

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

**Lecture No #21**  
**Backend Design Part-VII**

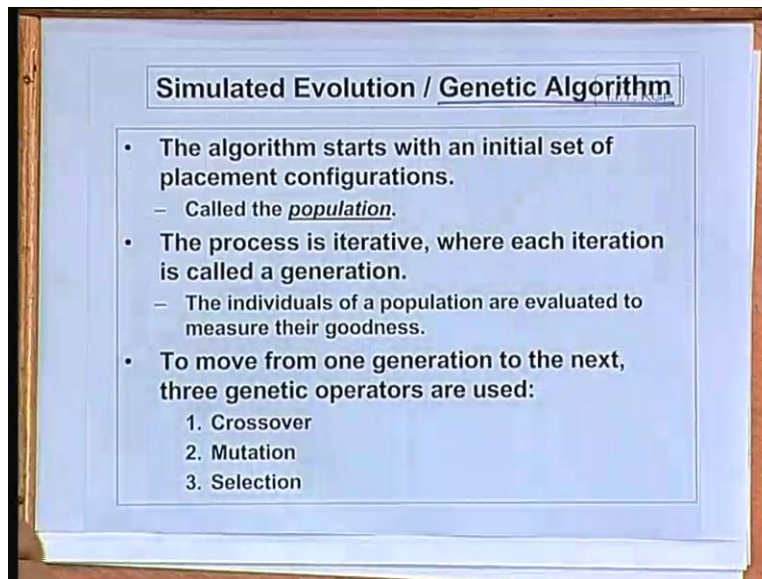
So in our last lecture we had talked about the simulated annealing method of placement.

(Refer Slide Time: 01:18)



Now today we shall be looking at a number of other methods which also have been attempted to solve the placement problem. The first method we look at is something called simulated evaluation.

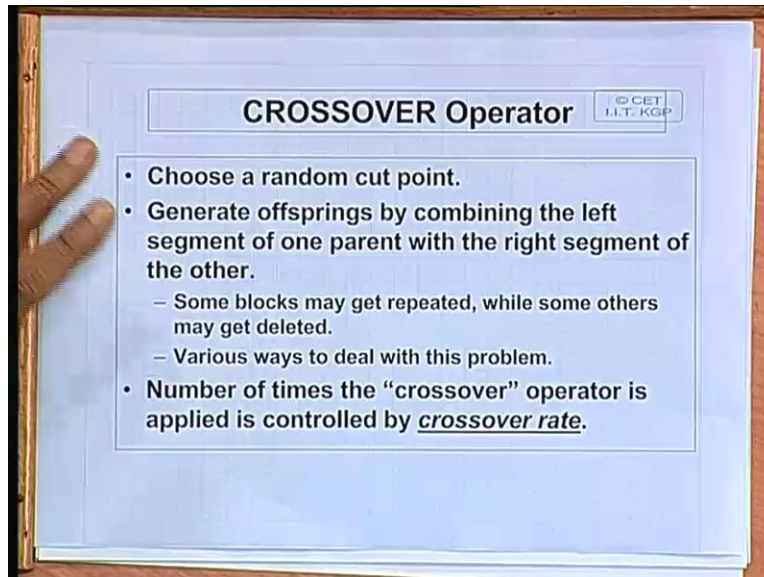
(Refer Slide Time: 01:36)



Well the way we present the method this is also a version of the so called genetic algorithm. So first let us try to understand what this general concept of genetic algorithm framework is and how the placement problem is mapped into this. Well the concept of genetic algorithm is simple. Well in a simulated annealing you start with one solution. But in genetic algorithm you start with a set of solutions. So with respect to placement there is a set of placement configurations to start with. So you have a set of solutions which you call the population. There is the concept of a population which is a collective set of solutions. There is some kind of a cost function with respect to which you can evaluate the goodness of each of the members of the population.

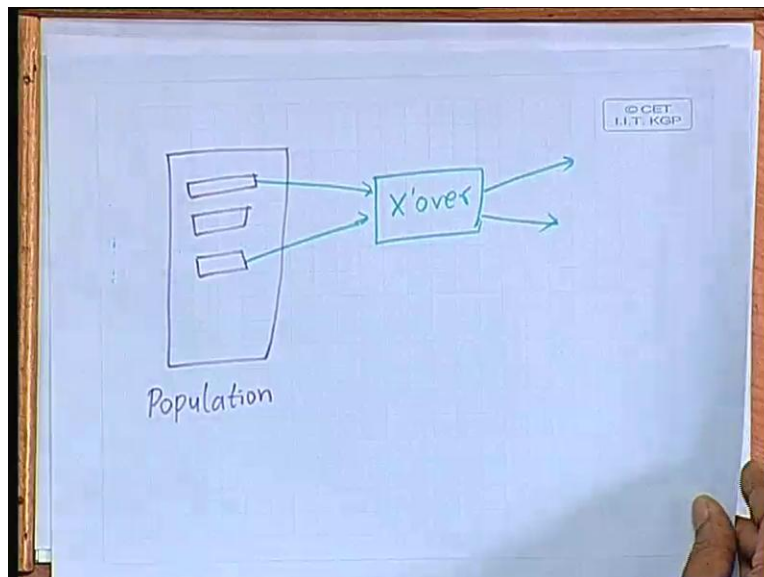
So you can measure the goodness which is sometimes called the fitness measure the fitness measure of the members of the population. This is also iterative method like simulated annealing there are three operations or processors which are carried out. These are called cross over, mutation and selection. So let us try to define and understand what these three operations are. Well the way we present this operations it will be specific to the placement problem. But if you want to apply this to some other problem you will just have to change the definitions but otherwise the framework will remain the same. First the cross over.

(Refer Slide Time: 03:36)



Well in order to explain the cross over operator just I am explaining first one thing.

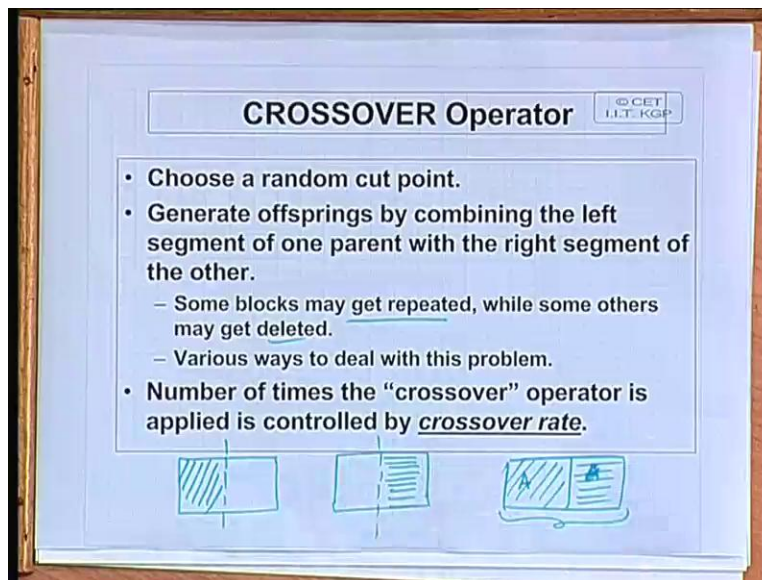
(Refer Slide Time: 03:43)



Suppose you have a given population ok. This is your given population which will comprise of several solutions. Now when I apply the cross over operator what I do is that I take any two solutions from the population ok. I apply the cross over operator on this I get two new solutions

this is the basic idea. I start with two solutions apply cross over on them and get two new solutions right ok.

