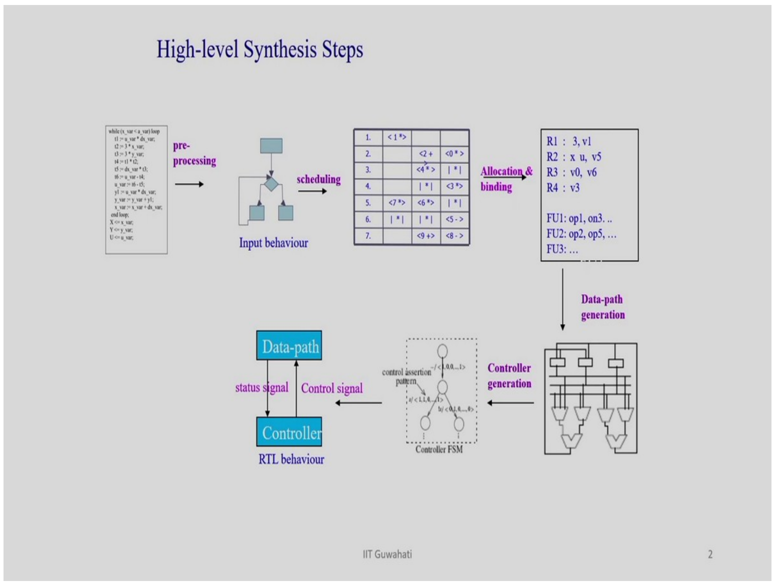
C-Based VLSI Design

Dr. Chandan Karfa
Department of Computer Science and Engineering
Indian Institute of Technology, Guwahati

Module - 05
C-Based VLSI Design: Allocation, Binding, Data-path and Controller Generation
Lecture - 16
Resource Allocation and Binding

Welcome back to my class. So, in today's class, we are going to learn about Resource Allocation and Binding.

(Refer Slide Time: 00:59)



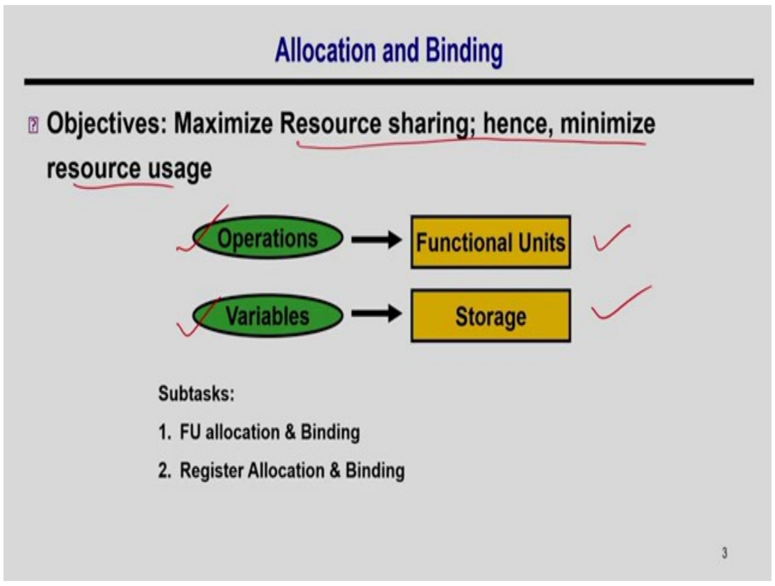
High-level Synthesis Steps

So, if you look back to this high-level synthesis, we have learned that high-level synthesis consists of several surfaces. Initially, we do pre-processing, followed by scheduling, allocation binding, data-path generation, controller generation, and then finally, we will get the RTL right and we have in the last previous few classes, we have learned about scheduling.

So, once the scheduling is over, we are going to do the allocation and binding right. So, scheduling assigns the timestamp. So, we know how many clock cycles, we need to execute the complete behavior and we know in every clock which operation is going to execute right.

So, what is our next task? The next task is to make sure that in the hardware these operations are going to map into FUs and the variables are going to map into registers. So, that is what I have to allocate the sufficient number of registers and function units so that I can map this variable to the registers and operation to the function unit and which satisfies the scheduling requirement. I am not going to change the requirement of what we have done in scheduling right.

(Refer Slide Time: 02:05)



Allocation and Binding

- Objectives: Maximize Resource sharing; hence, minimize resource usage

Operations
Functional Units
Variables
Storage

Subtasks:

1. FU allocation & Binding
2. Register Allocation & Binding

3

So, if you look into this allocation and the binding problem I am going to map the operations to the function units and variables to the storage as I mentioned here what is our objective? Objective obviously in the hardware, I want to use a minimum number of resources right. So, what does it mean? I want to use as less as possible function units to execute all the operations in the behavior.

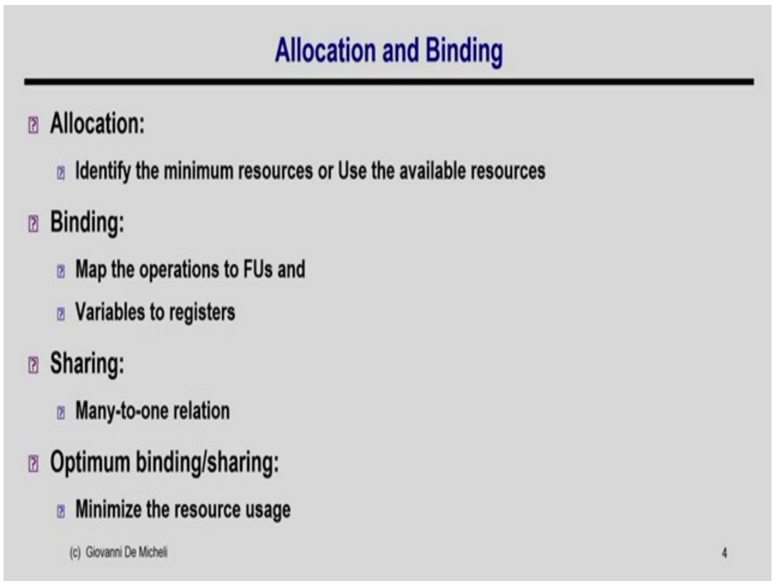
Similarly, I want to use a minimum number of storage which is register or RAM, or ROM to store all the variables and arrays of the program such that the overall use of these registers and functionalities is minimum how we can achieve that? Obviously, you have to share that right. So, unless you share, if you have each function unit allocated for each operation, you will not get any savings right.

If I can do that, that two operations can be executed by a single FU or say two variables can be stored into a single register, then I am having a saving. So, I want to maximize this resource sharing right and hence, minimize the resource usage. So, that is our objective right.

So, I want to maximize the resource sharing and hence, minimize the resource usage this allocation and binding phase has two subtasks if it is clearly understood because the operation has to be mapped to function units and variable has to mapped to the storage. So, it has two completely separated sub-tasks; one is FU allocation and binding and register allocation and binding.

