C-Based VLSI Design
Dr. Chandan Karfa
Department of Computer Science and Engineering
Indian Institute of Technology, Guwahati

Module - 03
C-Based VLSI Design: Scheduling
Lecture - 08
Multiprocessor Scheduling

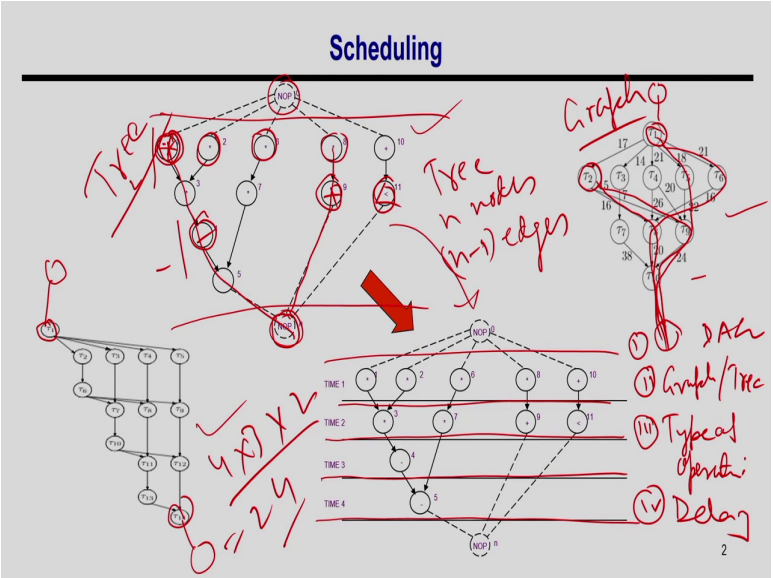
Welcome students. In today's class, we are going to discuss on Multiprocessor Scheduling. In last 2 classes, we are discussing on scheduling on high level synthesis. So, as you remember that scheduling is something, where we assign time steps to operations and it is needed because the actual initial C code is untimed. So, you do not know which operation to be schedule in which clock.

So, you have to decide that and that is what is happening in the scheduling steps and we have already discussed that and in the last class, we also discussed that the ILP formulation, how we can actually model this scheduling problem as a ILP and we have seen how to with example how those constraint will be represented and all. So, and you also seen this scheduling problem is in general NP complete.

So, NP complete as you might avoid that this is basically this means a complex problem and for this, there is no polynomial time. So, algorithm avail deterministic polynomial algorithm available. So, you usually do not have algorithm like n order of n, order of n square, n log n to solve this problem for to get the optimum results. So, you have to go for exponential algorithm n to the power k and so on. So, where n is the number of nodes in the behavior.

So, that we have already understood and then, so in today's class, what I am going to discuss is we try to take a special class of this scheduling problem and then, we try to see whether that particular class is solvable is still NP complete or if it is not then what is the particular subset or the subclass of this scheduling problem which is actually is polynomial time solvable and that is what is this multiprocessor scheduling is the topic.

(Refer Slide Time: 02:27)



So, again you remember that the way we modeled our behavior. So, we have a CDFG, Control and Data Flow Graph, where we have loops and then, if else branch everything and we have actually modeled as a control and data flow graph, where each basic block is a straight line of code and where there is no control flow and then, the way we have discussed that we are going to take one block at a time and we model them as a sequence curve.

So, basically it is the representation is like we break those operation into three address form and then, we identify the dependency among the operations and that dependency is basically a data flow graph and then, that data flow graph, we represent is basically as a sequence graph, where we put a sink node and source node for convenient representation purpose and in this particular this sequence graph is actually a dag; the direct acyclic graph.

So, direct acyclic graph in the sense, there is no loop and it is understandable because there is no control flow, we have a straight line of code. So, since the code is going to happen from the top to bottom manner, so there is no back edge. So, there is no backward dependency because it is a forward dependency you might have loop and that loop will be in the control flow; but within a basic block, there is no back edge. So, this is a direct acyclic graph.

Now, if you just try to understand, so here, I have taken various type of graphs. So, this is one graph, this is another graph and this is another graph and this is my source node and this is a sink node.

So, we can exclude this source node, sink node because they are the dummy nodes and we can actually for this graph, this is the source node and this is the sink node; this for this graph also this is the source node and this is a say there is a sink node . Now, let us try to understand this particular graph.

So, this is our input behavior and this is what I am going to schedule and I have seen that is basically scheduling is basically assigning time step, where I am going to say this operation is going to happen in time step 1; this is going to happen in time step 2; this is going to happen in time step 3 and this is going to happen in time step 4 and so on. So, this is what scheduling does.

