**VLSI Physical Design with Timing Analysis**

**Dr. Bishnu Prasad Das**

**Department of Electronics and Communication Engineering**

**Indian Institute of Technology, Roorkee**

**Week 01**

**Lecture 06**

# Spanning Tree and Shortest Path Algorithms

Welcome to the course on VLSI Physical Design with Timing Analysis. In this lecture, we will discuss about more graph algorithms that is applicable for VLSI physical design. So, we will discuss about the Spanning Tree Algorithm. So, we know that the electronic circuits consist of multiple components, and they are connected with wires or interconnects. So, there is a always tendency to optimize or minimize the wire length inside the design. So, there are several algorithms are there, but the Spanning Tree Algorithm helps us to minimize the total wire length inside the VLSI circuit design and that is useful to reduce the interconnect length in which in turn reduces the delay of the circuits.

We can model the electronic circuit as a undirected graph where the pins and the logic gates are modeled as vertices and the connection between the gates as connected or the wires or the interconnects are modeled as the edges of the graph. So, we have a graph G which is represented as V, E where V is basically the reference of pins and E represents a potential connection between them. So, here we have edge U, V has a weight W, U, V denoting the wire cost between the pins U and V. So, we have basically V is representing the pins and V is representing the pins of the circuits or the netlist and E is representing the connection between them. So, here what is the objective is that we need to find out a subset and that subset is a acyclic subset. Acyclic means that there is no cycle between the cycle inside the graph. The T is a subset of edges that connects all the vertices. So, that connects all the vertices. We need to find a acyclic subset that connects all the vertices and the second objective is to minimize the total where length or the weight of the wires of the subset T.

So, indirectly we are finding a minimum spanning tree. So, the solution forms a tree or that is called acyclic and that is why it is called a spanning tree. So, since there is no cycle in the graph that is why it is called a tree, and this is since it is connecting all the vertices in the graph that is why it is called a spanning tree. So, this tree connects all the vertices in the original graph. So, what is missing here? If I add a edge to that one then it will create a cycle.

So, what we are doing here we have a original graph we are removing or we are removing some of the edges to create the spanning tree. So, we have a problem called minimum spanning tree where we are finding the minimum cost of the tree. So, it is very useful for optimizing the wire length in case of a circuit design. So, we have two algorithms are there one is called Prim's algorithm, and this is the pseudo code of the Prim's algorithm. So, here we will go into an example here. What is happening here? We are starting with a node A that node is connected by two edges one edge cost is 4 another edge cost is 8. Since we are looking for the minimum edge so we will choose 4. So then, so the node B or the vertex B is connected to edge by weight 1. So, that is the minimum, that is why that edge is chosen. Then there is a connection between H and G so which weight is 1 so we connect we select that one.

Then we need to find the edge which is connected to the already existing nodes. So, if you can see here which is the minimum one 2. So, we have a edge from G to F. Now if you can see after 1 and 2, 3 is not there in the graph which is the next edge which is which can be part of the tree. So, that is the 4 the edge is 4 and it is connected to node D.

Now after this is selected after the 4 we have another edge which is having 5. So, the 5 is selected because it has having the minimum weight and we will select a edge from G to C. Now after 5 the next cost will be 6 but there is 6 is not there in the graph. Then the next weight is 7 and if I choose 7 then it is creating a cycle so that is not allowed because if I select this one it will creating a cycle this is a cycle that is not allowed. So, the 7 is not chosen 7 is 7 it is not allowed.

Then 8 if I select also it is creating a cycle 8 is also not allowed this is not allowed this edge is not allowed this edge is also not allowed because this is also creating a cycle. Then I have 9 so it is not creating any cycle, so it is allowed. So, basically, we have all the nodes already in the graph and they are connected by the minimum weight of the edges. So, that is why this is the minimum spanning tree. So, all the nodes are there in the graph and we have selected the edges which is having the minimum cost to create the minimum cost spanning tree.

This is using the Prim's method. Now we will look into the Kruskal's algorithm. Kruskal's is basically a greedy algorithm. So, the method what is followed for Kruskal's algorithm is that we need to choose vertices we need to choose vertices and first we will sort all the edges monotonically in the increasing order of the weight. So, we will create a list of edges in so we will create the sorted list of edges in increasing order based on the weight assigned to each of the edges.

So, then we will choose one of the edges at a time to create a minimum cost spanning tree. So, what we have to do? We have to use one of the edges and check that whether it is creating a cycle or not. If it is not creating a cycle then we will add to the list and finally we will see that if all the nodes or the vertices are present in the tree then our minimum cost spanning tree is created. Let us discuss this with an example. So, here this is a Kruskal's algorithm.

So, if you can see here we have sorted list actually sorted list of edges. So, if you can see which is having the minimum cost? Minimum cost is having which is having the age cost of 1. So, this one has having the age cost or the age weight is 1. Here the age weight is 1. So, these two is selected. After one these two so G and H is G and F is selected. So, this is selected. G and F nodes are connected. Then 3 is not there in the list. Then the fourth the 4 we have 2, 4 is there in the list.

So, we will take one at a time F to D then B to A this is selected. Then after 4 we have 5. So, G to C G to C is selected. Now after G to C basically we have after G to C we have then after 5 basically we do not have 6 in the list then we look into 7. If I select 7 which is connecting between node C and D that is creating a cycle.

