

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

Lecture No #28
Backend Design: Part – XIV

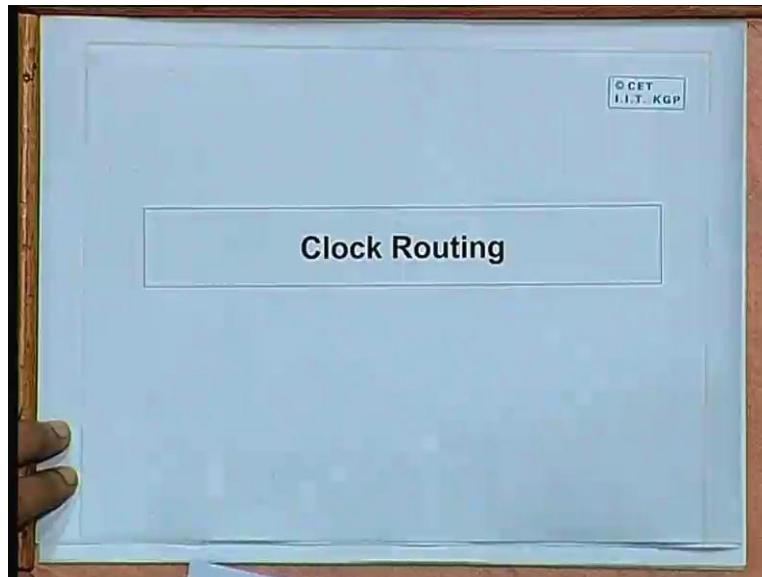
We will be talking about the problem of routing some special kind of nets. Namely we will be talking about two such net one is clock and the other is the power.

(Refer Slide Time: 01:07)

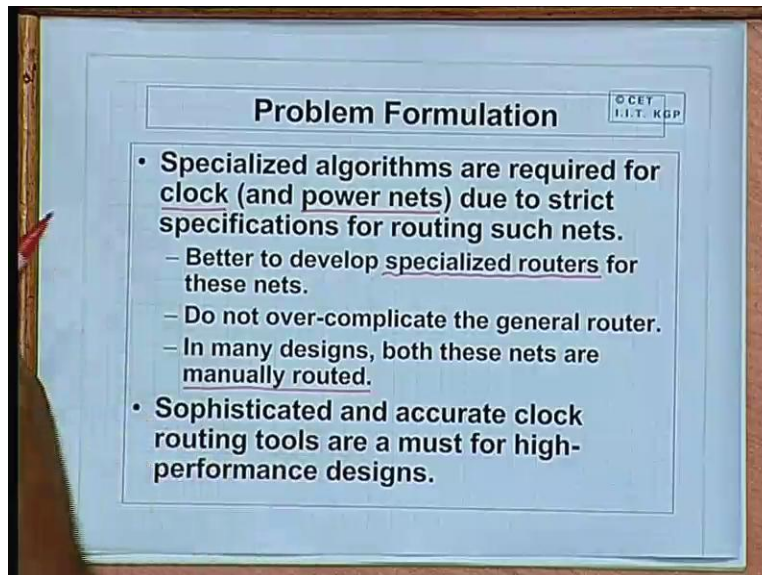


Now this clock and power nets are very important in terms of the overall performance of a system. So the issues of routing these nets are taken up separately because there are some stringent requirements for this nets which need to be satisfied. For a clock net there are some issues like clocks skew which need to be addressed. And for the power and ground ness nets that the issue like the current carrying capacity they need to be addressed and also the parasitic the impedance. So we start with talking about clock routing. So what are the issues involved and what are the different techniques which people have proposed?

(Refer Slide Time: 01:52)



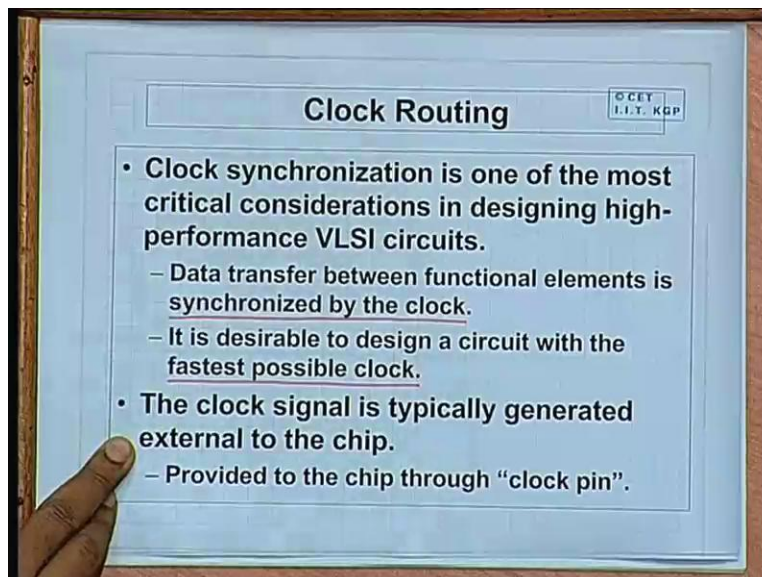
(Refer Slide Time: 01:53)



Now in clock routing in terms of the problem formulation just I am repeating. So we need special algorithms to route this nets. Well we are talking about power we shall subset clock. We shall subsequently be talking about the power nets as well. They are important and they need specialized algorithms because there are some strict specifications which are often different as

compared to conventional signal nets. So for this reason for these kinds of nets people have developed special purpose routers. So as to meet their individual requirements now the development of this special purpose routers will also simplify the general router because if we want to incorporate all the constraints in the single router. Then the router algorithm may become quite complex. Typically these two nets are routed separately this is done either manually or through a semi-automated power process. Normally clock and ground they are not entirely left to the automated router to complete and interconnect. Particularly for high performance design an accurate clock routing tool is essential.

(Refer Slide Time: 03:14)



So talking about clock routing well as you know clock synchronization is very important in VLSI circuits. Because in a typical VLSI circuits which work in a synchronous fashion, so all the data transfer and operations in the data path they are carried out in synchronism with the clock they are synchronized with respect to the clock. So if you can have a design with a fast clock well obviously the performance of the circuit will improve. So our aim is to have a design with the fastest possible clock. Of course there will be some hard constraint which will be coming in with respect to the delays of the combinational network which are there in the circuit because you cannot run the clock at a frequency higher than that.

Now there is another issue which needs to be addressed because in a typical chip the clock signal is typically generated external to the chip. So either you supply a clock input from an external oscillator or there is an oscillator which is located somewhere near the periphery of the chip. You connect a crystal externally so the oscillator generates the clock signal from one corner of the chip and then it gets distributed. So the idea is that the clock signal is coming from one corner of the chip and typically it has to go to every possible place in the chip. Because depending on the requirements from the different functional blocks which we have spread out. So say externally you have this clock pin. This is what we call.

(Refer Slide Time: 04:52)

Contd.

© CET
I.I.T. KGP

- Each functional unit which needs the clock is connected to clock pin by the clock net.
- Ideally, the clock must arrive at all the functional units precisely at the same time.
- In practice, clock skew exists.
 - Maximum difference in the arrival time of a clock at two different components.
 - Forces the designer to be conservative.
 - Use a large time period between clock pulses, i.e. lower clock frequency.

Diagram: A central point on the left is connected to three points on the right. The paths are labeled Δ_1 , Δ_2 , and Δ_3 .

And each functional unit inside the chip, which needs the clock. They will have to be connected to the clock pin. So we have something called a clock net. Now a clock net is also very similar to a conventional net in the sense that it connects several pins which are located at specific coordinates. But the main difference in the clock net is that the cardinality of this net is much higher as compared to conventional nets. Typically it is not uncommon that a clock net will have to connect to thousands of pins ok. So this is one issue which need to be addressed specifically and in order that that we may clock this circuit at the fastest possible rate. Well in the ideal

situation we desire that the clock should arrive at the different functional units all at the same time. If there is a variable delay between the, you can say in the different paths.

Suppose the clock is generated here, this is the source of the clock; this can be that external pin. I am talking about and as it is getting distributed. If the delays of the different paths are different δ_1 , δ_2 , δ_3 , then the clock edges will be reaching the functional units at slightly staggered **in this** in the access of time. So if the clock edges come staggered in the access of time. This is what is called clock skew. Clock skew means due to variable delay the clock edges all do not reach the different function units. At the same time there is a variation. So if there is a clock skew what is the requirement is that you will have to estimate the worst case clock skew the delay variation which is possible an effective clock cycle time will have to be enhance by that. So your effective clock rate will go down will become slower your clock frequency will go down.

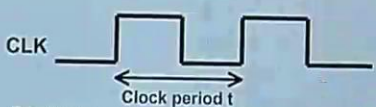
Because if you try to run the circuit at the maximum possible clock rate. There you are ignoring clock skew you are assuming that all edges come at the same time. But if you say that well there is a variation of one nanosecond between the two edges. Then you must add one nanosecond to a clock time period in order to take care of that skew. So just you must give provide for sufficient time. So that the skew may exist and still the different functional blocks may compute their outputs correctly, they are given sufficient time to do so. So if there is a skew the clock will have to be slowed down in order to take care of that. So from the designer's point of view it is essential to have a clock distribution network where the clock skew is minimized. So the main purpose of the clock network is to minimize the clock skew ok. Now talking about the various clocking schemes.

