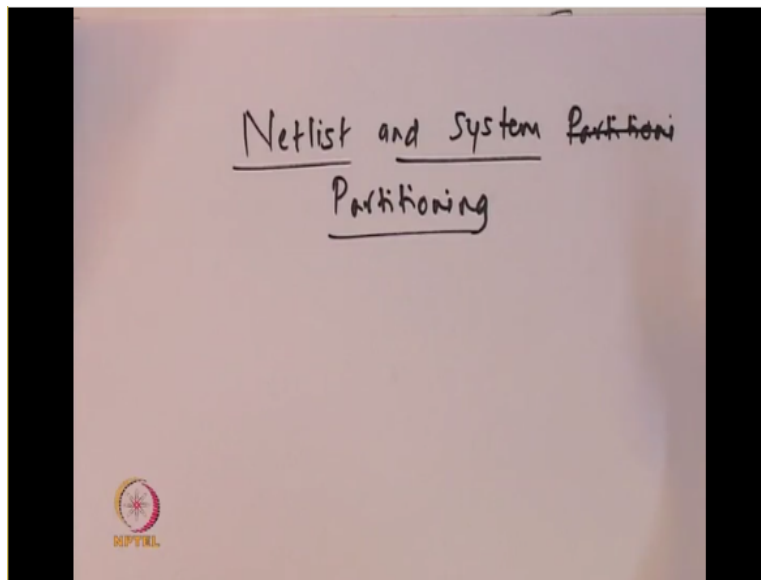


Advanced VLSI Design
Prof. Sachin Patkar
Department of Electrical Engineering
Indian Institute of Technology- Bombay

Lecture – 30
Netlist and System Partitioning

In this lecture, I will give a simplistic flavour of an important part of the design flow is an automation flow called Netlist and System Partitioning.

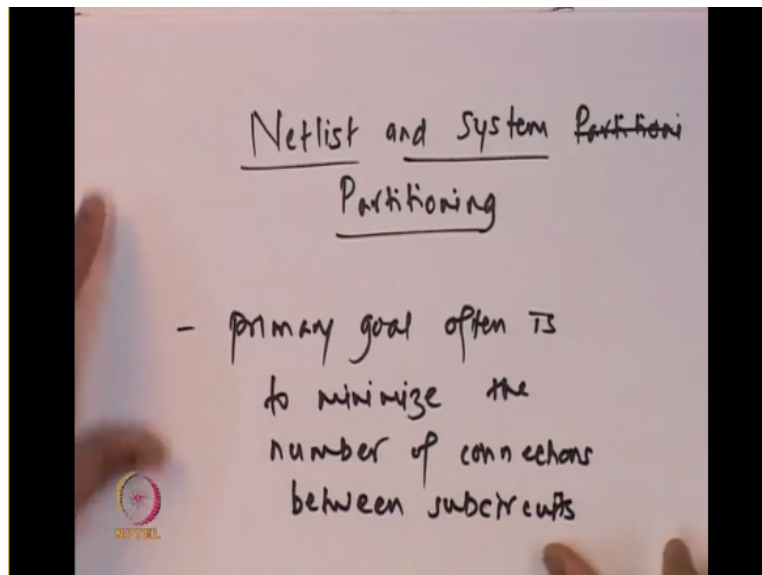
(Refer Slide Time: 00:35)



So, in digital systems, the designs are becoming increasingly complex, huge in size of the order of millions or hundreds of millions in some cases. So, one of the most pragmatic approach of handling such a huge complex it is to use divide and conquer approach wherein we partition the big design into manageable sub designs, so that each sub design meets some constraints or the resource constraints like each design should be fittable on the epigeous we have at our disposal or whatever.

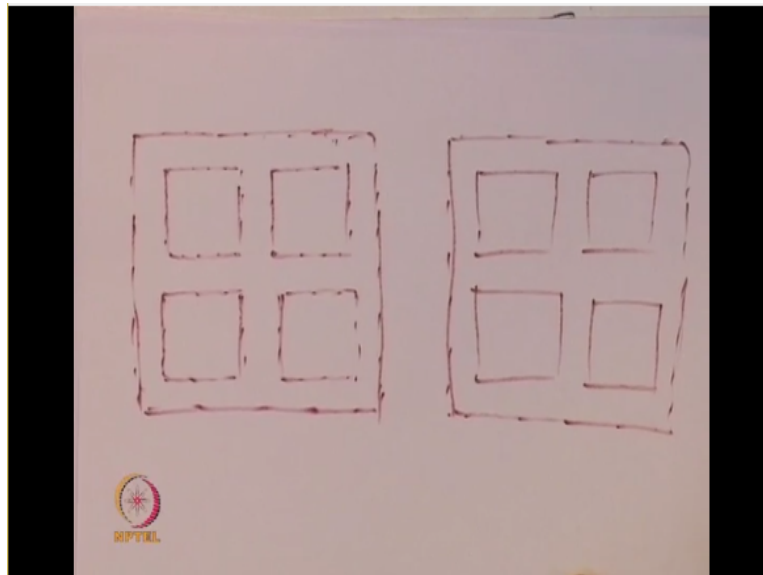
But one of the main optimization criterion is to try and minimise the number of connections which go across the sub designs. That is kind of communication that is going to be the interaction complexity. So, we would like each one of the sub designs to be as independent of the other one as possible. So, clearly the idea is to try and minimise the number of connections. That is one of the most natural objective to optimize in such partitioning approaches.

(Refer Slide Time: 02:05)



So, the primary goal of partitioning is to minimise the number of interconnections between sub circuits. For example, let us take this contrary example with say eight cells of NAND, NOR whatever kind. Very simple cells, each one of them can be implemented on the standard cell.

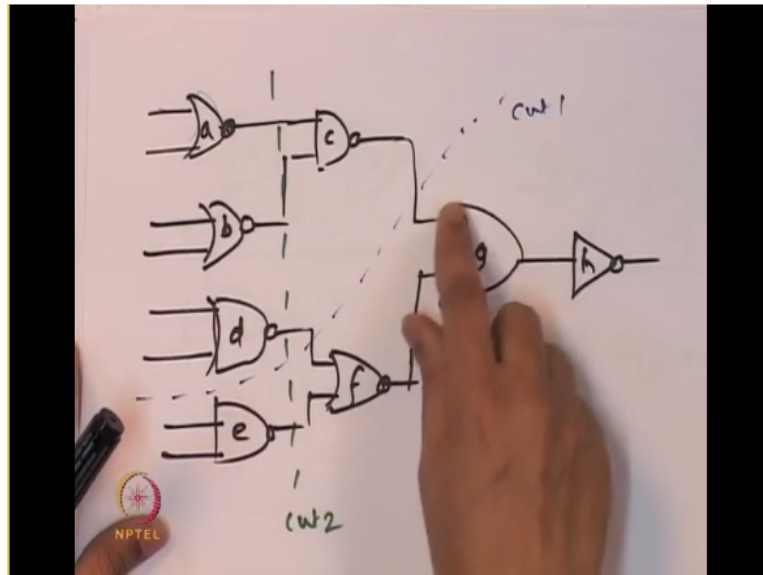
(Refer Slide Time: 02:55)



Let us say that we have been asked to implement this circuit consisting of eight gates onto two chips. For the sake of illustration, I assume we have on this chip four standard cells which can accommodate up to four gates and on this chip we have four standard cells which can accommodate up to four cells. So, in that Netlist we had eight gates, so this should suffice. Let us assume that we have plenty of area to route connections between these cells.