Here by register, I want to mean the memory which consists of both register and as well as memories like RAM and ROM ok. So, we are going to discuss first this FU allocation and binding, and then, in subsequent classes, I am going to discuss about the register allocation and binding ok.

(Refer Slide Time: 03:55)



Allocation and Binding

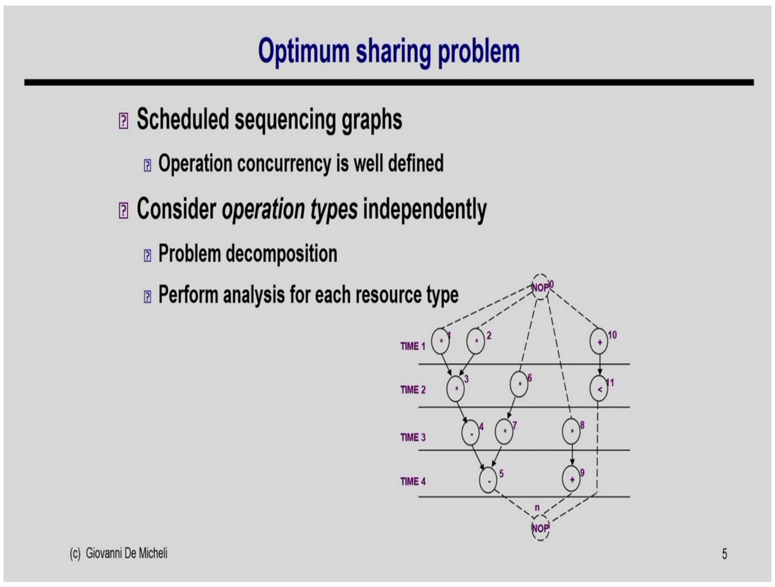
- Allocation:
  - Identify the minimum resources or Use the available resources
- Binding:
  - Map the operations to FUs and
  - Variables to registers
- Sharing:
  - Many-to-one relation
- Optimum binding/sharing:
  - Minimize the resource usage

(c) Giovanni De Micheli 4

So, as I already mentioned that this allocation is something you identify the minimum resource required, and sometimes, if it is a resource constant scheduling, then we already know the resource available for me right. So, you already have seen that for scheduling that if you have only two multipliers available or two-one ALU available right. So, once that is given or you identify what is the minimum requirement right, that is what you allocate.

You allocate a sufficient number of FUs and then, binding has mapped this operation to FUs and map the variable to registers right and our objective is to share. We want to share one FU for multiple operations and similarly, I want to share one register for storing multiple variables ok. So, that is our objective.

(Refer Slide Time: 04:45)



Optimum sharing problem

- Scheduled sequencing graphs
  - Operation concurrency is well defined
- Consider *operation types* independently
  - Problem decomposition
  - Perform analysis for each resource type

NOP 0
TIME 1
TIME 2
TIME 3
TIME 4
1
2
3
4
5
6
7
8
9
10
11
n
NOP

(c) Giovanni De Micheli

5

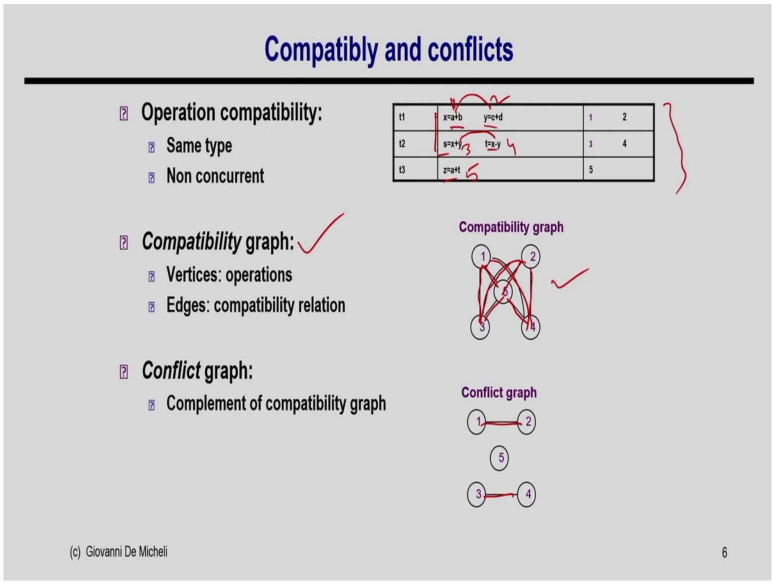
So, now let us only concentrate on the function unit allocation and binding problem. So, if you look into a scheduling graph you have taken the example that we consider for our scheduling discussion. So, this is the scheduled behavior that in time 1, I am going to execute these behaviors, this operation. In time 2, I am going to execute these three operations. In time 3, I am going to execute these three operations and in time 4, I am going to execute 5 and 9 right.

And here, you see there are two multipliers and one addition operation scheduled in time 1 right and similarly, two multipliers and one less than operation are scheduled in timestamp 2 and so on. So, once I am going to identify the total function unit requirement, I can consider this multiplier separately from other operations because once, I try to find out the minimum requirement of the multiplier, I do not have to consider how many adders are needed; how many other type of function units is needed right.

So, the important point to be noted here is that once I am going to consider about this function allocation and binding, I can consider the operations independently ok. So, it means I will consider only multiplier allocation binding. Once I am done, I will take this ALU allocation and binding because we have seen that ALU is a kind of operation I mean a function unit that can do addition, subtraction, greater than, less than operations right.

So, once I am going to do multiplier, I am going to do it independent of these other operations. One I am going to do ALU allocation and binding, I am going to do that independent of the multiplication or other operations right. So, this is something is the decomposing the problem into each type of operator and then, you do this analysis to find out the minimum requirement and then, you bind the operations and then, go for the next type of operation ok.

(Refer Slide Time: 06:31)



Compatibly and conflicts

- Operation compatibility:
  - Same type
  - Non concurrent
- *Compatibility* graph:
  - Vertices: operations
  - Edges: compatibility relation
- *Conflict* graph:
  - Complement of compatibility graph

t1	x=a+b y=c+d	1 2
t2	s=x+y 3 t=x-y 4	3 4
t3	z=a+t 5	5

Compatibility graph
1
2
5
3
4
Conflict graph
1
2
5
3
4

(c) Giovanni De Micheli

6

So, now, we will be going to discuss how we can solve this problem and we are going to show in the next few slides that this particular FU allocation and the binding problem can be mapped to either graph coloring problem or clique partitioning problem right. So, let us understand that. So, let us understand what is compatibility ok.

You consider a simple schedule behavior like this, where I am because I am taking all only one type of operator at a time. Here, all I am considering is only the operations that are of the plus right. So, all operations are having plus operation right and assume that in time step 1, this operation 1 and 2; this is operation 1 and this is operation 2, this is operation 3 and this is 4 and this is 5 right.