(Refer Slide Time: 04:36)



So how this is done with respect to the placement problem suppose I have two solutions one solution is say this is a placement solution this. I choose a random cut point say in terms of x coordinate I can choose randomly a cut point here for the other solution also in the same place. Well intuitively speaking what I am trying to do is that I am trying to take the first half of solution one and the second half of solution two put them together to get the new solution. Most of the crosses over operations for genetic algorithm implementations do it this way. But for this placement problem if you simply do this it can lead to a solution which is not correct because a particular block may be repeated in both the halves or some of the blocks may be absent. So you will have to make a small refinement to this basic idea what I say some block may get repeated while some others may be deleted.

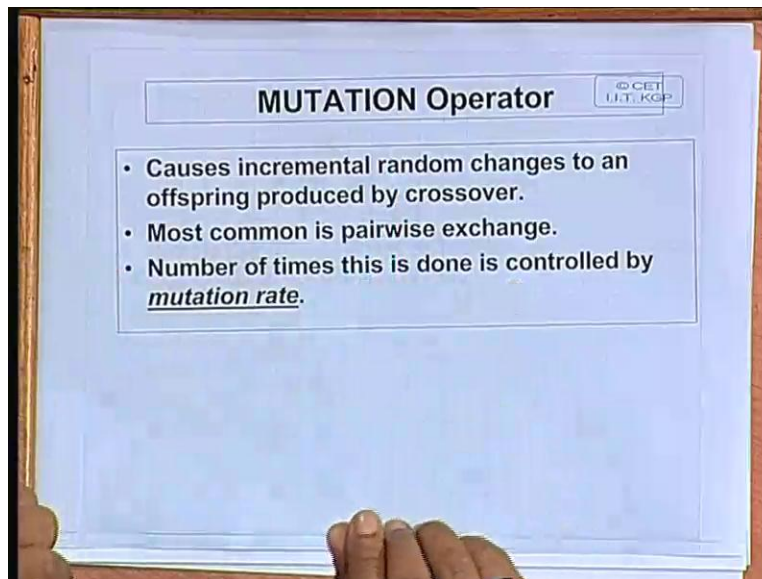
So one way to handle a situation is that well you take the first half of the solution you try to put to the second half of the second solution to this. But while trying to do this say you take the first half of this and the second half of this. You may see that some block say A which is here is also coming here. Naturally some block which was here is missing. So you try to find out that

through a search and you replace this block A by the other one. Do this to get a feasible solution out here. (()) (06:24) That is just a matter of implementation how you implement it. See the intuitive reason of the cross operator is like this. See I have a number of solutions available to me so I have computed the fitness of all. So the two solutions I take for crossing over will be the two which are good in terms of their fitness values this is what I always attempt to do.

Now if the two solutions are good and if I combine the first part of one good solution with the second part of another good solution possibly I will be getting a better solution. I am trying to take the good features of two solutions and combining them possibly I will be getting a better solution. This is again a random way of doing it. But I am using the fitness value of the solutions to select the good parents you can say to generate the good child's children. And also there is another parameter called the cross over rate well. We do not do this cross over blindly every time I take to and do a cross over. I take two then I compute random number and check against a given probability value if it is within the cross over rate, then we apply it otherwise you simply copy it to the next generation.

Given the population we apply cross over to get the population of the next generation. So we move from one generation to the next (())(08:03) Yes (())(08:05) Two at a time. (())(08:14) Yeah. I will just, I will. I will basically come to this point whether we are taking all or not I will come to this point little later. But we are taking two at a time. (()) (08:24) Cross over it is that suppose I take two solutions and I am just about to do this cross over before doing this I compute a probability and means if my generated random number lies within that probability value then I actually do the cross over or else I simply copy them just as it is. (()) (08:50) No cross overs only within a generation say once generation I plus one is computed from generation I generation I can be deleted ok.

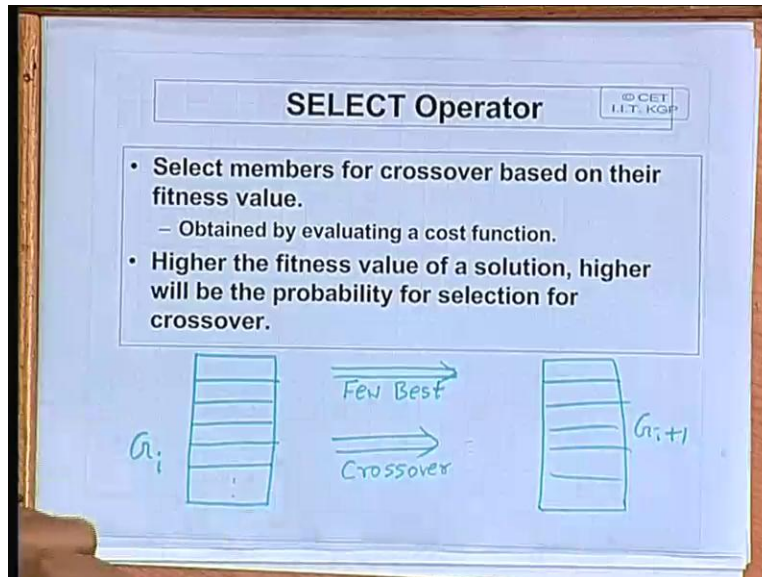
(Refer Slide Time: 09:06)



There is another operator called the mutation operator. (()) (09:08) Yes (()) (09:10) See some of the parents are good I do not want to disturb them let them be present in the next generation also. Because if I always (()) (09:25) Yes I will just I will basically come to that point I will just come to the point shortly. Now the second operator is called the mutation operator this basically makes some small random change to a solution take a solution make some random change and put it in next generation. So the one common way of doing is say you do a pair wise exchange of two blocks.

Now the frequency with which you carry out mutation again is determine by a mutation rate. Now the mutation rate is a very small number may be it is one percent or point five percent typically like that. So this mutation you carry out much less frequently as compare to cross over. Now the purpose of mutation is that you try to make a random change and possibly go to a new solution. From there possibly you will be getting some better solutions you have to be apply cross over again this is the intuitive justification.

(Refer Slide Time: 10:34)



Well now coming to a question the select operator actually selects the members for cross over. Now from experience what has been found is that suppose we have a generation out here this is generation I generation I contains several solutions. So you have to produce generation I plus one. So, each of the solutions will be having an associated fitness function or a cost function. Well experience shows that this algorithm works best if you do this. You select the first few best solutions simply copy them. Few best so the few best solutions simply get percolated to the next generation ok. Others will get generated through cross over. The way cross over is applied is that you select two solutions at random well not really at random their chance of selecting is directly proportional to their fitness value higher the fitness greater is their chance of their selection.