So, there are plenty of options, which of these four cells should go on this strip which four of those eight cells should go onto the other one, right.

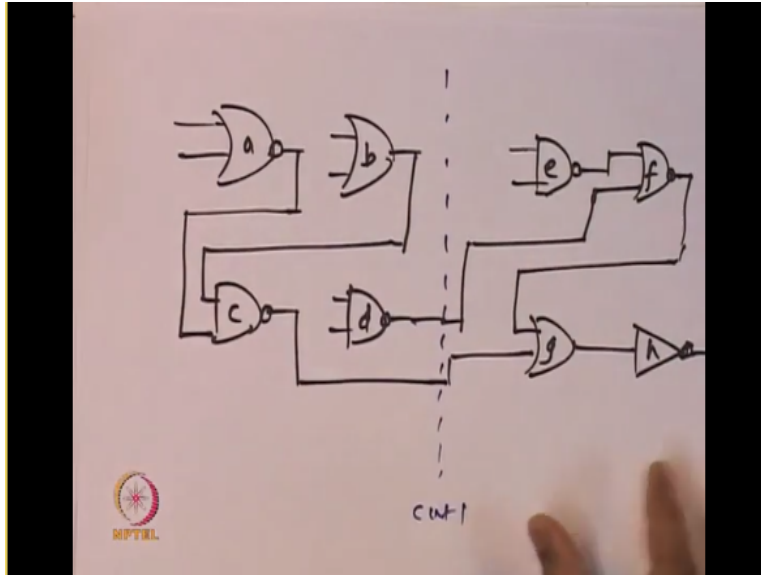
(Refer Slide Time: 03:56)



So, supposing we took the option of cutting this is a netlist with eight gates into this four A, B, C, D and the remaining four E, F, G, H. So, along this cutline we partition this circuit into two parts, one with A, B, C, D and the other one with E, F, G, H. Similarly, we could take other option, some other cut. For example, this cut 2 given by this green line in which we are going to A, B, D, E on one chip with those four standard cells and the cells or gates C, F, G, H on four standard cells of the other chip.

So, let us just visualize how it would look like. If we were to take the first option of partitioning along the so-called cutline one with A, B, C, D on one side and E, F, G, H on the other side.

(Refer Slide Time: 05:00)

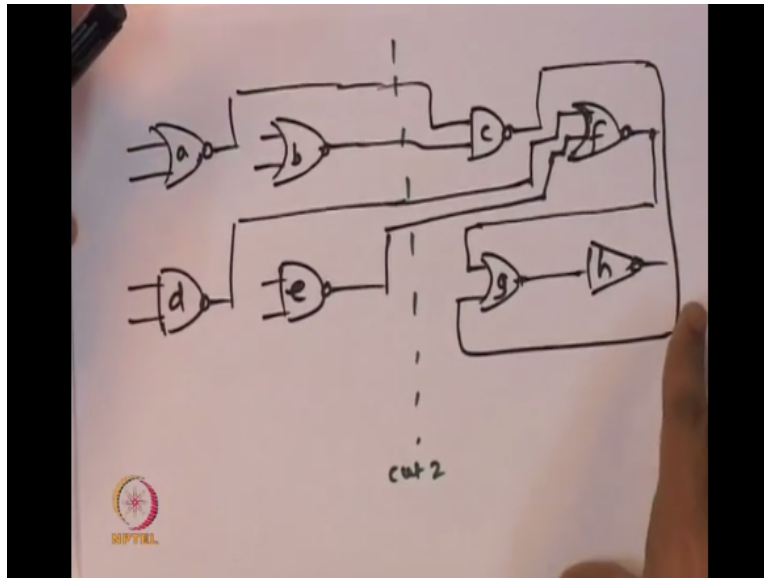


This gates A, B, C, D could be placed again using some algorithm, we could decide that A, B, C, D should be placed in the four standard cells in this fashion and E, F, G, H are placed on the other second chip on the four standard cells of that chip. Now, you see that while completing the routing, we realize that the cell D has to drive the inputs of gate F and cells has to drive the input of gate G and these two connections have to go across the chip.

Whereas all other connections like A to C is within the first chip. The other connection like E to F is within the second chip. G to H is within the second chip and so on. So, only this connection from D to F and from C to G have to cross the boundaries of this chip. So, we say that the cost of this particular cut separating this, implementing this eight-gate design into this two parts is two because there are two wires that have been cut.

Of course, this is very simplistic kind of example. This is important to build abstractions. On the other hand, if we were to take the other option cut number 2 as shown by this green cutline which is going to separate A, B, D, E from C, F, G, H. So, A, B, D, E we are going to put them on the left chip and C, F, G, H on the right side.

(Refer Slide Time: 07:00)

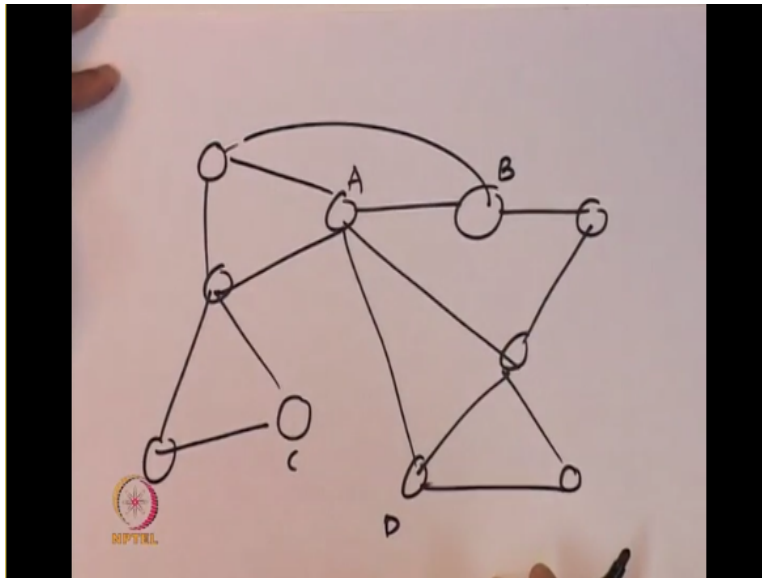


So, by some placement strategy that why we decide to put A, B, D, E on this four standard cells in this fashion, C, F, G, H on this four standard cells. So, this is chip number 1, this is chip number 2. Now, you see that to compute the wiring, you will realize that as many as four wires A to C, B to C, D to F, E to F. This have to go across this partition okay. So, the cost of this particular implementation would be four wires which have to be routed across.

So, within a chip the delays would be small. Across the chip, delays going to large. So, because of this reason, we would like to try and keep this a number connections going across small. There are various considerations why we would like to minimise the number of connections which have been broken. So, clearly as shown in this example, there are several choices. We have considered two of the choices and we saw that the choice number 1, cut 1 is better than the cut number 2.

The algorithmic problem is how to identify this best or nearly best choice of how to partition. So, to be able to algorithm, one has to use abstractions like graphs and hyper graphs for modeling netlist.