(Refer Slide Time: 08:17)

Clocking Schemes

© CET
I.I.T. KGP

- The clock is a simple pulsating signal alternating between 0 and 1.



- Digital systems use a number of clocking schemes:
 - Single-phase clocking with latches
 - Single-phase clocking with flip-flops
 - Two-phase clocking

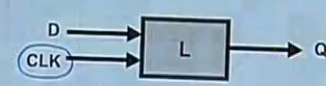
Well these things are fairly simple things. You already know this. That you can have a number of different options you can have single phase clocking with either latches or flip flops or you can have multiphase clocking typically two phases. Now if you have single phase clocking with latches the scenario is something like this.

(Refer Slide Time: 08:40)

Single-phase Clocking with Latches

© CET
I.I.T. KGP

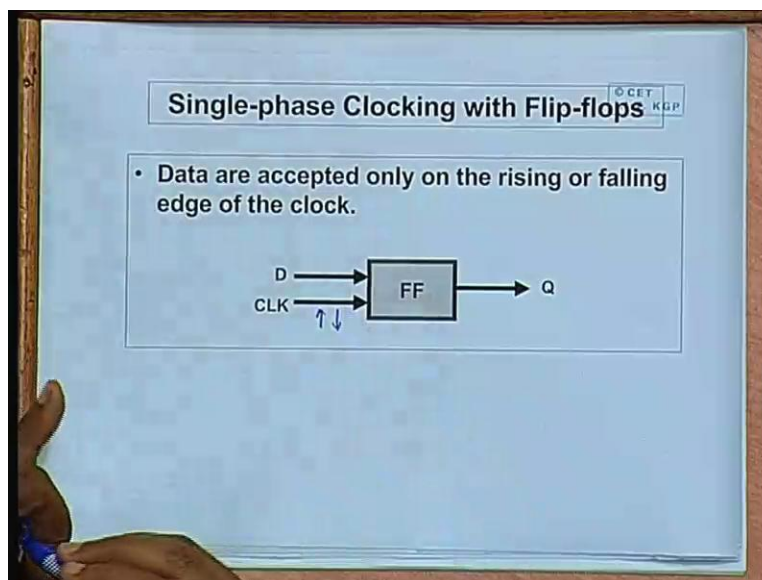
- The latch opens when the clock goes high.
- Data are accepted continuously while the clock is high.
- The latch closes when the clock goes down.
- Not commonly used due to their complicated timing requirements.
 - Some high-performance circuits use this scheme.



There is an suppose is a d latch d input this is output well all the way you are calling it a clock this is this works more like an enable. This latch opens when the clock is high the input data will get latched and the output changes continuously. Now as soon as the clock goes down the latch closes and the output becomes stable. Latches are sensitive to the level not to the edges of the clock. Normally in conventional designs we do not go for latches like this. We rather go for flip flops where clock edges trigger the data transfer they are easier to design.

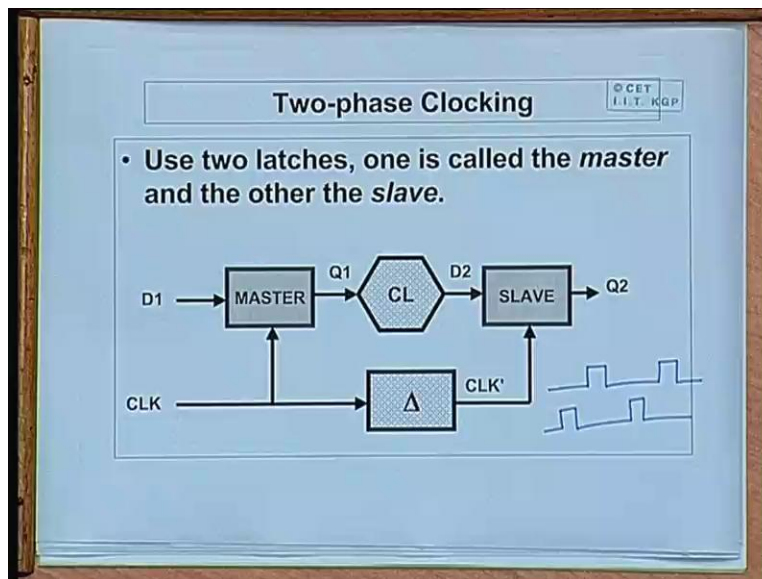
But latch based design also has its merits in particular for very high performance circuit for very high speed designs were you need some asynchronous sub circuits in our overall design. There this latch base circuits work better. Because when you have edge base clock circuits you are tied by that clock period you cannot do anything faster than the clock period. But if we have this kind of latch base circuit whenever or the or as long as the clock is high. So the any change in input will propagate to the output ok. But as I had said most of the conventional designs today they work with flip flops.

(Refer Slide Time: 10:05)



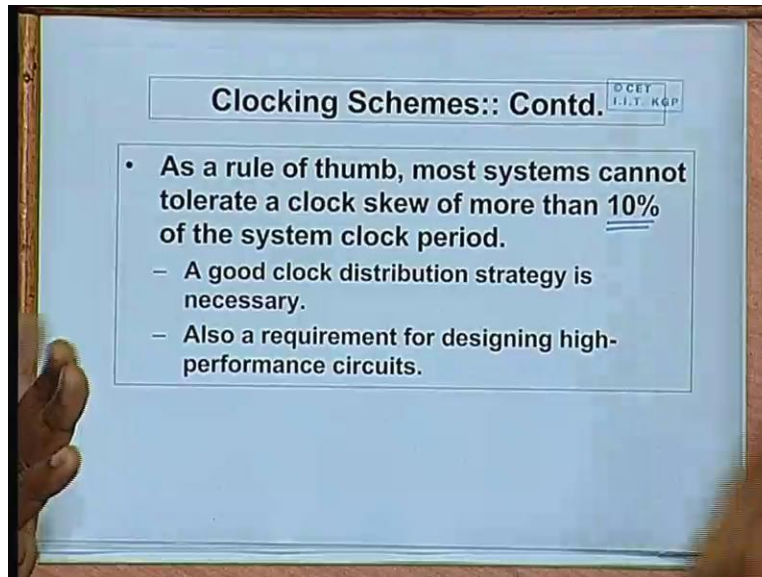
Where the flip flops is triggered either by the leading edge or the falling edge of the clock. So only at the edges the output will change state in response to the applied input. Touch if for greater flexibility and highest speed of operation also most of the complex circuits use multiphase clocks.

(Refer Slide Time: 10:31)



Well a two phase clocking is very typical because most of the flip flops which we use in circuits have a master slave kind of a structure which are often clocked with two different non overlapping clocks. Well one can be generate by inverting the other or this clock and clock bar can be two separate clock inputs which we generate. These are non-overlapping these are not necessarily complement of one another they are in general non overlapping one maybe like this and the other maybe like this. Ok. So I am not going into details of this all these things you know. So there whether irrespective of whether it is a single phase or two phase clocking the issue of clock skew remains equally important. So if we can minimize clock skew we can run the circuit at a higher speed ok fine.

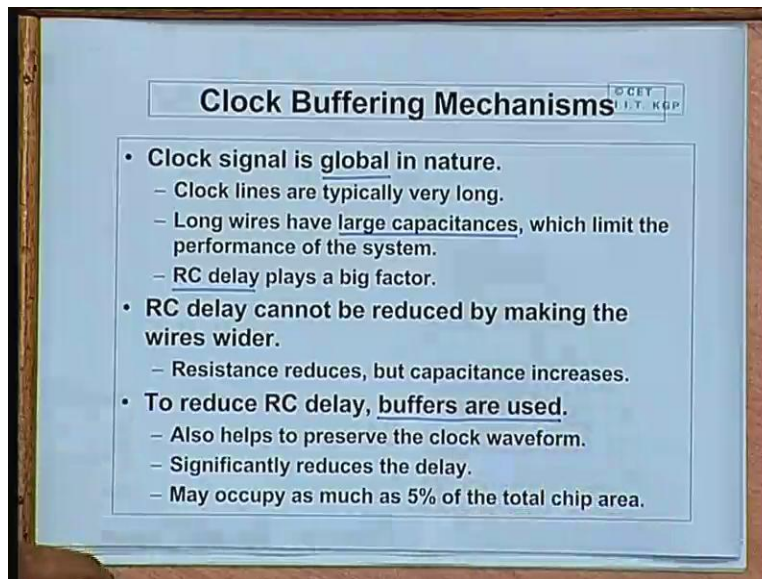
(Refer Slide Time: 11:33)



Now as the rule of thumb most of the practical systems, well 10 percent is a rule of thumb which you can say that this is the maximum clock skew I can tolerate. Well if my ideal clock frequency is f , well I can allow the clock frequency to go down by 10 percent but not more than that. Well if I say that it is going down by more than ten percent then my clock distribution network has some problem. So in order to achieve this we need a good layout of the clock network which is called the clock distribution strategy. Of course in conventional system this is important and in high performance circuit this is an absolute mandatory requirement. Because in case of high performance circuit a bad clock distribution circuit can destroy the very purpose of to design ok.

So clock distribution circuit is very important. Now the another thing I am mentioning that since the clock distribution is so important and we want to have minimum skew in the distribution it is often the case that the entire clock network is routed on a single layer. Because as you switch from one layer to the other through the wire connection, there are some additional parasitic effects which will enquire additional resistive and capacitive loads. So there can be some kind of variation in those loads and are the delay can again go on changing over different paths. So you prefer to layout the entire clock net in a planar fashion in a single layer ok fine.

