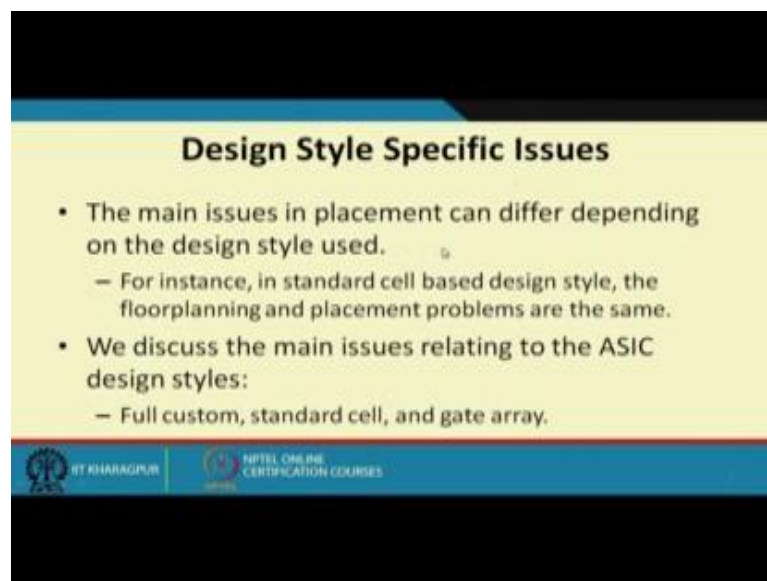


**VLSI Physical Design**  
**Prof. Indranil Sengupta**  
**Department of Computer Science and Engineering**  
**Indian Institute of Technology, Kharagpur**

**Lecture - 12**  
**Placement (Part II)**

So, in this lecture we continue with the discussion on placement issues in VLSI physical design.

(Refer Slide Time: 00:34)



**Design Style Specific Issues**

- The main issues in placement can differ depending on the design style used.
  - For instance, in standard cell based design style, the floorplanning and placement problems are the same.
- We discuss the main issues relating to the ASIC design styles:
  - Full custom, standard cell, and gate array.

IIIT KHARAGPUR | NPTEL ONLINE CERTIFICATION COURSES

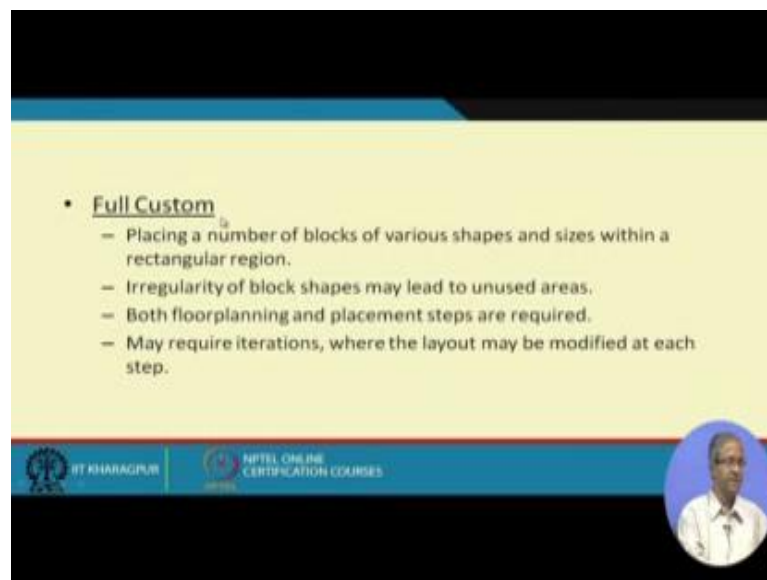
So, in this second part of the lecture first we talk about some of the design style specific issues in placement. Now I means I have repeatedly mentioned earlier also that many of the steps in VLSI physical design depends very much on the exact design style that you are following, in creating your final layout; while it can be a full custom, it can be semi custom, standard cell based, it can be gated or a FPGA based and so on.

So, here we shall see that with respective placement how the choice of the design style can impact or can affect exactly what we want to do, or what we intend to do during the placement phase. So, the main issues that arise out of the placement problem, this depends on the design style used you see this also I mentioned earlier that in the standard cell design style for instance the floorplanning and placement problems are the same; because you just recall that in a standard cell based design, the places where you can put a cell are limited, it is only along the rows in which you can place the cells. So, when we

talk about floorplanning you are tentatively assigning a location for these cells. Now that location itself will have to be within one of those rows. So, once you put a cell in one of the rows well you are as well completing the placement also, because placement will also mean the same thing placing the cells in one of the rows. So, here there is one design style example where floorplanning and placement does not make any difference actually right.

So, we discuss the ASIC design styles in particular full customs standard cell and gate array and the main issues involved there in for placement.