So using that kind of a randomized way you select two solution from this well may be you are selecting means one of those best no problem I can. You select two of this we apply cross over or do not apply cross over depending on the cross over rate. Now whatever is your decision those two resultant solutions you simply copy here. You go and repeating this till the next generation is full this process you repeat. (()) (12:36) Equal generation sizes are same the population sizes will be the same. Now the exactly what will be the size of the population? Well how much iteration will carry out this are all subject to experimentation? But this is the basic skeleton of the genetic

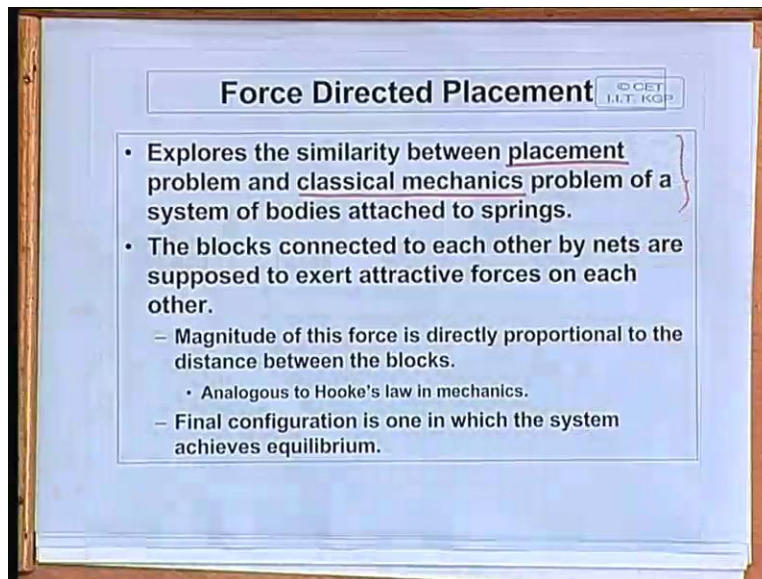
algorithm you have to represent the solution whether solution representation is important. Here it should be such that cross over operation can be applied very efficiently.

Because for this placement problem it is a little problematic the way of cutting and checking and doing this is not very efficient. So there are other algorithms where the solution representation and the way you do the cross over it can be done very easily fine. So this is an essential idea in genetic algorithm and some people have also tried to apply this. This is not used in commercial tools, but there have been a number of means number of works which have been reported where placement was using genetic algorithm the results obtained was good. But the time taken was not that good means the time taken to arrive at the solutions where much higher as compare to the simulated annealing.

So in practice people do not use genetic algorithm for placement. (( )) (14:02) Yeah. Possibly yes. Possibly yes. Yes if a standard cell library the swapping and other things are very easy. Yeah. Yes, yes. General cell it will be very difficult yes. Now one thing you understand that about 90 percent or even more than 90 percent of the design efforts that people give nowadays are to a standard cell only. The all the chips you design today that very rarely they will be it is a full custom kind of a design. Typically you pick up cells from the library you put them in standard cell arrays ok you are right. Ok next let us look at an interesting method.



(Refer Slide Time: 14:50)



Well which was proposed this method is interesting. But well because of the competition complexity this is applicable only to small placement problems but let us look at this method this is very interesting. See here the method is based on an analogy this analogy or a similarity between the placement problem and a problem of classical mechanics, a system of bodies which are connected by springs. See in a in the placement problem what is the thing there are several blocks which are connected by wires more the number of wires more is the attractive force. So the placement algorithm will try to bring them together. Similarly if there is string spring so as the blocks become further what the width the springs may be stronger or lighter depending on the strength of the string and the distance the attractive force will be different.

So the attractive forces that two blocks experience and the attractive forces well this body of springs experience they are very similar and they are guided by hooks law more the distance more will be the attractive force ok. So it tries to use this analogy magnitude of the force is directly proportional to the distance between the blocks. And the objective is that you try to obtain a final configuration which is the equilibrium condition where the net force exerted on each block is 0 that is the final configuration. So if you can find out a placement which satisfies

this that is your best placement. Yes so in terms of concept is very interesting. So let us try to formulate this problem first ok.

(Refer Slide Time: 16:50)

**Contd.** CET  
I.I.T. KGP

- A cell  $i$  connected to several cells  $j$  experiences a total force  
$$F_i = \sum_j (w_{ij} \cdot d_{ij})$$
where  $w_{ij}$  is the weight of connection between  $i$  and  $j$   
 $d_{ij}$  is the distance between  $i$  and  $j$ .
- If the cell  $i$  is free to move, it would do so in the direction of force  $F_i$  until the resultant force on it is zero.
- When all cells move to their zero-force target locations, the total wirelength is minimized.

*Diagram: Two rectangular blocks labeled 'i' and 'j' are shown. Block 'i' is on the left and block 'j' is on the right. Several lines connect block 'i' to block 'j'. The label 'Wij' is written above the lines connecting the two blocks.*

Say I have a cell I this is my cell I this cell is connected to several other cells lets call one of them  $j$  in general. So a cell  $I$  which is connected to several other cells in terms of the net list will experience a total force which will defined as the weight of each link this weight is  $w_{ij}$  this  $w_{ij}$  can be just be the number of wires connecting block  $i$  and block  $j$  weight multiplied by their distance Manhattan distance this will be the weight between  $i$  and  $j$ . So if  $i$  is connected to several other blocks like this then it will be the sigma of it summed over all  $j$ 's. This will be the total force exerted on block  $I$  by the other blocks in the surroundings. Now the idea is that similar to spring if the block  $I$  or cell  $I$  was free to move it will move in the direction of the force  $f_i$  until the resultant force on it is 0, this is what we are trying to do. We are trying to find the final location of block  $I$  such that both the  $x$  and  $y$  components of the force on it becomes 0. And this we iterate over all the blocks so that all of them try to move to their 0 force target locations. Ok let us see the calculation for block number  $i$ .

(Refer Slide Time: 18:33)

**Contd.**

© CET  
I.I.T. KGP

- For cell i, if  $(x_i^0, y_i^0)$  represents the zero-force target location, by equating the x- and y-components of the force to zero, we get
 
$$\sum_j (w_{ij} * (x_j - x_i^0)) = 0$$

$$\sum_j (w_{ij} * (y_j - y_i^0)) = 0$$
- Solving for  $x_i^0$  and  $y_i^0$ , we get
 
$$x_i^0 = (\sum_j (w_{ij} * x_j)) / (\sum_j w_{ij})$$

$$y_i^0 = (\sum_j (w_{ij} * y_j)) / (\sum_j w_{ij})$$
- Care should be taken to avoid assigning more than one cell to the same location.

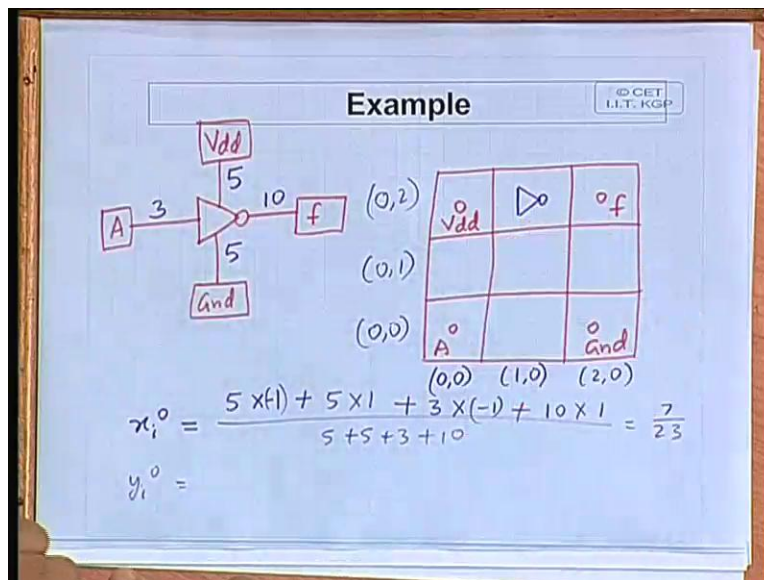
With respect to block number i, or cell number I suppose  $x_i^0, y_i^0$  represents the 0 force target location. Now these two equations will follow you see block number i was here block number j was here ok. This block number I has move to its 0 force location. Let us call it  $i^0$ . Now initially the force between I and j was  $w_{ij}$  into  $x_j$ . Now since has move to a new location. This it will now be  $w_{ij}$  into  $x_j$  minus  $x_i^0$ . (( )) (19:26) No. (( )) (19:34) This j has a coordinate of  $x_j$  and  $y_j$ . This new location of the block has coordinate of  $x_i^0$  and  $y_i^0$ . Now since this has move to this new 0 force location I am equating the x components and y components separately with respect to the x components the difference between the x component between this j and this  $i^0$  is  $x_j$  minus  $x_i^0$ .