(Refer Slide Time: 08:47)



So, supposing with the help of a graph, let us assume that this particular graph represents abstractly some netlist. Not really everything about the netlist is being represented in this graph, but enough information that we are going to use to partition this graph. So, I just want to illustrate one simple concept which we are going to use in the so-called Kernighan Lin Algorithm.

So this example is just about adequate for that. So, supposing currently at some intermediate snapshot of the algorithmic process to decide on a good partition into two parts, this 1, 2, 3, these five nodes have been grouped on into one partition and this remaining nodes have been grouped into other partition. So, this cutline is describing a particular partition of this graph.

So, now we want to see whether we can improve upon this, and in particular, we want to kind of respect some natural constraint, justifiable constant that the new partition, like this particular partition should also have five nodes on either side, same number of nodes on either side. It is like every node has equal amount of the area requirement, so we would like to make sure that neither of the partition becomes rather too big compared to the other one. Roughly, they should be of same size.

So, ideally, we would like the new partition which hopefully is better quality partition to be also of equal size on both sides. So, a natural idea would be to try and identify a pair of node on this

side and a pair of node on this side and try and swap them. So, that will help us keep the sizes also balanced but we will try to identify which of this swaps, 5 x 5 square swaps, 25 possible swaps we should do so as to improve this and the sequence of such swaps which we should like identify and commit to so that we have a better partition.

Note that this is a problem with only 10 nodes. We could have millions of nodes of this kind, so it is not going to be easy to do it by trial and error. There must be some efficient algorithmic approach to take care of this, to implement this. So, let us understand if are to analyze, what happens if we decide to swap A and B. So, note that A and B are across the cut. On the left side, A is in one part and B is in the other part.

So, notice that A is connected to two nodes is connected to two neighboring nodes on its own side and B is connected to only one neighboring node on its own side. On the other hand, while is connected to just two nodes on its own side, A connected to in fact three nodes on the opposite side and B is also connected to two nodes on the opposite side.

So, if A where to move to the other side what is going to happen that three edges which link A to the other side, they will become the internal edges of the new partition. So, basically moving A from here to here is going to drag these two edges into the cut. At the same time, these three edges which are currently in the cut would no longer appear in the cut. So, on the similarly moving B from here to here would drag this blue single edge into the cut.

So, definitely this blue edge will be dragged into the cut. This edge will, sort of vanish from the cut, but this edge which is currently in the cut, it will continue to remain in the cut because it is between A and B the pair of nodes which we are swapping. So, any edge which is between a pair of nodes which we are swapping, that will continue to remain in the cut.

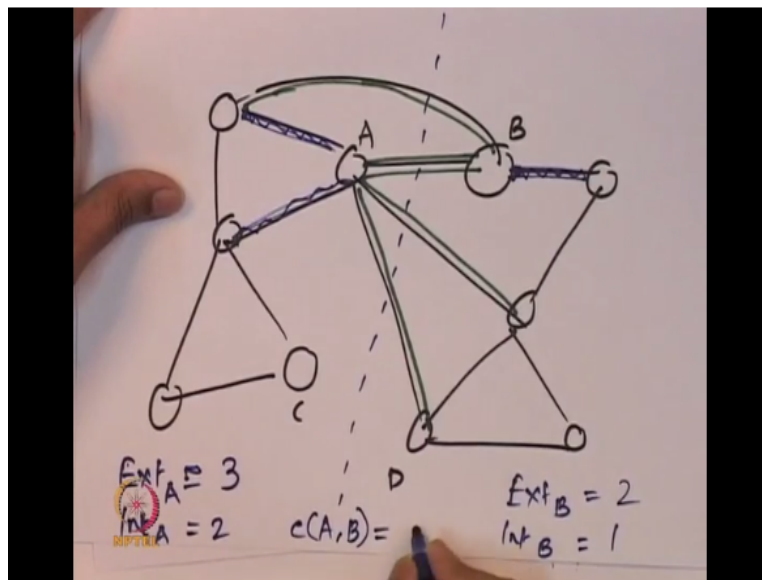
Any edge which is from a node being swapped into its own part, it will get brought into the cut and any edge which is across the cut and not between the pair that we are swapping, like this edge or this edge, this edge. These edges which are currently in the cut, they will vanish. So, revise cut will have slightly different cost and that will be based on these numbers.

How many nodes, like is A connected within its own part, how edges does A have incident on itself from its own side. How many edges does B have incident to itself from its own side. So, A has two edges incident to itself from its own side. B has exactly one edge incident to itself from its own side. A has one, two, three edges going to the other side, B has one and two edges going to the other side.

So, these numbers are going to play some role. So, we will define them. So, Ext of A. Ext stands for external. So, we will denote by Ext of A, number of edges which are going from A to the other side. So, that will be 3 here. And Int of A representing the internal connections from A to its own side that will be 2. Similarly, Ext of B is one and two. There are two links from B going to the other side. Int of B is 1, just 1.

Furthermore, the cost of connection A to B is 1. There is only one link that is joining A and B. Now, see what happens when we swap.

(Refer Slide Time: 16:51)

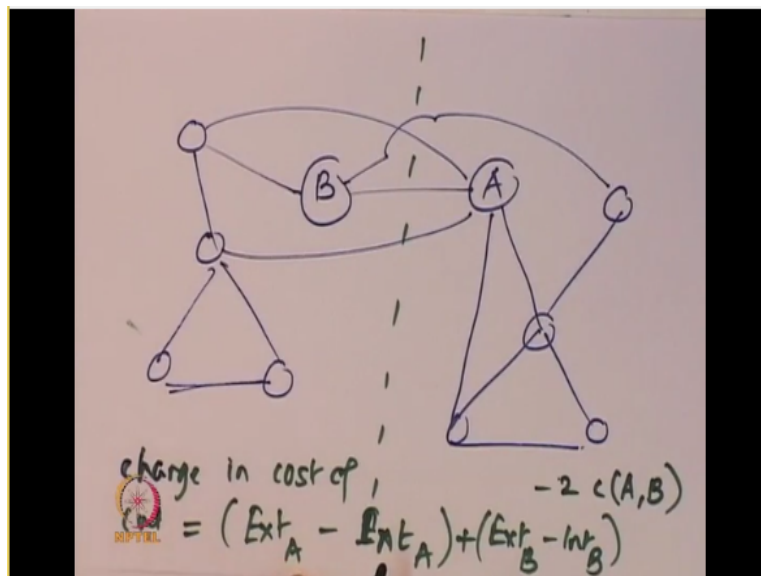


B has been brought here. A has been moved to the other side. So, A was connected to this pair of nodes and B is connected to this. Furthermore, A was connected also to B and B was connected to B as well as this node. This is the new partitioning after swapping of A and B and what we notice is that before we swapped, we had how many edges been cut 1, 2, 3, 4.

So, after swapping A and B, this is the resulting network. The change in the cost of cut will turn out to, one can verify or prove that it is going to be $\text{Ext } A - \text{Int } A + \text{Ext } B - \text{Int } B$. So, what does this first term means $\text{Ext } A - \text{Int } A$. $\text{Ext } A$ are the number of connections that A had to the other side and those kind of edges are going to get vanish from the cut. So, that is an improvement whereas $\text{Int } A$ is the number of edges connected to its own side. They are going to appear across the cut.