So, in operation time 1, operation 1 and 2 is scheduled; in time step 2, operation 3 and 4 is scheduled, and in timestamp 3, operation 5 is scheduled ok. So, what is the compatibility? Let us understand that. So, since this operation 1 and 2 is scheduled in timestamp 1, you have to have two different FU right to execute these two. So, they are not compatible with each other. So, they are conflicting with each other rights.

So, they have to be allocated to different FUs right; whereas, this 1 and 3, since they have to execute different timestamps, so I can execute both the behavior, both the operations using the same type of using the same FU right. It is the same ALU because in time ALU will perform this operation 1 in timestamp 1. In the next clock in time step 2, it can do operation 3. So, that means, that once these two operations are scheduled into a different timestamp and they are of the same type, then they are compatible with each other. Understood?

So, from that, I can construct a graph called compatibility graph ok. So, if you take this behavior, then what will be the compatibility graph? In the graph, you know we have to find out the nodes and the nodes are the operations. Since there are five operations here, I have five nodes ok, and what are the edges here? Edge defines compatibility. If two operations are compatible just as we have discussed, we will have an edge right.

So, when they are compatible? When the operations are of the same type, all are addition operations, addition or subtraction operation, and ALU operations, and they are scheduled in the different timestamps. So, I have so for with 1, I have 3, 4, and 5 scheduled because they have not scheduled in timestamp 1. So, from 1, I have an edge to 3, 4, and 5; but I do not have an edge from 1 to 2 because they are running in parallel right.

Similarly, from 2, I have I should have an edge over 3, 4, and 5 because they are compatible with 2. So, I have an edge over 3, 4, and 5, and for this operation 3, I have already so 1, 2, and 5 is compatible. So, I have an edge from 3 to 5, and 3; 5 to 1, and 5 to 2 are already there right. So, similarly from 4, I have edges to 1, 2, and 5; 1, and 2 are already there, so I have the edge here. So, that is complete the compatibility graph right.

So, this is how I can construct a compatibility graph from a given schedule ok. So, similarly, I can define the concept of a conflict graph.

It is just the complement graph of the compatibility graph because here, I am going to define the conflict. If two operations are scheduled at the same timestamp; that means, they are conflicting with each other rights. So, that means, I cannot schedule them into the same stamp. So, what I am going to do? In the conflict graph, I will have the same, again it consists of a set of nodes and the edges; the nodes represent the operations.

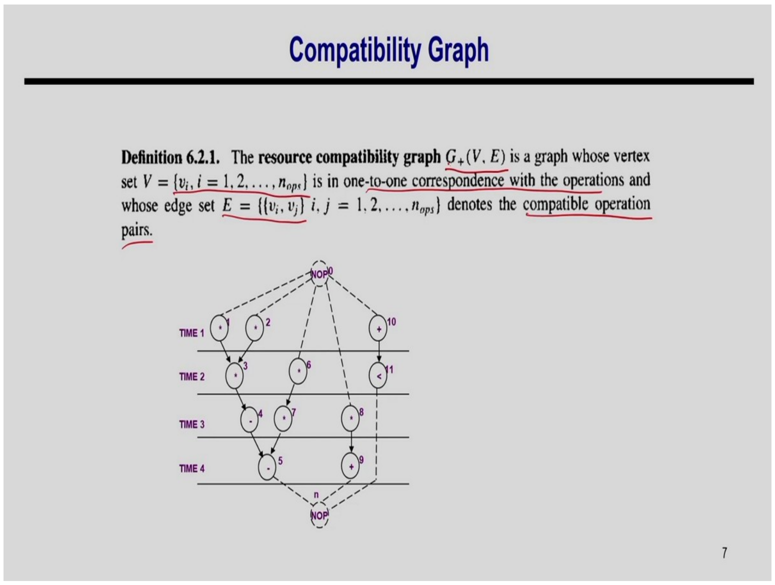
So, there are five operations. So, I have 5 nodes and if two operations are scheduled in the same timestamp and of the same type, then I have an edge between them. So, here, I have an edge between only 1 to 2 and 3 to 4 right. So, I have an edge from 1 to 2 and 3 to 4 and if you just take this graph and you do a compliment, you know what is complement graph is, I will have the same nodes, and whatever the edge is missing here, I will add them right. So, here if you see in this node of 5, only 1 to 2 and 3 to 4 are missing. So, that two is will be the in the conflict graph.

So, I can either can construct this conflict graph directly from the scheduling behavior or if I have given the compatibility graph, I can take the complement of that graph and which will be the conflict graph ok. So, this is how from the given schedule, I can construct the compatibility graph or conflict graph. But what is the use of that? We do not know right.

So, our objective is to identify the minimum number of resources and the binding and which is something we have not discussed yet right.

So, we only just argued that given a schedule, I can construct a compatibility graph and conflict graph ok.

(Refer Slide Time: 11:55)



Compatibility Graph

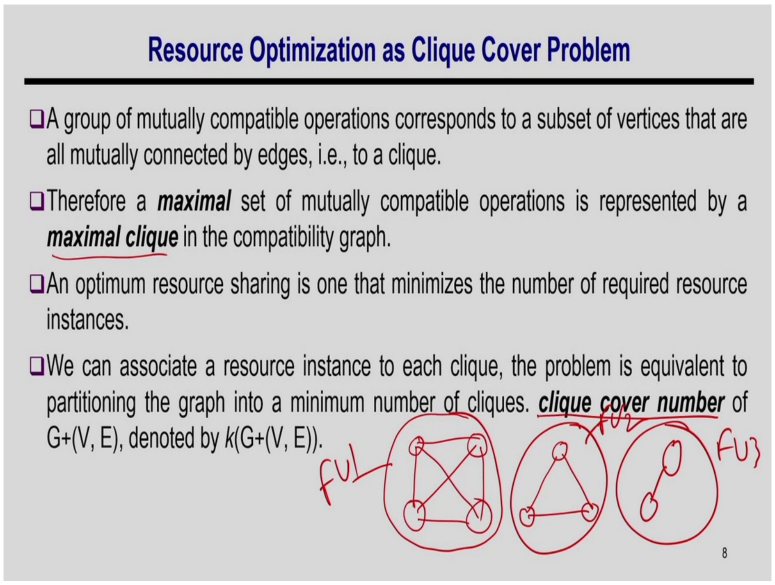
Definition 6.2.1. The **resource compatibility graph** $G_+(V, E)$ is a graph whose vertex set $V = \{v_i, i = 1, 2, \ldots, n_{ops}\}$ is in one-to-one correspondence with the operations and whose edge set $E = \{\{v_i, v_j\}\ i, j = 1, 2, \ldots, n_{ops}\}$ denotes the compatible operation pairs.