(Refer Slide Time: 02:50)

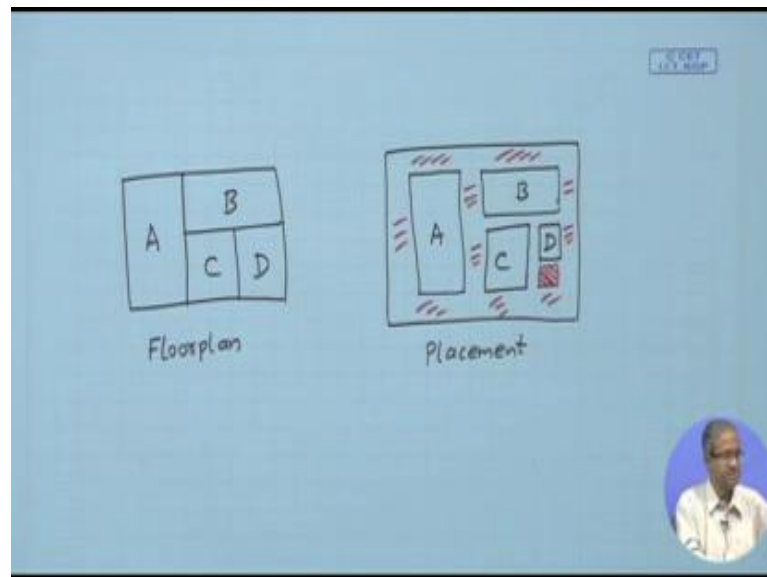


So, we start with full custom; now in full custom we have seen that this completely unrestricted kind of a design style that we have this silicon floor, and we can place any block anywhere you want to and also the shapes, the sizes, the aspect ratios of the blocks can vary, there is no restriction per se of course, you can mix the other design styles also some part can be a standard cell base, some part can be full customs etcetera. But in general for full custom base design style this is true. So, the placement problem will consist of placing a number of blocks of various shapes and sizes within the layout area that is available to you.

Now, the block shapes may not be very regular, because of which some area may remain unused. For full custom design style I mentioned earlier also that explicitly you require both the floorplanning and the placement steps, that however, depending on the

algorithm that we are using, you may need several iterations where you may start with a particular placement and you try to successively improve the quality of the placement in an iterative process for a (Refer Time: 04:22) step you are modifying the layout a little bit.

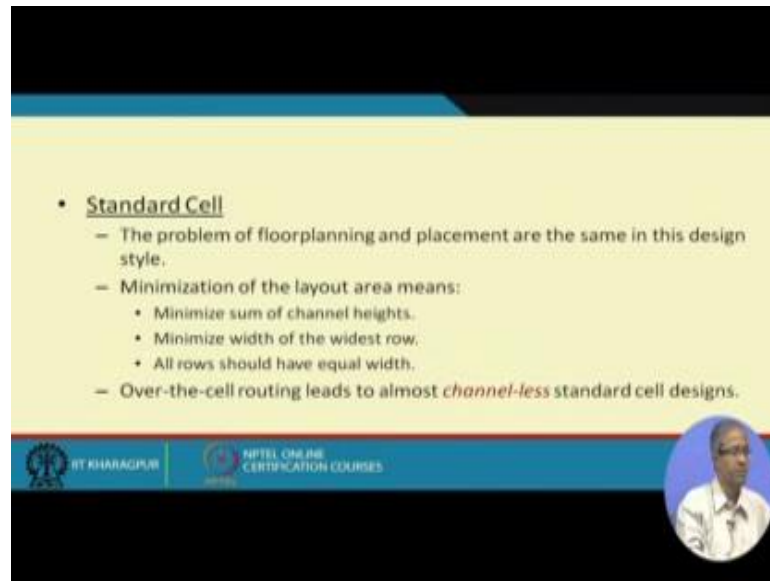
(Refer Slide Time: 04:35)



Now the floorplanning and the placement steps are required for full custom, let me just repeat it once more what exactly mean by that. Let us take an example again, let us say we talk about floorplanning; there is a block A, there is a block B, there is a block C and D this is one in instance or example of floorplanning for these 4 blocks.

Now, again finally, left next step of the placement, now the exact shapes and size of the blocks has been defined and also we have to keep some extra space for routing. So, now, your placement may look like this, this will be your block A, this can be your block B, your block C might look like this C here block might be smaller let us say like this. So, here you see in this example of a placement this is the placement. So, in this example of a placement you have explicitly steps and some spaces between the blocks and also around, using which you can complete the routing or the interconnections right and also you see because of the unevenness of the shapes and sizes of the blocks there are some areas like here which are unused. So, we are not really using this part of the area. So, this is one of the properties of placement in full custom design then there can be some area which is wasted.

(Refer Slide Time: 06:32)



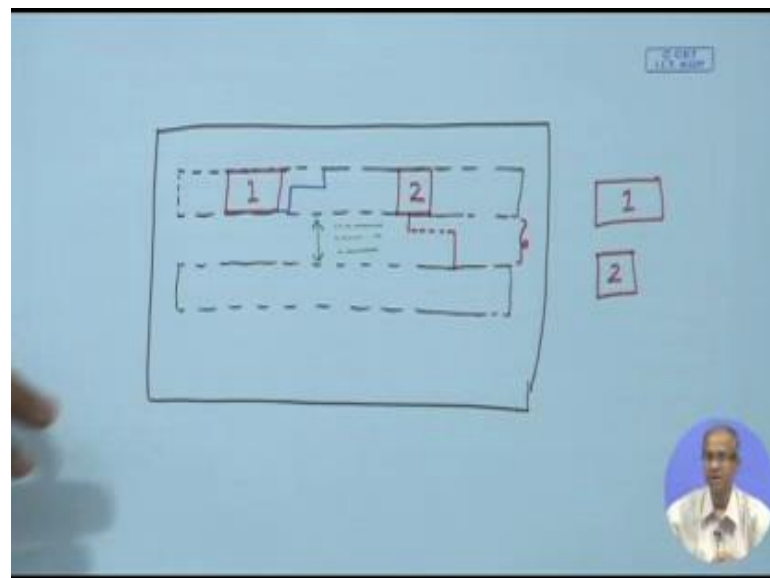
• Standard Cell

- The problem of floorplanning and placement are the same in this design style.
- Minimization of the layout area means:
  - Minimize sum of channel heights.
  - Minimize width of the widest row.
  - All rows should have equal width.
- Over-the-cell routing leads to almost *channel-less* standard cell designs.