So, this is the good thing and this is the bad thing, good thing and bad thing. So, this minus this is some kind of gain of moving A to the other side. This is the gain of moving B to the other side, but since we have done the swap, the connections between A and B would offset things to some extent and that will be minus two times the cost of number of connections between A and B.

(Refer Slide Time: 19:57)

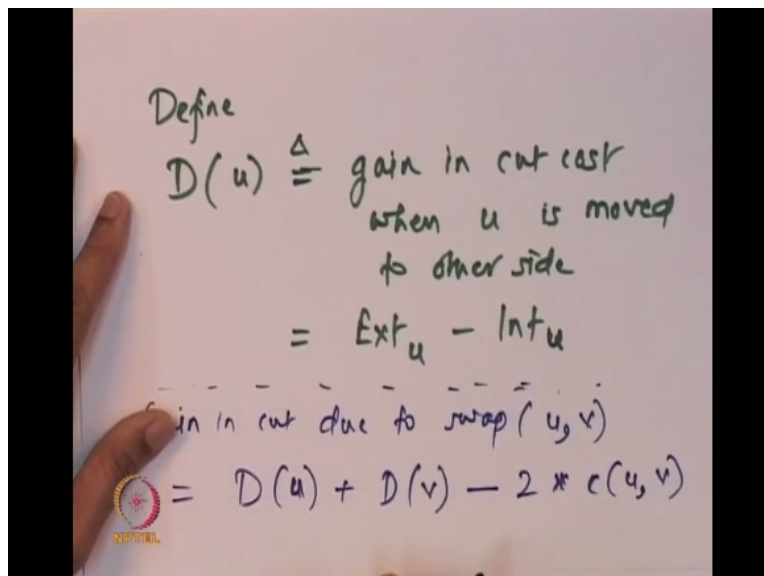


So, in this particular example, the $\text{Ext } A$ was equal to 3, $\text{Int } A$ was 2, $\text{Ext } B$ was 2, and $\text{Int } B$ was 1 and CAB cost is 1. So, what does this formula tell you it is $3 - 2 + 2 - 1$, so that is $1 + 1 - 2 \times 1$, so that is minus 2, so that is zero. In this particular example, the change in the cost is going to be zero. There is no improvement in the cost. So, it is quite simple stuff. In this model, we are not going to describe the algorithms in detail or study of a separate course on VLSI Design Automation.

So, it is just for the sake of flavour and to slowly proceed to the illustration, helping to develop your own insights into how this modeling is done with the help of graph and how certain simple calculations will be done by the algorithm. On the basis of it, the decisions will be taken and so on. I am just leading you to a very well-known heuristic called Kernighan-Lin heuristic which make use of such accounting approach.

With the help of these numbers which together contribute to so-called gain in the cut. So in general for node define the so-called D of U which is defined as gain in cut cost, that means imp, that means reduction in cut cost.

(Refer Slide Time: 21:30)



Define
 $D(u) \triangleq$ gain in cut cost
when u is moved
to other side
 $= \text{Ext}_u - \text{Int}_u$
Gain in cut due to swap (u, v)
 $= D(u) + D(v) - 2 * c(u, v)$

Gain means reduction when U is moved to the other side. Clearly this can be easily shown to be it is Ext of $U - \text{Int}$ of U where I have already illustrated the definition of what is Ext of U and what is Int of U . So, this is the number of edges which connect U to the other side and number of edges connecting U to nodes on its own side. See, only instead of swapping, we are just moving one vertice from one side to the other one.

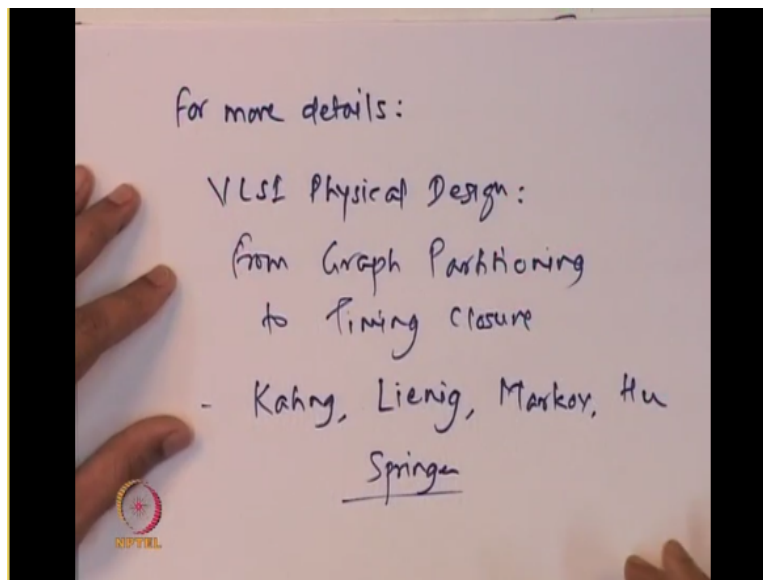
This will give us reduction in the cut because this many edges are going to vanish from the cut and this many edges are going to come into the new cut. So, now this is for just movement of single node. Our intention is to swap, right. So, gain in cut due to swap of nodes U and V , will turn out to be the reduction in the cut cost will turn out to be the gain due to movement of U . The

gain due to the movement of V but offset by two times the number of connections between U and V. It can be easily proved.

You can think of it as an exercise. so clearly but the supposedly you have been asked to find a pair of nodes U on one side and V on the other side whose swap will improve the cut that reduce the cut to a good extent, then you definitely look for node U and V such that DU is very high. It is this $Ext U - Int U$ is high, DV is high and hopefully the number of connections between them, there could be parallel connections of course.

It need not be just 1 or 0, this is small. So, then this particular reduction is as high as possible. So, greedily we would like to identify this pairs of nodes to be swapped. So, now let us work with an example which I think appeared about sometime last year.

(Refer Slide Time: 24:31)

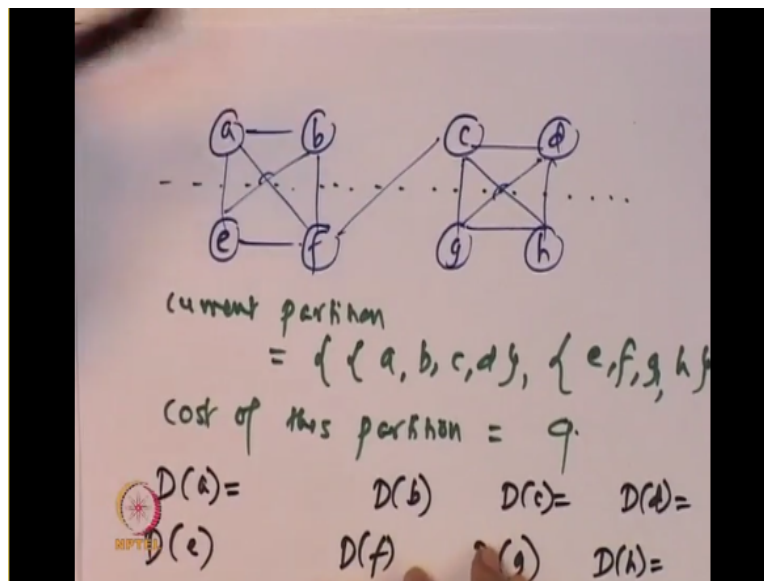


A very good book, compact book which gives very good illustration of basic algorithms, also some theory behind it. It is not an exhaustive book. There are books which are far more exhaustive on the subject or more specialized into certain sub-topics, but this will be a good book for starters, also very nicely written and good illustrations, exercises, and support for teaching.

