VLSI Physical Design with Timing Analysis

Dr. Bishnu Prasad Das

Department of Electronics and Communication Engineering

Indian Institute of Technology, Roorkee

Week 01

Lecture 05

Graph searching Algorithms

Welcome to the course on VLSI Physical Design with Timing Analysis. In this lecture, we will discuss about the graph algorithms for VLSI physical design. So, the content of this lecture includes the graph search algorithms. For example, we have two popular graph search algorithms like depth first search and breadth first search. So, what is graph search algorithm? In this case, we will follow the edges of the graph to visit the vertices of the graph systematically and discover each and every node in the graph. So, there are two popular algorithms which is used for graph search algorithms are breadth first search BFS and depth first search DFS.

So, we will discuss these algorithms in detail with many examples. So, in this depth first search algorithm, first of all we will use a stack data structure to keep track of the vertices that we are visiting. Then we will choose the vertices, stack vertices, then we push it to the stack and if the stack is not empty, then we will do these steps and we will repeat the step 2 and 3 until the stack is empty and we will basically understood this with the help of an example. And this is a pseudo code of the depth first search. pseudo code is not basically any it is a type of representation which is not basically particularly pertaining to any particular implementation of algorithm, but it give the overview of the algorithm, how what are the main steps of the algorithm. So, here what is happening is that you have a DFS. So, what we are doing here is that, so that graph has V, graph has V and E the V is the set of vertices, E is the set of edges, then that each vertices initialized with white and the parameters are basically attributes of the graph is nil initially and at will start the graph at time t equals to 0 and each vertices when it is visited, then the then it has a discovery time will be given to that one and similarly, we have a finish time given to that. So, let us go into the example. Let us discuss one example of DFS.

In this here we have a graph is given to us and the graph has basically denoted by usually all of you know that the graph is denoted by V, E and we are interested in the vertices the V. So, the V is basically set of vertices which is given in this graph which is basically A, B, C, D, E and all the edges are given, it is a directed graph. And what we are finding in this example is that in which order we should move in depth first search, which are the nodes we will visit in the depth first search and each the main goal is that we need to find out the discovery time and the finish time of each of the vertices. So, basically if you can see here, we have vertex A is there which is the start vertex. So, the start vertex or the source vertex which has a basically the discovery time the first is the discovery time and the second one is the finish time we will write it later is the discovery time or the start time we visited the node A first. So, at time t equals to 0. So, t is 1 here. Then if you see two node vertices A, we have three adjacent vertices are there B, C, B, E and D out of which we will choose one of them and in this case we have chosen B here, we will choose the B here. So, we will put the B in the stack. What is stack? It is basically a data structure which is called as last in first out data structure.

So, the last element entering the stack will come out of the stack first. So that is why it is called LIFO data structure last in first out. Now, the node B is visited, and the discovery time is 2 here. Then the adjacent vertices of node B are D and C. But we will choose one of them. So, in this case we have chosen D. So, D is now pushed to the stack and the discovery time is 3 here. Then the adjacent vertices of node B is E and it has two adjacent vertices E and C. We have chosen only the E. So, basically that discovery time is 4 here.

Then if you can see here the adjacent vertices of node E is B because it is a directed graph only one path is there from E to B. So, basically what we are doing is that B is already in the stack. So, we cannot visit the B again. So, it will go to the node B, but it will do the backtrack. And here the job of node E is already finished. So, job of node E is already finished. So, this is called the finish time of node E. So, after the node E is finished then it will go to the node the top element in the stack. After the node E is finished then it will go to the D node D or node and vertices are the same thing. I use node and vertices sometimes node sometimes vertices. So that means the same thing. So, the node D has adjacent node C is there. The only node C is adjacent to the node D. So, it will be put to the stack. So, the node has a loop there, cycle there and that cycle will always go to the same node. So, then the finish time we need to do the finish time for the node C because there is no adjacent node to the node C. So, the finish time for the node C is 7 now. Now after that it will do the which is the top element in the stack. So, the top element in the stack is your D then it will go back to the node D, and it will write down the finish time is 8 here. Then the top element in the stack is B.

So, then the node B will be visited, and its finish time will be 9. So here you have node A. Node A has multiple nodes adjacent to that one. But what are the multiple nodes connected to node A? E, D and B but all are already finished their job. So finally, the node A has a finish time of 10.

Now the my stack is empty and this DFS thing is complete now. DFS means the depth first search of the graph is complete. So, this is the one first example. Then the second example is basically here what we are doing is that we are also the same method. We are doing the we are maintaining a stack here.