NOP
0
TIME 1
TIME 2
TIME 3
TIME 4
1
2
3
4
5
6
7
8
9
10
11
n
NOP

7

So, here I give a formal definition of the compatibility graph that its graph consists of a G plus consisting of V and E, where V is the set of nodes that represents the operations and which has a one-to-one correspondence to the operations and the edges are representing from v_i to v_j denotes the compatibility operation pair. So, if the operations are compatible, I am going to add an edge to the graph which we have already discussed in the previous slide.

(Refer to Slide Time: 12:27)



Resource Optimization as Clique Cover Problem

- A group of mutually compatible operations corresponds to a subset of vertices that are all mutually connected by edges, i.e., to a clique.
- Therefore a ***maximal*** set of mutually compatible operations is represented by a ***maximal clique*** in the compatibility graph.
- An optimum resource sharing is one that minimizes the number of required resource instances.
- We can associate a resource instance to each clique, the problem is equivalent to partitioning the graph into a minimum number of cliques. ***clique cover number*** of G+(V, E), denoted by *k*(G+(V, E)).

8

So, what I am going to argue now is that this resource allocation and binding or the resource optimization problem can be modeled as a clique cover problem in the compatibility graph. So, let us just understand how I am telling that right. So, what I have discussed so far? In the compatible, if two operations, they are indifferent timestamp; that means, they are compatible to each other. What does it mean? So, these two operations can be mapped to the same FU.

So, what about three operations? If there are three operations and they have an edge between all of them; that means, they are all compatible with each other right. So, once they are all compatible with each other; that means, I can assign all these three operations into a single FU. Similarly, for 4, if there are 4 nodes are there, only when they have all edges right, so they are all compatible with each other; then, I can assign all these four operations into a single FU and I have already discussed for 3. So, similarly, I can talk about 4, 5 and 6, and so on right.

So, what I understood here is if these 4 nodes are from a complete graph, so this is a complete graph right; then, that particular subset of the graph is compatible with each other and then, can be mapped to a single FU. So, but in the whole graph, there are say n nodes and this is only 4 nodes.

So, this is a complete subgraph and what does it call? That is called maximal clique right. So, this is a clique. Clique is a complete subgraph in a complete in a bigger graph right because I have a graph of n nodes say 20 nodes and out of that, I found that there is a big 4 nodes which can form a conflict graph. So, this is a complete subgraph or clique.

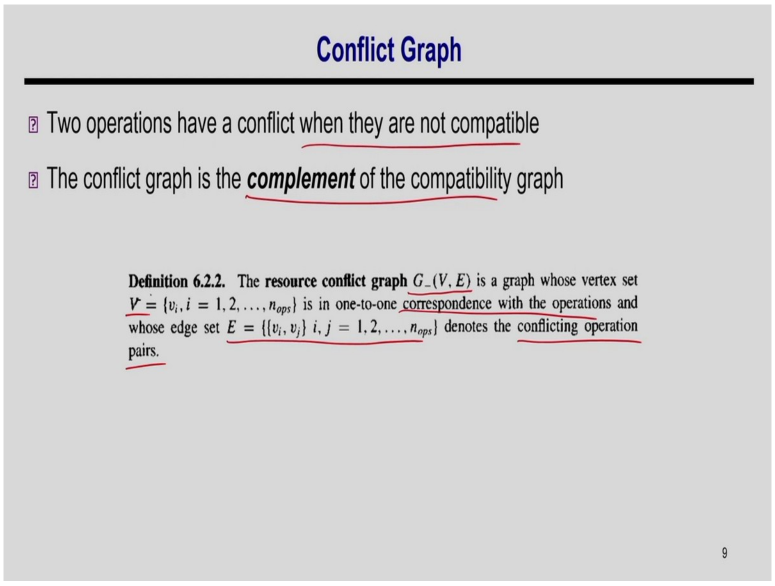
So, as I argued that once I found a complete subgraph or a clique we can be mapped to a single FU right. So, in the compatible graph, if I identify all such subgraphs because each of them is this is a FU 1, this is said FU 2 and say this is an only two nodes I found. So, this is FU 3 right. So, what does it mean? I identify the maximal possible subgraphs complete subgraph in the compatible graph and each of them can be mapped to a single FU. So, what I am doing here? I want to cover all the nodes of the graph using this clique right. So, what does it mean? This is a clique cover problem right.

So, it is a clique cover problem. That means I want to cover all the nodes of this compatibility graph using such maximal clique such that all the nodes are subsets of one of them and there be no overlap right. So, one node if occurs in one clique, it should not occur in other cliques. So, it’s an independent right. So, each subgraph is independent. So, if I identify such subgraphs and then, that each of the subgraphs is 1 FU. So, that means, identifying, so the allocation and the binding problem of FU map to the clique cover problem right. So, what is that?

We will construct a compatibility graph and then, I try to find out iteratively the maximum possible subgraph which is complete, which is a clique and each of them will be mapped to a single FU and all the nodes inside that subgraph will be mapped to that same FU. So, I will identify the number of FU needed because each clique represents 1 FU. So, the number of cliques needed to cover all the nodes defines the number of resources needed and how the binding will happen; inside one subgraph whatever node is present there, they will be mapped to the same FU.

So, what we understand from this slide is that this allocation and the binding problem is nothing but constructing a comparative graph and identifying the cliques which will cover all the nodes, which is a clique cover problem. So, that means, I have reduced that allocation binding problem into clique cover problem ok.

(Refer Slide Time: 16:55)



Conflict Graph

- Two operations have a conflict when they are not compatible
- The conflict graph is the ***complement*** of the compatibility graph

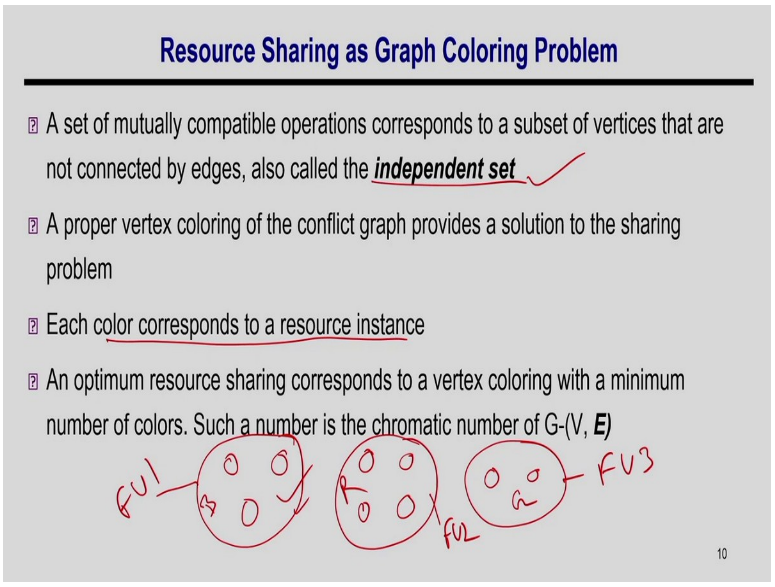
Definition 6.2.2. The **resource conflict graph** $G_{-}(V, E)$ is a graph whose vertex set $V = \{v_i, i = 1, 2, \ldots, n_{ops}\}$ is in one-to-one correspondence with the operations and whose edge set $E = \{\{v_i, v_j\}\ i, j = 1, 2, \ldots, n_{ops}\}$ denotes the conflicting operation pairs.

