objective function and using again Lagrange techniques or calculus nonlinear programming ideas, one can arrive at good solutions with these interesting approaches. Of course with some post processing and with combining these solutions with some other heuristics, one can get better solution. So, these techniques will not claim to be the best techniques by themselves.

They can be variously combined with each other or independently used for finding the best and so on. So, there is a fairly broad set of algorithm for placement.

(Refer Slide Time: 29:22)



Placement by partitioning

- Partitioning may be targeted for 1-D or 2-D layout of chip.
- Partition the netlist into two parts so as to place them into 2 parts which we've cut the chip into
  - Aim to minimize number of crossing wires
- Continue partitioning (but typically with alternating directions) …

So, in one lecture I am going to focus on placement by partitioning, slightly informal introduction or illustration would be given and that placement by partitioning is the top-down approach where a partitioning may be targeted for 1D or 2D layout of a chip and the netlist will be partitioned into two parts at each level of the recursion, with the objective that they should be placed into two parts which we have cut the chip into, maybe left and right part or top and bottom part.

The aim would be to minimize the number of crossing wires and if the sub-problems still happen to be complex in sense of size, then we recursively continue the same process, but the chip subareas will be partitioned with alternating directions, horizontal, vertical and so on so forth.

So, that is a rough idea about placement by partitioning. After this, I will move to overview of routing algorithms.

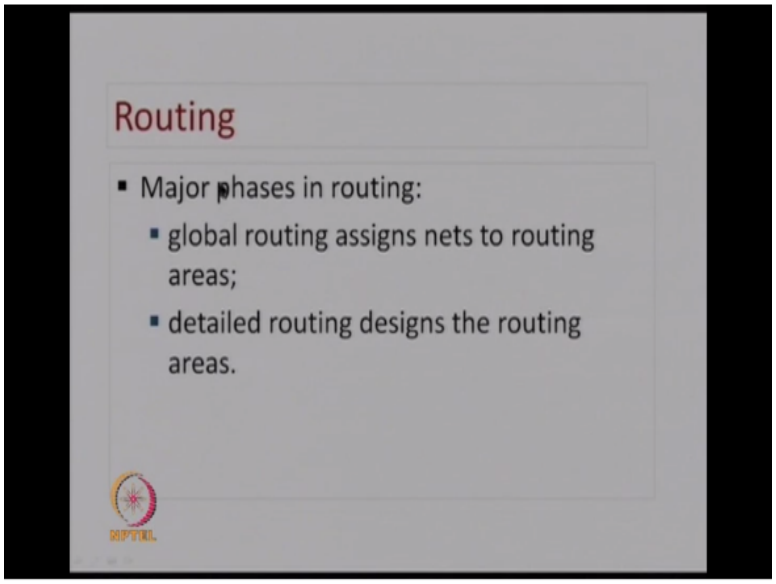
Here, I will have a couple of concrete illustrations but very informal, very sketchy, just highlighting the main ideas which are very intuitive. So, at the end of this lecture, I would have given you a couple of examples of routing, then in a couple of subsequent lectures there will be an illustration of placement algorithm, specifically placement b y partitioning with the idea of terminal propagation, separate lecture will be there on partitioning also.

Simple well-known algorithm called Kernighan–Lin will be discussed there and in another lecture, I would give you some idea of overview of the main core concepts of timing analysis. Again, how the graph model is used and how using the delay information the so-called actual arrival times and required arrival times are computed, that will be in a separate lecture and why timing is important from the perspective physical design flow that also will be remarked.

So, next I will be going to routing. After having taken a very brief look at the concepts of the partitioning and placement, I will just give a couple of examples of concepts, the algorithmic ideas involving in routing. Routing as I remarked before is about finding the layout for the actual wires. The actual layout for this signal nets because at the end of partitioning, we know the locations of the terminal pins of every signal net.

Some of them are coming from the periphery or boundary of the chip. Some are obviously driven by the cells. So, with this information, we have a fairly complex task of like routing possible millions of wires. So, it is a very large-scale problem again. So, again in some sense like of course we can look at several sub portions and routing separately, but routing itself is divided into two phases.

(Refer Slide Time: 32:37)



Routing

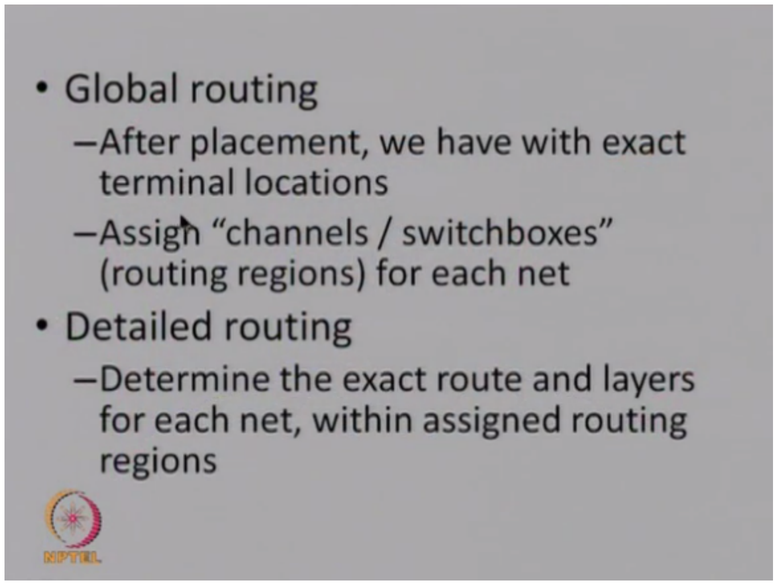
- Major phases in routing:
  - global routing assigns nets to routing areas;
  - detailed routing designs the routing areas.

