VLSI Physical Design with Timing Analysis

Dr. Bishnu Prasad Das

Department of Electronics and Communication Engineering

Indian Institute of Technology, Roorkee

Week - 04

Lecture 21

Partitioning Algorithms

Welcome to the course on VLSI Physical Design with Timing Analysis. In this lecture, we will discuss about different partitioning algorithms. So, the content of this lecture includes partitioning algorithm and there is some classification of different partitioning algorithms. Then we will discuss about different terminology involved in the partitioning algorithm. Then we will discuss the basic algorithm of the Kernighan-Lin or KL algorithm. So, what is a partition algorithm? So, the partition algorithm is a type of algorithm where it takes some inputs and produce some output. So, it takes the set of components and the netlist as input. So, it takes some of the set of components, the components basically the logic gates and netlist is basically the connection between the connection between the logic gate. So, this is the input to your partitioning algorithm. So, what is the final aim of partitioning algorithm? It minimizes or it reduces the number of nets basically are crossing the partition boundary. So, I have basically this block is there. This is the original design. I partition into two parts. Let us say this is P1 and P2 such that the number of connection between them should be as minimum as possible. So, number of nets crossing the partition P1 and P2 is as minimum as possible. So, here we have to have two things. One is called the objective function. This first one is the objective function. What is the main objective? Whether my total number of nets is the objective or whether my speed is the objective. So, there are different types of objectives are there and I have some set of constraints are there.

So, I have objective function. This is the first one and the second one is the set of constraints. So, the objective function can be reducing the number of nets or improving the speed of your design and the constraints might be the number of terminals and area constraint and also it depends upon the different design style. What is the design style and different design level? So, design level means whether you are going by architectural level of design or logic level of design or physical level of design that is your different design level. Then design style is whether it is a full custom or a semi-custom or gate array like all these different types of design style. So, depending upon all these design level and design style, we have to fix our objective function and constraint. So, the partitioning algorithm is basically classified in three ways. It basically it can be divided based on the availability of the initial partition. So, there are set of algorithms are used for finding the initial partition and there are set of algorithm is based on the nature of the algorithm, type of the algorithm and there are some algorithms are based on the process used for the partitioning. So, there are three different types we can divide based on how they are used to partition the overall design. So, based on the availability of initial partition, we have few algorithms and constructive algorithm is one of them. The first one is a constructive algorithm. So, what is the input to this constructive algorithm? You have the circuit components or the logic gates are the inputs and the netlist basically the connection between them that is input and the output is a set of partition. Set of partition means P1, P2, P3 so on and a new netlist. What is the new netlist? Because the original netlist is a netlist and what is the new netlist? Let us say I have the actual netlist is there in the actual design or original design, then it has a netlist let us say N. Then whenever I partition it to P1 and P2, P1 and P2. Now I have a netlist for P1 that corresponds to N1 and P2 has a netlist corresponds to N2. So, this original design has a netlist N. Now I have two new netlist N1 and N2, but they are connected by some interconnect between them. So, that is why this N1 and N2 are called the new netlist. So, these are used for the initial partition. So, this is used for some initial partition. But whenever I am thinking it is a initial partition that means that if I using a constructive algorithm I will not get a final solution. I get a partition which is something not close to my actual optimal solution, but I can get to that initial partition in less time. Now I have a iterative algorithm.

So, the iterative algorithm takes set of partitions actually. Here it takes a set of logic components. So, that is the difference and the netlist. Then what is it give you the output? It improves the set of partitions. Earlier partitions and the new partitions with a modified netlist. So, the each of the netlist is now modified. And what is the main difference between the constructive algorithm and the iterative algorithm is that in case of constructive algorithm I am giving logic gates and the netlist as input and I am getting set of partitions and the new netlist as the output. But in case of iterative algorithm I am giving a set of partition I am not giving the original logic gates actually. So, here I am giving a set of partition and the netlist and I am getting an improved set of partition with the modified netlist. And it iterates continuously until the partitions cannot be improved further. Whenever there it will it cannot be improved further there it will stop there it will stop. So, these are the two different types of algorithm based on the availability of initial partition. So, now we classify this algorithm based on the nature of the algorithm. Nature of the algorithm is that nature of the algorithm means that if you if you have some basically input given to a algorithm the output is repeatable. Repeatable means it will give you the same solution all the time. The inputs are I have a piece of algorithm is there same partitioning algorithm. So, if I give some input I will get a deterministic output. So, I will get a deterministic output. So, here so the output is basically if the input is same your output is same. So, this is called deterministic solution or repeatable solution. Why because we are using some kind of deterministic function here which is if you give the inputs it will evaluate the same value all the time and it generates the same solution for a given problem the same thing. So, here there is no randomness is there all are deterministic. But there is another so this is the first category based on the nature of the algorithm. Then this is the second algorithm on the nature of the algorithm that is called probabilistic algorithm. So, what is the probabilistic algorithm is that it make use of random functions. It make use of some random functions actually. So, I have some functions there some of the variables are generated from random numbers. So, those in that case seed will determine the output of your algorithm. The seed will change for each iteration and your output will also change because my input seed changes the output of the random function will change and the solution will also be different. So, here we are using random function and random function uses random numbers.