9

So, similarly, let us understand the conflict graph. So, this is I know this graph is the complement of this compatibility graph; that means if there is an edge between two nodes. That means, they are not compatible with each other; they cannot be mapped to a single FU right. And this is and so, they are not compatible when they are conflict right. So, formal definitions here.

So, it's a G_(V, E) graph, where each node again represents the operations and edges denotes the conflicting opt pair right. So, if the two operations are conflicting with each other, I will have an edge there ok. This is understood.

(Refer to Slide Time: 17:29)



Resource Sharing as Graph Coloring Problem

- A set of mutually compatible operations corresponds to a subset of vertices that are not connected by edges, also called the ***independent set***
- A proper vertex coloring of the conflict graph provides a solution to the sharing problem
- Each color corresponds to a resource instance
- An optimum resource sharing corresponds to a vertex coloring with a minimum number of colors. Such a number is the chromatic number of G-(V, ***E)***

Now, I am arguing that this resource allocation problem, this resource sharing problem is nothing but a graph coloring problem for this conflict graph. So, let us understand why. So, in this particular conflict graph, if there is an edge; that means, they cannot be mapped to the same FU. But what about the other thing? If there are 2 nodes, that have no there is no edge between them, I can assign them to the same FU because there is no edge between them.

So, what about 3 nodes, that have no edges between them? So, this is one part. Similarly, what about say 4 nodes, where there is no edge between them, or say 2 nodes, where there is no edge between them. So, what is this call? This is called an independent set right because these nodes are independent of each other, there is no conflict among them and this particular set of nodes can be mapped to the same FU.

So, similar to the previous problem, if earlier I try to find a complete graph, here I am trying to find out the independent set; that means, the nodes that do not have any edge. Because this is a conflict graph and if there is no edge; that means, they are compatible with each other. So, I can assign them to different FU. So, this can be FU 2, this can be FU 3 right.

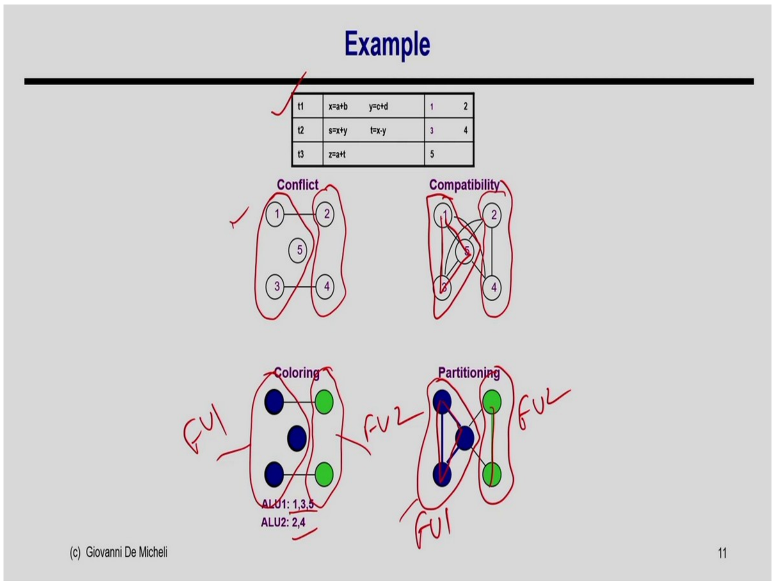
So, you know what is coloring, graph coloring problem right. So, the graph coloring problem is what? I have given a graph, I want to assign the color to the nodes such that if there are two nodes which have adjacent to each other, have a different color. Two nodes, that are adjacent to each other cannot have the same color, they should be assigned to the same different color and my objective is to use the minimum number of colors to color all the nodes of the graph right.

So, what will happen to these independent nodes? Since they do not have an edge between them, I can put the same color for them. So, I have to assign blue to them. Since this set, they do not have an edge between them, I can put another color same color on them. So, I put say red to this and since these nodes, they have do not have any edge, I can assign the same color which will not violate my coloring problem. So, I say I assign a green here ok. So, what does it mean? And now, my objective is what? My objective is to use a minimum number of colors.

So, what is my objective? I try to find out the maximal independent set right. I try to find the bigger independent set so that I can put the same color to all the nodes right. So, this problem is now reduced to identifying the minimum color needed. So, identifying such an independent set and you put a color to them and each color now represents a resource right; 1 unit of FU and the nodes that have the same color, can be mapped to the same FU.

So, now, you understand that once I construct the conflict graph, the conflict problem is now reduced to coloring the nodes of the graph such that no two adjacent nodes have the same color and then, the number of colors needed represents the number of FU needed for me and each color represents each FU and the nodes that have the same color can be mapped to the same FU. So, what do I understand from here? I understood that this resource sharing or allocation and the binding problem can be mapped as a graph coloring problem ok. Interesting right?

(Refer Slide Time: 21:07)



Example

t1	x=a+b y=c+d	1	2
t2	s=x+y t=x-y	3	4
t3	z=a+t	5	

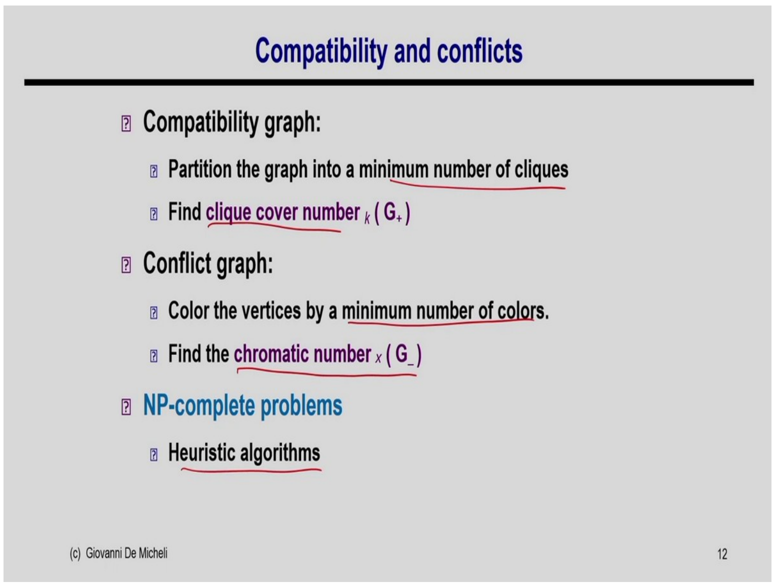
So, let's take an example. So, say I took the same example again and this is my conflict graph that we have already drawn earlier and this is the coloring right. So, what I am doing here is say these three nodes; 1, 3, and 5 forms an independent set; maximally independent set because there is no edge between 1, 3, and 5 and I can put them the same color say blue and this 2 and 4 is another independent set, where there is no between them and I put the same color to them.