Similarly y component is  $y_j$  minus  $y_i^0$ . These two equations I am writing separately. One is the weight between these two multiplied by the difference in the x components weight multiplied by the difference in y coordinates. These sigma over all j's will be separately 0. Ok the x components and the y components of the force here separately equating to 0. Now (( )) (20:34) Well  $i^0$  is the coordinate well this is not really  $i^0$ , I am saying that this I which was  $x_i, y_i$  this has move to a coordinate  $x_i^0, y_i^0$  which is the 0 force target location. But is actually same block I am I have written this  $i^0$  but is the, but it actually the same block I which is moved here.

$(i)$   $(20:57)$   $W_{ij}$  is the number of wires that connects block  $i$  with block  $j$  that will remain same in the new location also.

Now in this equation if you just move it here and there you can easily solve for  $x_i^0$  and  $y_i^0$ . This will be the final expressions.  $\sum w_{ij} x_j$  divide by  $\sum w_{ij}$ . Similarly  $\sum w_{ij} y_j$  divide by  $\sum w_{ij}$ . So you can use these two expressions to find out the 0 force target location of a particular block right fine. And of course one thing you have to keep in mind explicitly you have to check. That means you are not leading to block overlap two or more cell should not be assigned to the same  $x_i$  locations. This is something you have to check separately ok fine. Let us take a simple example to illustrate it.

(Refer Slide Time: 22:04)



Suppose I have a problem like this a simple inverter. These are the io pins or io pads whatever you call A is the input f is the output Vdd is the power and ground. So this gate has to be connected to this four things right. Now what I am assuming that I have a three by three grid I am taking a small example just for the sake of illustration I have a three by three grid I am assuming that these four terminals or pads are located in the four extreme corners. So here I have Vdd say here I have f here I have A, and here I have ground. Now I can place this inverter in any

one of the remaining five cells. Now my objective is to find out the best location for this inverter in these cells so that the force so that it corresponds to its 0 force location.

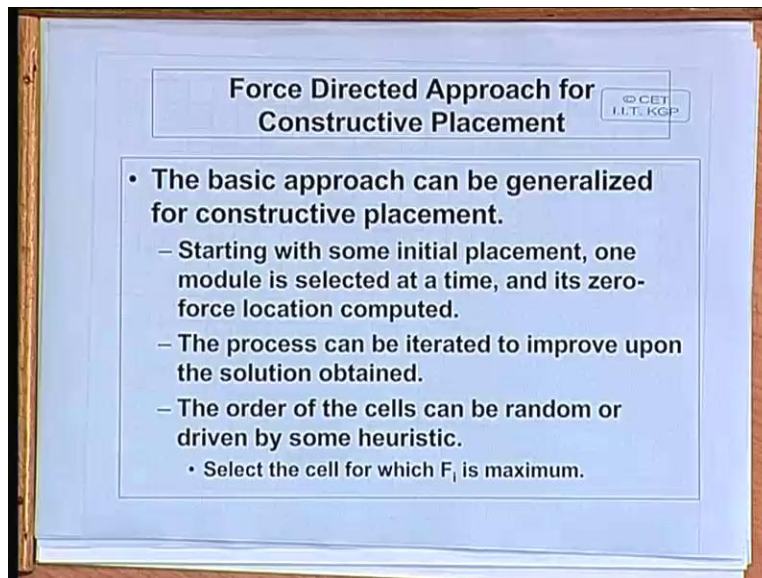
Now what we assume is that some weights let us assign some weights to these lines also. Well, these weights may be depending on the number of lines or depending on importance. This so these I am just assigning arbitrarily. So there is no justification behind this numbers these are the numbers I am assigning and these are the coordinates 0 0 1 0 1 0 2 0. This is again 0 0 in this side 0 1 0 2 fine. Now on this problem we can compute using the expression we have just shown before this  $x_i$  and  $y_i$  we can simply compute the  $x_i$  and  $y_i$  expressions.  $X_i$  is 0. See  $x_i$ , I can calculate the weight multiplied by the you can say  $x V_{dd}$  ok. Now we assume that the gate is initially placed here in this location this is your  $i, j$  from here I want to find out the 0 force location. So initially the gate, gate, gate is placed here.

So in terms of the x coordinate the weight of  $V_{dd}$  multiplied by the difference in the well in the x coordinate how much is the x coordinate difference x coordinate difference is 0, 0 to one, one or one plus ground multiplied by difference between ground and this in terms of x coordinate this also one plus. (( )) (25:54) Now, say it is the distance it is the distance we are not talking of the sign really. (( )) (26:02) Ok, ok, ok, ok, ok. Minus one  $V_{dd}$  into minus 1 right?  $V_{dd}$  into minus 1 5 into 1 plus in terms of A. A will also be minus one plus  $f$  will be 1 divide by 5 plus 5 plus 3 plus 10. How much it comes to? Minus 5 minus 8. 7. 7 by 23. Similarly  $y_i$  will be computed to something similar see this after  $x_i$  and  $y_i$  are computed will be getting some values. Now you try to approximate them to the nearest integer. Nearest integer and try to see that whether you can place it in that particular location that will be the best place to put it in. Basically this algorithm will work like this. (( )) (27:14)

No see this is the 0 force location you try to place it as close to that position as you can that is the idea because you cannot place in a fractional place. So it has to be it has to be aligned with the grid because we are assuming some grid  $x, y, z, x, y$ . So this your approximate it to the nearest grid point we are trying to place it there ok. Fine so this is the essential idea be had constructive placement you have a problem you try to take one of the blocks at a time and try to place it in the most desirable location you can go on iterating. Now this force directed placement can also be

used for some kind of constructive placement like you are trying to build up a placement starting from scratch. Here I am assuming that placement was already given, you are trying to improve a point this is iterative refinement. But as an alternative you can also have a constructive placement mechanism where, starting from scratch you are going on building up the placement. How?

(Refer Slide Time: 28:41)

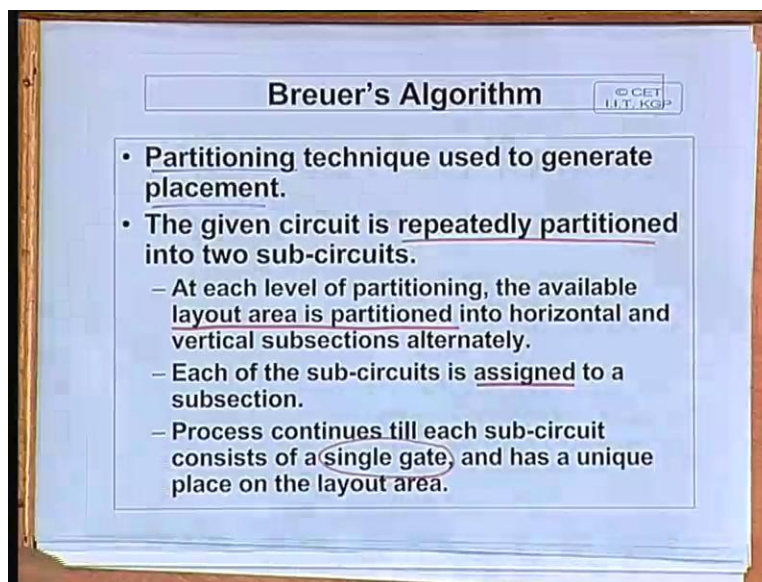


You can see you can basically start with the fixed blocks you take one of the modules at a time try to find out the 0 force location put it in take the next one put it in in this way. You build up it and then you can iterate a few more times to improve upon the overall thing. So this force directed placement is used both for iterative refinement as also for constructive placement both it has been used. But as you can see since it since it involves quite a bit of computation to find the 0 force location. So it is useful only for small designs and for big designs often we will find that the x y coordinate you are computing they will be mapping into the same grid location for multiple blocks there will be overlaps.