BT KHARAGPUR | NPTEL ONLINE CERTIFICATION COURSES

Next let us conclude this standard cell design style; so here as I had mentioned here also the problem of floorplanning and placement are the same.

(Refer Slide Time: 06:47)



The diagram shows a rectangular layout divided into two rows by a horizontal channel. The top row contains two blocks labeled '1' and '2'. The bottom row is empty. To the right of the layout, two separate boxes labeled '1' and '2' are shown, representing the blocks to be placed. A small inset in the top right corner reads 'SCD (1) 2007'.

So, again let me just clarify this point. So, in this standard cell design style your layout area will look like this, where there will be several rows in which you are expected to put in the standard cells one by one; so I am show you two rows. Now suppose I have some standard cells. So, one shape like this get one let me let us write. So, I want put them somewhere. So, in the fourth lining I have to find out a tentative place for this blocks let

us call this block 1 and let us call this block 2. Suppose I decide to put my block 1 put here during floorplanning space, block 2 I may decide to put here.

Now, you see during floorplanning because of the constraint in the design style, we are forced to plan our these blocks to be placed only along this standard cell rows. In the placement also we have to do the same thing, this is these are the only places where you can place the block. So, what you have done during floor planning is exactly what we need to be during the placement also, so the two problems are the same in case of standard cell design. Now here of course, your objective will be the minimize the overall layout area, layout area means several you have allocate the blocks in a such a way that your interconnection complexity is minimized; that means, the number of tracks will require to interconnect; that means, the channels minimize some of the channel heights, like again coming back to this diagram channel height is this.

This is the place where the interconnection was to be laid out; depending on how many tracks will be laid out here the height of the channel will be fixed according to that right. So, one objective will be to assign the block in a such a way that you require less number of tracks therefore the height of the channel will be less and of course, we want to minimize the width of the widest row; like in the standard cell design it is desirable that all the rows are of the same width, that will provide you with an optimum utilization of the area, but again there can be a slightly means unequal blocks at the end so there can be a little mismatch.

So, even there is a little mismatch it should not be too much difference between the widths, we are trying to minimize the widest row in the standard cell layout. This is what I mentioned and the last point there is something called over the cell routing, this is also an option in sophisticated a routing algorithms, here you can get something called almost channel less designs. So, for this means let me just try to tell you with the help of this diagram again, now you see in this diagram you have kept a separate area for the interconnections. So, as I have said in this case what you do, you layout the horizontal and the vertical lines on two different layers or you complete the connections like this.

Now, these connections are created on metal layers which are typically on a higher layer surface as compared to the actual transistors which have been fabricated down below using the polysilicon diffusion in these layers. So, the metal layers are anyway means

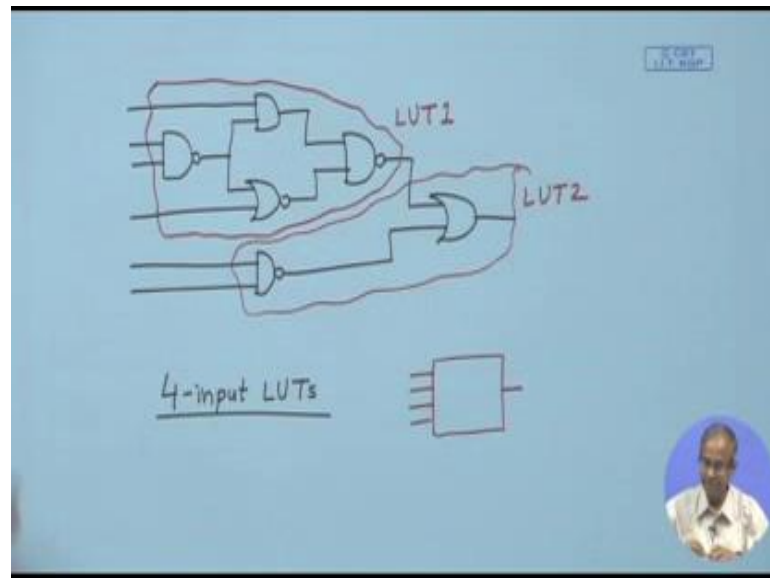
they are created on our layer which is above the transistors. So, because they are above the transistors, so no one is preventing you from creating interconnections like this. Let us say some of the point here you can make an interconnection like this, this over the cell you are drawing the interconnection lines above the cell on a different layer of course, this is called over the cell routing. Now if you can complete your routing means in this way over the cell manner, then you do not need this separate channel at all. So, you will be getting a virtually a channel less layout of standard cell, which will be much more compact again fine.