So, what does it mean? This is one FU, this is one FU. So, this is ALU 1, where operations 1, 3, and 5 going to execute and this is FU 2 in which operations 2 and 4 are going to execute. So, this is a graph coloring problem. Similarly, if you consider the compatibility graph, what I have told you? I have to find out the clique. So, you see here this is a clique right.

So, this is a clique of three nodes right 1, 3, and 5. This is a complete subgraph and this is a maximal possible clique, I cannot have a clique or a complete subgraph using four nodes in this graph. So, I can actually put the same color or I can actually put this clique into FU 1 right, and similarly, this is another clique of two nodes and I can assign them into a different FU right. So, this is my FU 2. So, both give the same solution.

So, either you take the compatible graph and identify the cliques or you take the conflict graph and you try to color them and you have the same solution. So, this is clique 1, this is clique 2 and one clique represents 1 FU, and allocation and binding are done ok.

(Refer Slide Time: 22:51)



Compatibility and conflicts

- Compatibility graph:
  - Partition the graph into a minimum number of cliques
  - Find clique cover number $_k$ (G_+)
- Conflict graph:
  - Color the vertices by a minimum number of colors.
  - Find the chromatic number $_x$ (G_-)
- NP-complete problems
  - Heuristic algorithms

(c) Giovanni De Micheli 12

So, what we understood summarized in the discussion so far is that lets the problem of either doing this allocation and binding is nothing but identifying the compatibility graph and doing the clique cover problem or you would construct the conflict graph and you identify the graph coloring. What is the chromatic number? It is the minimum number of colors needed right. So, identifying the minimum color needed to color the graph is nothing but this chromatic number and this minimum number of such color is the FU and similarly, the minimum number of cliques is the number of FU needed.

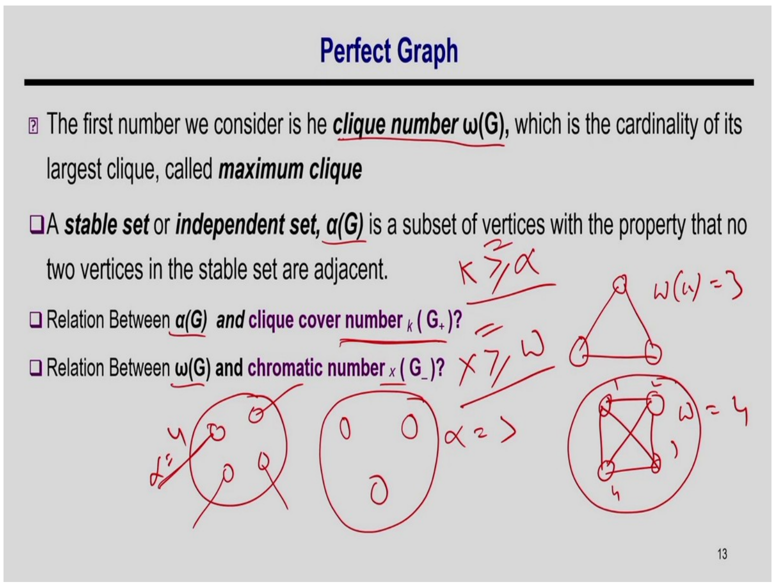
So, this is how I map this allocation and binding problem to either the clique cover problem of finding the chromatic number of a graph ok. But the problem of these two is well-known to be NP-complete problem ok; both the problem clique cover problem or chromatic number finding of graph coloring problem is known to be NP-complete problem.

What does it mean? For a generic graph, now, it means I do not have a polynomial-time algorithm to solve this problem. So, that means, I do not have an efficient algorithm to solve this problem. If I have to go for an optimal solution, I have to run for an exponential algorithm either 2 to the power n and alright. So, it should be an exponential algorithm to solve this problem.

So, usually, for such an NP-complete problem, what do we do? Usually, apply heuristic algorithms right. So, a heuristic algorithm is something that is working in logic that most of the time, gives an optimal solution, and most of the time, it gives a good solution; it may not be optimal, but it is a good enough right, but it's very fast right. So, we will learn in these subsequent classes, how to apply to solve either this graph coloring problem or clique cover problem heuristically. But in today's discussion, we try to figure out a subset of the graph for which this problem is polynomial-time solvable.

So, what I conclude here is that in general, this problem is an NP-complete problem; but if the graph satisfies a certain property, then this particular problem is polynomial-time solvable. So, we will discuss in the rest of the class, what is the subset for which this problem is polynomial-time solvable. In the subsequent classes, we will discuss the generic solution ok.

(Refer Slide Time: 25:17)



Perfect Graph

- The first number we consider is he ***clique number*** **ω(G)**, which is the cardinality of its largest clique, called ***maximum clique***
- A ***stable set*** or ***independent set***, ***α(G)*** is a subset of vertices with the property that no two vertices in the stable set are adjacent.
- Relation Between ***α(G)*** ***and*** **clique cover number** $_k$ **(G$_+$)?**
- Relation Between **ω(G) and chromatic number** $_x$ **(G_)?**

13

$k \geq \alpha$
$\chi \geq \omega$
$\omega(G)=3$
$\alpha=4$
$\alpha=3$
$\omega=4$

So, let us understand some basic terminology in the graph theory and we will try to define what is perfect graph ok. So, we first try to define a terminology called clique number. So, I told you that a clique is nothing but a complete subgraph. So, this is a complete subgraph of 3 nodes and its clique number defines the number of nodes present on this right.

So, the clique number ω(G) of this graph is 3. So, this is a graph with a complete graph with four nodes right. So, the ω(G) is 4 on this graph. So, the clique number ω(G) says the number of nodes present in that particular maximal clique. Similarly, I told you the independent set, the set of nodes that have no edge between them that is called an independent set and I define a term called α(G), the number of nodes present here ok.