(Refer Slide Time: 13:25)

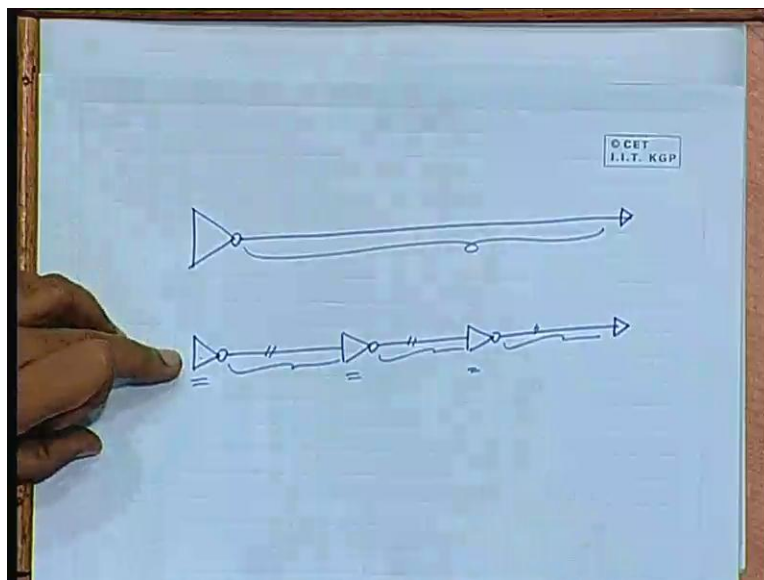


Now typically we use clock buffering because as I had said the number of destinations of the clock network can be many it can be of the order thousands. So a single source cannot feed the clock net to 1000 destinations. We need some good buffering mechanism to provide the required drive current drive typically. And this clock signal is global in nature. Global in the sense that it is a signal which is common to the whole chip so it should go to all corners of the chip. Since it has to go to all corners of the chip clock lines are typically very long and as we have long wires they will have larger capacitances and also larger impedances resistance.

So RC delay of this clock nets will play a big factor. So we will have to take care of the RC delay to find out that means how much delays encountered in the different you can say segments of the clock network. Well it does not matter that much if a clock signal is delayed with respect to what is entering the chip and what you are sending to a function unit. But it will suffice if I can ensure that the delays of all the paths are approximately equal. Maybe I am applying it to the input of the chip at time t_1 there all reaching the function units at a time t_2 . T_2 minus t_1 maybe large but I do not care about that. I am happy if the clock reaches all the functional units equally delayed in time.

Well equally delayed means there is no skew ok fine. And what it shows that RC delay you cannot reduce by simply making the wires wider because if you make the wires wider R will decrease, but C will increase. So by making the wires thinner or thicker you really cannot change RC delay appreciably to reduce the RC delay buffers are used this is one. This is one point I also mentioned earlier. Because in an interconnecting line connecting two points well there are distributed resistive and capacitive loads. So you can treat this as a something similar to a transmission line and it can be shown that the total delay in that line is proportional to the square of the distance ok. So if you can break the line into smaller segments and insert buffers in between then the total delay can decrease.

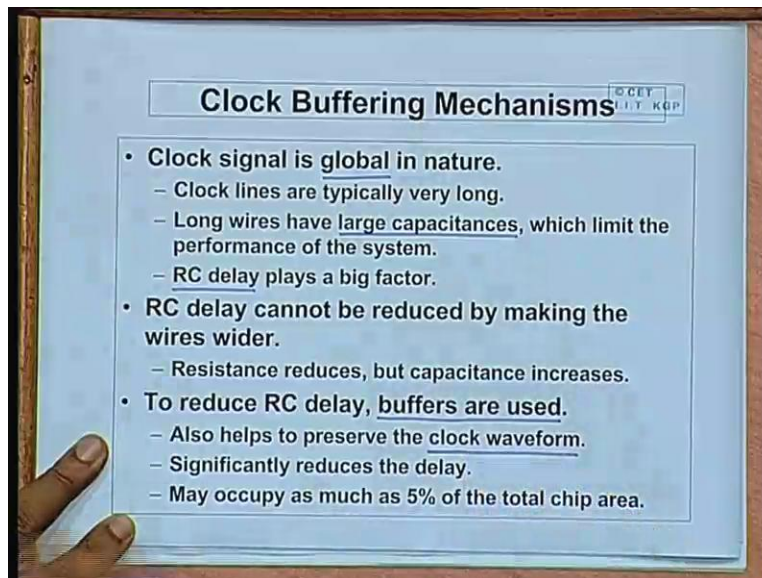
(Refer Slide Time: 16:18)



What I am saying is that rather than using a single buffer, suppose this is a buffer an inverter acting as a buffer distributing it over a long distance to a destination here. If you use a relatively smaller buffer break this line up into several segments and then distribute it then the two things. Since this buffers are small their delays will be less since this line segments are small some of their delays, will be much less then the delay of the over line because the delay is proportional to the square of the distance. So if I break it up in this way the delay in this partition buffer network

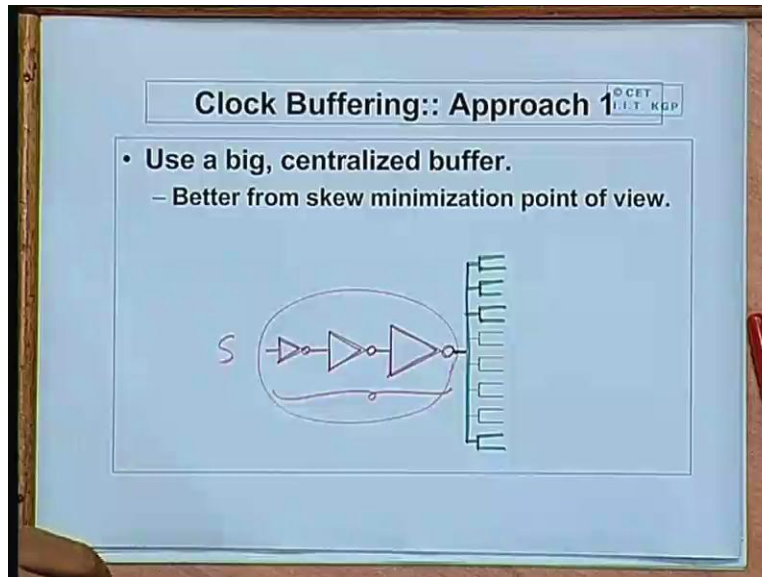
will be less in general as compared to a network without any distributed buffers ok. [Students Noise: 17:16] Well for the time being I am assuming that this is the generator and this is the destination I will come to skew yes this will also take care of skew.

(Refer Slide Time: 17:34)



So buffers need to be used buffers have two purposes one is of course the direction of delay as I had mentioned and second is that it will also preserve the clock waveform. Because as it traverses over a distance the rising edge and falling edge gets degraded. So if we use buffers in between the edges get again sharp. So you again regenerate the clock signal so signal degradation is also controlled. And in fact these buffers alone may occupy as much as five percent of the total chip area. So this is not something which we should underestimate. For distributing the clock network clock network was the big network. I told you; you must insert buffer in crucial places are all through out and the total area occupied by buffer can be pretty large. Now there are two broad approaches to clock buffering I am talking about these. The first approach is something which uses a centralized buffer.

(Refer Slide Time: 18:41)



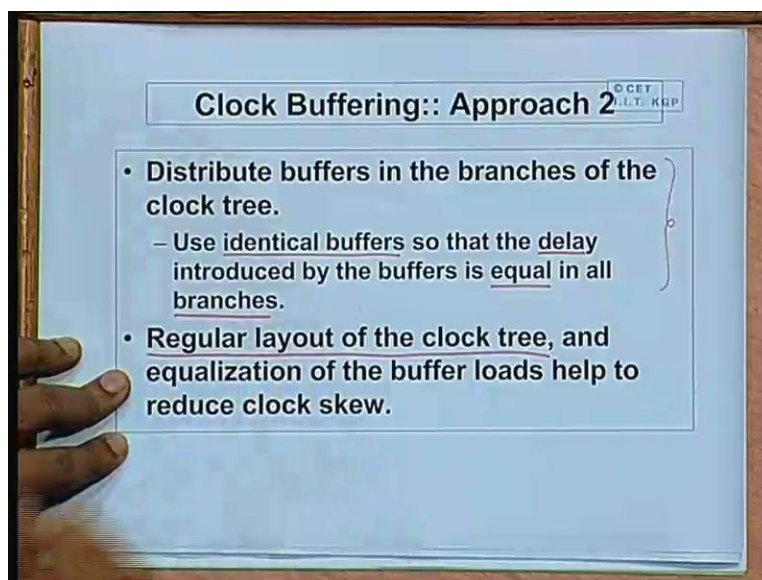
So here you have a central buffer which has a very high current driving capacity. Typically when you need a buffer with a very high current driving capacity instead of designing it as a cascade of several buffers with the size increasing slowly, that is how I have shown here. This is a central buffer which is near the source of the clock. This is the source. Now once you have generated this clock signal with sufficient driving strength through this buffer, you can distribute it in some kind of a tree network. Now you must layout this tree network in a suitable way so that the distance of each segment of the tree up to the leaves is equal from this point.