(Refer Slide Time: 11:59)

- Gate Arrays
  - The problem of partitioning, floorplanning and placement are the same in this design style.
  - For FPGAs, the partitioned sub-circuit may be a complex netlist.
    - Map the netlist to one or more basic blocks or LUTs (placement).

So, continuing with the next one gate arrays; gate arrays here what we have we have an area of gates. So, when you do partitioning you are rating the partitions you are trying to put each of the partitioned elements somewhere, floorplanning you are doing the same thing, placement also you will be the same thing. So, ultimate goal of all these steps will be to place a gate in to a particular location in the gate array. So, for gate array design style the partitioning floorplanning placement all this 3 can be merged together. They really do not mean until different, but for FPGAs; however, the problem is slightly different. So, here you are not mapping of particular gate like.

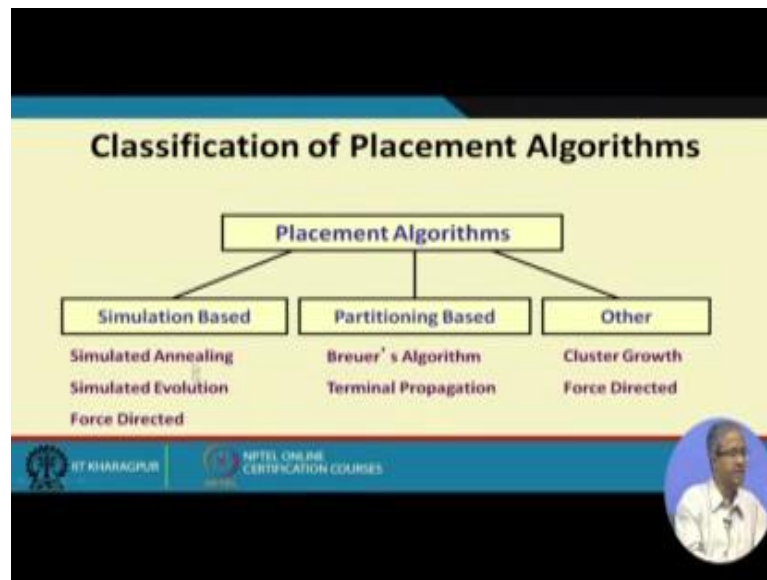
(Refer Slide Time: 12:55)



Let me take an example for FPGAs (Refer Time: 12:58). Suppose I have a circuit netlist like this. So, I am taking an example.

Suppose I have a circuit like this, there are several gates here; now suppose I want to map it into an FPGA where you have 4 input lookup tenures right. So, as I said for this kind of a scenario it is not that you are mapping one gate into an LUT, but 1 u general circuit or sub circuit, which will be having 4 inputs and 1 output can be mapped into an LUT. So, in this example for instance you can map this part of the circuit into one LUT let us call it as a LUT 1 and the remaining part again which will have a 3 inputs and one output into the second LUT. So, the mapping for FPGAs is much simpler. So, you take your netlist tree and from the netlist tree you actually find out sub circuits from the graph which will be having upto 4 inputs and 1 outputs and do a some kind of graph partitioning. So, there are many good algorithms for graph partitioning, which exists right. So, this is what is done for FPGAs.

(Refer Slide Time: 14:51)



Now, talking about the placement algorithms, the placement algorithms can be classified into broadly these classes one are based on simulation base; simulated annealing, simulated evolution or genetic algorithm force, directed placement these also a very interesting method we shall be looking at this. There are methods where you are creating placement out of partitioning process, there are methods like Breuer's algorithm and some modification terminal propagation and there are some other methods also cluster growth force directed also you can categories in the other category. So, it can be used to create an initial cluster, which can be cluster or means some kind initial placement which can be improved during the simulated annealing process for example, so let us see some of this algorithms.



(Refer Slide Time: 15:53)



**Simulated Annealing**

- Simulation of the annealing process in metals or glass.
  - Avoids getting trapped in local minima.
  - Starts with an initial placement.
  - Incremental improvements by exchanging blocks, displacing a block, etc.
  - Moves which decrease cost are always accepted.
  - Moves which increase cost are accepted with a probability that decreases with the number of iterations.
- Timberwolf is one of the most successful placement algorithms based on simulated annealing.