But what is the characteristic of this particular sequence graph? So, if you see this graph, if you just exclude the source node or sink node is basically it is a tree. So, what is the tree? Tree means if there are n number of nodes, you have n minus 1 number of edges; n nodes, n minus 1 number of edges.

So, and what is the special property of the tree? It is basically you can assume that from this any node to the sink node, there is exactly one path. If you come from this node, there is exactly one path to the sink node. You take any node, there is exactly one path. Then, this is basically a tree. So, your sequence graph or the input graph can be a tree in a special circumstance; but in general, it may not be. You consider this graph, so here you consider this node.

You have how many paths to this sink node? I have one path through this; I have one path through this. So, there are more than one path, you consider this node from this path, there are many paths. So, this is one possible path, this is another possible path and so on.

So, this is not a tree, this is a graph. So, this is the first property that this sequence graph in specifically can be a tree or it in general, it is actually a graph. Here also there are many paths. So, there are we can consider from this node to this node, there are 4 into 3 into 2

which is basically there are 24 paths. So, this is definitely not a tree. So, this is the first thing. So, your graph in general, this dag is a acyclic graph, there is no cycle, there is no back edge. But it in general, it is a graph; it is not a tree. But in a specific case, it can be a tree as well.

Next is we have seen that this the nodes can actually perform many operations. This is a performing multiplication, this is performing subtraction, and this is performing plus, this performing greater than equal to and all. So, these are all different-different type of operations and it is understandable that in general in hardware, this all operations are not execute in the using the same module.

But you can have different function units like you have a multiplier, you have a adder, you have a divider and so on and so, this is something that means, you have the operations which have of different type. So, that is one aspect.

So, one is the structure of the tree which is graph or tree, the structure of the sequence graph. So, the first thing is it is a DAG that there is no confusion. So, all is all are DAG, there is no cycle. Second is it can be graph or tree. Second thing is the type of nodes type of operation. So, what is the distinguishing factor that I am try to identify?

The fourth is the delay of the nodes. Delay of the nodes in the sense, so for example, I mean we have already formulated the problem with this multiplier can have a delay of 2, this adder can be delay of 1. So, basically it is not that all operation will take same amount of time in hardware. It might have some complex operation it might take say D cycle.

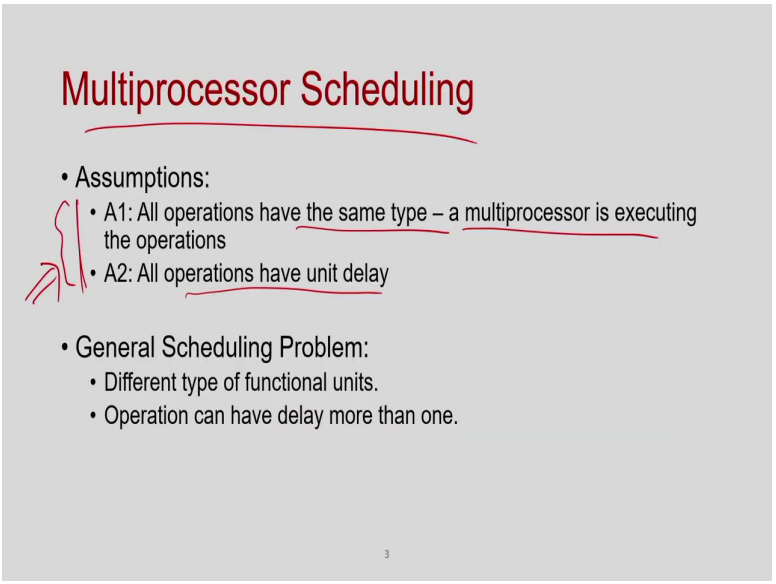
So, for example, here I am just giving a graph, where actually I have just tried to show that this node say take 15 cycle, this node says take 7 cycle, this node takes a 20 cycles and so on. So, it can happen. So, if this operation may not be only a multiplication is a set of operations as well. So, you can represent say complex operations using this node as well.

So, these are the characteristics of this general sequence graph. So, if we try, so the problem is that if in general if this graph is a it is a graph, if you have different type of operations, if you have different-different node at different delays, the that the problem is actually NP complete the scheduling problem .

So, the scheduling problem whether it is a latency minimizations under resource constraint or it is a resource minimization under latency constraint, both are NP complete problem; it is a complete hard problem . So, if we try to assume that this particular graph has some special characteristics that this is like a tree or say I have the type of operation can be same or the delay is unit.