This random number has a seed that seed will determine the random number value and the random number will determine the output of a random function. So, here we will get different solution for the same inputs even if my logic gates are same even if my netlist is same my output is different at each runs of the algorithm. So, output is different. So, solution is different here. So, there is one more category which is based on the process used for partitioning which is called a group migration algorithm. So, this is the one category called group migration algorithm. So, it starts from some randomly generated initial partition. So, we have some algorithm we discussed like constructive algorithm and deterministic algorithm. So, we discussed about those deterministic algorithm and the constructive algorithm to find the initial solution. So, that initial solution is now input to the group migration algorithm. It is going as a input to the group migration algorithm. So, what happens inside a group migration algorithm is that we move component between the partition. So, let us say I have some logic gate in one partition. Let us say partition 1, I have partition 2. Let us say this is a G1 one gate, G2 is another gate.

I am just giving one example G3. So, let us say this is a G4 and G5. So, let us say this is connected with this and this is connected with this. So, and this is connected with basically let us say these two are connected. So, if I move G2 to partition P2, now I have basically cut cost will be reduced by 1. So, we are moving let us say the G2 to partition 2, then my number of cut will be reduced by 1. So, here what we are doing? We are moving a component or a logic gate between the partition to improve the partition. So, the G2 will move to the partition 2 to reduce the number of cut to 1. So, this is called your group migration algorithm. But one of the problem with the group migration algorithm is that it will fall into the local optimum.

What is local optimum? Let us say I have I can draw it here. Let us say I have this is my number of solution state 1, 2, 3, 4, 5 just I am giving an example and here the number of cuts. So, let us say if I get some profile like this. So, these are my local minimum. So, these are called local minimum. However, I have one global minimum is there. So, this is my global minimum. And these others are local minimum. So, basically it will fall into the local minimum which is not a good idea because I always look for a global solution to get better partitioning. So, now I will discuss about the simulated annealing or evolution based algorithm. Basically, the simulated annealing or the evolution based algorithms are based on random functions. So, it has a cost function which classifies any feasible solution actually and it allows set of moves that allows the movement from one solution to the other solution. So, whenever it moves one of the component from one partition to the other partition then we evaluate the cost function. So, what is the advantedge of one of the method followed in simulated annealing is that in case of deterministic algorithm, we always look for improvement all the time. But in case of simulated annealing we accept a move even if the solution is not good.

So, in case of deterministic algorithm we always look for optimal or better solution in each iteration. But in case of simulated annealing or evolution based algorithm we accept a solution even if it is a bad solution. Why we accept a bad solution? Because we have a hope that if I accept a bad solution I have a chance of getting a better solution in the future iteration. For example, let us say if I have accepted this solution at this point and I have a chance to go to the global solution. So, but in case of deterministic algorithm so this is not allowed to take the solution, but it is possible to take the solution in case of a simulated annealing or evolution based algorithms.