So for so for a general designs this method will have limited utility. But the method is interesting but it has limited utility in terms of it is you can say it is in practical even say practical way of

tackling the placement problem. Because of placement problem it purely depends on mathematics its tries to optimize a function it sets a 0 tries to find out x, x and y values but it does not check whether there is a overlap in the process right. And means when you are talking about the constructive placement you can also you can also have some heuristic. You can start with the block which in the default case will be having the maximum force that will be a first block you select to move to the 0 force target this way you can iterate. So either you can start with an empty scratch or you can start with all the blocks in place take them one at a time so it's a nu number of ways you can handle it. Ok so, number of other methods are, there I am very briefly talking about it I am not going in to details of this methods.

(Refer Slide Time: 30:54)



See brewers' algorithm is interesting in the sense that it combines partitioning with placement. So here partitioning and placement are going hand in hand. The idea is very simple I have a net list I have my floor plan I cut the net list I also cut the floor plan. So a part of the net list will go here other part will go here. I repeat the process I cut the net list I cut the floor plan I cut the net list I cut the floor plan. Finally means each piece of my net list will become small enough and each portion of my floor plan will also become small enough. So piece I can put in to that small part of the floor plan this is some kind of a divide and conquer approach ok. So breuer's

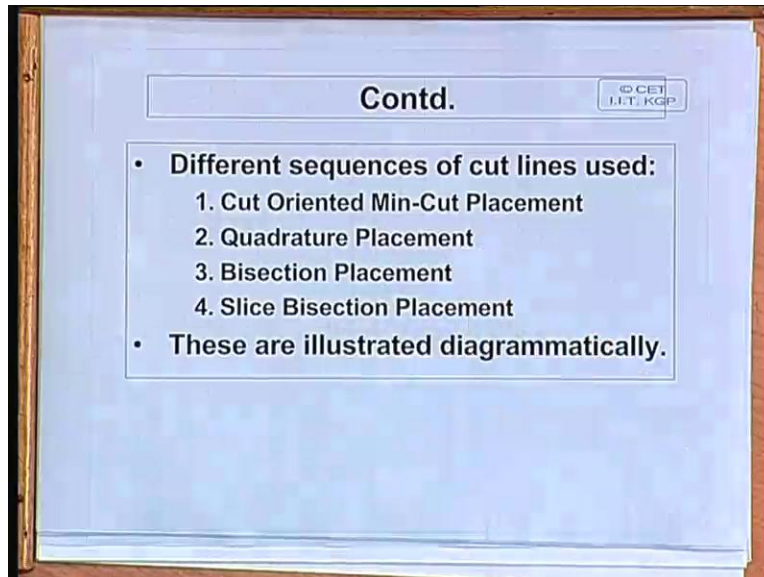
algorithm is essentially this the circuit is repeatedly partitioned and at each level of partitioning the available layout area is also partitioned.

Now how they are partitioned a number of different ways of slicing have been proposed I will show them. And each of the sub circuits which are obtained is assigned to a sub section. This repeats until each sub circuit is small enough may be a single gate or a few gates which can be directly placed. This method is good, but in general it will not lead to very good placement. Because you are basically cutting it with respect to certain criteria. But the final solution will be getting in the process may be one but the net length may be pretty long. You are not putting enough emphasis on the interconnection. You are putting more emphasis on the gates or the basic components. The interconnections you are not putting much emphasis on.

By doing this partitioning and placing them individually may be the interconnection wire length will become very long. (( )) (33:03) Yeah. But here interconnection is not being taken care off partitioning the graph you are assigning them. But of course well partitioning you are trying to partition in such a way that the cut size is minimized but still during the process of further partitioning the two blocks may be going further apart which were initially strongly connected ok. (( )) (33:28) This will produce a correct solution of course (( )) (33:34) Yes, yes, yes. There are number of methods which give you a very rough and a gross solution which is not very good, but ok. It is acceptable that can be starting point for the iterative refinement methods yes. So this breuers algorithms defines four different types of cuts.

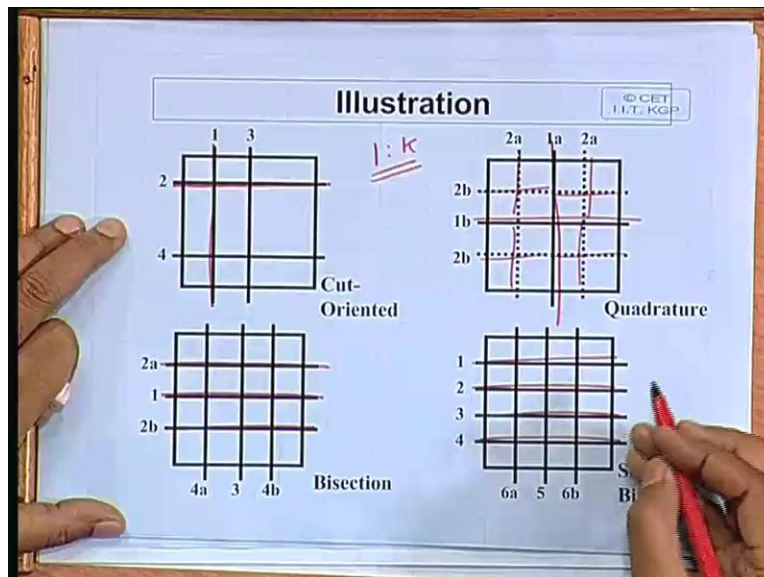


(Refer Slide Time: 33:58)



Cut oriented min cut quadrature bisection and slice bisection so I am showing you what this cuts mean.

(Refer Slide Time: 34:06)



Well in the first kind of a cut the sequences of cuts are like this this is the layoff floor, floor area. Well here you are assuming that the partitioning algorithm you choose it is not a bi

section algorithm it does not divide net list into two balanced partitions it will be unbalanced. The size of the partition will be one is to k for example. So you first make a slice like this one part will be small one part will be bigger. You do this repeatedly then you slice like this. One part will be very small the other parts will be there then three then four then five then six in this way you go on. This is called cut oriented this is one method. The second method quadrature quadrature says that in each step you divide the square into four smallest squares the first step you divide like this. And the second step you divide each of them into four.