This is one approach; you use a big buffer and a passive interconnection network. But of course one problem with the passive interconnection network is that, if these lines are pretty long, then again the signal degradation will occur. Well, if we are careful with the total length of these lines, skew can be controlled. But signal degradation is there if the rise time and fall time degrade significantly. Then also you have to keep some tolerance in the clock period. So something similar to clock skew also occurs, so the speed falls. [Students Noise: 0:27] Yes. [Students Noise: 20:29] No. The idea is not that. See this is a standard method of designing means high current driving capability.

Suppose you have a requirement to drive a very high capacity. Ultimately in terms of VLSI chip these are all typically CMOS gates which need to be driven. So from this point you can look at it as driving a very high capacity. Now, it can be shown mathematically very easily that instead of using a very big buffer to provide the required current driving capa. See big buffer means the resistance value is common is small so RC time constant is within limit. So if you progressively increase the sizes of the buffers and connect them in cascade each buffer is f is bigger than the previous buffer by a factor of f . Then it can be proved that the total current driving capability and the total means RC time delay gets controlled to a great extent.

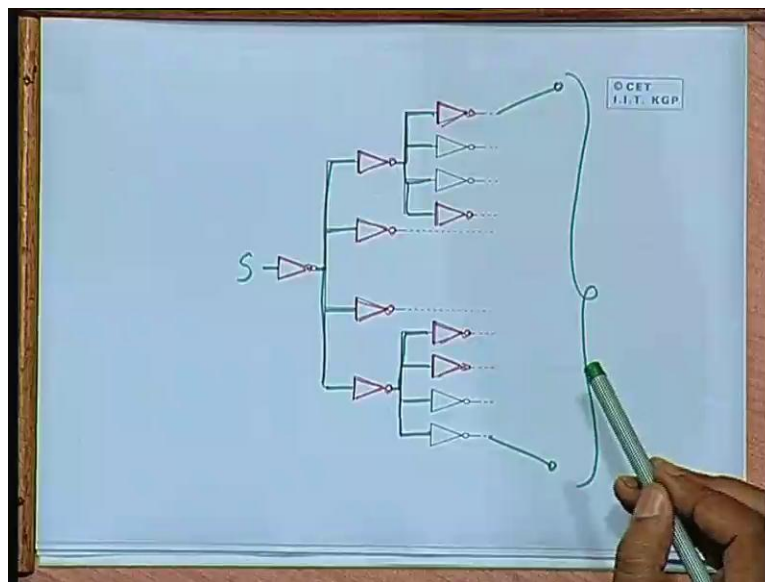
There is a formula there is a way to calculate how many buffers you need in that cascade and what is the sizing. This is a standard VLSI technique I am not going into the detail in this class. But if you are interested to know about this you can look at any standard book on VLSI design you will get the design of current driving buffers why these are connected in cascade with increase in sizes ok fine. So here we use a single buffer big buffer. That is why we need this kind of a cascade, a very high current drive capability where the other approach says that well we distribute buffer in the branches of the clock tree.

(Refer Slide Time: 22:24)



Do not use a single buffer. Now here it says that you use all identical buffer, there will be many buffers distributed all through out the clock tree and all of them will be identical in terms of the sizes right and if we use identical buffers. And if the tree is symmetrical with respect to the length of the segments up to the leaf nodes then the delays introduced in all the branches will be equal. Well this demands some kind of a regular layout of the clock tree. This we shall be looking at very shortly that how the clock tree layout can be made regular. But for the time being, let us see that how this buffer distribution can be done over the clock tree. Well this again you can conceptually look at it like this.

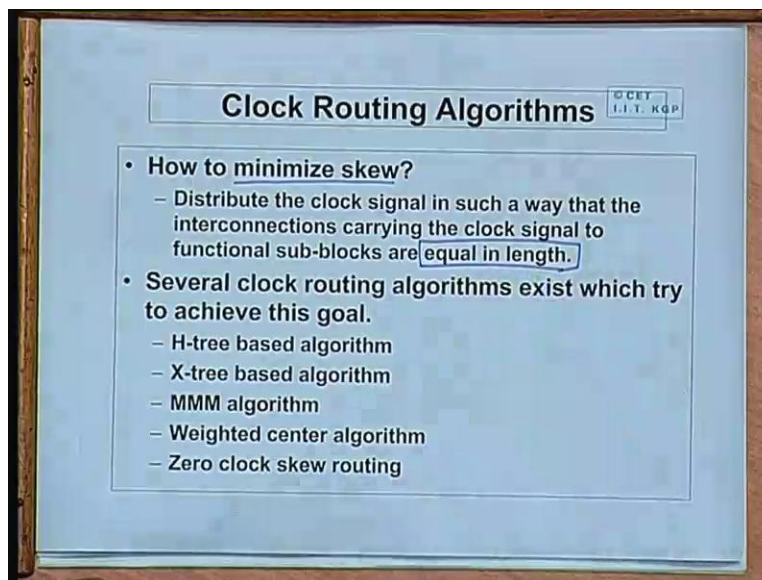
(Refer Slide Time: 23:25)



These are the buffers these are all distributed on the different segments of the tree and so on and from the clock source you distribute them like this. Now the buffers are called identical. Because for example, in this example I have shown the fan out of each of the buffers is four in the first stage as well as in the second stage, as well as it will be in the third stage. So the final buffer also will be driving four clock pins. So if you can ensure a symmetrical network like this. Then ultimately the different clock pins which will be at the leaf level they will be encountering equal delays from the source right. [Students Noise: 24:30] They will have to be equidistance. Yes, see

this is a conceptual diagram I have drawn. But now we shall see exactly in terms of the geometrical coordinates that how this can be laid out. This is just a conceptual picture I am saying that all segments of trees are equal in length then the delays will be equal. So now we would be looking at the actual clock routing algorithms which will talk about how to layout these wires on the surface of the silicon.

(Refer Slide Time: 25:03)

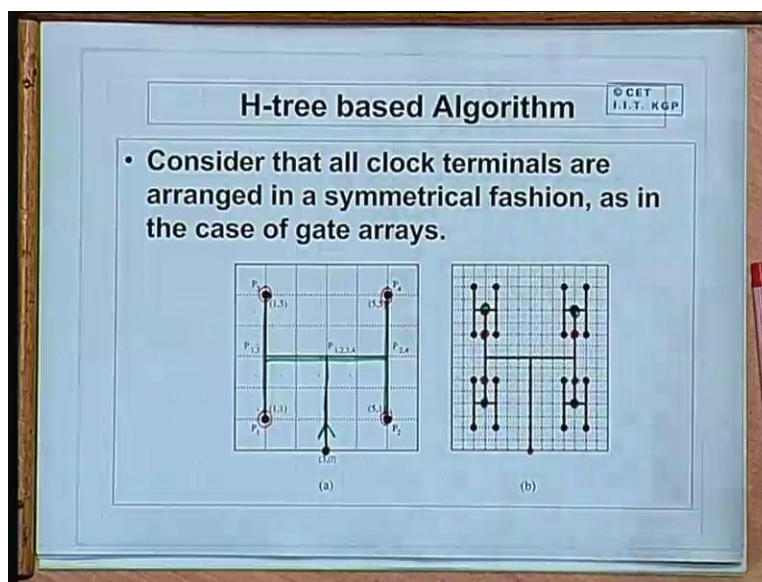


So clock routing algorithms of course the main issue here is to minimize this skew. So the clock routing algorithms that have been proposed they talk about distributing the clock signal in such a way that the interconnecting paths from the source to the final destination or the leaf nodes in that tree they are equal in length. So we have to ensure somehow that from the source of the clock to the destinations where you are connecting finally to the functional pins they are all eq. Well length is not the correct term I should say they are equal in delay the delay from that point to all those pins should be equal. But as the first approximation we can say length and delay are propositional.

So we can also consider length as a measure. Now there are a number of algorithms which have been proposed. Well some are general which can be applied to general designs. Some are special

purpose which is not applicable to general designs but applicable to special cases only. So we would be quickly looking at some of these algorithms. H-Tree, X-Tree method of means and medians weighted center zero clock skew these are some of the method which have been proposed. Now this H-Tree based algorithm is a very interesting and you can say this is a way in which you can layout the clock net in a very regular way. See first we will show how the clock net is distributed the buffers you can introduce later.