So, in that particular sub case or the sub structure is this still problem is NP complete. So, that is something, let us try to understand.

(Refer Slide Time: 09:02)



So, in the multiprocessor scheduling, we have certain assumptions; so, on these four parameters. We assume that this particular all operation of the same type; that means, there is a multiprocessor which can execute this operation. So, that means, although this nodes can be plus, minus, multiplication, division; but you can assume there is a kind of processor which can perform any operation.

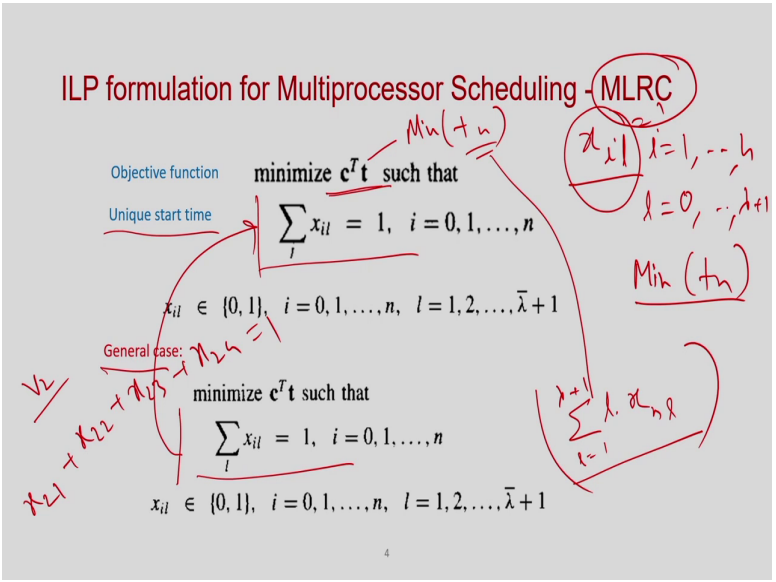
This can be generic purpose processor as well which can do any operations. So, the first, so basically what I assume that this is not there. So, all operations are of same type. So, I do not have to bother about whether it is a multiplier or divider. So, I can consider all the nodes as a same type, they are not different types.

The second one is that this all operations have unit delay. So, unit delay means as I mentioned that earlier this multiplier will take 2 cycles, this adder will take 1 cycle and so on. So, I am also assuming that there is no delay constraint. So, all are of delay one.

So, every operation can be executed using single clock. So, multi-processor scheduling assume these two constraint that I do not have to take different type of functional units, since I have a single multiprocessor which can execute all type of operations and then, I can actually consider all nodes of the same type and also assume that this operations are of a unit delay . So, delay of 1.

So, under this particular assumption, let us try to formulate this problem as an ILP because in the last class, we have solved this is in ILP. So, if we have these assumptions, how this constraint will change, let us let us try to understand that.

(Refer Slide Time: 10:40)



So, you if you remember that this ILP formulation, we have taken a variable called x il, so which is basically because what is our objective i? Our objective is to identify the time step, where this operation is going to start its execution and that is unknown. So, I have just taken a variable this i l; where, i varies from 1 to n representing the nodes and l representing the time step, which is basically 0 to lambda plus 1.

So, if a operations i be i, the schedule in time step l, then this will be 1; otherwise, it will be 0 and this is what I am going to identify through ILP formulation that I mean I hope you understand or you have you remember that. So, within that assumption that you know this formulation.

So, if we take this ml MLRC problem, where we try to minimize the latency under resource constraint, our objective function is that is basically c t, where basically its nothing but minimize the schedule time of the last node. I think that you remember that if we schedule the last node as early as possible; so, if the time step schedule time of the last node is early because of the dependencies, all other node will be executed before that and hence, this is the minimum solution.