So, initially they started begin with some random solution actually. So, it starts with some random solution over time basically we move. So, basically what we do here is that we basically have some random solution. So, basically whatever I told in the previous slide is that I have some. So, this is my x axis is a solution state and this is the cost function y axis is a cost function. So, if these are my local solutions, these are my local solutions. So, in case of simulated annealing whatever I told earlier, so if even if I have some I can accept the bad solutions so that I can go here and finally I can go to the global solution. This is my global solution. So, the global solution I can get there is a chance of getting global solution in case of in case of simulated annealing if we run the algorithm for a sufficient amount of runs this is a global solution. So, this is very popular in industry to adopt this simulated annealing is one of the algorithms used in partitioning. So, there are other types of partitioning algorithms like we have performance driven algorithm then we have metric allocation. So, there are basically we discussed about several different types of algorithm which is divided into group migration is one category then you have simulated based is another category then we have performance basically the speed driven algorithm basically the delay or the performance is one of the objective function in this algorithm then we discussed about the metric allocation algorithm. So, then we in case group migration algorithm we have several algorithms are there K L this is called K L algorithm or kernighan lin algorithm is a popular algorithm and fiduccia mattheyses use is also fm algorithm is another popular algorithm. Goldberg burstein is another popular algorithm component replication is also used in case of group migration then ratio cut is another algorithm which is falls under the category of group migration then simulated annealing is a have two different category called simulated annealing and simulated evolution. So, this is used and these are the other different categories of algorithms.

So, now we will discuss the K L algorithm in detail. So, the K L algorithm has we have a we need to represent the basically the net list the logic gates or the components with the nets as a graph G (V comma E) where the mod of V is 2 n. So, mod of the V is a even number of nodes then we need the node V belongs to V has the same weight actually. So, each of the node is assigned the same weight value. So, this is the one of the constraint we use in case of K L algorithm then all the A's should have non-negative edge weight. So, all the adges has basically non-negative edge weight for these two are the heuristic used in this algorithm. These are the heuristic or assumption used in this algorithm. So, here whenever I am doing this scale party algorithm partitioning I divide this V which is having the all the gets into two disjoint subset. Disjoint means there is no overlap between the subset A and B with a minimum cut cost. So, A mod equal to B mod equals to n. A mod equals to B mod equals to n. So, let us take one example here. Here we have a total number of node or vertices is basically 10.

Okay. Total number of nodes is 10. Okay. And your n basically each partition should have basically 5 nodes. Each partition should have 5 nodes. Now this is a one partition. Okay. This is one initial partition. So, this is a group A. So, this upper side is a group A and the bottom side is called the group B. Now we have to define some cost function. So, this is the cost function. Okay. So, let us say if I move a node from one partition to other partition then what is the how I can evaluate its cost. So, this cost the cut cost or the cost of D of V of moving a node V is defined by two parameter mod of E c of V minus E n c of V.

$$D(v) = |E_c(v)| - |E_{nc}(v)|$$

E c of V is a set of these incident edges that are cut by the cut line. That is why it is denoted by c. Then E of n c of V is a set of these incident edges that are not cut by cut line. That is why it is called n c not cut. Okay. So, this is not cut and this is cut. So, here if I take an example here. Okay. So, node 1. Let us consider a node 1. So, node 1 has how many incident edges are there. So, there are three edges connected to node 1. Now, how many are cut by the cut line? How many are cut by the cut line is comes the E c of V. So, here this is 1 and this is 2. Okay. So, E c of V is 2. Is n c of V which is not cut by the cut line. Cut line means this is the cut line actually. This is your cut line. Which is not cut by the cut line is this red line. Okay. So, this is 1. So, n c of V is 1. So, this node 1, d of 1 is basically E c of 1 minus E n c of 1. So, this is 2 minus 1 is 1. Okay. So, this is for another example that node 9 is 3 minus 1 is 2. So, I need to erase this side to get some more information. So, here so, there are two points are there. Here the point 1, the point 2 is this one. Okay. So, high cost when d is greater than 0 indicates that the node should move. Okay. If you have d of that node is larger then you should move that node to the other partition. Okay. And if the low it d is less or negative indicates that the node should stay in the same partition. Okay. If the d is negative then it should stay in the partition. Because here if it is positive means it should move. Now, we have a conditions that in case of KL algorithm, we have a condition that each partition should have same number of nodes. Same number of nodes or gates, logic gates whatever you can say. So, if it has same number of nodes then I cannot move one node at a time. I can move one node to the other side, other node to the this side. So, I have so, I cannot move one node to the other side. I need to create a equal balance. So, let us say if I want to move G1 to this side, G2 should come to this partition. So, if I do like this then I need to find another cost function. So, what is that cost function if I move one node or one logic gate to the other and other logic gate to the this side. So, that is denote this cost function or the gain this. So, now I have a cost function, new cost function which is denoted by gain delta of G. If I am moving one node to the other and other node to the this partition then what is the my gain. So, gain of swapping a pair of nodes. If I do that I have to find d of A plus d of B minus 2 times C (A, B).

$$\Delta g = D(a) + D(b) - 2 * c(a,b)$$

Okay, this one is clear because I need to add the cut basically d if I move one node to the other what is the improvement that I need to add, but why I need to subtract that is very important.