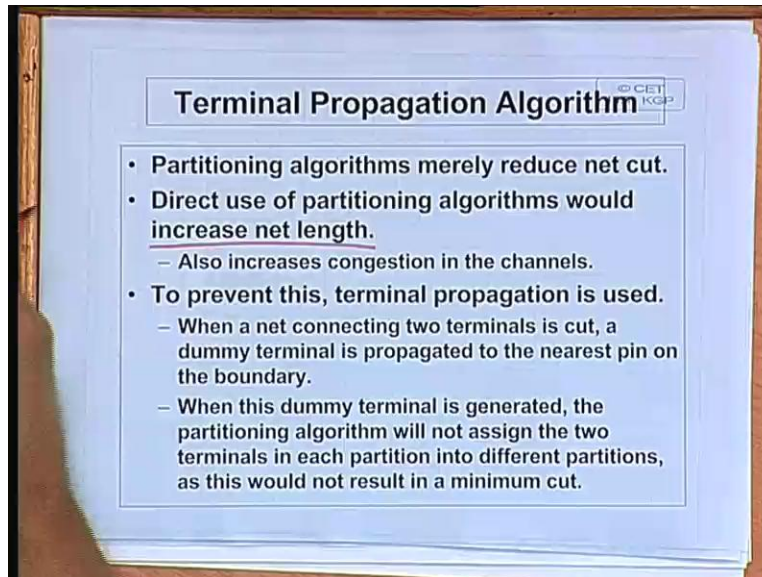
So here the sub circuit at each step is divided up into four equal pieces. Well this you can you can use (( )) (35:38) in for the purpose just apply it twice one after the other first divide it into two then two into further 2 and 2 ok. So quadrature placement can be applied for general cell arrays like you can say for means customs and designs. You have the layout space for anything can be placed anywhere you on partitioning and going on placing. But there are some partitioning techniques which are targeted more towards standard cell base designs likewise in the method of bisection. They do like this first you cut like this then you cut like this then you cut each of this into half first to slice using horizontal lines. This you continue until each horizontal slice is sufficient to hold one row of the standard cell.

First you do this and if you also do the partitioning of the original net at the same time you know that which sub circuit you will have to put in which standard cell row. Then you try to map the corresponding net list using technology mapping from cells of the library then you do vertical mappings vertical cuts and place them along the rows in suitable points. Similarly the slice bi section is similar you cut it like this not using half and half from one side you go on cutting. This each row width or height is sufficient for one row of standard cell there there on the other side you do bi section.

First you define what will go in each row of the standard cell. But while placing that you do bi section five then half then half. So I have I have given you just a rough idea but actually breuers algorithm is more complex than this. This actually this involves a number of other heuristics also. But breuers algorithm works fairly well for standard cell placement because it uses this kind of a partition. But it assumes that you do not have any floor planning to start with, you start with

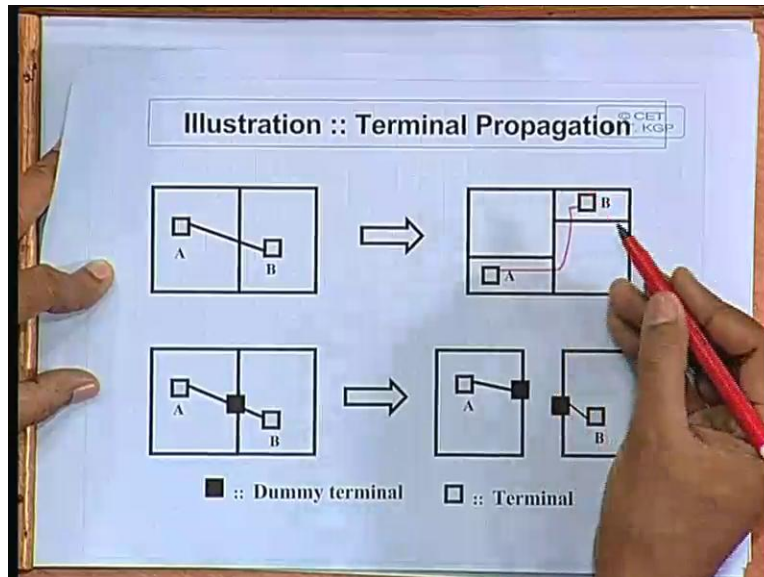
partitioning and placement both together this is a technique where partitioning and placement can go hand in hand ok.

(Refer Slide Time: 38:26)



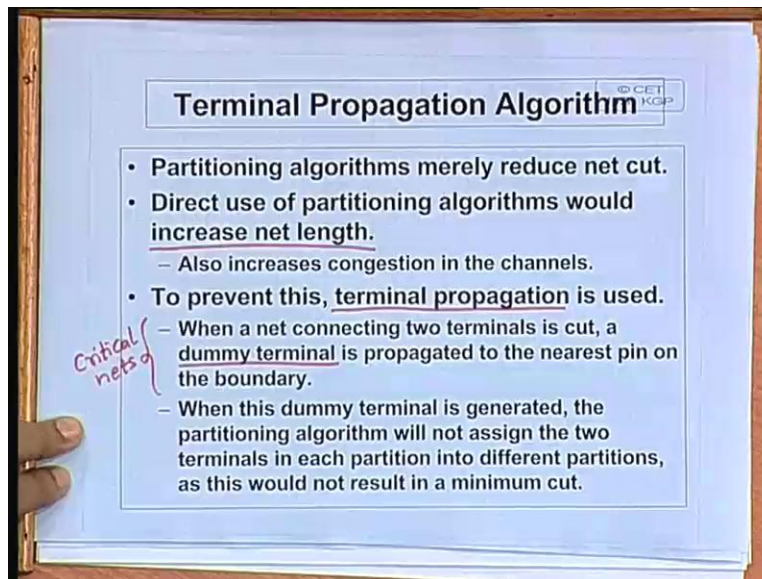
Alright another approach this is called terminal propagation this is also I mention very briefly. See algorithms like breuers algorithm if you applied blindly I told you sometimes net length may be increasing. Why just I have an example in next slide I am showing this.

(Refer Slide Time: 38:50)



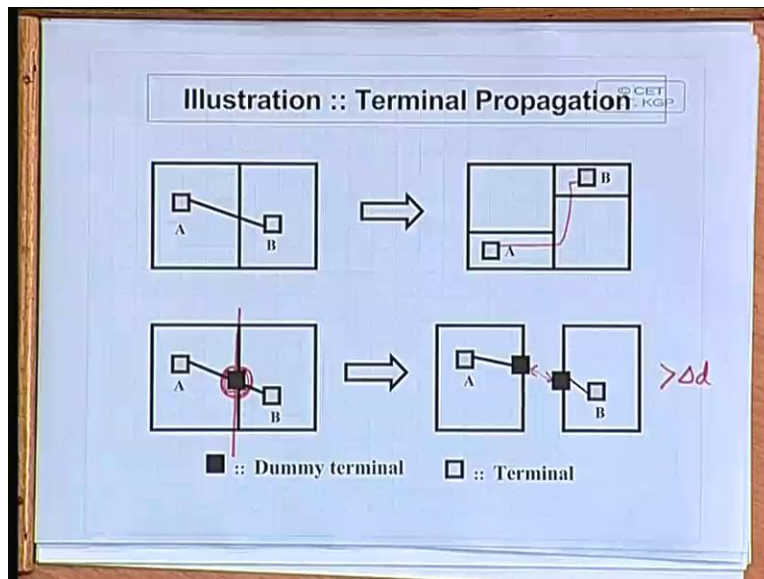
Suppose this was a scenario there was a block A block B which are interconnected. Suppose after a sequence of cuts suppose initial cut was like this they are separate two part then again further cuts are made. So it is a possible that after a sequence of cuts this module A will come here and module B will go there which means that the length of this net will become may become longer. So there is no guarantee that two blocks which were connected together they will still remain closer close. They may go to two extreme corners and the length of the wire connecting them may become very long and if this is one of the critical paths you are in trouble right.

(Refer Slide Time: 39:42)



So to avoid this kind of a problem well a heuristic like called terminal propagation is used. Terminal propagation says that whenever a net connecting two terminals is cut you introduce a dummy terminal. Now see this it is up to you do you do you apply this technique for all net cuts or only for critical nets typically this is used only for critical nets. The concept of dummy terminal is something like this.