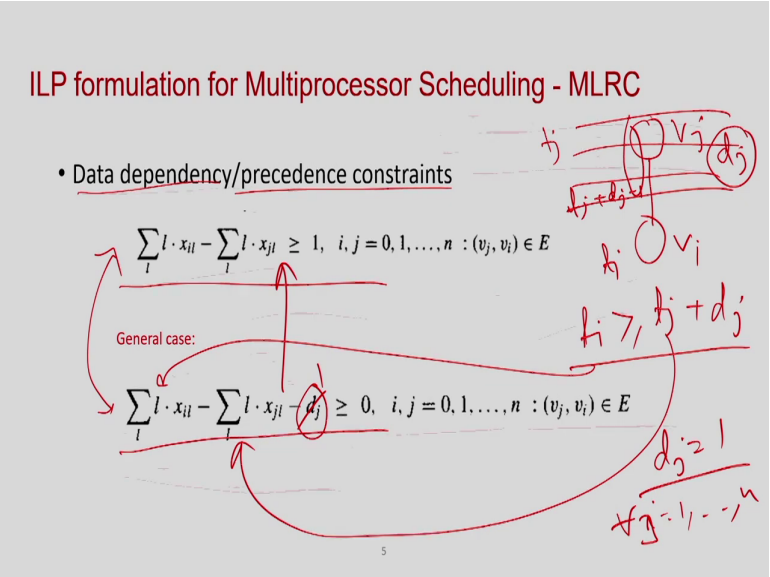
So, our objective function for this minimization of latency is that it is basically nothing but mean that start time of the last node and we have to represent this t n in terms of x because there is no t n which I can represent as like this that x n l . This is and this is l into this, summation of this l which is basically 1 to lambda plus 1. So, this should be minimum.

So, this is something we try to achieve. So, and what was the constraint? So, the first constraint was the unique start time that although there are many variables possible for each node, we have to assume that one operation can start only once in the behavior.

So, and that is given by this expression that we already remember that for say a node say. So, for a node say i. So, it can say if there are 4 say time step and this is a node say 2. So, 21 plus x 22 plus x 23 plus x 24 if I am just talking about v 2 and there are say 4 time step. So, this must be 1. So, that means, it only can schedule in 1 time step and since here for these two assumption that does not change anything. So, these two assumptions that all operations of same type and operations are unit delay has no impact on this constraint.

So, in general case, where there is no such restriction, the constraint is this and for my case, our case, where we have these two assumptions, this constraint will be same because this these two assumptions does not impact on this particular constraint. So, for this unique start time remains same whatever the general case it will remain as it is.

(Refer Slide Time: 13:53)

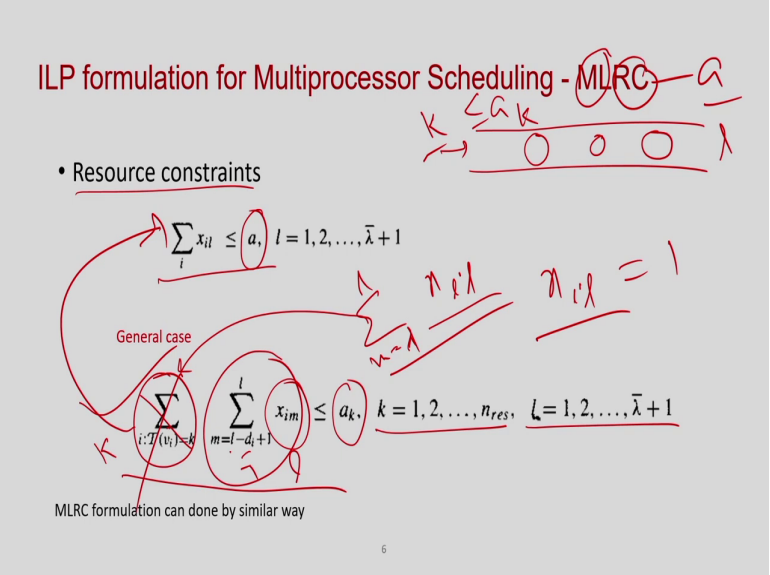


So, next one was the precedence constraint or data dependency. So, if you remember that if there is an edge from v j to v i and this node has a delay of d j, then what is the idea that if this is schedule in t j and it will take say till t j plus d j minus 1 . So, it will execute in this time step. So, the t i the time step of this t i must be greater than equal to t j plus d j , that we have understood and we can represent this t j, this t i by this and t j by this.

Because I have to represent everything in terms of x ij, x il and so, I just rewrite this expression like this in general case. So, what is the difference here? So, in generous case, this node has can have a delay of d j, but in our specific multiprocessor scheduling my d j is equal to 1 for all nodes for all i or for all j 1 to n. So, since this is 1, so I have to just represent rewrite this by this and I can get this expression.

So, for multiprocessor scheduling, my data dependency constraint will be this instead of this. So, this is the correlation understood.

(Refer Slide Time: 15:18)



So, now, I will move on to the next constraint. In the next constraint was the resource constraint, you remember and the resource constraint says that in each time step, the number of operation that are running in that time step of set 1 type of resource k must be less than a k; that means, the number of resource available of type k.