There is something called global routing which is a bit course level routing and there is something called detailed routing, so these are the major phases in routing. The global routing would have the task of assigning signal nets to routing areas. After placement, we know exactly where the cells are and we know which areas of the chip are free for routing the signal nets. So, these routable areas are divided into subareas called routing regions.

So there is a vast collection of routing regions. Maybe they are like rectangle or square like shape regions. The global routing would identify for every net which of this routing regions, the signal net should be laid out. Detailed routing is going to actually find out the particular track inside each region along which the wires layout will be fixed. So, a global routing is going look at big problem, the whole chip, the whole set of routing regions and all single nets and for every night.

It is going to find a set of routing regions through which a net will ideally pass, so that the routes are as short as possible hopefully. Detailed routing is going to look at each subregion and within the subregion it is going to fix the position of the wire, the track on which the wire will, on which layer, on which horizontal track, on which vertical branch, all that will be decided by detailed routing.

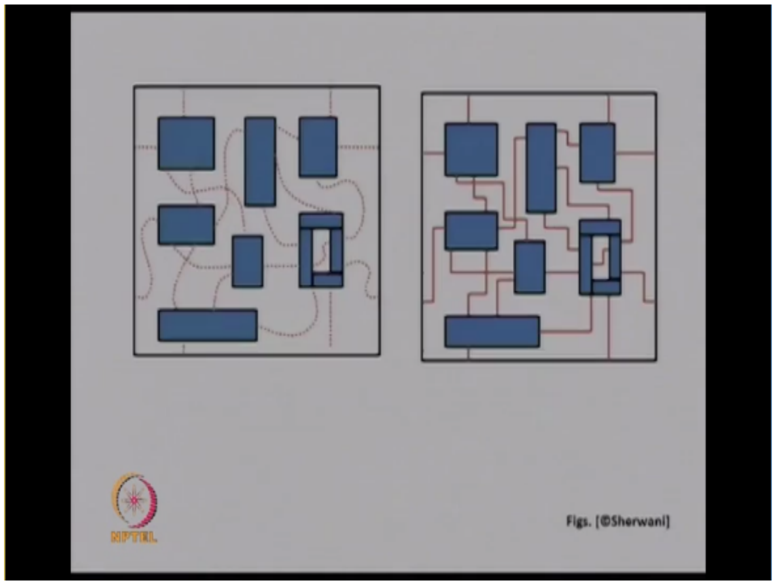
(Refer Slide Time: 34:33)



- Global routing
  - After placement, we have with exact terminal locations
  - Assign "channels / switchboxes" (routing regions) for each net
- Detailed routing
  - Determine the exact route and layers for each net, within assigned routing regions

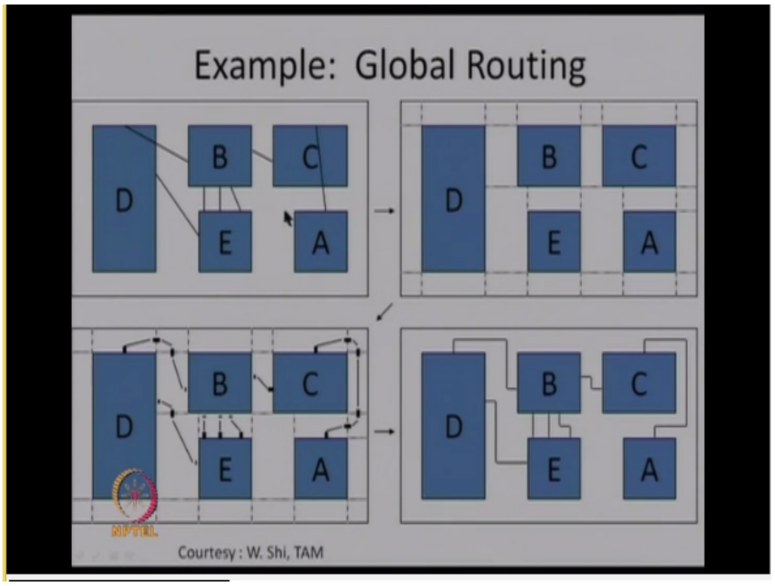
So, global routing after placement we have the exact terminal locations and we want to assign this routing regions which could be either channels or switch boxes. You are going to find which of these channels and switch boxes should be used to route a particular net and this has to be done for each net. Detailed routing will determine exact route and layers for each net within the assigned routing region. Okay, that is what I just remarked.

(Refer Slide Time: 35:01)



So, here is one illustration from the book of Sherwani. So, these blue rectangles are the regions where the cells have been placed and this dotted red lines are like approximate picture of how the wires will be routed and then the detail routing is going to fix exact location the horizontal tracks, vertical branches along which this wires should go. This is still just abstract picture, but this is one way of visualizing what global routing and digital routing mean.