BT KHARAGPUR | NPTEL ONLINE CERTIFICATION COURSES

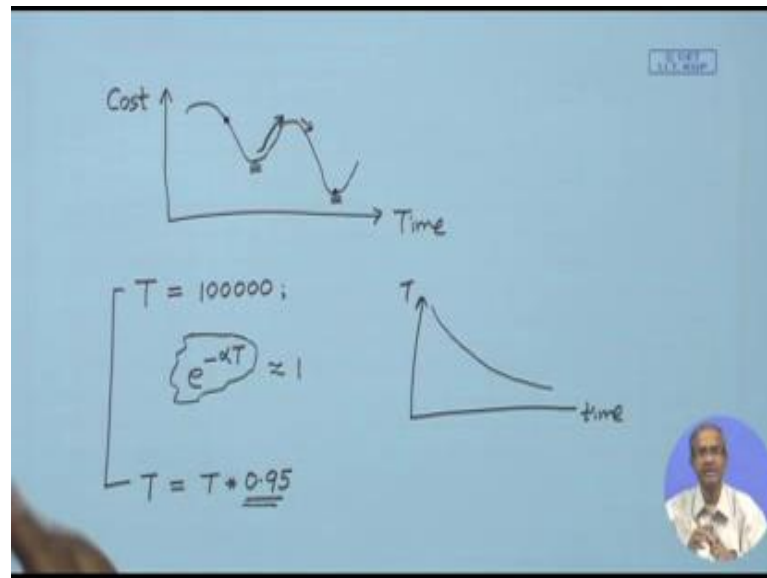


Simulated annealing there is an algorithm which has been very popular with respect to the problem of placement. We had talked about simulated annealing earlier also, let us spend a little time to recall what you had said about simulated annealing and let us say how we apply this tool you can say tool or algorithm whatever you say, to this particular problem of module placement. So, simulated annealing I mentioned this that this is a process which simulates the annealing process in metals or glass. So, when we create metal or glass crystals what you do, we melt to a high temperature for it is converted into liquid and then you follow a cooling sequence. We slowly cool the liquid state that material and the material will be finally, crystallizing and will take the shape of the material that we want it to have.

So, the things is that at high temperatures the atoms and molecules are very volatile, they are moving around very fast, but as the temperature is going down they are getting more and more stabilized. So, something similar is done here, some analogy is drawn that initially things might change, but as you move in time things will become more and more stable. So, the kind of analogy that is drawn is like this, that you start with the initial placement well; that may represent that initial liquidly of state of the material; you start with an initial placement which you want to improve. Then you defined a number of moves like for example, exchanging to blocks, moving of block from one position to another, changing the orientation of a block etcetera, which might resulting incremental improvements.

So, if you get an improvement you accept it, moves which decrease the cost are always accepted, but; however, if you find that you make a move, but the cost is increasing so it is not that you will always be rejecting such a move. So, you can reject that move, but that move will be accepted with a probability that decreases with the number of iterations, this strategy helps in avoiding the process getting trapped in a local minima.

(Refer Slide Time: 18:53)



I can see in any optimization problem, so usually for complex problem you will be having a behavior like this, cost with time so as you carry out the optimization, in your find that your cost is decreasing again increasing again decreasing. So, there can be a several minima points. So, if you are say here, and then you are using a purely gradient approach to decrease the cost then you will get stuck here. But if you follow some over snoop, it is likely that you may make a move in this direction, so you move on the other side, and you can possibly fall in this valley and reach this solution which has a even lower cost.

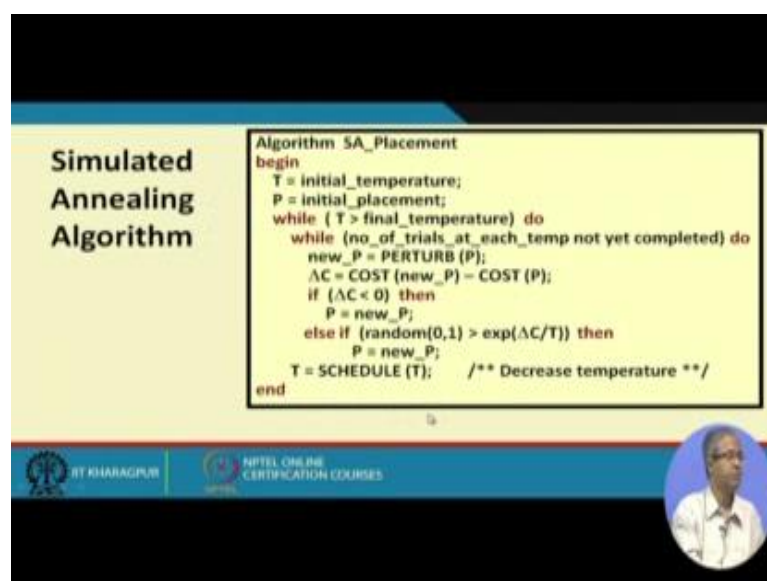
This is the basic idea behind saying that you are avoiding local minima coins, these are local minima coins. Now another thing is that I said that you accept the words moves with the probability that decreases with time. So, how do implement this? Say if we implement in this way while in terms a programming, you can keep a variable that represents the temperature. Let us say I start by initializing this temperature to high value let us say 100000 right and at the end every iteration, I modify the temperature let us say

T equal to T multiplied by let us say 0.95 let us say. So, in every step I will be decreasing this temperature progressively like here 100000, after one iteration it will be 95000, after 7 iteration it will be even less and so on.

So, if you plot temperature verses time, this will also go down following certain slope at the curve, depending on this multiplication factor and in between you can use of a parameter, let us say  $e$  to the power minus  $\alpha$  into  $T$ ,  $\alpha$  is a constant. So, you accept a words move with probability this. So, you see as the  $T$  is high for this probability means,  $e$  to the power minus a very small number. So, it is this number is actually goes to 1, but as temperature decreases this will become less and less. So, it will be these probabilities were actually go down. So, the idea is that there is initially towards the initial parts of the iteration, you are saying that you may accept words moves many a time, but as time progresses the temperature drops. So, here we shall be reducing the probability, the chance of accepting a words move becomes lesser and less this is the idea.

So, one of the very popular simulated annealing tool that was developed was called timber wolf. In fact, there was a pointing time for timber wolf has the best known placement algorithm, and it was mainly used for creating placement for standard cell base designs and of course, it can be used for general full custom designs also.