(Refer Slide Time: 40:22)



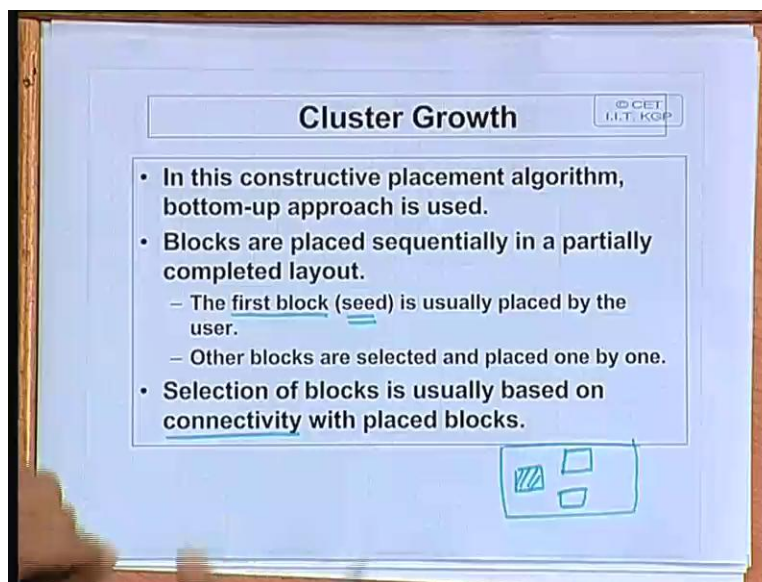
See whenever you cut this you introduces a dummy terminal out here on the boundary, boundary of the cut this is the dummy terminal ok. Now when this dummy terminal is generated the partitioning algorithm will not assign the two terminals in to different partitions it will try to keep them the in same partition as far as possible well I am giving an example. Suppose this was a scenario that when this partition was generated this dummy terminal was created. Now after this partition these two halves will be moving away from each other. Now if they move away from each other these terminals well a copy of the terminal on each boundary will also start moving away from each other. Now these two are dummy terminals these is kept separately track off.

So if you keep track of this dummy terminal separately you will not allow these terminals to move apart greater than some specified distance  $\delta$ . This will be an additional constraint which we impose that these dummy terminals will always remain close together. So you will so you will not allow this blocks a, and b to be placed anywhere you wish there will be some additional constraint you are imposing. So using this terminal propagation people have (( )) (42:00) try to solve this problem. So this A and this dummy terminal will always belong to the same part this is this was actually what was said. Not assign the two terminals in different

partition this A and this dummy terminal and this B and this dummy terminal this two as a pair will always remain in the same partition for all future partition.


So you cannot partition it like this or like this. This is again just a heuristic which is which are use to tackle the critical nets nothing else. Just you are using partitioning but instead of un constraint partitioning you are putting some additional constraint. You cannot cut this or even if you cut this you cannot move this away some constraint like this. Well finally a very simple algorithm which is used more as the initial point of iterative placement algorithm this is called cluster growth.

(Refer Slide Time: 43:06)



**Cluster Growth** ©CET I.I.T. KGP

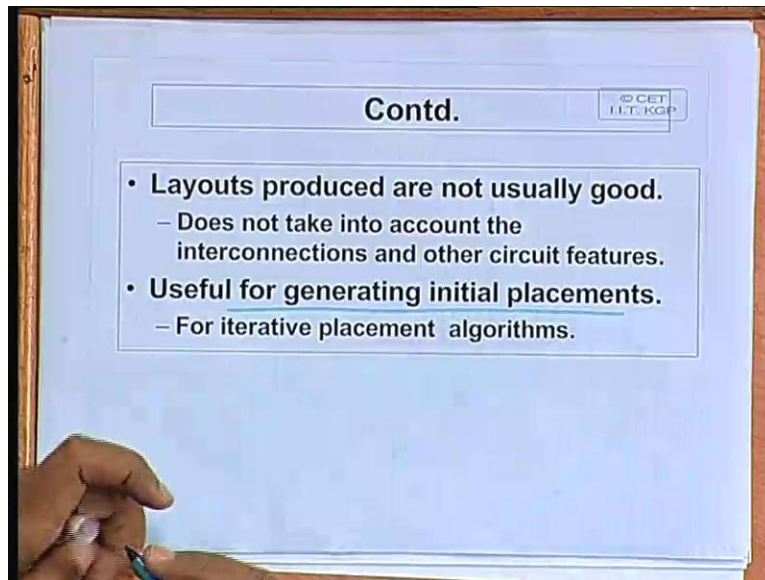
- In this constructive placement algorithm, bottom-up approach is used.
- Blocks are placed sequentially in a partially completed layout.
  - The first block (seed) is usually placed by the user.
  - Other blocks are selected and placed one by one.
- Selection of blocks is usually based on connectivity with placed blocks.



This is this is this is a very simple algorithm where starting from an empty layout you place the block sequentially based on certain criteria. The first block which is called the seed is usually placed or chosen by the user and other blocks will be placed by the algorithm one by one and the criteria of selecting the other blocks is usually based on connectivity with the already placed blocks. So you have a layout you first put one block this is the seed then you select a block which is most closely connected to this most strongly connected to this place it. Then take

another block which is strongly connected to this pair place it. So this why you go on but there is no provision of moving a block around the placement you go and sequentially placing it.

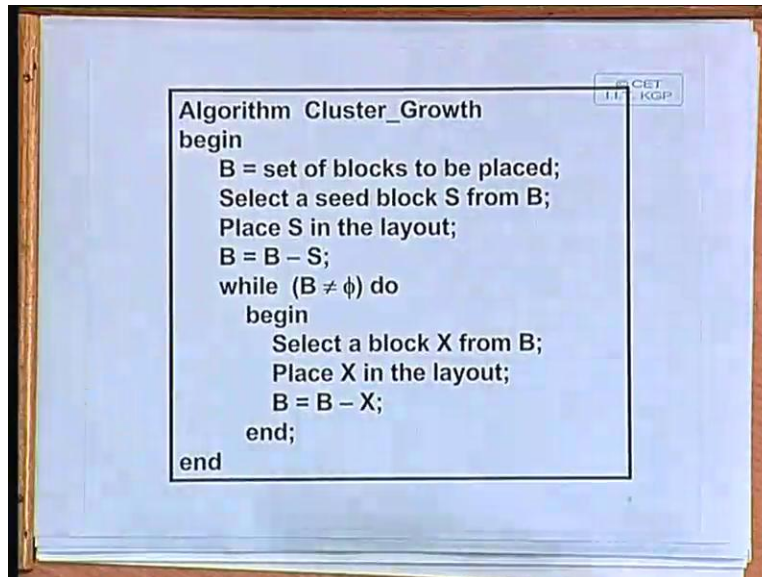
(Refer Slide Time: 44:13)



Now obviously since you are not improving iteratively on the solution layouts produced are usually not good. So interconnections you are not taking into account only the on weights of connections you are taking into account. So this is useful for generating initial placements only. Some of the blocks which are strongly connected possibly you have place them side by side, that is all nothing else.