(Refer Slide Time: 35:45)



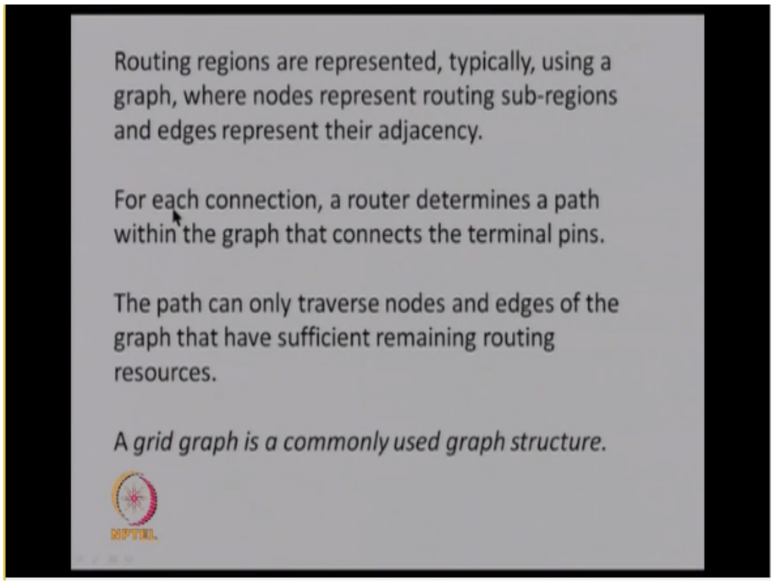
Example: Global Routing

Here is another picture of example taken from the net from the course of Texas University. So, here is the set of cells and some wires connecting them. Then, the regions outside the placed cells is the routable region and that region is broken up into rectangle shapes which are called channels. So, for example, this is one channel, this is another channel, one more channels here. So, so many channels and few switch boxes like this. These are called switch boxes.

So, these are regions are identified by the preprocessing part of global routing and then for every net, everyone of this connect signal net, the global router would identify that this particular net which goes from D2, this particular pin of B should go through this channel and then should go through this switchbox and then through this particular channel and onto this particular pin and so and so forth.

So, for every net, such sequence of channels and switch boxes would be identified. Then detailed routing is going to look at each channel and each switchbox and fix the positions of the wires tracks, which track and which branch, vertical branch, which horizontal track that will be fixed by have detailed routing, that could be something like this.

(Refer Slide Time 37:14)


Routing regions are represented, typically, using a graph, where nodes represent routing sub-regions and edges represent their adjacency.

For each connection, a router determines a path within the graph that connects the terminal pins.

The path can only traverse nodes and edges of the graph that have sufficient remaining routing resources.

A *grid graph is a commonly used graph structure.*

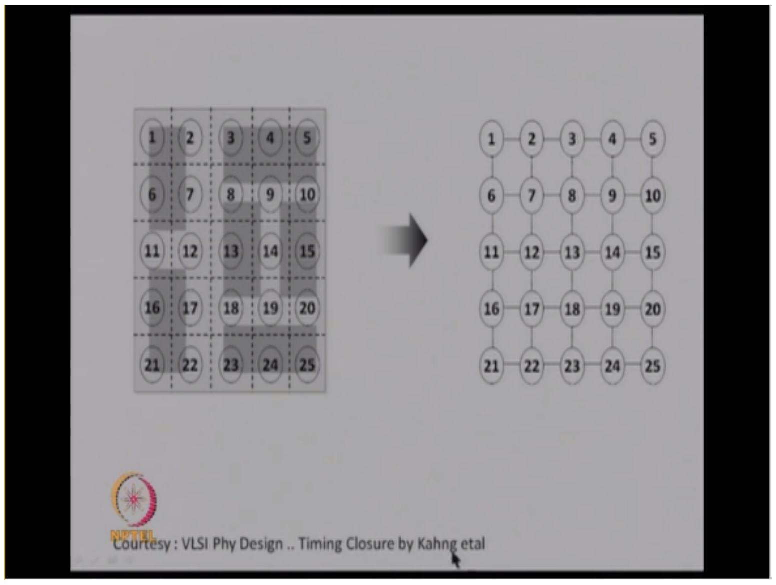
Then to model the algorithm, to kind of describe the algorithms, routing regions are represented typically using the graph model. Again, graph is a very popular data structure in this combinatorial algorithm for Physical Design Automation. The notes of the graph would represent routing sub-regions, like channels and switch boxes and they just represent the adjacencies.

For each connection, each net, the router that determines the path within the graph that connects the terminal pins. Maybe like something tree, not just a path as illustrate in the next example, but the important thing is that the path can traverse those notes and edges of the graph which has sufficient remaining routing resources. For example, this is a sequential process one net at a time is routed maybe, just one net at a time and then after approximate routes for each one of this nets done so far.

We should reduce the capacities of those regions because some of them have been committed for the layout of some of the segments of the nets. So, the resource availability of each routing region or each channel is going to be updated and that information would be used while figuring out the global route for the next net that would be considered. So, this graph models have the notes and edges with capturing the appropriate information about routing regions.

And their adjacencies, but some kind of resource capacity information is also available which will be decremented to keep track of how much is the available routing resource within that region. One of the specific kinds of graph used in this global routing is so-called grid graph. So, this grid graph is a commonly used data structure or graph of structure. The example is over here. This example is from Kahng's book.

(Refer Slide Time: 39:40)

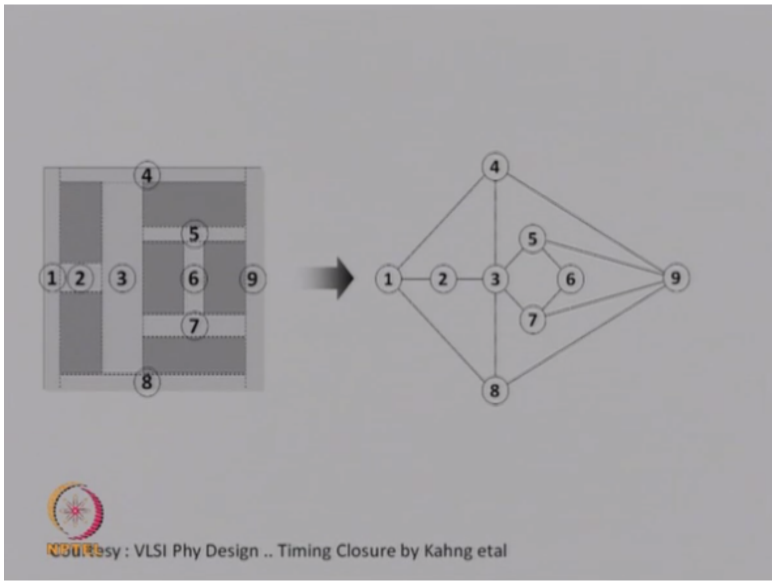


So, whole chip area is divided into this grid. This size of the grid will be chosen appropriately. If you want to use this idea for global routing essentially course approximate routing, then this can be coarse. If you want this idea of grid graph to be used for detailed routing, like you want to get more accurate idea, then the grid should be much finer.