The C (A, B) is a connection between A and B. So, if there is a connection between A and B that should be subtracted otherwise it will overestimate my gain. So, I need to subtract that value and it has been added in both the places that is why I have to multiply it by 2. If an edge exists between A and B then C (A, B) is 1 otherwise C (A, B) is 0. Okay, so basically this is C (A, B) should be 1 otherwise otherwise your C(A, B) is 0. If there is a node connected between G1 and G2 let us say like this if I write draw it in a different color okay if I have a line between them then I need to consider 2 times C(A, B) in this case.

Okay, so now I need to basically this gain whatever I have the gain delta G indicates how useful between if you swap 2 nodes between the 2 partition. If you have larger delta G then your total cost in the partition will reduce. Okay, so there is 2 nodes here this is 1 and this is 2. If the gain delta G indicates how it is beneficial for moving a node from one partition to the other and taking vice versa take the other node from one partition to the other. So, if what is the gain so that is there if the delta G is larger that your total cost will be reduced. So, basically I have considered 2 nodes here 2 and 8 okay. So, what is the D of 2 actually if you can see is how many edges are there to node 2 I have 1, 2 this is 3 and this is 4 okay. So, 4 lines are there how many are connected with the cut line this red line 3 are connected with the red line. So, this is 3 how many are not touching the cut line this is 1. So, this is 2. Similarly, here if you can see node 8 has 3 lines 1, 2 and 3. So, out of which 2 are connected basically touching the cut line. So, D of 8 is basically 2 minus 1 so it is 1. Now, I need to find out delta G okay. So, this delta G is between two nodes D is for each node. So, if I swap 2, 8 so my gain will be D of 2 plus D of 8 minus 2 times C 2, 8. So, is there a line connecting between 2 and 8, yes there was a line this line is connecting between 2 and 8. So, this line so I need to subtract 2 because it is used for 4 and it is used for 2. So, this 2 it is used here and it is used here so I need to subtract. So, if I can do that so here it is 2 plus 1 minus 2 this will be 1. So, basically this is the delta G so we have a delta G is greater than 1 so we are moving 2 and 8. So, if I move that 2 and 8 now this is my new partition. So, this new partition will have basically 2 and 8 is swapped that 2 will go to one partition and 8 will go to the other partition. So, this will reduce the total cut cost by 1 okay. So, earlier if you can see here I have 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11 correct. So, this many cuts are there. Now if I go by this approach I have 1, 2, 3, this is 4, this is 5, this is 6, this is 7, this is 8, this is 9, this is 10. So, if I swap the 2 nodes the cut cost will be reduced by 1. So, the maximum positive gain okay so this is a another basically factor because if I am doing this delta G for each pass what is the maximum gain is denoted by G of M okay for a pass. So, this maximum positive gain is basically the gain at a particular iteration where I am getting a maximum gain in changing the nodes okay or the swapping the nodes in a given pass. So, the maximum positive gain G M is the best prefix of M swap within a swap of sequence for a given pass. For one pass this G of M is giving the maximum positive gain among all the all the delta G. So, that is my the point I should take as the best partition in the given pass of the KL algorithm.

So, there are M swaps okay lead to the partition within the minimum cut cost encountered during the pass. So, then G M is computed as a sum of delta G K basically what we are doing is that whatever the delta G is coming for each of the passes that is summed over to find the G of M okay each iteration of your algorithm I am getting delta G I.

$$G_m = \sum_{k=0}^{m} \Delta g_k$$

Then I am every iteration it is accumulating the gain to find the gain at each iteration. Then I need to find the maximum which iteration has the maximum gain that should be taken as the best partition. So, in this lecture we discuss about the partitioning algorithm different types of partitioning algorithm. Then we discuss about the some of the terminologies needed for doing the partitioning.

Thank you for your attention. Thank you.