So, I have to identify the number of operation that are running in the time step l and I have to say what is the type if their type is k, I will only take those operations just type k. I have to make sure that the number of such running operations must be less than of the resource bound of type k and that was given by this expression. I hope you remember.

So, these actually identify the operations that are running in time step l and these actually give me the things which are of type k. So, this is basically gives me the things the operations that are running in time step l of type k and that is must be less than equal to a k, where this k will vary for all type of resource and l time vary for all time of time step.

So, now, how this two assumption will impact this particular expression? So, what was the assumption? You just to recall that I do not have different type of resource, I have only one type of resource. So, I do not have this k type of resource, I have only one; so, this will go.

This I do not need because this is its basically one type of resource are there and next was that here, this is coming because we have to identify the operations that are running. It is basically identified by this expression, but since for my case, it is basically my d i equal to 1.

So, the delay is 1 unit delay. So, this summation will become m equal to so this l minus 1 plus 1, so m equal to l to l. So, which is that this summation is not there. So, it is basically x il. So, basically what does it mean? It is basically this operations that are l to l; that means, this is the all the operations that are running in a time step.

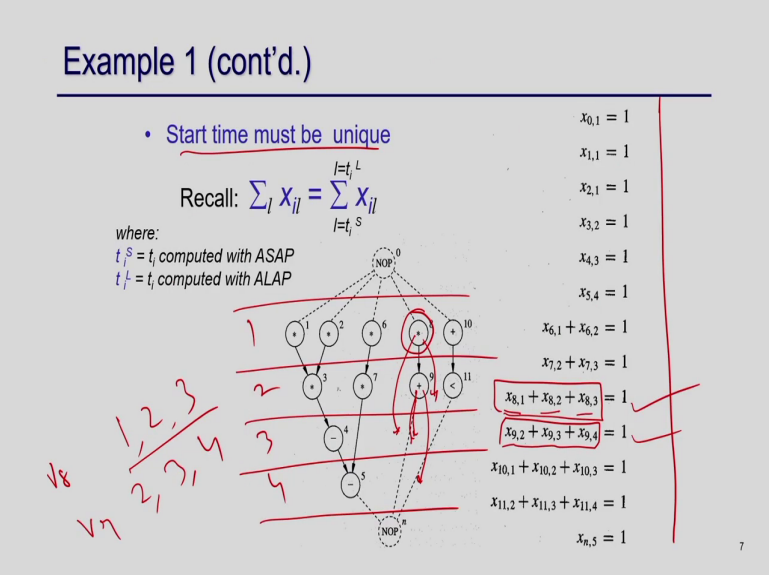
So, what just to summarize that here all type of, so all the nodes that are running in this time. So, it is basically x il. So, I have to identify all the x il which is 1 in this time step and that must be less than equal to a. So, this is how I this expression reduced to this expression.

Because I do not have to check for all type of resource, I do not have to identify the running operations; only the operation, if it is schedule in this time step. So, all the operations that are scheduled in this operation in a time step l that must be less than equal to the total resource bound because I have only one resource bound.

Because these I have only one type of operations. So, this is how this will move. This objective function will remain same because it is minimization of the latency. So, this is how I can actually formulate this multiprocessor scheduling as an ILP, where my objective function as it has the general case. Unique start time would not change, it is the same constraint.

Data dependency all same, only this delay will be replaced by 1 and this resource constraint is be simplified. Because here there is no running or percent concept because this all operations are of unit delay and what we have to just check? The all operation that actually schedule in a particular time step must be less than of the resource bound of that state that is given by this.

(Refer Slide Time: 18:52)