Along the grid, we will have there will be places where the cells are placed and based on that part of this grid cells would have reduced capacity for routing purpose and so on. For example, the cell number 12 has plenty of capacity routing capacity, cell number 11, this is not cell of the circuit, but cell of the grid. This grid cell of the chip will have more routing capacity whereas something like grid cell number 13 will have very little area for routing and so on.

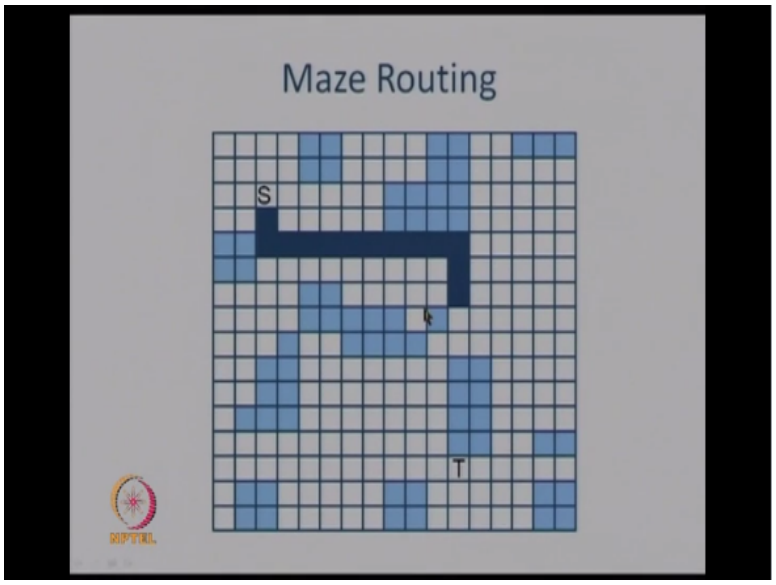
The adjacency of this grid cells is captured by edges. The grid cells are represented by notes of this graph and some shortest path or variations of shortest part of algorithms are going to be implemented on such the graph structures which will lead to the solutions of global routing problems.

(Refer Slide Time: 41:05)



As an alternative to grid graph, there is something called channel adjacency graph where every channel, say channel 1, 2, 3, 4, each one of the channel is represented by a note of a graph and depending on which channels are adjacent, there is an edge between the notes of the corresponding channel. So, channel 1 and channel 2 are adjacent, so there is an edge. Again, there could be some capacity information of this notes and edges reflecting the routing capacities or some other factors.

(Refer Slide Time: 41:38)



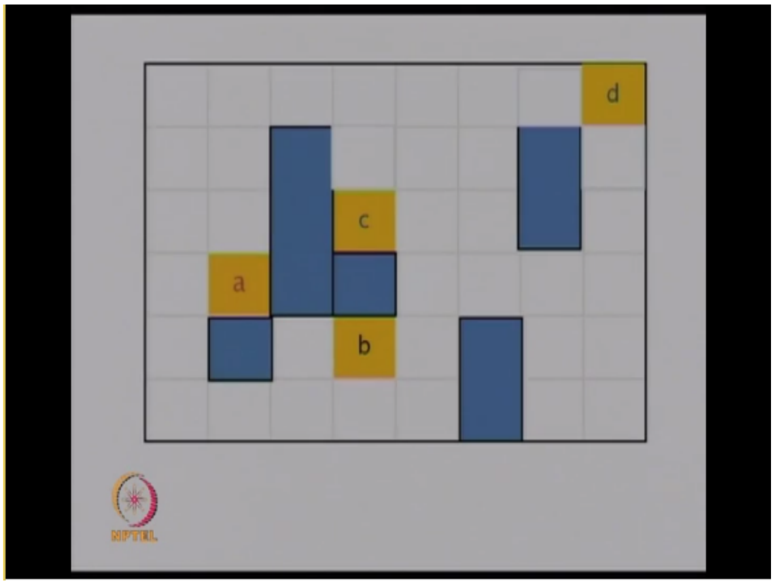
Maze Routing

But for the sake of illustration, I am going to lead you through one very simple paradigm called Maze routing. In various like other context you might have heard about Maze routing. It is very popular. It is one of the earliest algorithm for the routing problem. So, as shown over here, Maze routing will use some kind of grid and this light blue portions represent some obstacle in the maze and these are indeed the places where logic circuit cells have been placed. So, these are not available for routing.

Whereas this empty white background grid cells are available for routing. Supposing a connection has to be made from somewhere over here to someplace over here, then and correspondingly we will to find some kind of shortest path through this grid, the maze and that would evidently be this, okay. If you just look at other alternative which will be bit longer than this.

So, there are more possibilities than this, but one such the good as short as possible or often the shortest possible sequence of such grid cells is used for the purpose of eventual routing of wire from a S to T, and once that is chosen, then the capacities of each one of the grid cells along this path would be decremented to reflect that some partial use of those routing regions.

