VLSI Physical Design with Timing Analysis

Dr. Bishnu Prasad Das

Department of Electronics and Communication Engineering

Indian Institute of Technology, Roorkee

Week - 04

Lecture 22

Kernighan – Lin (KL) Algorithm

Welcome to the course on VLSI Physical Design with Timing Analysis. In this lecture, we will discuss about Kernighan-Lin or KL algorithm. The content of this lecture includes different steps of the KL algorithm, then we will discuss partitioning of a circuit using KL algorithm. We will take an example and discuss how the KL algorithm is useful for partitioning. Then we will discuss about the what is the speed of speed of running the KL algorithm. So, this KL algorithm has multiple steps, we will discuss one at a time. So, step 0 is basically we have a initial partition. So we have a initial partition and see this a and b are the two partitions and your number of node is or the number of vertices is basically 2N. So a has n nodes, a or b has n nodes. So each partition has same number of nodes. So then we will go to the step 1. In the step 1, we have k equals to 1. So here we have to calculate the cost function d of v. For all nodes v belongs to v. So here we need to let us say we have 2N nodes, we have to calculate the v of v 2N times for one time for each node. So now after we find out basically v of each node, then we will define another cost function that gain delta G of k. So this is delta G of k which is defined as d of a plus d of b minus 2 c (a, b).

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

c (a, b) is a comes when there is a edge between a and b. So we need to find out where my delta G of k is maximum. So then only we can swap a and b and we will not change that a and b in the next iteration.

So in this process every iteration we are fixing 2 nodes because after the swap we are fixing those nodes. So the time will come when there is none of the nodes need to be moved. There is no nodes means all the nodes will be fixed. If all the nodes are fixed then we have to go to the step 4. Otherwise we will compute the d values for each node connected to a

and b which are not fixed then we will increase the value of k plus 1 then we will go to step 2. Step 2 will do what? It will find the delta G of k this one. For all possible combination of the nodes which are not fixed then we will find out which is giving me the maximum gain maximum value of delta G then that will be taken for moving from one partition to other and those nodes will be fixed. So this process will continue till all the nodes are fixed then we will go to the step 4. In this step 4 we will find a move sequence 1 to m. So m is something between 1 to k and each m it is basically accumulation of previous gain we are doing the accumulation of the previous gains.

$$G_m = \sum_{K=0}^{m} \Delta g_K$$

So your delta G of k will be G of m is summation k equal to 0 to m delta G of k. If your Gm is greater than 0 then you go to step 5. Greater than 0 means if your Gm is positive. If your Gm is positive then you have to continue the process will continue. If Gm is negative then stop. So these are the two case 1 and case 2. So then execute m swaps and reset the remaining nodes and go to step 1. So this is for the step 5. Now we will take an example and see how these things are happening. So this is the first path. So this is basically having basically your number of nodes V equals to 10 here. So your V is 10 here and you have a partition A this side is let us say partition A this side is partition B. So each one of them are having 5 nodes and all the nodes or vertices are not fixed all the nodes 1 to 10 are not fixed. Then how many cut cost is there? How many this is the cut line. Cut cost is basically the number of edges passing through the cut lines.

So here if you have seen 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11. So number of cuts cost is 11. So now we will go to calculation of D of V. So whenever I am calculating D of V in the first k equal to 1. So how many times I need to calculate D? Number of times the D is evaluated at this k equals to 1 is 10 which is proportional to 2N which is proportional to 2N. Now let us say how the D's are evaluated. First we will calculate the how the D's are evaluated. Let us take the node 1. Node 1 if you can see here I have 1, 2 and 3 edges. So the 1 and 2 are passing through the cut line and the 3 which is the which I wrote in the blue color is not passing through the cut line. So my D of 1 is basically 2 minus 1 is 1. Similarly I can calculate for any other node D of 9. Let us say this 9. So D of 9 is basically here 1, 2, 3 are the edge cut by the cut line which is 3. Then this edge is not cut by the cut line. So this will be 1, it will be 2. D of 9 is 2. So we have to calculate for all the D's. So now I need to choose which is giving me maximum gain delta G. So if you can see here ideally you have to do it for all the combinations but here looking at this evaluation I can see D of 2 and D of 9 has cost of D of B is highest.