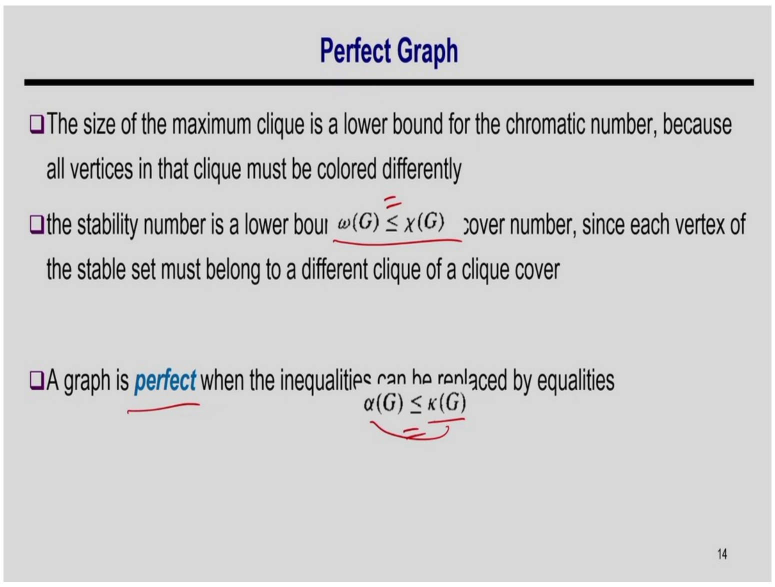
So, again for this independent set α(G) is 3; for this set, α(G) is 4 ok. Now, let us try to understand what is the relation between this α(G) and the clique cover number which is $_k(G_+)$ of the graph, and the what is the relation between $\omega(G)$ and the chromatic number $_x(G_-)$. You try to understand one point here.

So, if I told you that if there are four now nodes in the independent set in a graph, what does it mean? They cannot form any clique. So, they must go into different cliques because you cannot have a clique considering them. So, you need at least have a maximal independent set that has say 4; that means, you need at least four numbers of the clique to cover them because you cannot have a clique that will cover two of the nodes from here because there is no connection between them.

So, they will all go to a different clique. So, what does it mean? So, if I have found a maximum independent set, then the number of cliques must be at least this number right. So, you have you can have more, but you cannot have less than this right. So, this $\alpha(G)$; so that means, this $_k(G_+) >= \alpha(G)$ right and similarly, for chromatic numbers $_x(G_-)$, so if I found a maximal clique of size 4, what does it mean? They are all connected and I must assign different colors for all these nodes because they have all edges between this right.

So, there is my color 1, 2, 3, and 4. So, if I found a maximal clique of size 4; that means, I need at least 4 colors. I might need more than 4 because of the other part of the graph, but at least 4. So, what I can say is that $_x(G_-)>=\omega(G)$ ok.

(Refer Slide Time: 28:09)



Perfect Graph

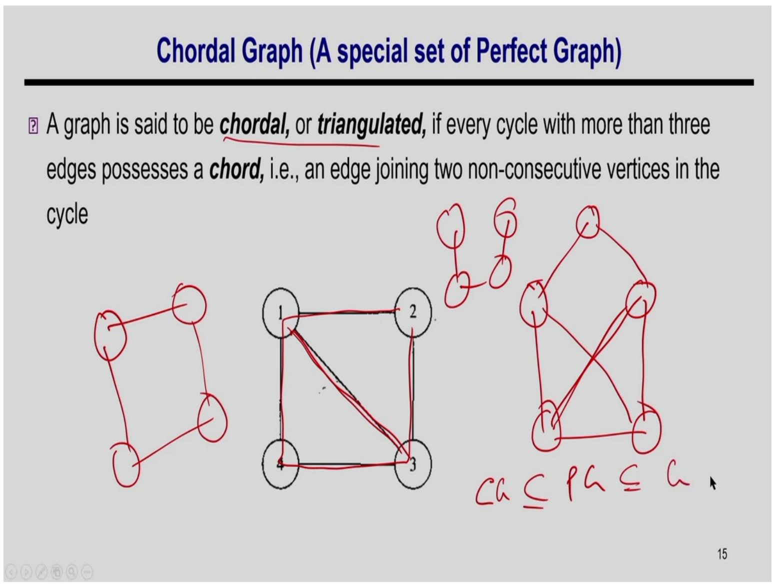
- The size of the maximum clique is a lower bound for the chromatic number, because all vertices in that clique must be colored differently
- the stability number is a lower bou $\omega(G) \le \chi(G)$ over number, since each vertex of the stable set must belong to a different clique of a clique cover
- A graph is ***perfect*** when the inequalities can be replaced by equalities $\alpha(G) \le \kappa(G)$

14

So, that is what I told her that this $\omega(G)<=_{x}(G_{-})$ and $\alpha(G)<=_{k}(G_{+})$. But in a perfect graph, this relation becomes equal ok. So, in the perfect graph, the number of colors needed to color the graph should be equal to this clique number $\omega(G)==_{x}(G_{-})$. So, the maximum number of nodes here.

So, this will be equal and similarly, so the maximum size of the independent set this $\alpha(G)$ should be equal to this number ok so,$\alpha(G)==_{k}(G_{+})$. So, then, I will say this is a perfect graph ok. So, this is the definition of a perfect graph.

(Refer Slide Time: 28:55)


Chordal Graph (A special set of Perfect Graph)

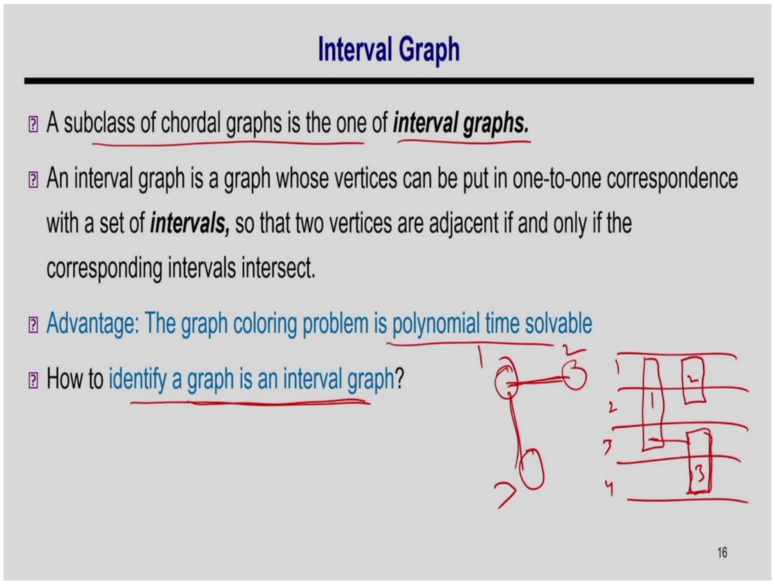
- A graph is said to be ***chordal***, or ***triangulated***, if every cycle with more than three edges possesses a ***chord***, i.e., an edge joining two non-consecutive vertices in the cycle

Now, I will come to another kind of graph which is called a chordal graph and this is a subset of a perfect graph. So, the chordal graph is something is a graph, where if you consider a cycle of size 4; that means, you have at least four edges in them, then you should have a chord ok. So, this is a chordal graph because it has only one cycle and the cycle have a chord ok.