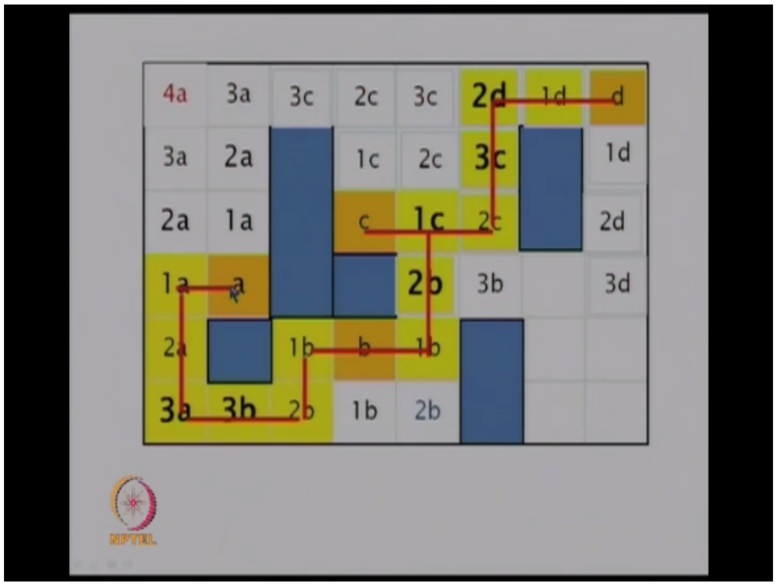
(Refer Slide Time: 43:34)



Now, I will take you through an illustration of a simple idea. So, supposing here is a chip area which is divided into this grid and this dark blue portions are the regions where you cannot do any routing because this is where this circuit cells have been placed and let us say in this grid cells marked A, B, C, D, we have some pins which need to be connected by a single net, okay, maybe that net is getting driven by A and is driving some pin of a gate in this zone C or in zone

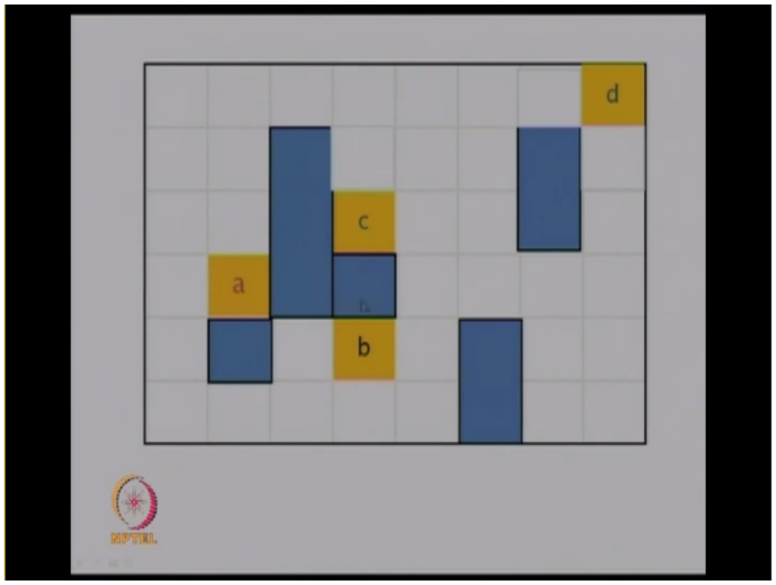
B as well as zone D. These are the grid cell zones.

(Refer Slide Time: 44:24)



So, we would like to eventually arrive at some so-called a net of this kind. A driving it, the signal flowing to B as well as to C as well as to D. Okay, so how is this to be computed. So this is the initial data.

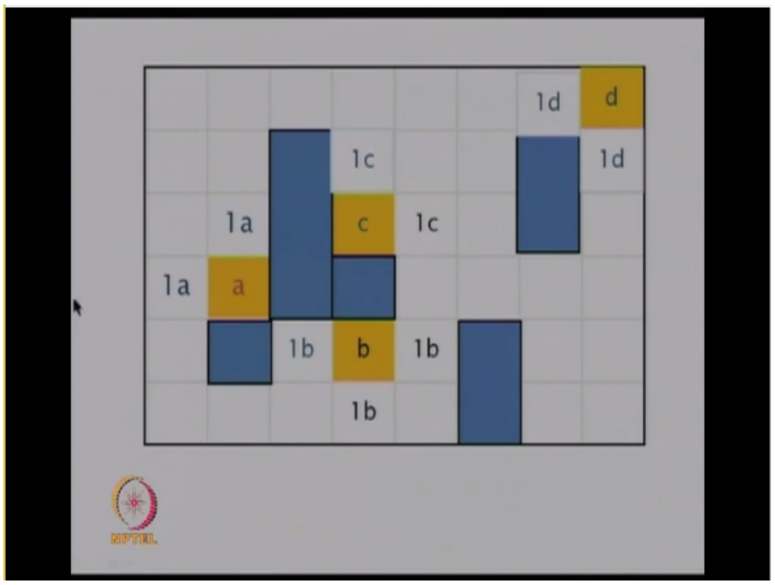
(Refer Slide Time: 44:38)



For every net which could be multi terminal net you would have such routing cells marked as the terminals to be connected, terminal cells or grid cells to be connected, so main idea in this algorithm is so-called wavefront and very natural idea. So, what we are going to do is propagating waves and in synchronous fashion.

So, from every cell, you will start out a wave which will like more one distance one neighbouring cell further and when this wavefront meet each other, that is the time we know that a short connection is found out and a bit of bookkeeping is required with the help of so-called the front of the wavefront and that will give us the final answers, help us in getting the layout of a net connecting A, B, C, D.