So let us take those nodes for gain calculation or moving from one partition to the other. So if I take these two nodes D of 2 is 2 and D of 9 is 2 here my gain because there is a 2 and 9 there is no edge between them C of A, B is 0. C of A, B at this point is 0. So now

delta G of 1 is 2 plus 2 minus 0 which is 4. Now this 2 and 9 should be swapped. My gain is basically G of 1 is delta G 1 because it is a first iteration it is 4. After we do this then which node will be fixed? The 2 and 9 will be fixed. In this case 2 and 9 are swapped so the 2 and 9 are fixed and rest of the nodes are not fixed. So we have 8 nodes not fixed, 2 nodes are fixed in iteration 1. Now we will calculate D of B of each node.

So if you can see here if I look into this D values the number of times the D is evaluated is basically 2N minus 2K. So 2N in this case is 10 minus 2 which is 8. So you have number of times your D is evaluated is 8. So 1, 2, 3, 4, 5, 6, 7, 8. So now we can see which of the nodes will be swapped. If you can see here D of 1 is giving me a gain of 3 and D of 10 is giving me a gain of 3. So these two nodes are showing is basically giving highest gain. So I need to swap them. So ideally you have to do it for all nodes if you are doing algorithm but here we are visually finding the nodes. So this node 1 and 10 need to be swapped. So now in this slide if you can see so the C of 1, 10 because node 1 and node 10 there is no edge between them so this is 0. So the gain this will be 0 because of this, this 3 is coming because of this 3 and this 3 is coming because of this 3. So my delta G2 is basically 6 here. Now I need to accumulate all the previous gain with this present gain. So my previous gain is basically G1 plus delta G2. So my overall gain is basically 10. So now my number of cut cost will be reduced by 10 after doing this move of these two nodes. Now if I can see I have 4 nodes are fixed, 1, 2, 9, 10 are fixed and the rest of the nodes are not fixed. So here the K is basically 2 here. So now we have 4 nodes fixed, 6 nodes are not fixed. So now if I can see number of times D is evaluated in this iteration is basically 2N minus 2 into K. So basically if you can see 10 minus 2 into 2 here it is 6. So you have 4 nodes are fixed for them you do not need to calculate the D but rest 6 nodes you need to calculate the D. So 3, 4, 5, 1, 2, 3, 4, 5, 6 I need to calculate the D. See if I can look into this which node is giving me the best swap is basically all are negative.

So the maximum negative number will give me the best swap. So the 3 and 8 nodes are chosen actually minus 1, D of 3 is minus 1, D of 8 is minus 1. So if I do this my gain is now basically minus 4 means I am not getting any improvement in the cut cost if I go by this. However, I have to check if I can get a better solution in the future. So your G3 is basically G2 plus delta G3 which is giving me 6.

So now this is the new partition after this iteration. If I can see I have basically fixed node 1, 2, 3, 8, 9, 10 these are the fixed nodes and these 4 are not fixed nodes we need to calculate the D's for them. So number of times D is evaluated is basically 2N minus 2 into K. So 2N is basically 10 minus 2 into 3. So this will give me basically 4. So here you have 1, 2, 3, 4. So D is evaluated for the on fixed nodes 4, 5, 6, 7 in this case. Now that once the D is evaluated then I have to choose pair of nodes to get a maximum gain. So in this case if I can see 4 and 7 will give me the maximum gain. So the 4 and 7 is chosen. Now the gain becomes G4, the delta G4 is basically minus 4. My cumulative gain is basically G4 is basically 2. So this is after doing this pass my all of the nodes which are fixed is basically

8 nodes are fixed and 2 nodes are not fixed. So then I need to calculate the D for the 2 nodes. Number of times the D is evaluated is basically 2N minus 2 into K.

So 2N is basically 10 minus 2 into 4. So this number of times the D is evaluated is 2, this is 1, this is 2. So for these 2 on fixed nodes we need to help calculate the D. So there are 2 options, only 1 pair of nodes are there. We do not have any choice. So we have to choose these 2, 5 and 6 for the swap. So after you do the swap my gain becomes 0 here. There is no improvement in the cut cost whatever the 11 cut cost was there. Now the same cut will come at the final cut. So there is no improvement. Now we have this final step where K equals to 5. I have total cut cost is becoming 11 and all the nodes are fixed and I do not have any node to do D's evaluation. So I have basically iteration K equal to 1, K equal to 2, K equal to 3, K equal to 4 and K equal to 5. My x axis is basically K and y axis is basically my gm. So if I plot this one which iteration is giving me maximum gain that I should choose. So if you can see here my K equals to 2 is giving me my maximum gain. So this is corresponding to this node, this point in the graph. So that is the best choice. So if you can see here my gm is greater than 0 and the first m equal to 2 because first 2 swap I am getting the best result. So in the K equals to 1, the 2 and 9 nodes are interchanged or swapped and K equals to 2, my 1 and 10 nodes are swapped.