(Refer Slide Time: 22:41)



**Simulated Annealing Algorithm**

```
Algorithm SA_Placement
begin
  T = initial_temperature;
  P = initial_placement;
  while ( T > final_temperature ) do
    while ( no_of_trials_at_each_temp not yet completed ) do
      new_P = PERTURB (P);
      ΔC = COST (new_P) - COST (P);
      if ( ΔC < 0 ) then
        P = new_P;
      else if ( random(0,1) > exp(ΔC/T) ) then
        P = new_P;
      T = SCHEDULE (T);  /** Decrease temperature **/
    end
  end
```

BT KHARAGPUR | NPTEL ONLINE CERTIFICATION COURSES



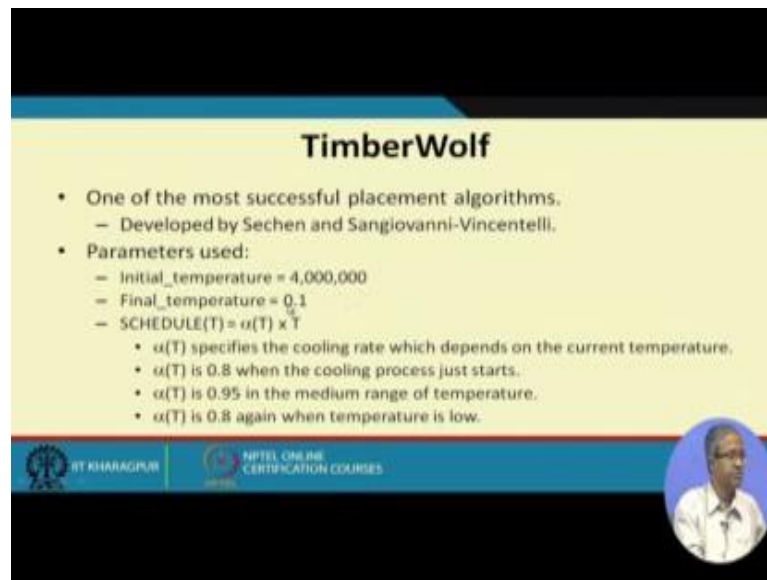
Let us say how timber wolf works; this is the overall pseudo code of this simulated annealing process or algorithm, let us see the different steps. As I said we use a variable  $T$  indicating the temperature, to initialize it some large value like the example I said 100000 that will be the initial temperature; and you start with an initial placement of the blocks. So, initially we can start with the random placement may be or some placement which has been created to some kind of intelligence will see those later. Now in this while loop I can create a final temp I can define a final temp let us say final temperature I can define as a 0.5. So, in every iteration I am reducing the temperature  $T$  equal to schedule  $T$ , this can be as I said  $T$  equal to  $T$  multiplied by 0.95. So,  $T$  is decreasing. So, at point in time  $T$  will be less than the final temperature, then you will be going out of the loop otherwise you will be repeating.

Now, at every temperature value  $T$  you are carrying out a number of inner books or iterations, while number of trials at each temperature not yet completed. Let us say you decide to create 100 trial moves, it means at every value of  $T$ . So, for every value of  $T$  the inner value will looping 100 times. So, what you do? You make a trial move you call a function call perturb, which makes small changes to your placement  $P$  that will be your new placement.

So, you have a cost function, you applied to your new part new placement and also the original placement and see the difference in  $\Delta C$ .

You see if  $\Delta C$  is negative; that means, you have found out a better placement. So, you accept it and make it as your  $P$ , the place  $P$  by this new placement. But otherwise your cost is increasing is positive then you use a function like this, if a random number between 0 and 1 generate a random number is greater than  $e^{-\Delta C/T}$ , this creates that condition that if your probability towards the initial part of iteration is larger, but as  $T$  increases the probability will be a smaller, this condition takes care of that. So, with this probability the random is greater than that then only you accept this words move I know repeat this process; look at this is very simple the process is very simple right.

(Refer Slide Time: 25:50)



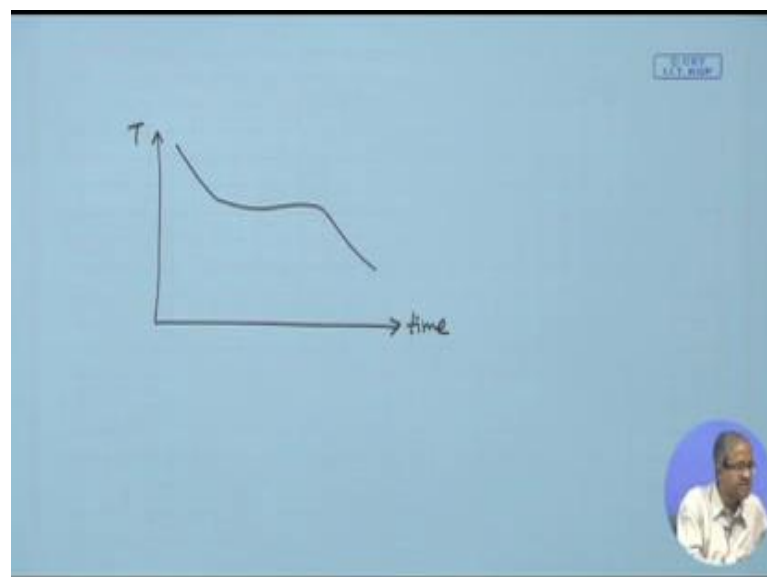
**TimberWolf**

- One of the most successful placement algorithms.
  - Developed by Sechen and Sangiovanni-Vincentelli.
- Parameters used:
  - Initial\_temperature = 4,000,000
  - Final\_temperature = 0.1
  - SCHEDULE(T) =  $\alpha(T) \times T$ 
    - $\alpha(T)$  specifies the cooling rate which depends on the current temperature.
    - $\alpha(T)$  is 0.8 when the cooling process just starts.
    - $\alpha(T)$  is 0.95 in the medium range of temperature.
    - $\alpha(T)$  is 0.8 again when temperature is low.