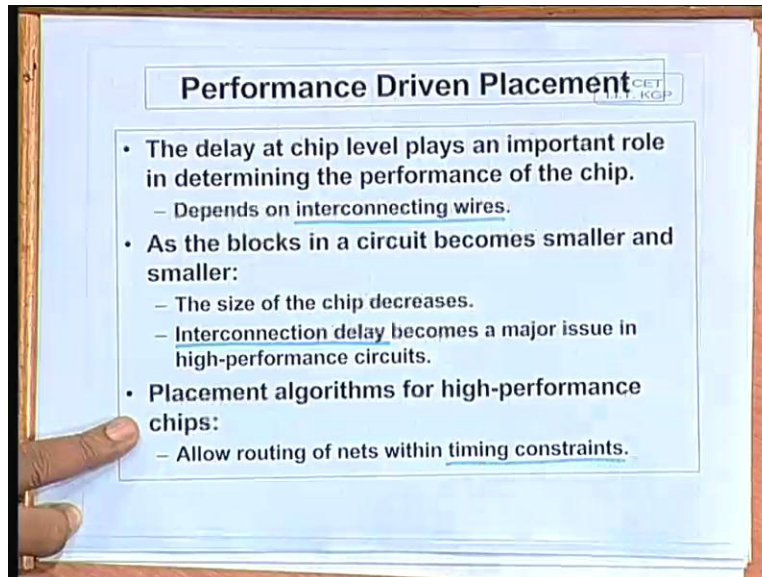
(Refer Slide Time: 44:40)



So the algorithm will look like this B is set of blocks to be placed you select one block S from B place it in the layout remove that block S from B. So while B is non null repeat select a block X from B the selection will be based on that weight I told you place X in the layout in a suitable place. Take it out from B and iterate simple ok. So to summarize the algorithms that we have seen so far. Well this cluster growth and a version of the force directed method this can be use to generate an initial partition starting from an empty layout. You can systematically grow the placement to get a tentative you can say final placement.

There are the simulated annealing and simulated evaluation algorithms we have seen which depending on probabilistic search techniques improve on a improve on a given solution or a set of solutions and you have seen that those algorithms typically give very good results for placement. And finally we have seen some other methods based on partitioning breuers algorithm and the terminal propagation that heuristic. Now there actually many work which have been reported also which can be broadly categorized into something called performance driven placement. Well we are not only worried about the area. We are not only worried about the total inter connection length but also the overall performance.

(Refer Slide Time: 46:39)



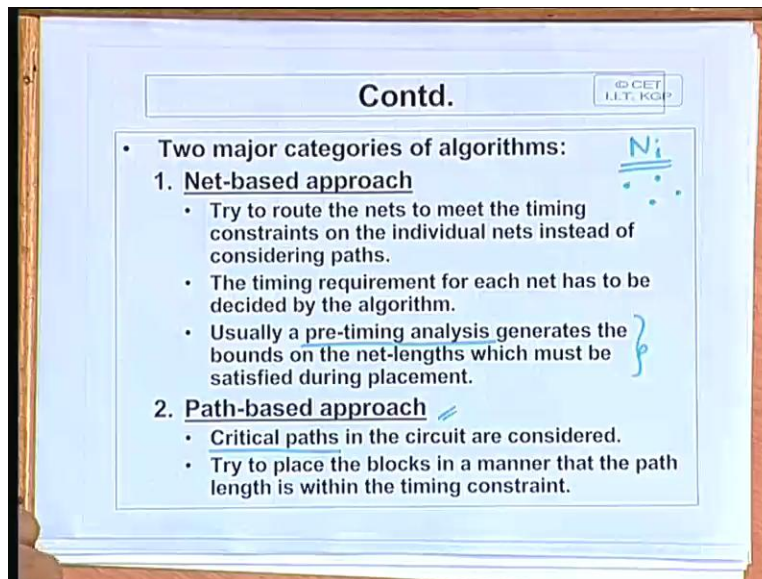
So here we are giving more emphasis on interconnecting wires and their lengths in particular the critical nets. See this performance driven placement has become important in the modern (( )) (46:57) because today in a VLSI chip the size of the basic components or the gates or model whatever you call they are reducing in size their speed is also increasing. Now in comparison the interconnecting lines which have also become very narrow and they are very closely space together they are some other parasitizing delays which are coming into the picture. So finally it will be the interconnections which will be dictating the final speed of the chip actually how fast the chip will work on. So interconnection delay this has become a major issue if you want to make a circuit which will be running as fast as you want to and Yeah.

Heat generated we did not take into account that is of course an issue. See normally what happens is that many placement tools like you can look at some of the tools which are having which are available commercially. So while doing a placement it will not take care of the heat. But there is a tool using which you can observe a heat profile of the total floor area you will find that some portion of the chip will get more heated than the other then through manual intervention. You can modify the placement. This is a semi-automated process of course there are some tools which have come up now which also try to do it in an automated way. But most

of the existing tools will handle the heat problem in a semi-automated way. Well if you find some part are becoming red. Red means hotter than the other you may have to take out some blocks on some place. Place it somewhere else change some placement and again go back and iterate.

See this placement and routing this will consume most of you can say design time in the total VLSI design process. Because you might have got the net list. But after placing you will find that there are number of problems are coming in you are not meeting the performance requirements the heat problem is coming up. So in order to tackle this you may have to go back make some changes again come down again go back again come down this you will have to iterate. Most of the tools will not be able to handle this automatically ok fine. So the placement algorithms for high performance chips will allow routing to carry out within some specified ts. Now these algorithms are still not very well established some method have been proposed.

(Refer Slide Time: 49:50)



Broadly there are two major categories of such algorithms. One is based on the individual nets here you take one net at a time. Net means you take a net  $N_i$  which connects a set of points and with respect to this net you see that whether the timing constraints are met. See the idea is like this the user have specified some kind of an overall timing requirement or constraint ok. There is a tool which will take the user level constraint as input and will try to generate net level constraint as the output that be in order to meet this performance requirement I have to limit the total delay of this net to this much these are or range of delays should be between this and this. These kinds of constraints are generated and then for each of the individual nets the tool will try to place it or try to meet the timing constraint 2. 2 means within those ranges ok.

So the net based approach the individual nets are taken an attempt is made to route the nets so that those constraints are met. So the timing requirements of that set these are decided by the algorithm given the user constraint and the tool I just talking about this is called a pre timing analysis tool. This pre timing analysis tool will give you the bounds of the net delays well net delays will finally get transmitted into net lengths because lengths and delays are proportional. Ok. So this is broadly one approach where the pre timing analysis tool will give you a bound of

the net lengths and you try to meet the net lengths. (( )) (52:02) Timing constraint is given by the user I want to run my circuit at hundred megahertz for example. (( )) (52:10)

See, this kind of an algorithm this cannot be a placement algorithm individually. So here placement and routing should go hand in hand. Here just giving an estimate using semi perimeter method anything that will not tell you exactly what is the real delay. We will have to find out the exact routing path and see that whether the timing constraints are really met. So for performance driven thing placement and routing must also go on together. And the other approach is the path better path based approach where you again analyze the circuit and you try to identify the critical paths and you concentrate only on the critical paths try to optimize them as far as possible. These are two broad appr of course net based approach is more general in the sense that it can meet the user constraints in a much better way. But critical paths as I said that paths which are critical in one scenario after some modification they may not remain critical some other path may become critical.

Because as you move blocks around the lengths of the interconnections paths may also go on changing ok. So these algorithms as I said these are not independent algorithms they must go hand in hand with routing. Because when you are saying critical path when you are saying the net length you must have an exact estimate of that not rough estimate. Because by the rough estimate you may see you may say now that well I am meeting the constraint. But later on after routing you will receive that you are failing to meet the constraint ok. So continuing from here in our next class we shall be moving on to the next step in the physical design automation namely routing we will be looking at the different sub problems of routing. And we will be trying to see that what are the different algorithms people uses and what are the well issues are challenges out there. Thank you.