So this is my final partition after the pass one. So if I can see here my gain is positive. I need to iterate it till my gain is negative. So since my gain is positive more paths are needed until my gain is negative. gm should be less than equals to 0. So I have to go to the second pass. So the output of the first pass is the initial partition for the second pass. So here the cut cost is 1 and I have all the 10 nodes are unfixed. So the same procedure whatever was followed in the pass one the same procedure will be repeated here. So I have to calculate all the d's and from them which is giving me the best result that will be best gain that will be taken and swapped. So that method will be followed. So I am not discussing all the steps but this is the results for K equals to 2 and the gain is K equal to 2 is basically your g is basically minus 8. So here what is the basically your K values are your x axis is basically K and gm is basically your y axis for different values of K. So if you can see here your gain is becoming negative and finally it is settled to 0. So since your gain final gain and all the gains are negative so I can stop my algorithm here I do not need any further pass. So since I am gain overall gain is 0 so this is my initial partition my gm is there is no gm is 0 finally and so there is no further passes are needed we can terminate the algorithm here and this is my final partition. And in this case my cut cost was 11 now in this final partition the cut cost is become 1. So, this is the final partition is the best partition because I have reduced the cut cost by 10. Now we will discuss about the runtime of the K-L algorithm. So this runtime will tell my speed of a pair algorithm how fast my algorithm is working. So the K-L algorithm is basically determined the speed of the algorithm is determined by two factors one is my gain update and the pair selection.

So we will discuss how the algorithm speed depends during the gain updates and the pair selection. So here we have basically that so if I have whatever I discussed in the first iteration K equal to 1 I have to evaluate d basically 2n times K equal to 2, my d is evaluated basically 2n minus 2 basically this the first one is 0 and this is K equals to 1. So I have basically 2 K so it will be evaluated 2 times less than the 2n. So similarly, if I go in this method finally the d will be evaluated 0 times. So here what is happening is that my 2n, 2n minus 2i like that it is going.

So now basically what is happening is that my d number of times the d is evaluated is reduced by 2 in each iteration. So if I make a sum of that because how many times the K should run? How many times the K should run? The K should run for number of nodes by 2 which is n. So that is why your this is this n comes into picture. So you have a summation I equal to 1 to n 2n minus 2i.

$$\sum_{i=1}^{n} 2n - 2i = O(n^2)$$

So i is basically index for K. So now this is basically order of n square. Now we have basically your gain evaluation delta G. So the delta G is evaluated how many times? When the K equals to 1 the delta G is evaluated how many times? The delta G is evaluated 25 times. So it is basically n square. Your number of nodes is 2n and n is basically half of the nodes actually.

So when K equal to 2 your delta G is evaluated how many times? It will be basically 4 this side 4. So 16 times, n minus 1 square like that. So how long it should run? It should run till basically 5. So your delta G will be running 1. So it will be 1. So basically if I can have total number of times the delta G is evaluated basically n square plus n minus 1 square n minus 2 square dot dot dot 1. So this we can represent it in a series as this one we can represent this one we can represent as a series as summation i equal to 1 to n, n minus i plus 1 whole square. So this is basically because already one square is there there is another loop is there so it is order of n cube. So we have two operation we are doing one is de-evaluation is doing one is your gain evaluation your delta G evaluation de-evaluation is happening or order of n squares and this gain evaluation is happening order of n cube.

So it is out of both which is the highest that is order of n cube. So the speed of this algorithm is basically order of n cube. In this lecture we discuss about the steps of the KL algorithm we discuss also about one example how we can find out the D and delta G to find a final partition using the KL algorithm. Then we discuss about the complexity of the KL algorithm how it is useful we should know the time complexity of the algorithm. So we discuss that in detail.

Thank you for your attention.