So, the example that I am doing is borrowed from this book, chapter number 2 which is on graph partitioning. You can get more details from this book. So, the example that I am going to lead

you through is a very simple example. Again highly contrived, but illustrates ideas very well. So let us assume that out of a netlist, we have abstracted out of graph.

(Refer Slide Time: 25:38)



The graph is like A, B, C, D, E, F, G, H. So, this is eight node graph. Assume that these eight nodes, each one of them is representing one gate or one standard cell and let us say we have, maybe initially or at some point in the algorithm, we have arrived at this particular partition. The partition A, B, C, D on one side and E, F, G, H on the other side. What is the cost of this partition is equal to, you can see that it is 1, 2, 3, 4, 5 and plus 4 equal to 9. It is a high cost, right.

It is a bad partition. Clearly like you will see that just by inspection you can see that if you were to partition this eight-node graph into two equal parts, this would be best. This would be the one which would minimise the number of edges being cut that will precisely 1. So, things are not going to be so simple to visualize or solved by inspection.

So let us use this simple example where we have good intrusion of whether the algorithm, the so-called Kernighan-Lin algorithm, is it proceeding with some good natural intrusion or has some natural justification or not. So, it should be intrusively like obvious algorithm. Supposing, let us proceed along the natural idea that since we have to keep this partition balanced, these two parts balanced, we will commit to doing swaps one at a time.

So, without any further clue, the obvious thing is to try greedy. Supposing we have been told by experts that this problem of partitioning graph into two equal parts so that the number of connections going across is minimized, this problem is known to be NP hard.

So, it is good to think in terms of heuristics and then develop further intrusion inside, and then try better approximation algorithms more theoretical investigations but let us use orientation first and try and design a good heuristic, so that heuristic, one natural approach seems to be do swaps in a Greedy way. So, what would be the first swap that you would go for with the intention of improving this cut as much as possible.

Right now the cost of the cut is 9. So, for that again some kind of bookkeeping will help and that bookkeeping will be in terms accounting information or whatever it would be, maintaining this D information for every node. DA equal to how much, DB, DC, and so DA will be how much. For example, what is DA. The cost of moving, rather not a cost, in fact the gain or reduction in the cost if we were to move A to the other side. Of course, we are not just going to move, we will swap.

But if we are allowed to just move A to the other side, A is connected to just one node on its own side, that is internal cost of A is one but A is connected to two nodes on the other side, namely E and F. So, the gain in the cut if you move A to the other side is going to be $2 - 1$ that is 1. Similarly gain for B is going to be 1. Gain for C will be how much. C is connected to only one node on its own side.

But C is connected to 1, 2, 3 nodes on the other side, so gain if we moved C is going to be 2 and for D it is just 1. For E is 1. For F like C it is going to be 2. G it is 1. H it is 1. From this information and the information about the number of connections between any pair of nodes that is already available in this graph. From this, we will be able to identify a pair whose gain will be highest which when swap will be highest.

So, recall that gain in the cut or reduction in the cost of a cut due to swapping a pair U and V assuming that U and V are on the opposite side is the gain due to movement of U, gain due to

movement $V - 2$ times. This is the offsetting thing. But fortunately in this example, it is quite obvious that since C and F have the highest gain due to their movement and C and F although they are connected by 1, it will turn out to be the best pair to swap. One can verify that by comparing it with all other pairs.

(Refer Slide Time: 32:01)

9 Best pair to swap is (c, f)
 \therefore reduction/gain in the cost of cut $= 9 - \text{gain}(c, f)$
 $= 9 - (D(c) + D(f) - 2 * c(c, f))$
 $= 9 - (2 + 2 - 2)$
 $= 9 - 2 = 7$
 consider swap (c, e)
 $\text{gain}(c, e) = D(c) + D(e) - 2 * c(c, e)$
 $= 2 + 1 - 0 = 3$

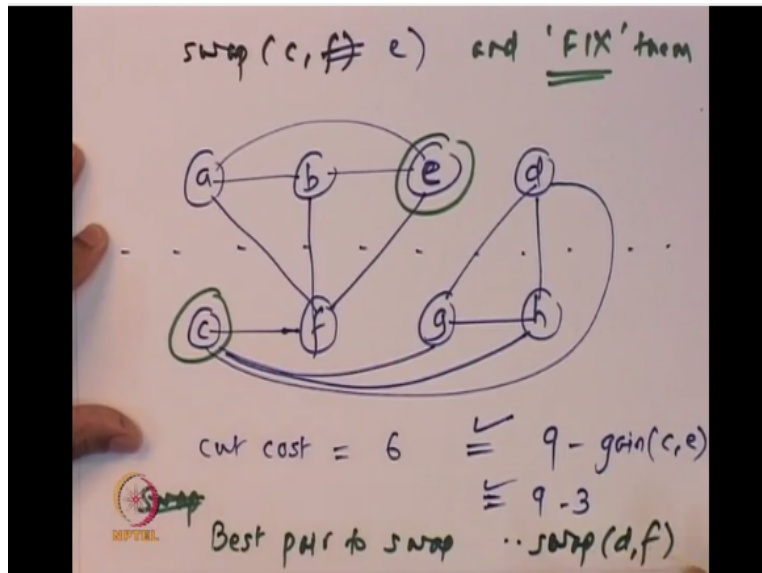
Best pair swap is evidently C and F. Once we do that what will be reduction or gain in the cost of cut that will be earlier cost that is 9 minus the gain due to C, F which is 9 minus D, C plus D, F. I think I have made a mistake. So, let us put a question mark here because C and F have very high, both of them have the highest gain of just movement to each, will it give us the best pair to swap. This will be 9 minus gain will be 2 plus 2 minus 2 that is 9 minus 2 equal 7.

On the other hand, consider swapping C and say E. So, gain due to the swap of C and E will be D, C plus D of E – two times C. So, this will be 2 plus 1 but here it is zero, so this is 3. So, here the gain was just 2, that helped reduce the cut cost from 9 to 7, but here, if we swap C and E instead of swapping C and F, we would have the improvement in the cut cost by 3. So, the cut would reduce to 9 minus 3 that is 6 which is better.

So, like maybe we should not simply try, we will have to work out for every possible pair the gain and hopefully that is not too much of competition and then find the one that is best. So, in this case we can verify further and confirm that, C and E is the best pair to swap. So, C and F are

earlier guess was wrong.

(Refer Slide Time: 34:51)



So, this is the new picture across this cutline and the new cut cost is going to be 1, 2, 3, 4, 5, 6 and confirm that it is 9 minus gain of C, E that is 9 minus. Now, let us mark here that C and E are the ones which have participated in the swap and like to decide the next swap or subsequent swaps, we will now not allow them to play the role. We will freeze them in their respective parts. They will not play any role until we consider the other swaps as a sequence of swaps which will hopefully improve the cut cost in the best possible way.

So, I say that now what my remark over here is that swap C and E and fix them. So we are not going to allow them to participate, fix this pair of node C and E. Not allow them to participate in future swaps. So, now in this partitioning that has that has resulted after this first swap, we again have to find the next best pair of swaps which do not involve either C or E. So, could it be F or could it be D, E, or whatever.