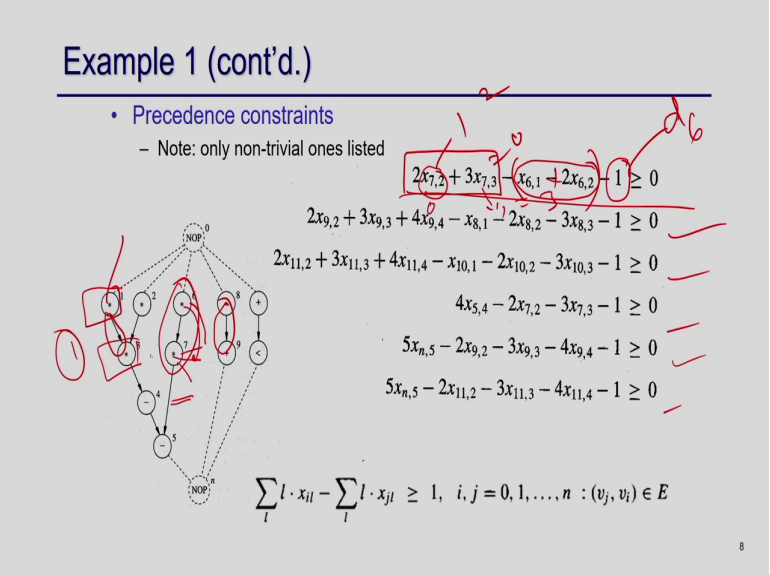


So, let me take the same example, since the resource constraint does not change, it will remain at as it is. So, this is sorry unique start time constraint does not change, whatever was for generic case, it will remain as it is. I will just explain 1one of them say. So, let us say take me this 8.

So, since this if I am assuming, this is actually going to be schedule in say fourth time step; so, 1, 2, 3, 4. So, this even if I it come here or here, the total time step would not differ. So, similarly if this operation comes here, this operation comes here, if this operation comes here, this operation can be schedule come here.

So, that means, this 8 can schedule either time step 1, 2 or 3 and 9 can schedule; so, this is v 8 and 9 can schedule either in time step 2, 3 or 4 . So, I have to say that this 8 schedule either in time step 1, 2 or 3; but it can schedule only one of the time step that is given by this. Similarly, 9 is something either 2, 3 or 4; but only one of them, it can schedule only one of the time step. This is how I can actually identify the unique start time constraint for this ILP formulation.

(Refer Slide Time: 20:03)



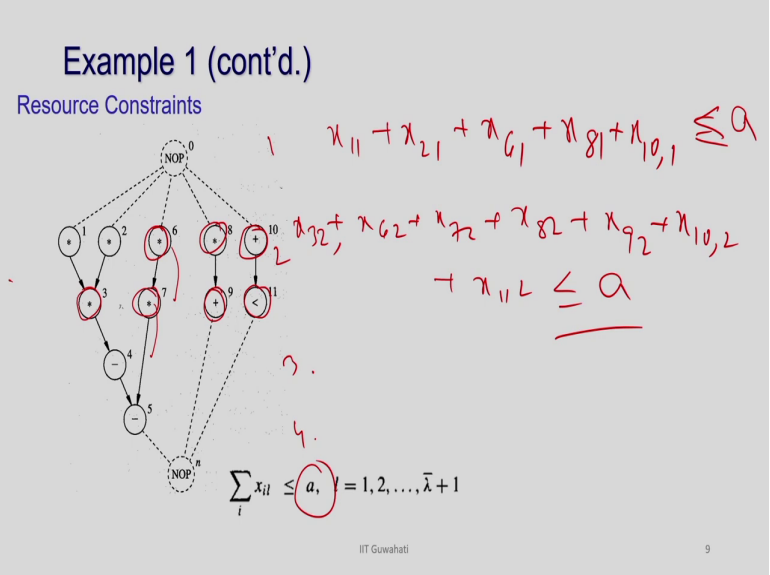
For the precedence constraint, it is basically for every edge. So, if you just take an edge here, so this is I have to check the possible start time of this and the possible start time of this, but the difference will be 1 because the this operation has a delay 1 now. So, what are the possible start times of this node say 7? So, let me take this example. So, 7 can start here or here, without touching the delay and if 7 work here, the 6 can come here.

If 7 schedule here, 6 has to schedule here. So, then, so this is the possibility of the time step, where 7 schedule either in time step 2 or time step 3. So, if it is schedule in time step 1, this will be 1. This will be 1, so it will become 2 and this will be 0; so, this will set 2.

If it is schedule in time step 3, so this will be 1 and this will be 0; so, it will effectively means 3 . So, this is how can I did define the time step of the start time of the node 7 and this is actually given the start time of the node 6, it is either in time step 1 or time step 2. And it saying that the and it is the delay of the node 7, so start time of 6 must be a start time of 7 must be the start time of this node plus 1 . This is what this expression says.

So, now, I have to take each such edge and I have to represent them as in the constraint. The only difference from the generic case, here it is basically d 6. In previous case, it was d 6; here, it is just 1. So, that is the only difference from the generic case.

(Refer Slide Time: 21:43)



And for resource constraint is basically I have to see what are the node that can be schedule in time step 1 because ah. So, what is the node can schedule in time step 1? So, 1, 2, 6, 8 and 10. So, that means, x 11 plus x 21 plus x 61 plus x 81 plus x 10 1 should be less than equal to t; it should be less than equal to a. Because this is the resource bound.