(Refer Slide Time: 27:00)



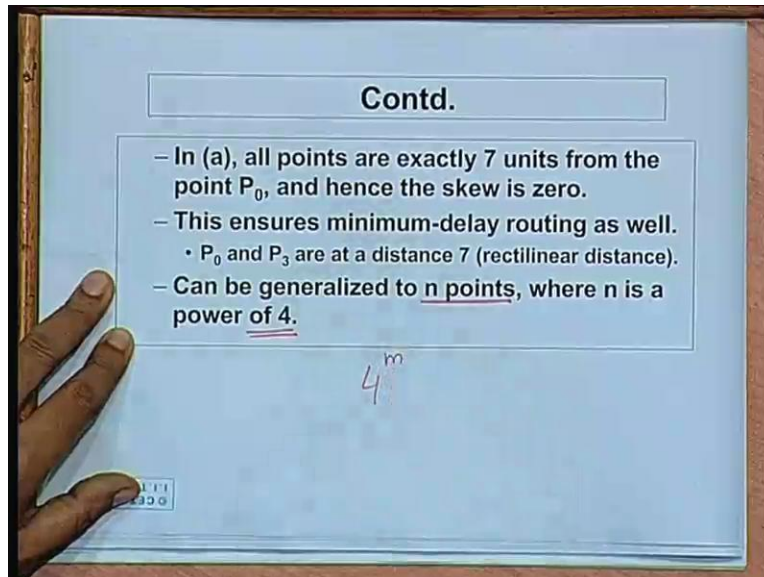
So the H-Tree algorithm is simple in concept. The tree can be constructed recursively. The tree is constructed recursively at every stage of the iteration you have a segment of the tree which looks like an H. That is why this is called H-Tree based algorithm. Now you assume that you have an H like this and your clock is being fed to the middle of this H. This is the source of the clock and these are the four points where you are finally producing the clock. These are the destinations you can say in the first level of iteration these are the four clock destinations. Now if you lay this out in a regular way, we will see in terms of the grid lines 1, 2, 3, 4, 1, 2, 3, 4, this is laid out as a 4 by 4 grid.

So now you see with respect to the source each of the destinations are at a distance of 1, 2, 3, 4, 5, 6, 7, 1, 2, 3, 4, 5, 6, 7. So all the four corners of the h they are at the same distance from the source. This is the first level of iteration. Now the second level; level of iteration says, well now we have been successful in generating the clock in these four points. Now from these four points these are the four points. Now you treat these four points as the well in fact not this and actually these points are this. So these will be treated as this source of the clock signal for the next level of iteration. So the next level of iteration, you have four smaller H'es in these four corners.

One h here where this is the source, there is another h. This is the source another h this is the source another h this is the source. So now from four we now have 16 destinations. If you count from the master source the distance to each of the destinations are still the same. Because from here the distances to these were same and from these points to the four new point the distance will again be the same. But of course the resolution of the grid we are increasing at every step of the iteration we are increasing by 4 times. So the next level iteration we go from 16 to 64. In the next level from 64 to it will go to 256. So this is the way you can systematically build up the clock tree.

Now suppose after 4 and 5, 4 or 5 iterations you will have sufficient number of clock points which should be distributed regularly throughout the chip. Because if you go on carrying out this iteration these points will be slowly and uniformly getting distributed all though out the chips. So in so in all parts of the chip you will be getting some clock pin which is very nearby right. Now in terms of the buffer, well if we have a clock tree you can introduce a buffer at the center so you can do it uniformly. So you can put buffers in the centers of this H'es. So in such now buffer is also very regular here once we have the layout of clock tree. So this mechanism well.

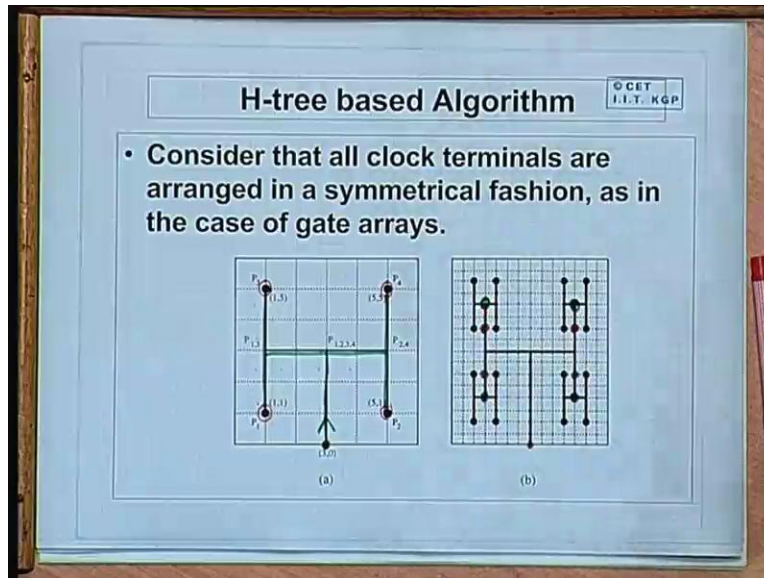
(Refer Slide Time: 31:24)



This I have said the distances are same this can be generalized to have n points where you can have the clock where n is some power of four. Because at every step from each point you are generating four new points. So it will be some power of 4 this m is the number of iterations you are going through depending on the number of iterations the number of clock points will be 4 to the power that. Now there is one issue in these method well this method is apparently very regular it has a very nice geometrical property so that all through out the chip you can get the clock pins. But in a practical design do you need the clock pins uniformly distributed through out the chips usually no. You have such a requirement for FPGAs or gate arrays where the cells or pre placed.

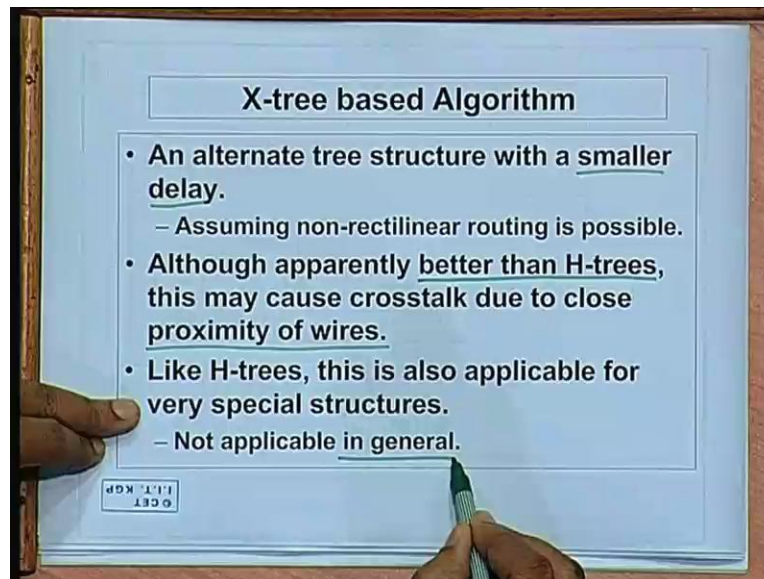
And they are pre placed in very uniform locations and each cell in the FPGA typically also has a clock input. So you need to distribute the clock to uniform points throughout the chip. So far those kinds of environment this H-Tree method can prove beneficial. But for general standard cell or custom cell based designs there this this H-Tree may not prove too much beneficial. Because from the final destination you will find that the point you will have to draw the clock that can be at some distance. So you may have to draw a line from there more over some of the points maybe wasted you do not need them ok. [Students Noise: 33:04] minimum delay well.

(Refer Slide Time: 33:17)



You see with respect to this this is the source of the clock this is these are the destinations. The way we have calculate the destination these are based on rectilinear distances Manhattan distances. So if we assume that the clock routing net will compose will be consisting of only horizontal and vertical segments of the lines. This is the shortest distance because these are all Manhattan distances. Say from this to the furthest pin you cannot have any net which is smaller than this [Students Noise: 33:50]. Coming back you can you could have a shorter one but in order to have that equal length limitation we have laid it out like this ok. Now normally since the clock net is laid out on a separate layer as I had said, so there need not be an hardened fast restriction that the lines should be horizontal and vertical always. So there is a variation of the H-Tree based algorithm this is called x tree.

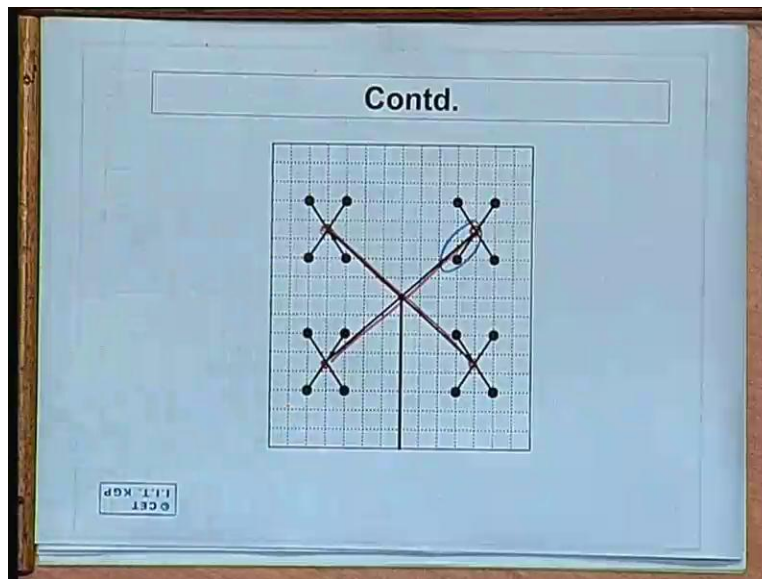
(Refer Slide Time: 34:21)



This is very similar in concept here instead of that H kind of a structure growing recursively we have an x shaped structure. Now X shaped structure means the lines can run at an angle of 45 degrees. So any line running 45 degrees will join two points which will be smaller than the Manhattan distance. So typically in this kind of an X-Tree based algorithm the delay will be smaller. Delay means the delay of the clock signal from the source to the destination. Now in the sense this can prove to be better than h I mean example we will show sure diagram this can be better than H-Tree in terms of delays. But one problem arises because in the H-Tree because of horizontal and vertical segments the line layout are very uniform and some minimum spacing can be ensured.