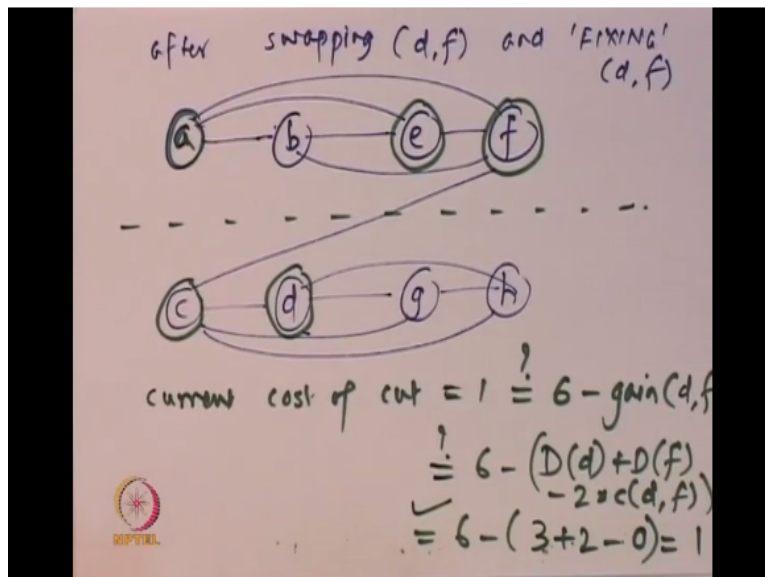
So, what do you think what would be the best pair of nodes to be swapped. Again, we have to calculate the D values, the gain values which reflect the gain by movement of a node onto the other side, but we have to correct it by that suffice this connection C times U, V and all that. So, we can here compute the gains for every allowed swap and find the best one. So, it will so turn out that, so if you are just in fact you can notice that these are node which is disconnected from

everything else on its side.

So, most naturally you know, so you can start guessing that D should participate in swap because there is nothing to lose. Everything it has to gain, all the edges which are connected to D are to the other side. So, by moving D onto the other side, you are going to gain all that much and lose nothing. Similarly, on this side it looks like there is no such node like D which is isolated, but F is next best thing to isolate it.

So, if we move F to the other side we see that only one edge will come into the cut see as many three edges will vanish from the cut and thirdly, there is no connection between D and F. The gain due to the swap of D and F is going to be the gain due to movement of D plus the gain due to the movement of F minus no correction because there is no link between D and F. So, next best pair to swap will turn out to be swap D and F.

(Refer Slide Time: 38:49)



That will give us, in place of D we have brought F and in place of F we are going to bring D, F was here. Please let me with the green circles mark that early C had participated, E had participated in swaps and now D and F have participated in swaps, so they are going to be frozen. So after swapping D and F and fixing of D and F, we get the following diagram following graph.

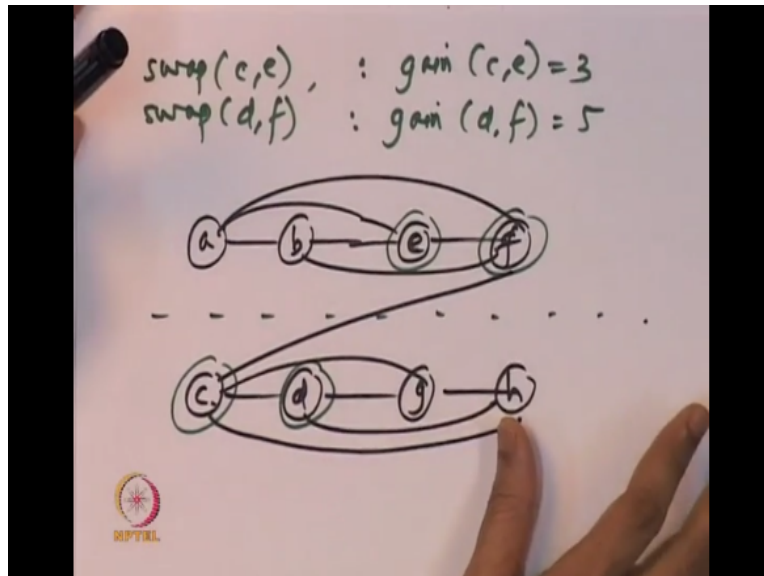
So, we note that current cost of cut is equal to 1. We would like to check that it is earlier cost, that is 6 minus the gain due to which pair we have swapped D and F. What is gain due to D and F, that is $D \text{ of } D \text{ plus } D \text{ of } F \text{ minus } 2 \text{ times number of connections between D and F}$, you can verify that 6 minus. It is D of D, how much is D of D, number of external connections from D that is 1, 2, 3 minus number of internal connection that is zero. So, D of D is 3.

What is D of F, number of external connections to the other side is 1, 2, 3 minus internal is 1. So, D of F is 2 minus 2 times 0. So, this is indeed 6 minus 5 that is 1. So, it is confirmed. So, that particular formula that you can easily prove, we have been verifying that validating that with this example. So far we have met, locally greedily swaps, the first we executed a swap C, E, fix those nodes. We are not going to allow them to participate in any future swap until the so-called pass is over.

I will tell you what the pass is. Then we again by looking at revised partition, we figured out that the pair D and F is the best pair to swap that we swap, fixed it and that has resulted in a very good-looking partition with only one edge across it. So, you know that this is something the best that we already obtained, but in general in a big algorithm, when this algorithm is run on a very big circuit, very big design.

You are not going to be able to easily guess that this is the best or this is nearly the best that you have reached and you need not waste your time further. So, the algorithm should go ahead and arbitrary stop here with some kind of confidence. That could be premature right. So, let us go ahead.

(Refer Slide Time: 42:37)



Now, we already arrived after swap of C, E and followed by swap of D, F. This gave us the gain of 3. This gave us the gain of further 5. So, with the help of these two swaps, we have reduced the cut from 9 to 6 and then further to 1 and we have fixed the node C, D, E, F, not going to allow them to take any further. Now, we know intuitively that if we try to do any further swapping, things will not get any better.

This is the only best cut but the algorithm would not have known that so early. So, the main idea, the natural idea that we have like Kernighan-Lin develop was start with some initial equal size partition and just go through a sequence of swaps and everytime while deciding the swap, decide the best possible swap in terms of the amount of reduction that the swap would result in the cut cost and find the best possible pair to swap among the nodes which are not fixed yet.

And keep on doing exercising the swaps and fixing the nodes which are involved in the swaps so that they do not take part in the swap and finally after N by two such steps where N is the number of nodes, we would have like know tried out enough swaps which would complete a so-called pass. Because after we swap a pair of nodes, we do not let them participate in any future swap in the so-called pass.