I have only one type of resource and for time step 2, so if you now just think about the time step 2. So, obviously, 3 can schedule; see here 7, 9, 11 and also, I have told you that this can come here and that time 6 can come here, 6 also can schedule in time step 2, 8 also can schedule in the time step 2, 10 also can schedule .

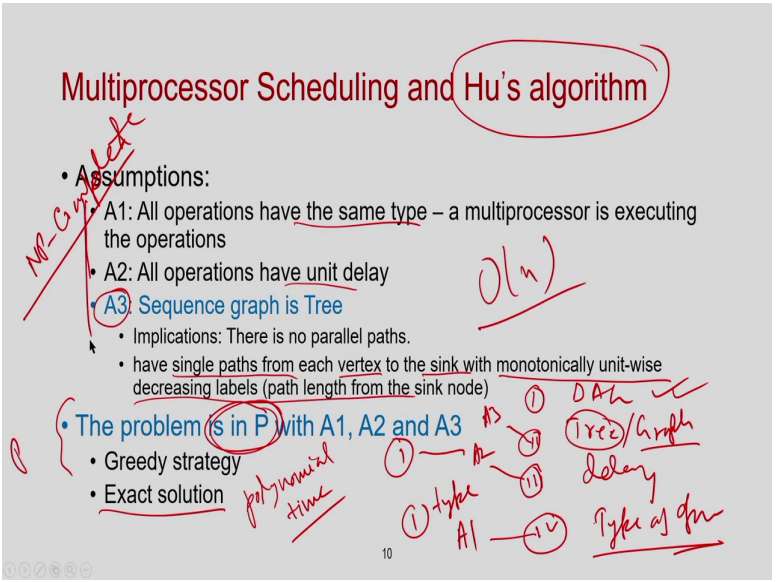
So, I have to write x 32 plus x 62 plus x 72 plus x 82 plus x 92 plus x 10 2 plus x 11 2 should be less than equal to a. So, there are many operation which is actually likely to be scheduled in time step 2; 1, 2, 3, 4, 5, 6, 7 operations can be scheduled; but only if say a equal to 2, only two of them can get scheduled in time step 2 . So, this is how I have to write constraint, resource constraint for each time step.

This is for 1, this is time step 2, this is for time step 3, this is 4 time step 4, I have to write such operations. So, this talks about the resource constraint.

So, if I just give this constraint 3 constraint and with that objective function, it will actually give me a solution which is actually minimize the latency because my objective is to

minimize the latency under a given resource constraint a and since I have only one type of resource, I have only one resource bound . So, this is what this problem is and this is the ILP formulation. I hope you understand?

(Refer Slide Time: 23:48)



So, as I mentioned in the start of the class that the objective of this class to identify a sub class of my sequence graph which is polynomial time solvable and if we assume this the graph has all operations of same types that there is no different types that I have already assumed and the all operations have unit delay, still this problem is NP complete . So, even if this particular two assumptions are there, still these problems are NP complete.

So, as I mentioned this types are there are four characteristics I have considered; one is the DAG, which is always true. It is always true and then, it is a tree or graph and then, the third characteristic was the delay. Then, node delay can be anything and the type of operation. So, what we have taken this is so I have assumed that this saying the type of operation.

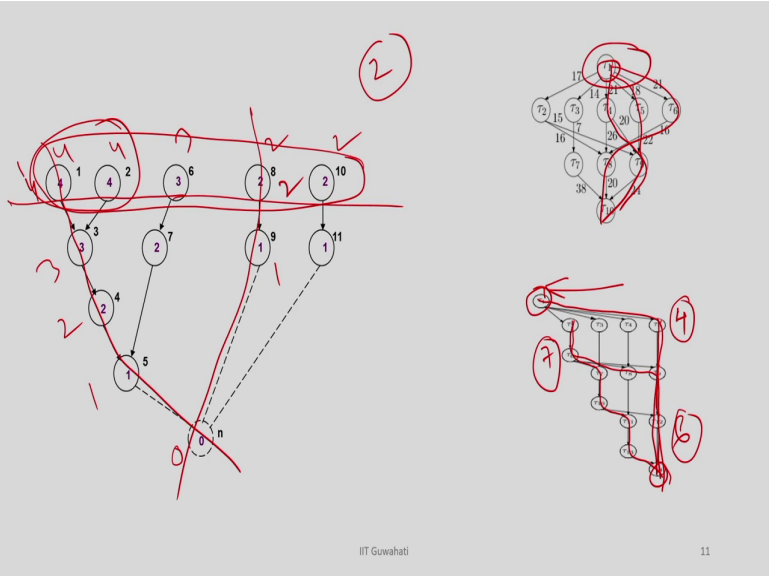
So, this A 1, assumption 1 saying that I have only 1 type of operations 1 type . And this delay, so this A 2, so A 2 actually restrict this delay also, say that the delay is also 1 for all nodes. But still even if this assumption was taken and this sequence graph can be graphed, the sequence graph can be a graphed, then still the problem is NP complete.