But in x shaped wires if you go on recursively generating that kind of a structure you will see that in some corners some of the wires may come very close together and in some other corner wires are not that much close together. So the capacity of coupling of the wires may not be uniform all through out this is one drawback of this X-Tree based. An X-Tree based algorithm is also suitable for distributing the clock signals regularly. So it is applicable to structures like FPGAs or gate arrays not applicable in general ASIC's ok. So let us look at an X-Tree structure.

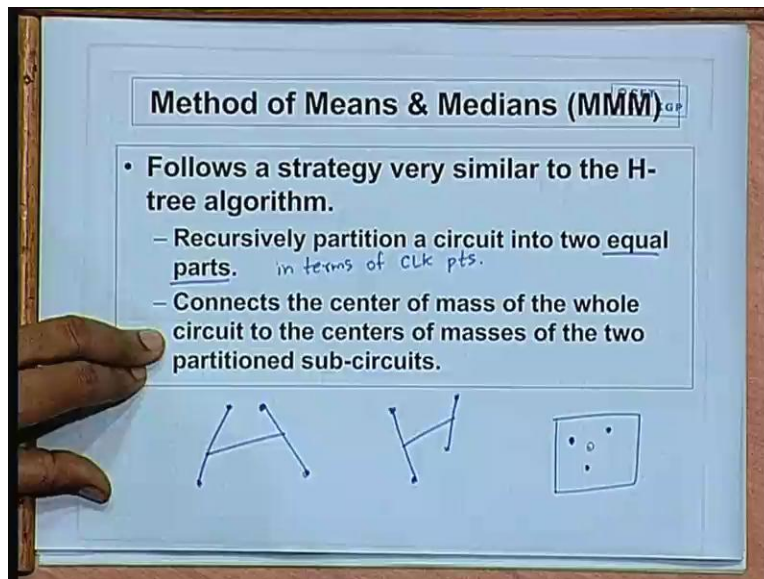
(Refer Slide Time: 35:59)



Well I have shown you the structure at the end of iteration two. So at the first iteration this was the x switches constructed, this is an x which connects these 4 points and the clock source was the center of the x. So in the next level iteration from this center you generate another x, another x, another x, another x, this will go on extending. Now as I had said that as you do this there are some pairs of well like here these two lines are so close together. But the other lines are not close together. So there will be unequal coupling between these pairs of lines. So the delays of the different segments may still slightly different ok.

So this h and x are very similar in terms of the layout but only the wires here are running in a non rectilinear fashion they are running at angle of 45 and one 35 degrees ok fine. [Students Noise: 37:14] Single layer for the clock yes normally all the clock routing algorithms they route the clock on a single layer. But now let us look at a scheme which is based on a non-regular clock distribution requirement. Well you have an ASIC. It can be a standard cell, it can be **it can be um it can be** a custom design that the clock net points maybe distributed across the chip there one method which you can use is called the method of means and medians.

(Refer Slide Time: 37:53)

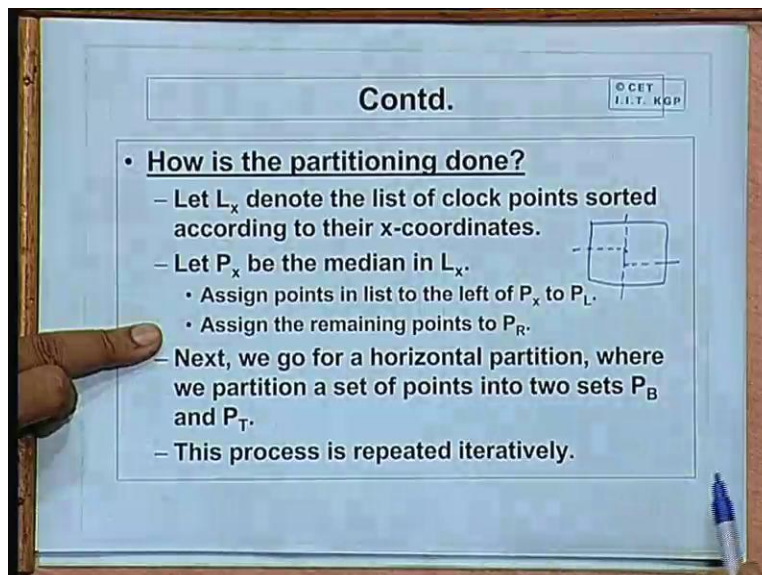


This is called MMM in short. Now in concept it follows a strategy very similar to the H-Tree algorithm. But here the issue is that the arms of the h may not all be parallel. Well you can, depending on the requirement of distance for you can have an h which looks like this or like this. So the shapes of the H's are not regular. So you are making these adjustments depending on which are the points you need to distribute the clock net to. In concept this is similar to H-Tree algorithms because your whole interconnection network will look like an H-Tree network. But with the lines or zigzag kind of lines this will not be rectilinear lines again.

Now the way it processes is that we recursively partition a circuit into two equal parts. Now this equal part says typically in terms of size. But in some cases you can also partition in terms of the number of clock points. This equal part which I am saying this equal part may also be in terms of the number of clock points you need to connect to right. Now in each of the partitions you compute the center of mass. Suppose in one partition you have a requirement that you have a clock pin here clock pin here clock pin here. Given points there are well known algorithms to compute the center of mass well here the center of mass will be somewhere here.

You treat this as the center of this partition and you connect the center of mass of the whole circuit to the center of mass of the partition sub circuits and this you go on doing recursively. This I should I have an example to show you. But first let us look at slightly more into the detail of this that how we do this partitioning. [Students Noise: 40:22] Yes, yes. They are all identical the masses of the individual points are same. Yes [Students Noise: 40:29]. They do not have weight but in some cases you may choose to assign weights to the points if some particular point is more important than the others. But normally they are equal because for the clock net in particular they are treated all equal fine.

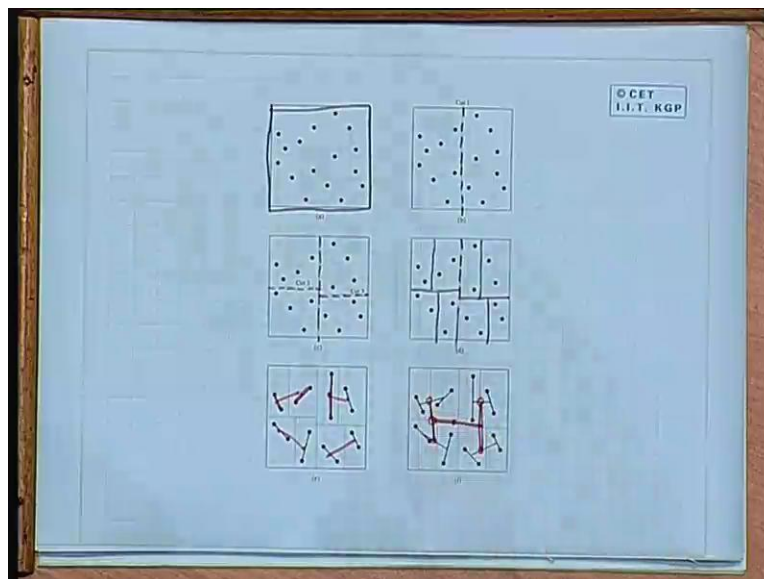
(Refer Slide Time: 40:46)



So the partitioning is normally done using a simple algorithm like this. This L_x is a list, it is a list of clock points which are sorted according to that x coordinate from left to right you sort them. P_x is the median in L_x that means that is the middle point there the number of clock points to the left and right are equal so at P_x you can do the partition. So you have the layout the clock points are all distributed you find P_x and you decide to make a partition like this. So all points through the left of P_x you call them P_L left all points to the right you call them P_R right. Yes [Students Noise: 40:31].

Yes you have the placement done you have on each block some pins where you will have to feed the clock those are the points fine. So we do a vertical partition like this then in the next step we do a horizontal partitioning. Now horizontal partitioning may not be uniform all throughout it may be like this also you partition this portion partition also this portion I have an example I will show you. So there again I call it P_B and P_T bottom and top and this why you go on recursively doing it once vertically once horizontally. So this process is repeated iteratively till the well basic block has only two points which you can connect directly. So I have an example to show you. Suppose you have an example like this.

(Refer Slide Time: 42:32)



This is the original problem where you have the clock points distributed. So you find out a cut point based on the median so the numbers of clock points are equally distributed in the left and right. Then you partition maybe the horizontal. Now this was the vertical partition now for the left partition the median will lay here for the right partition the median will lie here. They may not be in same place you continue doing this now with a vertical partition. So now we have a situation where in each partition the numbers of clock pins are two so you can directly connect them in each of the partition with a line.

Now the center of mass that I was talking about, now you can back track from this one and go on computing the back computing the center of mass and complete the tree. See now we have these individual line segments now with respect to each line segment the center of the line segment will be the center of mass. So the center of each line segment will be the corresponding center of masses. Now the last partition was by these vertical lines. So you connect the center of mass of this to the center of mass of this center of mass of this to the center of mass of this this to this and this to this.