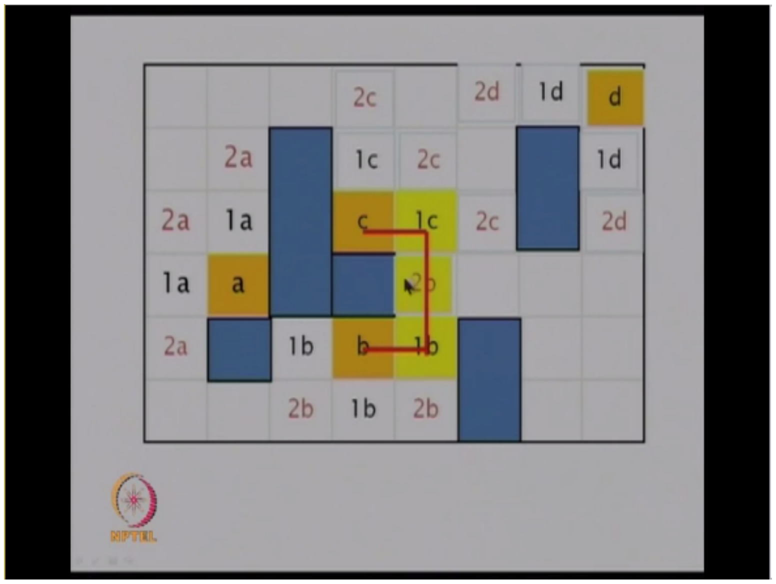
(Refer Slide Time: 45:37)



We start the wavefront. In the first step, the wavefront would be at the distance one from the sources, so this is grid cell marked 1A indicating that it is a distance one from A. It is part of the wavefront that is started from A. Similarly form B, the wave front reaches this cell as well as this and this. From C it reaches here, it is marked 1C and 1C and from D it reaches these two. In the next step, note that the wavefront are not still like touched each other.

Whereas in the next step from 1C and this cells, the wavefront will reach this, this, this, as well as this. But before growing the wavefront of C, we would have grown the wavefront of B and these wavefront would have reached over here. So, this would have been marked 2B.

(Refer Slide Time: 46:34)

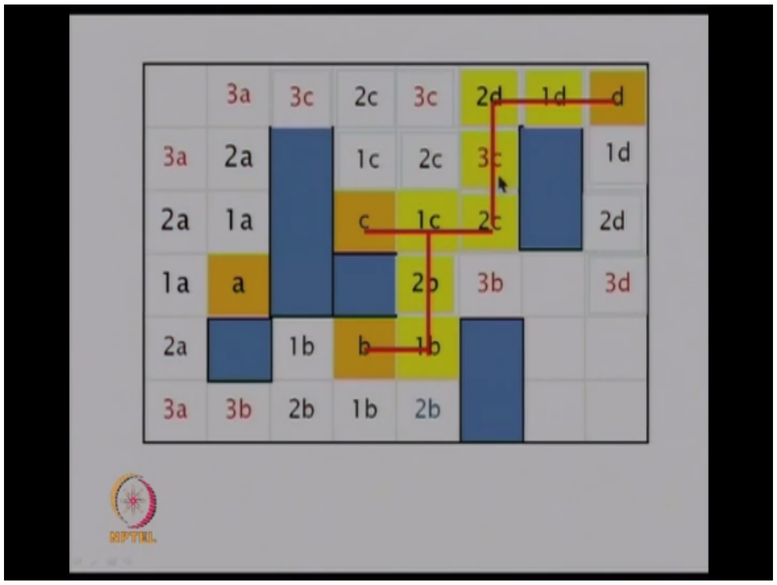


So, in the next step, B wavefront would have reached here, marked in yellow 2B. A wavefront would have reached this cell is marked 2A, 2A, 2A. This wavefront would have reached here, here, here, as well as here. Now, when you start like you know updating the wavefront for C, you notice, this becomes the new grid cell on the wavefront, this also is on wavefront of C, but while like growing the wavefront from here to here.

One notices that we are already adjacent to the wavefront touching the wavefront of B and that indicates that now we have identified a sequence of grid cells through which a short connection can be made between C and D and that we fix in this. Similarly, the wave front of D has been extended to cover visit 2D and this particular cell. So, only this connection has been made.

Whereas in the next step, this wavefront of A is going to reach such cells this, this one, and this one. Wavefront of B will reach here. Wavefront of C will reach here, here, here and over here also and when we start growing wavefront of D, we will notice that wavefront of C which has reached here is touching the wavefront of D. So, D and C would get connected in the next step as shown over here.

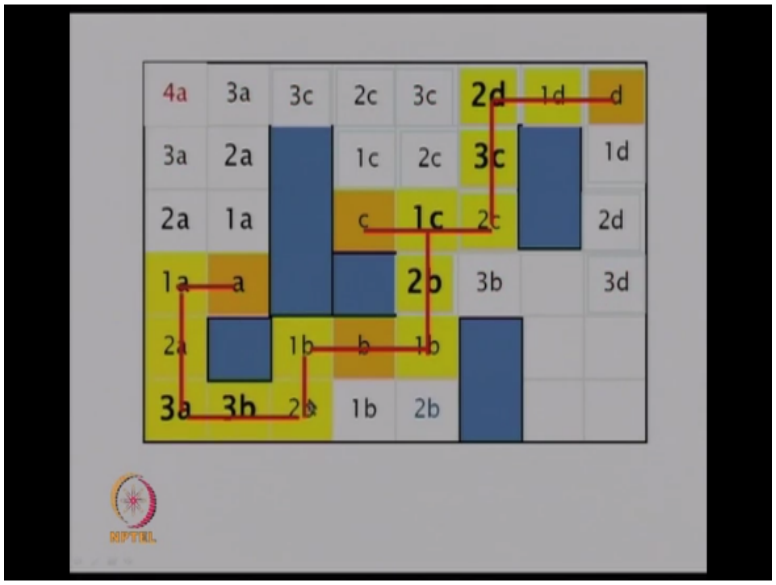
(Refer Slide Time: 48:10)



The wavefront of C would have reached here and when we start growing the wavefront of D, we released that this is already adjacent to the wavefront of C and that means between D and C a good connection has been found out, that means a good set of sequence of grid cells would have been identified for the purpose of detailed routing which is to be done later. Now still we have, so far managed to connect B, C and D.