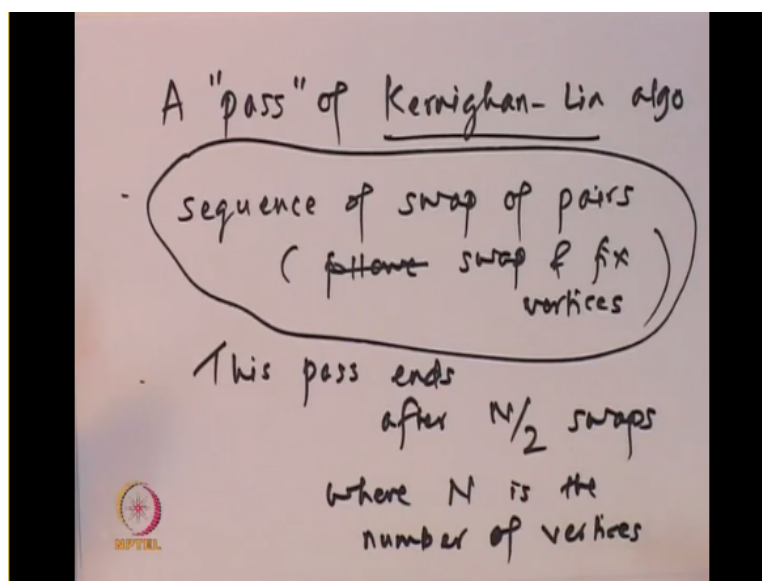
So, after N by two such swaps, we would have come to this so-called end of the pass and in this end of the pass, we would have been like based on the gain of the pair that has been swapped, we

will be getting the new partition of a different cost. If the gain is positive, we would be getting the partition of improved cost. If the gain is negative, that is surprising why would we bother about swapping a pair of nodes with negative gain.

But supposing we do that or we have to do that as required by the algorithm, then negative gain would mean that next partition would have higher cost, so that is actually worse. So, so far we have seen two swaps. Here, swap C, E that was the first one after which we commit fix those word as C and E and then swap D and F and in both cases, the gains were positive. We are lucky, so we improved the cut from 9 to 6, 9 minus 3 equal 6 and then 6 minus 5 equal 1.

That is the current situation, but what about the non-fixed nodes like A, B, D, G, H, we should optimistically try, although by inspection we know that nothing will yield us anything better, but algorithm would not like try and get any certificate, like at this current time do I have the best one. Not let halfway or before the pass is over. So, we have the so-called notion of pass.

(Refer Slide Time: 46:23)

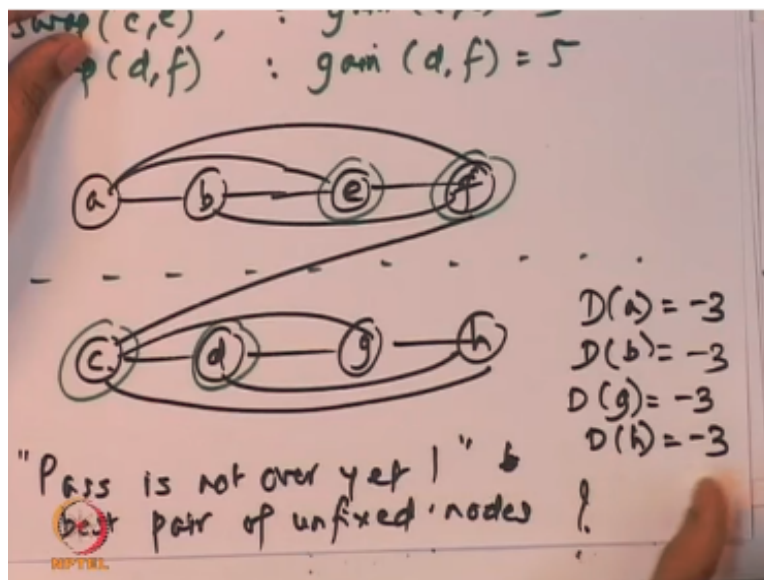


It is basically a sequence of swap of pairs followed by the swap and fix vertices. So, this pass ends after $N/2$ swaps where N is the number vertices where is assumed to be 1. These are all, we should not worry about these restrictions because one can always add some dummy nodes and make number of nodes even and all that, and also these are some just basic ideas from which one can think of lot of variations and improvisations.

So that is why in this overview lecture like this or it just suffice to highlight the basic ideas and simplest such algorithms. There are far more interesting variations of this which are practical as well as efficient, practical as well as interesting by itself. That we could leave it to the next course or some other specialist course in VLSI CAD.

So, this is the concept of pass that is a sequence of swaps, how many N by 2 swaps because after every swap we are going to disable the swap pair of nodes from participating in any future swaps. So, after N by 2 swaps, we would have fixed all the nodes for that pass and then after that we can take a pause. So, far we have not reached that end of the pass, so we should continue okay.

(Refer Slide Time: 49:08)



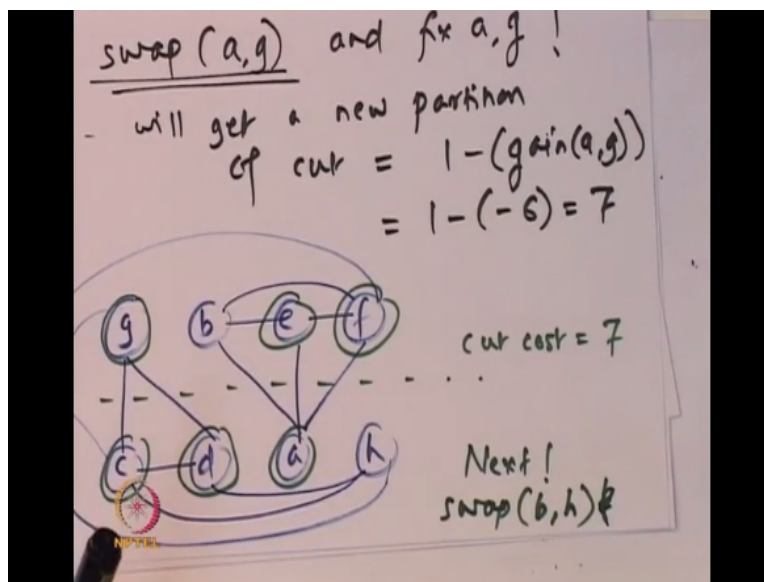
So, if we continue from here, so pass is not over. So, the question is next best pair of unfixed nodes. So, what will it turn out to be. We have A, B from this side and G, H from this side eligible to participate in swaps and it looks like we can take any pair, AB, GB, BG, or BH whatever because it is all uniform now, same thing because D for A, the gain not the cost.

The gain resulting in movement of A is going to be 0 is the external cost and everything is internal. So, the gain is going to be negative. In fact, it is going to be as bad as minus 3. D , A is minus 3, right. Why, because external cost is external connections zero. Number of internal

connections is 1, 2, 3. So, (()) (50:27) is -3. Similarly, Db is going to be -3 and Dg is going to be -3, Dh is -3 alright and also we see that there are no connections between any pair of nodes, one from here and one from here.

Neither connection A and G, H. There are no connections, so by the gain formula you will see that the gain by swapping is going to only make things worse. For any pair, the gain is going to be minus 6, very negative gain. So, it is going to in fact worsen the cost of next partition, but still we are committed to continuing with the pass and we let us say A, G.