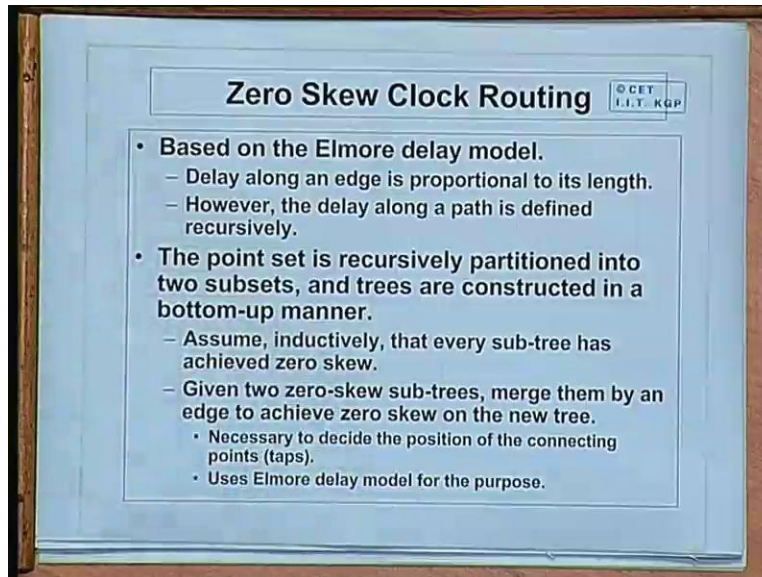
In the next step these lines using which you have connect now in this sub circuit you again compute the center of the mass. That will it can be shown that center of mass will lie one this connecting straight line somewhere on that. So in the next step you connect them like this.

[Students Noise: 44:40] It will be on the center. Yeah. It will be center it means all are identical I will be on the center of that line also. Similarly these two will be connected like this. So finally you will be having network which looks like this. Well it is an h but the h edges are all zigzag. In the H-Tree algorithm the lines are constraint to a horizontal and vertical only but here you can have any arbitrary [Students Noise: 45:03] and [Students Noise: 45:05]

Clock skew will be there but you are trying to minimize it. See here you are not making an exact estimate and try to make it equal you are trying to see that in all portions of the circuit. See this each junction of this will be having a buffer ok. There will be a buffer inserted in each of these junctions. Now if you do this, the clock skew will be less not means zero you are not trying to make it zero here. There will be clock skew unlike the H-Tree algorithm were the path lengths are equal. But here the path lengths are not equal but you are trying to make an uniform distribution all throughout the chip say how we can lay out the clock.

[Students Noise: 45:54] Cross talk is different yes that is there. But one thing you can ensure that say from the middle point of this line the distance to each of these will be approximately equal. Since you are taking the median at each step means you are taking the center of mass at each step and you are connecting like that. See exactly equal you can never do that is an ideal scenario. Yeah. But one thing people have talked about and in fact they do it for a very high performance circuit design.

(Refer Slide Time: 46:31)

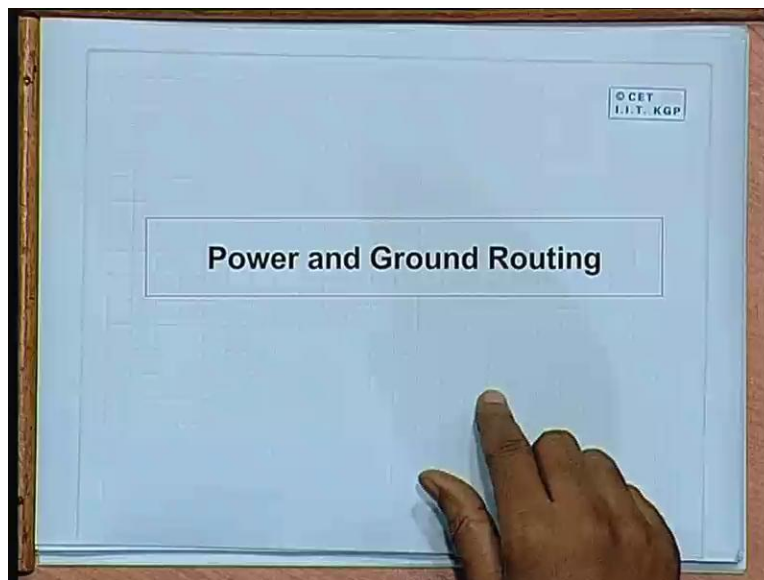


This is something called zero skew clock routing. See [Students Noise: 46:36] buffers in the net you can put them at the junctions of see in the H-Tree algorithm what you are doing. If your h was like this, then you are putting the buffers at each of the junctions. Say one here one here one here one here and finally one in the middle here. [Students Noise: 47:06] Buffers can be laid out on a single layer sing single layer layout of buffers are available ok there is one thing. I did not mention single layer layout of buffers are available. I mean the interconnection that means the input and output. Yeah. But there will have to go through the polysilicon diffusion layer because they are active components.

See but here the delays are more important in the buffer the delay should all be equal the unequal delays we are eliminating. [Students Noise: 47:41] More realistic algorithm true. True. But in some in some very high performance designs people talk about zero skew clock routing. See in order to estimate the exact delay you have to go for a very accurate model for the interconnecting lines and estimate the exact delay of that. Now one popular delay model is the Elmore delay model. So in the zeros skew clock routing what it says is that you estimate the exact RC delays of the interconnection lines by modeling them using the Elmore delay model.

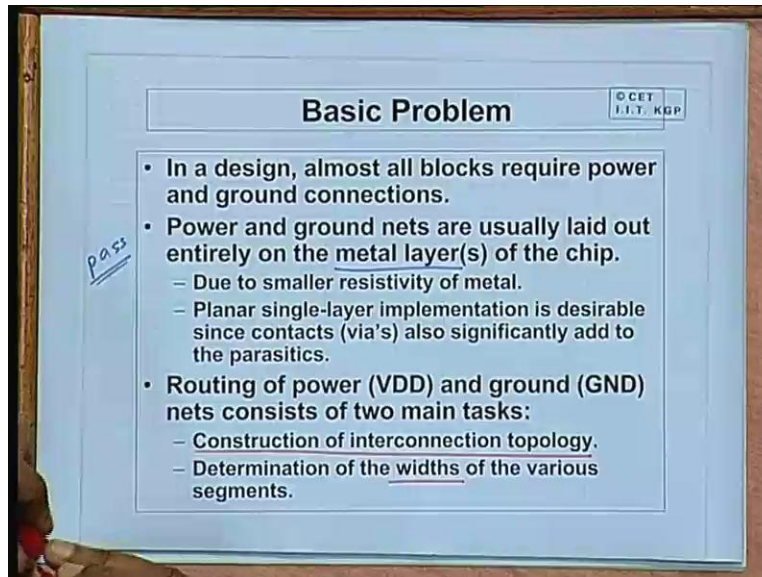
And well you can use an extension of the previous algorithm you are you were simply computing the center of the mass but center of the mass computation will not do. You will have to calculate the delays of with the individual lines and based on that you will have to take the center. Some lines are close in proximity some lines are in not so we are putting everything into the account and we have to find the zero skew clock routing point at each level of the iteration and interconnect them step wise. So if it is a complex process it requires lot of computation. So, but this is what people do for very high performance designs ok fine.

(Refer Slide Time: 49:19)



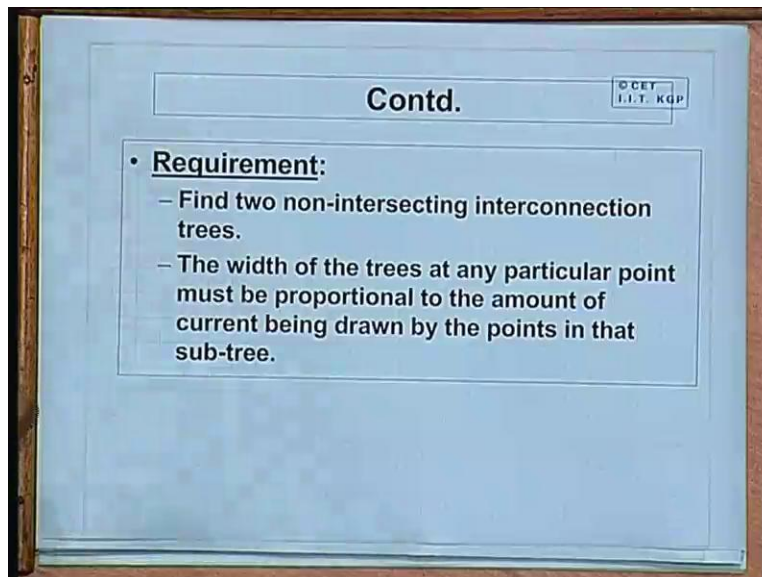
So we quickly look at the issue of the power and ground routing as well. Because power and ground routing is also important. They have to carry quiet a high current drive.