A is still remaining but now looking at the wavefront, we realize that in the very next step, it is going to be evident that evident that you know when we try to grow the wavefront of A from 3A to this, it would realize that it is already touching the wavefront of B, so connection between A and B would also be found out in the next step, that would be this.

(Refer Slide Time: 49:01)



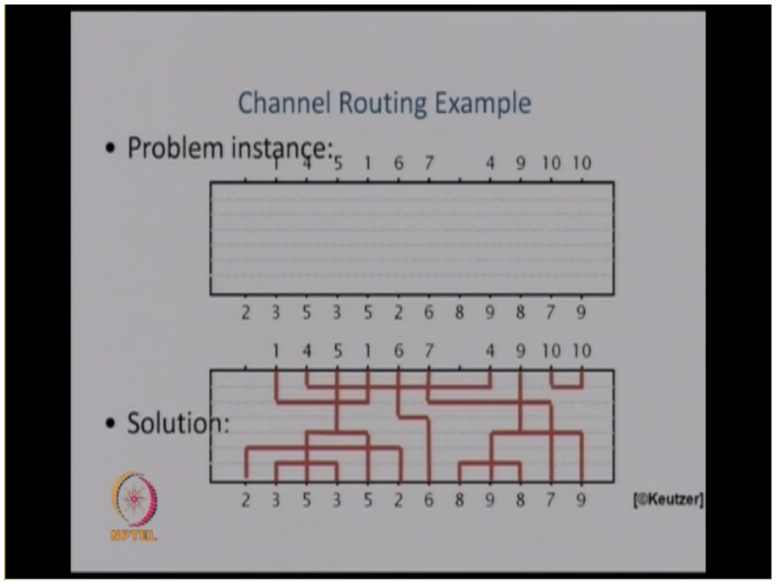
So, this will complete an approximately good. This problem of finding such so-called Steiner tree connecting a set of terminals is again an NP hard problem, even in the rectilinear Manhattan kind of setup. So, one does not expect optimal answers for large problems. So, this can regarded as one clever heuristic which is very easy to implement on obvious data structures and also fairly parallelable, has some benefits in terms of elegance and simplicity.

So, such a heuristic is going to give you some decent quality multi-terminal Steiner tree layout of a net and the variations of these ideas. In fact, what I shown here is not the basic idea that was due to Lee and Moore called Maze routing idea which was essentially for two terminal like nets where a simplified assumption was made that every net is just having two terminals, a source and a target, and this wave propagation idea was used to identify the shortest path through this grid cell from source to the target.

Then, some extensions of this idea where made for multi-terminal nets. So, just to indicate that various similar ideas are possible, I have given you outline of one idea that is possible. I am not giving any pseudo code for that. It would be an interesting exercise to correct and capture it as a well-defined heuristic and how the bookkeeping is precisely done and when exactly the labels are updated.

When does one realize that wavefront are touching and it is time to capture that information and record it as a good connection and when to stop and so on. All that would be an interesting exercise to figure out and convert this into a pseudo code and then into a rigorous implementation and maybe parallelisation. So, this can be regarded as a simple example of global routing, like identifying the grid cell regions through which a multi-terminal net should be laid out.

(Refer Slide Time: 51:30)



Channel Routing Example

- Problem instance:
- Solution:

Then, at a detail routing, one would have several such problems where we have different channels and for each channel we know where the net enters through a pin and channels are those regions where the pins are on either side, the longer side, opposite sides, and we have specific specification of which pin is to be connected to which one. It could be multi-terminal pin or could be two terminal pins.

And within a channel we have some number of horizontal tracks and you have to figure out exact horizontal track that should be used for the layout and corresponding vertical connections from the pins to that track. One of the natural optimization pattern would be to use minimum number of tracks, so at the time of compaction of the layout, this channel can be reduced in size.

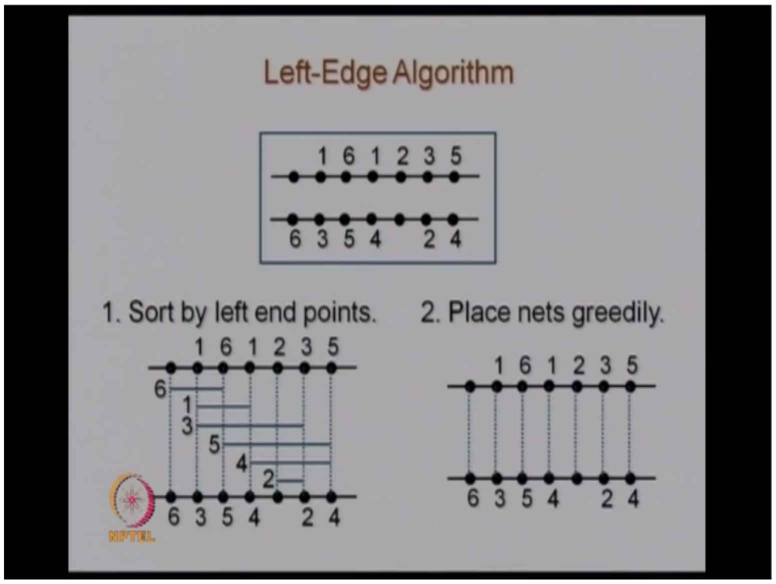
In the beginning with some pessimism, we would have allowed a big enough channel with sufficient many tracks and if we optimize well, the number of tracks used would be as few as possible, would help us string this channel into a narrower channel at the time of compaction which would improve the size or the area requirement of that particular layout.