IT KHARAGPUR | NPTEL ONLINE CERTIFICATION COURSES

So, in timber wolf, the initial temperature was set to 4 million, final temperature was 0.1 the schedule was not as simple as  $x$  at 0.95 into  $T$ , but it was some multiplicative factor who changes with time like when the cooling process just starts; you start with alpha  $T$  equal to 0.8, but in the medium range of the temperature you make it 0.95, towards the end you again make it 0.8.

(Refer Slide Time: 26:28)



So, what will happen is that the cooling sequence will actually be like this initially it will be cooling at a certain rate, then it will be leveling up, then again it will cooling until

here. So, it will be something like this temperature verses time curve. So, it this is has been decided or arrived the point through experimentation, so by using various different kinds of the schedule function, the authors Sechen and Sangiovanni-Vincentelli they found that this kind of a function works the best for placement.

(Refer Slide Time: 27:00)

The slide is titled "The PERTURB Function" and is set against a light yellow background with a blue header and footer. The main content is a bulleted list of three moves: M1, M2, and M3. M1 is "The displacement of a block to a new location." M2 is "The interchange of locations between two blocks." M3 is "An orientation change for a block," with two sub-points: "Mirror image of the block's x-coordinate." and "Used only when a new configuration generated using alternative M1 is rejected." The footer contains the logos for IIT KHARAGPUR and NPTEL ONLINE CERTIFICATION COURSES, along with a circular portrait of a man in a white shirt and tie.

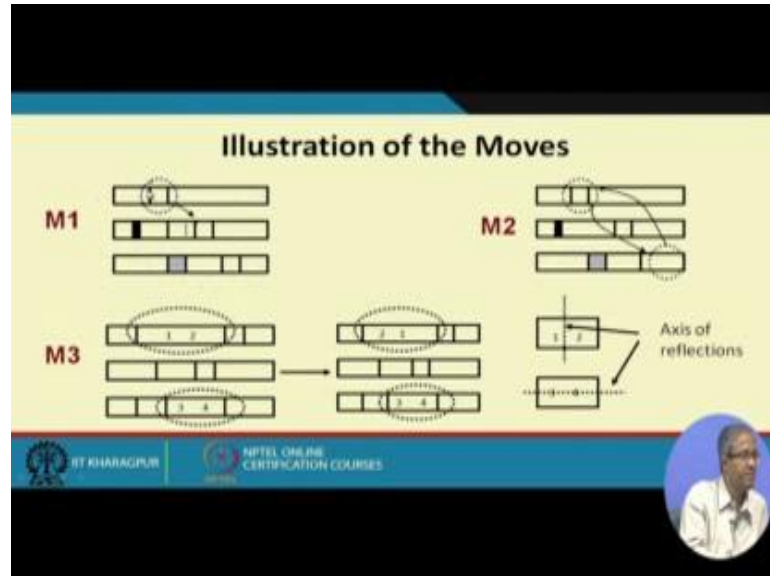
So, in the perturb function they used 3 different these ultimate notes, this M 1 M 2 and M 3, this M 1 indicates the displace a block randomly selected block to a new location. See this block and also the new location both of these at typically selected randomly.

The second move M 2 says you pick up two random blocks you interchange the locations. The third move says you pick up a block at random, you change the orientation. Orientation means you take a mirror image, mirror image means you have a block you either take a mirror of it in the x direction rotated it in the x direction, or rotated it in the y direction well.

Rotating in the y direction is applicable for full custom design, for standard cells you rally cannot do it because you recall in standard cells with the runs on the top and the ground runs on the bottom; if you rotate in this way, ground will go up and with the will go down, but for general full custom you can do this alright fine. And these moves M 1 M 2 M 3 are chosen at random again and m 3 is chosen (Refer Time: 28:28) was done that only when this M 1 whatever you are trying to do this was rejected; for example, you randomly select a new location to move, but you find that already the new location is