Now, if we assume the third assumption, I am assuming that this particular graph is a tree ; it's not a this sequence graph is not a graph rather it is a tree. So, that means, there is unique path from each node to the sink node under this three assumption, this problem turns out to be is in P. That means, it is a polynomial time solvable.

So, that means, I have an efficient algorithm which will be complexity of n to the power k, where k is some constant with these three assumptions. So, this is very important. So, when we actually assume this tree; A 1, A 2 and A 3, then I have this problem become P and I have exact solution in polynomial time and I can actually come up with some greedy strategy to solve this problem.

So, let us try to understand why this after once we assume this is tree, why this problem become easy.

(Refer Slide Time: 26:15)



So, let us try to understand (Refer Time: 26:24); I am not going to prove it, but I am just going to give you some intuition about that. So, the point here is that so because when you try to do the scheduling, you will start from say start step 1; you try to schedule the time step 1, then 2 and 3. So, when you try to take a decision on time step 1, if you have you can actually take a greedy strategy that if I take this node to be schedule here that will give you the best solution.

Then, it's this greedy strategy works and for tree, it this greedy strategy works, why? So, this is because of this property. So, if you just think about a single path from any vertex to the sink node, you can actually have a monotonically unit wise decreasing level which is basically the path length.

You can see here if you just think about this graph now, in this graph, so if you just take a path, if you think the length of this the node have some level which is say 0 here and the length distance from the sink node is 1, for this node is 2, this is 3 and this is 4. So, the this is the length of the path and this is in monotonically increasing . So, you can think about. So, you just take any path here I am also giving this is that this is 1, this is 2.

So, because of that, so when are you actually taking this node because at the time step, you are going to take the node which has predecessors are input . So, among these nodes, you see here the delay is 4, for this node it is 4, this is 3, this is 2, this is 2. So, you can actually choose because so if you see if that the distance is more; that means, it has more operation to be scheduled in the subsequent part.

So, you can definitely if you have a resource boundaries 2, you should must take these 2 nodes because that has the maximum criticality. In the sense that the distance from this node to the sink node is maximum. So, that is something. So, that since that is a unique path from each node, so there is always this is this value is always unique and you can actually take a greedy strategy that at this point, if I have 2 resources available, I must schedule this node with level highest level.

So, this kind of greedy strategy will works and I can actually take a local decision, without loing into the other part of the circuit because and that actually ensures that it is the best decision; whereas, you think about this kind of graph, So here you see here there are lot of paths and the distance of the paths are many.

For example, if you just think about this in this graph, there is a path from here which is 1. What is the distance? 1, 2, 3, 4, 5, 6, 7. There is another path from this node which is like this; so, 1, 2, 3, 4, 5, 6. So, if you think about this (Refer Time: 29:10) node which is basically of; so, if you think about this path which is delay 1, 2, 3, 4.

So, from this node to this node, there are many paths; 24 paths are there. Some of the delay nodes paths have delay 7, some of the path has delay 4, some of these 5, 6 and (Refer Time: 29:31). So, now, the point since there are many such nodes, so at this point, it is not something your decision some local decision at this point may turn out to be a incorrect decision because you do not know the structure of the rest of the graph .

Similarly here, so suppose here you can see here from this node there are many paths . So, there is one path like this, there is one path like this, so there are many paths and every path may have different-different delay. So, just taking a decision at this point because of there are many paths of different not unique delay, your decision at the start, a greedy decision at the start may not be correct. So, that is why in general, this is the intuitive idea that once you have a generic graph, yours local decision would not work.

You have to explore all possible choice, then only you can actually come up with the best solution; whereas, for this graph which is basically special type of graph which is a tree, since all path have unique there is no such confusion. So, you can actually take a local decision and which actually give you the best solution. So, that is why once you have this three assumption; this A 1, A 2 and A 3, your problem become P and you can actually have an efficient solution.

In next class, I am going to explain this one of such algorithm is called HUS algorithm which solve this problem in order of n time very fast, when these three assumptions holds for a sequence graph. So, with this, I conclude today's class.

Thank you.