(Refer Slide Time: 51:35)

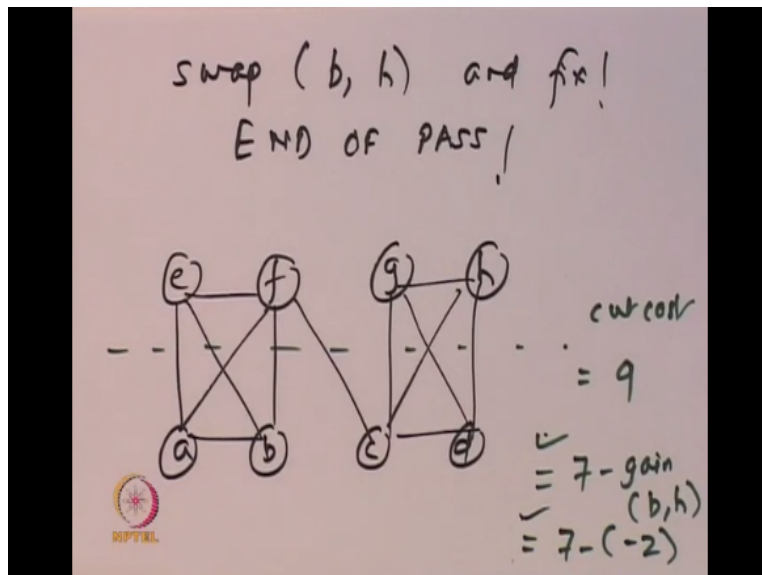


So, swap A, G and fix, so what we get as a result of this. We will get a different partition of how much cost. We had previous partition of really good cost 1 but we have to subtract from it the negative gain that we get out by swapping A, G which is going to be 1 minus 6 that is 7. So, let us confirm. So, this going to be the resulting pictures. Verify that number of edges cut is 1, 2, 3, 4, 5, 6, 7.

Confirming that with our formula, so things have really made this thing worse, but that is the interesting thing about algorithm. If you have to proceed, optimistically hoping that in future things will again get better. But in this particular example, it will not. So, now let us just complete this pass. We have only after this we would have frozen G, E, F, C, D, A. So, only pair to swap is B and H. So, next there is no choice, swap B and H and fix and that will be the end of

pass.

(Refer Slide Time: 53:53)

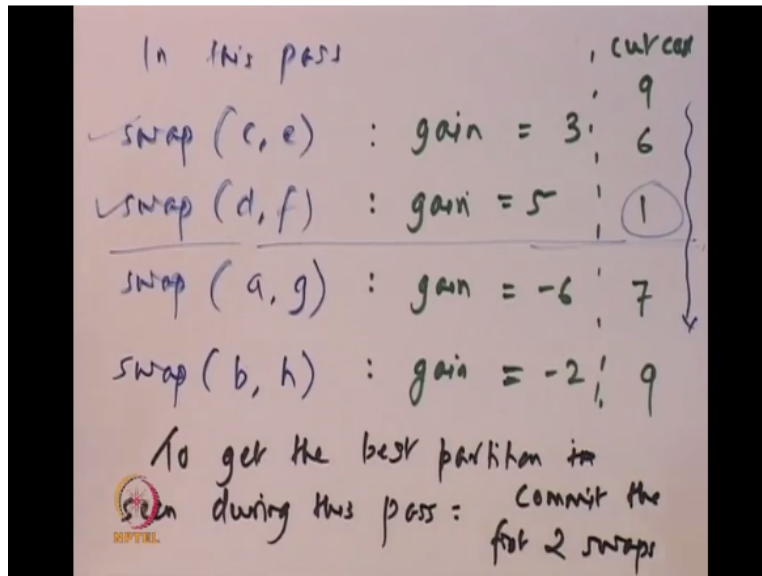


So, what do we get. We do not even need to draw this. Now, swap B and H and fix end of pass. Notice that what we had done at this last step of the pass is like swap the last possible swapping pair of nodes and fixed pair of nodes and what we will simply have is the same picture just reflected that we started off in the beginning.

So, we had A, B, C, D on this side, but A, B, C, D have been swapped with E, F, G, H and so with some re-arrangement, we will have. I have taken the liberty to just re-arrange this position of nodes, ensuring that the A, B, C, D is on one side and E, F, G, H are on the other side. This is the resulting picture with some bit of re-drawing. So, cut cost is going to end up as it was in the beginning 9.

So, this must have been you can verify that it will be 7 minus gain of the last swap that is B, H. You can verify that this will be gain of B, H is going to be minus 2. So, we are expected to get back the same initial partition at the end of the pass. So, need not actually go up to the last. We would know, we do not need to compute anything explicitly, but this is the result that is inevitable.

(Refer Slide Time: 56:14)



But now let us look the whole pass what we have done. In this pass, we started with a swap C, E, then swap D, F, then swap A, G, last inevitable or redundant swap the gain was 3, gain 5, gain minus 6. We did not calculate it, but can be verified. So, cut cost was 9 in the beginning. After this swap, it became 6. After this swap, it became 1, then it became close to 7 again and further worsen back to 9.

So, you see that during the pass, we have this particular sequence of partitions and there cut cost and then we simply like decide the best point. After having seen the whole swap, we can now see that this would have been clearly the best partition. To arrive at this partition with this particular best cut costs, we should only commit to get the best partition seen during this pass. Clearly we only commit the first two swaps.

Because exactly after this first two swaps, we get a best partition with this best cost and then after that it will make things worse. It might improve again, but it will never be as good as what we have got at the end of this. So, looking at this sequence of partitions and their cut cost, we can decide and this is that is what we call the result of this pass. This is the best partition that we got at the end of the pass.

Then, the next thing that we can do or the Kernighan Lin Algorithm does is uses the best partition as the initial partition of the next pass. It optimistically hopes that there might be some

improvement. In fact, if it were the case that at the end of this pass, whatever the best we get is the global best, then this problem would have tractable but in fact it has been proved to be a hard problem. So, we will have counterexamples where, although in this example after a single pass we got the best one.

In most general examples you will not get necessarily the globally best partition as the best partition of the first pass itself. So, algorithm should go to the next pass, but there what might happen is that the next pass may not improve the partition. It might so happen that any pair or sequence of swaps in that pass is going to make the partition cost worse, in which case we just accept that whatever we got at the end of the previous pass as the best one.

So, that is the main idea behind the algorithm or lose informal description of the algorithm. It does one or more passes within each pass it does a sequence of swaps, swap and fix. Once you swap a pair of nodes where the pair of nodes is chosen, greedily on the basis of the gain in the cut cost. You fix that pair of nodes so that they do not participate in any further swaps. In that pass, at the end of N by 2 such swaps the pass is over.

Then, you take a look at the sequence of partitions and the cut cost, stop at the best, look at the best point that you had reached during this pass. Commit only those swaps like in this example, just first two swaps and that is a best partition out of this pass. Then, use this best partition as an initial partition for the next pass and if things improve fine, then go onto the next pass. If the things do not improve, stop. Maybe you are stuck in some kind of local optima.

That is what might happen often. There are ways around it, but still getting global optima for large size problem is going to be brutally hard. So, do not hope for that too easily. So, this is hope is something that describes the essence of Kernighan Lin Algorithm and the main idea is try and think of your variations with natural variations which make this algorithm more efficient or which make this algorithm more general and so on and that is how the things proceeded.

From KL, there were various variants and that they become more standard better algorithms which are still used in most of this CAD tool packages and the research is still going on. There is

now more focused on performance-based partitioning and performance-based placement. So, objectives are changing and like lot of technology implementation related objectives are being brought in. This was purely topological number of connections in cut. Things can be far more general than that.