So, here is an example where we have a channel with the pins marked 1, 2, 3, 4 and so on. Some of the pins are not marked indicating that those pins are not to be used. There is no net being connected to this pins. You see that sometimes net is connecting pins on the same side 1 and 1. There is no pin number 1 on the other side. Sometimes net connects something like say pin number 5 to pin number 5 on the other side, as well as there is another pin number 5.

So, this is a multi-terminal net connecting this pin with these two pins number 5, so you can have different types of nets and a solution would look like this which would make use of horizontal tracks like this, which are the main trunks of this the net layouts and vertical so-called branches and so on. So, this net has this particular trunk on this horizontal track and connected using this vertical branch and this vertical branch.

So, there will be a layer of horizontal tracks and they will be maybe one or two layers of vertical branches; why more, that would become clear because sometimes the nets can have overlapping positions and the vertical branches might overlap; to have them separately laid out, we might require more than one vertical layer. So, channel routing is also very, very investigated problem and one of the very basic idea like for starters is the so-called Left Edge Algorithm.

(Refer Slide Time: 54:32)



Left-Edge Algorithm

1. Sort by left end points.

2. Place nets greedily.

So, this is a very simple, very Greedy algorithm which is where one begins the study of such algorithm because it is based on simple Greedy algorithm strategy which can be shown to be optimal in the absence of certain constraints, in case of very highly simplified assumptions. For example, if we assume that all the nets are just two terminal nets.

So, net 1 is connecting this pin with this and nothing else. Net 2 is connecting this pin with this one. so all the nets here are just two terminal nets and we also assume that we have plenty of vertical layers, plenty means two or three what that should suffice. So, it is just a matter of then that because we do not have any restriction in number of vertical layers for fixing the vertical branches. The problem really boils down to like assigning the horizontal trunks to appropriate tracks

So, net number 6 should have the scope from this left end point to this right end point, because this is the leftmost pin of net number 6 and this the rightmost pin of net number six. So, net 6 has to be laid out in this zone, maybe on this track or some other track. Net number 1 will start at this X coordinate and will go up to this. It need not be on this track. This track may not be the best choice for net number 1 one and so on. So, what this algorithm does is sorts the nets by their left end points.

So, here net number 6 will be the first to be considered because it is left end point is the earliest. Then, net number 3 or 1 will be considered next because their left end point is the next earliest and so on. So, now we look at them one by one. So, net number 6. So, that is the first one in the sorted list because it starts at the earliest leftmost end point.

So, this is laid out on the first available horizontal track and obviously connection is made with the help of one vertical metal wire here and one of the vertical wires available from here to here. Wire will be placed over here connecting the vertical segment with this horizontal. After 6, we will choose 3 or 1. Let say 1. 1 will be connected over here, then on this horizontal track and then connected again by vertical metal wire from here to here and connected wire will be here as well as here.

So, the horizontal scope of this 1 is from here to here. Just notice that the track for 1 has been, we have to choose a new horizontal track for 1. We could not a layout 1 on the same horizontal track as net number 6 because there would be an overlap and we are assuming that we have only one horizontal lap but maybe two or more vertical layers. So, we cannot have the layout of net number 1 and net number 6 on the same horizontal track because of the obvious overlap.

So, net number 1 has to be opened up on a new track, but similar considerations has to be put on a new track, next track, track number 3. So, we seem to be using one track in every such case. Let us look at the next one, number 5, that is next one in the sorted order. Again we have to open up a new track but now when you look at net number 4 which is next, we realize that its scope is from pin number 4 to this pin number 4 and you see that it can be accommodated on the first track itself.

Because the first track now is empty, the net 6 has finished its scope and net 4 can now start. So, you see next that net number 4 can be laid out on one of the earlier utilized tracks, specifically track number 1 itself. So, this is how we are now hopefully the algorithm is optimizing as compared to this picture where we have used one track one per net. We will be reusing some of the tracks for more than one nets.

After that, net number 2 again can be accommodated since its scope is from here to here. This horizontal scope is just this much. It can be fitted on the second track because net number 1 which was laid out on that track is now over, so 2. So, that is over. So, we have laid out all six horizontal tracks of the six nets using only four tracks as compared to using six tracks over here. If you had been less careful, we would have used six tracks or five tracks.

But this can be shown to be optimum solution because of assumptions that we are making because this particular example has only two terminal nets and I assume that there would not be any problem in laying out the vertical wires. They will not overlap. If they overlap, we can use different layer different vertical layer for the vertical branches. In the absence of so-called vertical constraints, this algorithm will give you optimal result for a set of two terminal networks which are to be laid out in a channel.

It is more sophisticated variations or more practical enhancements, there is something called Greedy Channel Router (GCR), there is Dogleg router. Various channel routing algorithms have been investigated and most of the standard text will give you good treatment of this algorithm. So, the purpose here was just a very sketchy, simplistic overview and in next couple of lectures, we will address the other algorithms for placement on partitioning and time, so I stop here.