So, this is not allowed. Now after 7 I have 8. So, 8 if I select here that is also creating a cycle between if I B C is selected then it is creating a cycle. So, the 8 is not allowed. Similarly, if I look into this 8 A and H. So, this is also creating a cycle. So, this is not not allowed. Then I have D and E which is 9 which is not creating a cycle then we can select that one. Now if I select 10 then it is creating a cycle not allowed, 11 is also not allowed, 14 is also not allowed. So, finally we got the minimum cost spanning tree using Kruskal's method. The Kruskal's method is a greedy based algorithm which basically looks for the minimum cost edge and add it minimum cost edge and add that one to the tree.

Now we will discuss about the shortest path algorithm. It has lots of use in case of VLSI routing and basically it is it has lots of use in VLSI routing physical design routing. So, let us discuss about the Dijkstra's shortest path algorithm. It is basically a single source and single source shortest path problem on a weighted directed graph.

So, we have a weighted and directed graph is given to us. We need we are starting from a single source, and we are going through all the edges to find the shortest path from the source to the final node. And but requires non-negative weights on all the edges. So, all the weights should be non-negative. Of course, in case of a VLSI physical design the weight of the wires the delay of the wires is always positive.

So, that is not a problem. If we apply this algorithm for VLSI routing algorithms. So, let us look into this how it is working. This is the pseudo code of the Dijkstra's shortest path algorithm. So, this is the Dijkstra's shortest path algorithm. So, in this case we will start with a source node.

In this one this is the source node actually. So, from the source node we will find out the shortest path from that node to all the nodes. How we can find it out? So, initially what we have to do? We have to assign infinite to all other nodes. Apart from the source node all other nodes has a word actually the distance from the source node to all the other nodes which is not discovered or not visited basically that will be assigned infinite. Infinite is a very big number to compare. So, all the other nodes are basically given infinite. So, the source node has the distance basically 0 and rest of the things are infinite. So, if I go to basically t the distance is basically 4. So, the 4 is less than infinite.

So, 0 plus 4 is less than infinite. So, I will replace I will choose which is having a minimum distance from the source node. Since infinite is a very large number and 4 is the small number. So, I will remove that infinite and basically add 4 to that node t. Similarly, if I go to the node z which is having weight of 8. So, the 8 is less than infinite. So, this z has the weight of 8. Node z has a distance of 8. Now if I do another iteration, then what will happen is that till this point is basically 4. Now if I add 1 to this one, I can reach z in 5. So, the z node, node z has basically earlier it is 8.

Now the node z is 5, which is smallest, the smallest is 5. So now I will replace remove 8 and put 5, add 5 here because that is the shortest. So, here I do not need to go to the source node, I need to work from the present node. The present node here is the t node. t node calculation has already been done, then we are adding what is the distance from t node to z node is 1. So, the distance from t node to z node is 1 and if I look into the overall path delay that is automatically calculated which is 5, which is less than 8. So, your z node becomes 5, distance of the z node becomes 5. So now after z is 5, now I have t has other paths also. So, from t earlier it is this node weight is 8.

So this will be 4 plus 8 is 12. This is 12, then 4 plus 14 it is basically 18. 4 plus 14 it is 18. So, this is here, here there is a very interesting point will come. If you can see here, here till this point is the z is 5 and if I go to y node by this path, so it will be 6.

So which is less than 18. So, in the next iteration, when z will be 5 plus 1 is 6 and y node distance is 6 which is less than 18. So, the y node will be replaced by 6. So here if you can see now we are exploring the y node. Now we are exploring the y node. So, whenever we are exploring the y node, what are the nodes connected to the y node? So, it is connected to u and x. So, if I go to x node, 6 plus 2 is 8, x node will be 8. Then your u node basically, u node is basically 6 plus 5 is basically 11. If I go by this method, I will reach by 11. So, if I go by this edge, it is 11. If I go by this edge, it is 12. So, which is the shortest? The 11 is the shortest. So, I need to remove this edge, I need to add this edge. So, from node x, node x I will visit the other nodes. From node x, if I go to u node, it is 18. If I go to node v, it is 12. If I go to node w, it is 19. So, if I can see here, so you have 18 is there, 11 is there. 18 is there, 11 is there. Which is the minimum? 11 is the minimum. So, this path will not be selected.  This path is not selected.  And about this 4, because there is only one path, 12 is less than infinite, so it will be replaced to 12.  Similarly, here w 19 is less than infinite, so it will be replaced by 19.  So here if you can see, this will be 12, correct. This path will be selected and this 19 will be selected.  But this 10 will not be selected.  Now I have, if I go by u, from u it is 18.  So here if I go, it is 18. Earlier 12 is there.  Which is minimum?  12 is minimum.  So, this path will not be selected.  This path will not be selected.  Now after that, if I have 12 here, 12 plus 9, 12 plus 9 is what?  It is 21 and here it is 19. So here it is 21.  Now already it is 19.  So, 19 is the smallest.  So, 19 will be selected, 21 will not be selected.  So, this path will not be selected.  So, this is the Dijkstra shortest path algorithm.  What is the advantage, main advantage of this algorithm is that it finds the shortest path and here all the age weights are accumulated in the node. We do not need to traverse from the initial source node to the destination node or the sink node all the time.  Since all the node has their weights and we are comparing and storing the minimum cost value in the node, it finds the shortest path in efficient manner, which is very useful for VLSI physical routing in case of VLSI physical synthesis.  In these two lectures, we discussed about graph search algorithms, depth first search and breadth first search and we also discussed about the minimum spanning tree algorithm using two popular spanning tree algorithms like Prim's algorithm and Kruskal's algorithm.  The Kruskal's algorithm is basically a greedy algorithm and we discussed about shortest path algorithm like Dijkstra's shortest path algorithm with one example and these algorithms has tremendous applications in the field of VLSI physical design flow.

Thank you for your attention.  Thank you.