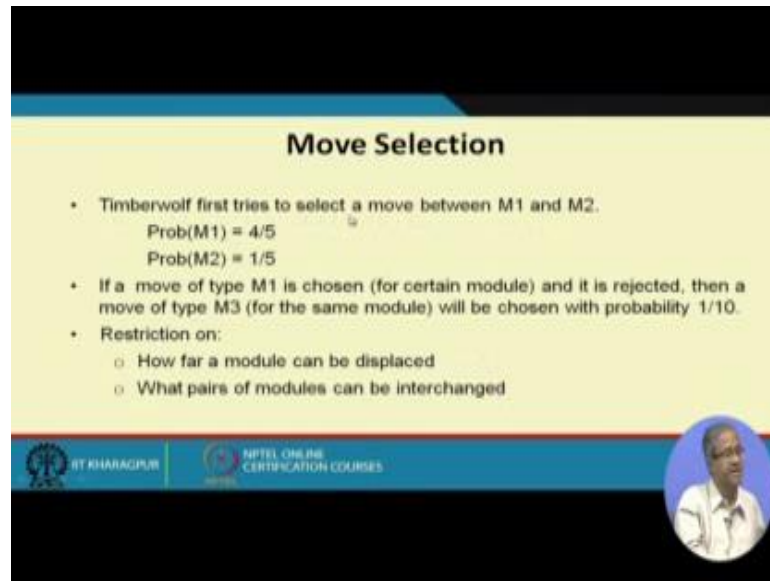
occupied, you cannot make the move then only you try out any 3, this was a strategy which is followed.

(Refer Slide Time: 28:50)



So, some illustrations, these illustrations have standard cells. If M 1 move says; so I pick up a block at random, I move it to a new location let us say this block will be moved here. This M 2 says I can pick up two blocks at random say this and this, and if space permits I will exchange them; I will going this block here and I will move these block here and M 3 as I have said if I have a block like this, I will take a mirror image this one is left to is a right it will become 2 or 1 or I can also reflect it along the y direction. So, this 3 4 will remain 3 4, but vertically its orientation will get changed. So, the block will rotated in this fashion got it, but again I said for standard cell this kind of vertical rotation does not make a sense, this rotation is this will be allowed only for full custom design style placement for there, fine.


(Refer Slide Time: 30:05)



**Move Selection**

- Timberwolf first tries to select a move between M1 and M2.  
Prob(M1) = 4/5  
Prob(M2) = 1/5
- If a move of type M1 is chosen (for certain module) and it is rejected, then a move of type M3 (for the same module) will be chosen with probability 1/10.
- Restriction on:
  - How far a module can be displaced
  - What pairs of modules can be interchanged

BT KHARAGPUR | NPTEL ONLINE CERTIFICATION COURSES



Now, selection of the moves was done like this, first timber wolf selected and moved between M 1 and M 2; for the probability of M 1 was a 80 percent 4 by 5, probability of M 2 was 20 percent 1 by 5. Now as I said earlier that if you select M 1 and if it is rejected; that means, you cannot move it then a move of type M 3 is done with probability 0.1 10 percent probability, but when you make the moves also there is something you want to look at, there are some restrictions we want to impose how far a module can be displaced and what pair of module can be interchanged like as I said to simulate the properties of the annealing process, initially I am allowing lot of volatility blocks may move over longer distances, that as time progresses I will not allow such random or drastic moves, only blocks in the closer virility can move around this is the idea.




(Refer Slide Time: 31:15)

### Move Restriction

**Range Limiter:**

- At the beginning, R is very large, big enough to contain the whole chip.
- Window size shrinks slowly as the temperature decreases. In fact, height and width of R  $\propto \log(T)$ .
- Stage 2 begins when window size are so small that no inter-row modules interchanges are possible.



Rectangular window R

BIT KHARAGPUR | NPTEL ONLINE CERTIFICATION COURSES

So, here we use some kind of a range limiter, the either is this range limiter is like a rectangular region like you imagine a rectangular space, rectangular window. So, initially you start with this R which covers the entire layout area, which means the moves are allowed anywhere within R, but as the iterations progress this size of the R will becomes smaller and smaller. So, the range like in one stage it can be mismatch; that means; only you can move within this region indicated by this rectangular area, not beyond that. So, window size shrinks as the temperature decreases. So, this is the idea.


(Refer Slide Time: 32:06)

### The COST Function

- The cost of a solution is computed as:  
$$\text{COST} = \text{cost1} + \text{cost2} + \text{cost3}$$

where cost1 : weighted sum of estimated length of all nets  
cost2 : penalty cost for overlapping  
cost3 : penalty cost for uneven length among standard cell rows.

  - Overlap is not allowed in placement.
  - Computationally complex to remove all overlaps.
  - More efficient to allow overlaps during intermediate placements.
    - Cost function (cost2) penalizes the overlapping.



BIT KHARAGPUR | NPTEL ONLINE CERTIFICATION COURSES

So, the cost function there are 3 components to it, first one is the weighted sum of the estimated length of the all nets, we have already seen earlier how to estimate the cost of the nets. Cost two is during placement you are not here, you are not actually this allowing overlapping, but you know that overlapping is not allowed. So, instead of disallow overlapping, what you would; you are saying the blocks can overlap, but the penalty for overlapping will be of very high cost. So, anywhere when you are minimizing the cost the overlapping cases will get eliminated automatically ok.

So, cost two is the penalty cost for overlapping, and cost 3 is the penalty for uneven length across the different standard cell rows, because you want that all rows must be approximately of this same width, but if there is a difference you incur a penalty of cost 3. So, this is what I mentioned the cost function cost 2, penalizes the overlapping and similarly cost 3 penalizes the unequal lengths of the rows.

(Refer Slide Time: 33:32)



So, to summaries timber wolf was one of the very successful placement tools, that was used for standard cell base designs, but this with some modification can also be used for the other design styles like full custom. So, we come to the end of this lecture, in our next lectures we shall be looking at some other techniques or algorithms for placement.

Thank you.