(Refer Slide Time: 49:32)



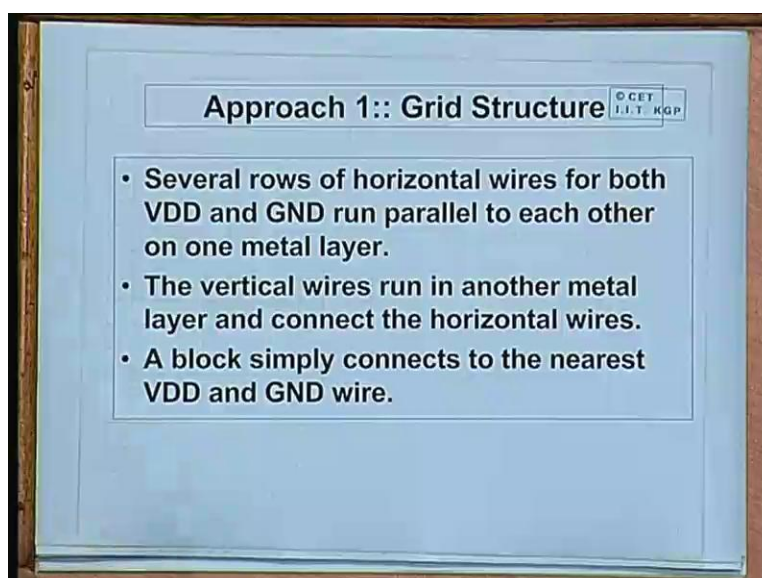
And all the functional blocks in a typical circuit require power and ground connections typically ok. Unless if there are power if there are basically some are pass transistor networks they do not need power but most of the other blocks they need. Now power and ground nets are again usually laid out on the metal layers of the chip. Not necessarily on a single layer maybe on a single layer maybe on a two layers also. But only on the metal layer due to the lower impedance. So we are we shall looking at some of the from the interconnection topologies. So we will be basically looking at two things construction of the interconnection topology. This is one issue and after the topology it is constructed we will have to do another thing. Here you will have to estimate the current that need to flow thro through the different segments of the tree and accordingly size the widths of the individual segments. So the power and ground nets are one where the widths of the different net segments may vary depending on the total current carrying capacity. Now for VDD and ground obviously the requirement is that.

(Refer Slide Time: 50:51)



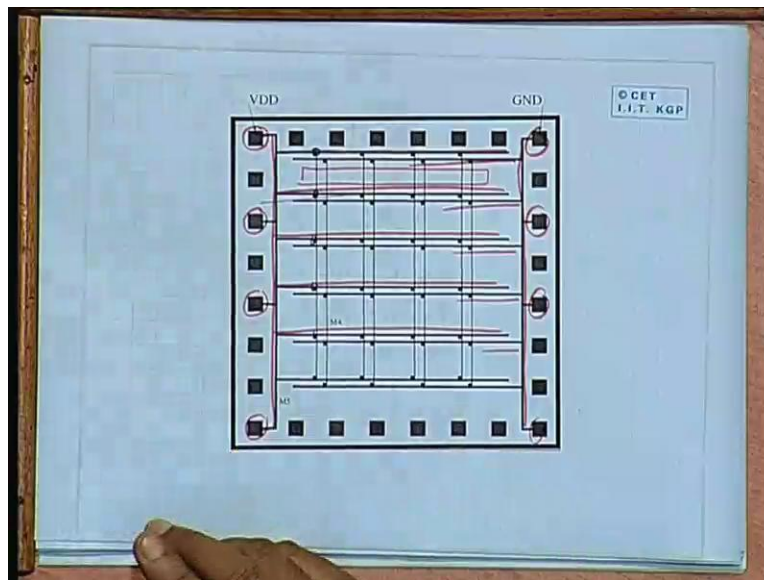
We will require two non-intersecting interconnecting trees. The trees will be non-interconnected obviously; the widths of the trees will be variable depending on the total current carrying capacity at that point of the tree. Now there are several approaches which have been proposed we would be looking at couple of them.

(Refer Slide Time: 51:13)



First one is a very simple approach this is suitable for standards cell based designs. Here we assume some kind of a grid structure several rows of horizontal wires for VDD and ground run on one metal layer the vertical wires run on separate layer. So I am showing with the, I am explain with the example a diagram say this.

(Refer Slide Time: 51:35)




This is the layout of a chip these are the input output pads. I am assuming that on once on the left side of the chip there are several IO pads which are feeding VDD. Well instead of a single VDD pin you can have several so that the drive is more. So you have a net on the left side this one from here several rows running like this these are the VDD rows. Similarly from the other side there is a ground net there are several points again on the other side for ground. There is a vertical line like this and from here you have several such ground line running. So it is like a comb from both side a comb is coming and if it is a standard syllabus design for the cell rows which are laid down between you can uniformly get VDD and ground in all rows in uniform bases.

So this is a very common kind of a layout for standard cell based designs and between the horizontal and vertical connections you can have those wire connection not necessarily in one point. But in all points in order to reduce the resistance. These are all uni-potential nets VDD and ground so there is one VDD line here and there are number of VDD lines coming and again in the next stage. You can have vertical lines running down below again another level of comb ok. So this you can continue so in a standard cell design you may not need this second level that depends on your requirement ok. And the second approach that we talk about this is a more general approach.

(Refer Slide Time: 53:31)

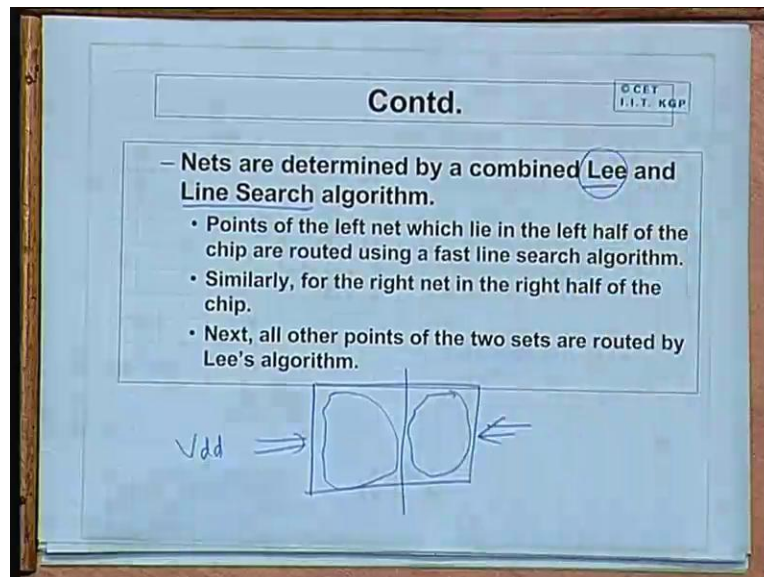
Approach 2:: Using Interdigitated Trees

- Tends to route nets in an inter-digitated fashion.
- Extends one net from the left edge of the chip, and the other from the right.
 - Routing order of the connecting points is determined by the horizontal distances of the connecting points from the edge of the chip.

Vdd ⇒  ⇐ *GND*

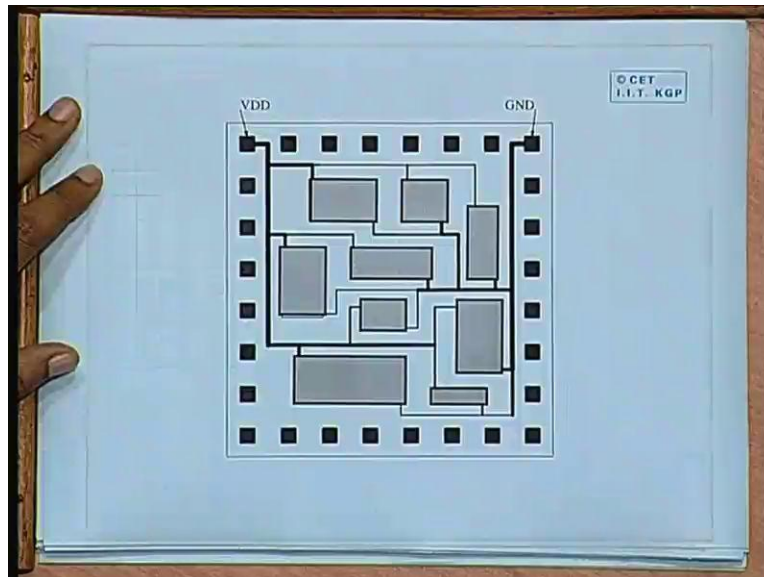
Where the clock and ground net maybe required to be routed to arbitrary locations this is called inter digitated trees. So the idea is that VDD net and the ground net, say the VDD net you start routing from the left edge of the chip. And the ground net you start from the right edge and you start routing them. For example this is your chip, you divide the chip into half from this direction you start with VDD from this direction. You start with ground right and the way this VDD and ground nets are connected to the destination points.

(Refer Slide Time: 54:19)



Here they are done using a combination of lees and line search algorithms. See if you start the VDD net from the left this is the middle point. So the rule of thumb says that for the VDD points on the left side of the chip we use the line search algorithm there will be done very fast. There is there are no other nets which are interfering. Similarly for the ground net coming from here we use the right half and we use the line search algorithm for routing them. But as this nets crossed the VDD net crossed the right side the ground nets crossed the left side. There will be now lots of obstacle on the other net so now better use the lees algorithm. This is how it is done and one typical example of an inter-digitated tree layout is this.

(Refer Slide Time: 55:19)



So for one side you have the VDD net starting from the other side you have the ground net starting. So this is I am showing the final layout but where I am saying that half of this was done using line search then half of this was done using leers. That is what is typically done. And the sizing and the width of the nets this have to be you computed separately using a power estimated tool and you can just do this our size sizing separately. So we have looked at the issue of clock and VDD ground routing that means what kind of routers are needed for this. Now in our next class we will be continuing with this and we would be looking at some other specialized routing techniques like the over the cell routing one thing I mentioned earlier and we would also be talking about layout compaction so after this layout how we can do the compaction. Thank you.