So, the first element in the stack is U we will put it into the stack. Then the second element adjacent to the node U is basically X and V but we have chosen V and we put that into the stack and the start time or discovery time for the node V is 2 now. Then the adjacent node for the node V are Y. So, it has only one adjacent node. So, that will be put to the stack. After the node Y is there then adjacent node for the node Y is X. So, then the adjacent node for the node Y is X. And if you can see here the adjacent node for the node X is basically node V it will go to the node V but and see that it is already in the stack. So, it will do a backtrack from there and the node X will be having basically the finish time of 5. Then the next node basically your Y here if you can see that there is no path from adjacent node for the node Y.

So, the node Y is not connected to W because it is a directed graph, this path is not there. So, the then the finish time of node Y is 6 then the finish time of the node V is 7, then it will go to the node basically the U which has two paths, and both the path has completed their job. So, the node U has a finish time of 8 then basically the graph is not complete. Now we have to look into the nodes which has not been visited so far and the first node we will enter is the W and the stack time for the node W is 9 and basically the finish then what are the adjacent node to node W is that your Y and Z but this Y and Z is basically we need to choose one of them but Y is has already finished their its job. So now we will look into the Z we will put the Z into the stack then there is a loop is there or the cycle is there in the node Z and then what we have to do is that the node Z basically has to finish its job.

So we need to that visit of the node Z that is the finish time will be 11 then we have the last element in the stack is your W and that W has a finish time of W whenever it is going to the W it has two path and both are already it has two path but both are already visited so the W will get a value of 12.

So, this is the basically the breadth first search algorithm here we will describe this with a example this is the pseudo code of the breadth first search algorithm these two are the pseudo code of the breadth first search algorithm. Let us take an example here so here in case of a depth first search we have maintain a stack data structure but breadth first search algorithm will use a Q data structure. So, if you can we have to define a stack node or source node here. This is the source node from where we are starting. So, from the source node we have what are the adjacent node connected to the source node here we have basically U, V and T are connected to the source node.

So, the all the three nodes will appear in the queue. Q is basically is a first in first out data structure. Q is a first in first out data structure. The first element entering the Q will come out of the Q first so that is why it is called first in first out data structure. So, if you can see here the elements to the Q are T, U and V which are adjacent to the node S. So, we need to the first element come out from the Q is T like all of you know that there is a queue in railway ticket counter or if you go to any place the bus or something there is a queue is there.

The first person who is there in the queue that will go to be served first. So, the same concept applicable is here. So, we have a which is the first element here the first element is the T. So, the T node we need to visit first. So, the T node and one more thing is that since all the three are visited at the same time they have denoted by 1 all the adjacent node connected to S will be denoted by 1.

Now we will go to T. What are the adjacent node for the node T? W so W will be denoted by 2 this is the second level. Then we will see that what is connected to U. Next element is U but what are the adjacent element to the U are V, S and T but all are visited. So, we do not need to do anything.

Then we have to go to node V. Node V has also two nodes connected to that one adjacent node of node V and both are already visited so we do not need to do anything. Then we will go to the node W. W is connected to what are the adjacent node of W is X and Y. So, the X and Y are the adjacent node for W. So those will be assigned 3. Now we need to check for the top that element X. So, the element X what is the adjacent node for element sorry node X. Node X has Z and Y. Y is already there in the Q, but Z is not there in the Q. So, we need to insert Z in the Q and basically the label assigned to Z is 4. So now we visited Y, Y is already visited then Z we have already visited. Now the Q is empty and our basically BFS is complete. This is one more example here basically this if you can see here the start node is S what is the adjacent node for the connected to S are R then U and V. They are assigned 1.

All of them will be assigned 1. Then what is the top element in the Q or the last element means the first element in the Q is the R. What is the adjacent element to the R? R basically the W and T. So, W and T. So, the W and T if you can see here will be given 2. Why 2? Because this is the second level. And basically, the W and T then the next element is your U. U is basically connected with your Y. So, Y will also be 2. Then you have V. V is also V is not connected to anyone which is already visited.

So, V is basically connected to W and Y which is already in the queue. Now we will go into the level 2. Let us say if I draw this is the source, I have three things. One is your R is one vertex. Then you have U then V. Then the next one is connected to R is basically your W. Then your T. W and T and this is level 2. level 2 this is level 1. This is level 2.

Now what are the nodes connected to T? P is already visited. W is already visited. W is there. So, W is connected to your X and Z.

So now X and Z will be inserted into the Q. Now X and Z will be given 3. If you can see here your basically W is connected to node X and node Z, and this is level 3. So, these are the three levels of breadth-first search. Now you have visited X. There is no more nodes are there. Then you visited Z. There is no nodes connected to Z which is not visited. Then your queue is empty. Then BFS algorithm is completed. So, this is the tree BFS tree whatever you get out of breadth-first search.

In this lecture we discussed about the breadth-first search and Depth- first search algorithm and breadth-first search and Depth-first search algorithms are very useful for VLSI physical design algorithms. What we will discuss in our future lectures, and it has applications in timing analysis, it has applications in global routing all these things we will discuss in future classes.

Thank you. Thank you very much.