So, this graph is chord is not a chordal graph. So, this is; so, this is also a chordal graph because you have whenever you have a cycle of size 5, you have a chord there, but this is not a chordal graph because this has a cycle 4, but it has no chord here right. So, then, so, I will say this is a chordal graph if, for any such cycle of size 4, we have a chord right. So, this is also a chordal graph because there is no cycle right.

So, this is also a chordal graph because there is no cycle of size 4, then still and although there is no chord because there is no cycle of size 4, this is also a chordal graph ok. So, what do I understand? So, the chordal graph is something subset of a perfect graph right. So, the chordal graph is a subset of the perfect graph CG ⊆ PG and the perfect graph is a subset of the generic graph PG ⊆ G right.

(Refer to Slide Time: 30:15)



Interval Graph

- A subclass of chordal graphs is the one of ***interval graphs.***
- An interval graph is a graph whose vertices can be put in one-to-one correspondence with a set of ***intervals***, so that two vertices are adjacent if and only if the corresponding intervals intersect.
- Advantage: The graph coloring problem is polynomial time solvable
- How to identify a graph is an interval graph?

16

And then, I will define another class of graph which is called interval graph. Again, this is a subclass of the chordal graph and this particular graph is very interesting in our context because what does this graph says? So, I have a graph and I can map this node to some interval say suppose I have this graph, I have this say graph and then, I can define some intervals.

So, these are the intervals. So, this is my node 1, 2, and 3. So, one is saying interval is this and so, 2 is overlapping with say this is my say 2, this is my 1, this is 2 and this is my 3 right. So, I can map these nodes to some interval such that if these two-node has an edge their interval will overlap. So, I have an edge between 1 and 2, so the 1 and 2 intervals is overlapping right.

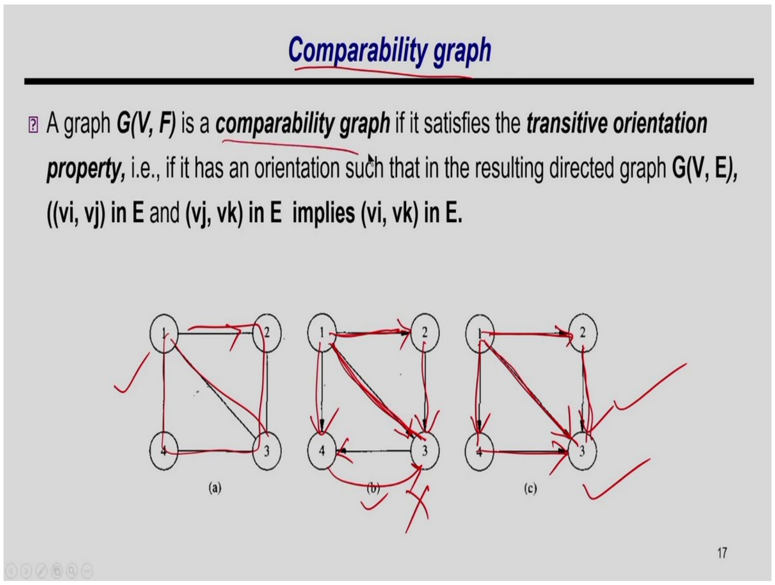
So, this is what you can think about this is the time right and 1 and 3 also have an edge. So, there is they are overlapping 1 and 3 is overlapping right. But there is no edge between 2 and 3, so these 2 and 3 must not overlap. If they overlap, then they do not hold. So, given a chordal graph, if I can map this node to such intervals such that whenever there is an edge, the corresponding interval will overlap. So, then I will say this particular chordal graph is an interval graph right. So, I found this interval graph which is a subset of a chordal graph ok.

So, why this particular interval graph is so important? Because for this interval graph this graph coloring problem is polynomial-time solvable. So, if I know, that if I found that a particular subcase, this graph that conflict graph that we have obtained is an interval graph, then I will have an efficient algorithm or I can solve the when I can color the node using polynomial-time ok. So, this is very interesting.

So, why this problem is important? Because in this case, this particular graph is polynomial-time solvable, the graph coloring problem is polynomial-time solvable right. So, the question here is that how do we identify the graph as an interval of a node? Because this is not so easy and sometimes for example, if you have in some cases probably, it is not possible to map them into these intervals right.

So, the question here is how do we identify whether a particular graph is an interval graph or not? Because if I identify this graph as an interval graph, then can I solve this problem easily or efficiently ok.

(Refer to Slide Time: 33:11)



Comparability graph

- A graph ***G(V, F)*** is a ***comparability graph*** if it satisfies the ***transitive orientation property***, i.e., if it has an orientation such that in the resulting directed graph **G(V, E)**, **((vi, vj) in E** and **(vj, vk) in E implies (vi, vk) in E.**

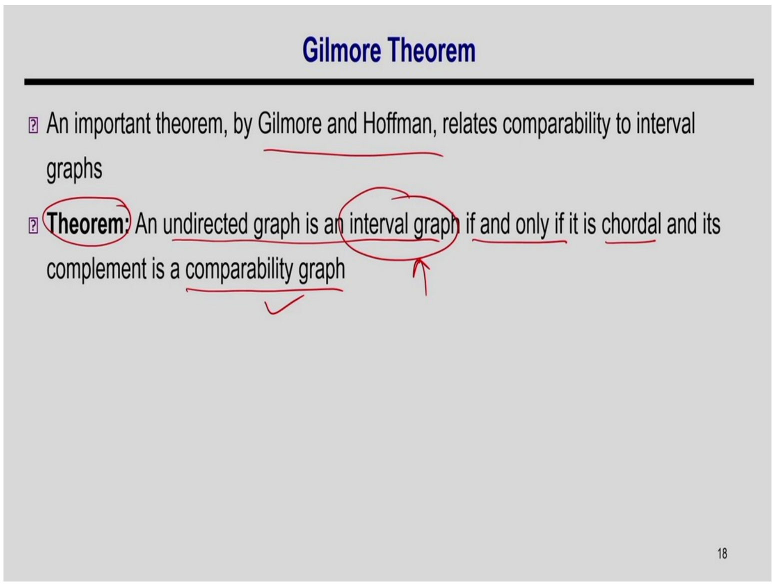
So, for that let us try to understand another class of graph which is called comparability graph. In this graph, so this is what you have a graph and so far you understand that the conflict graph and comparability graph are undirected. Now, I want to orient the edges; that means, I want to add direction to the edges. So, I want to make this undirected graph into a directed graph in such a way that it holds the transitivity property. What does it mean? So, I have this graph we already know is chordal.

So, I mean we have seen that because there is a cycle of 4 and we have a chord here. Now, I want to give direction to these edges right. So, I have given say direction like this. For this, I have given a direction like this and now, see I have an edge 1 -> 4 and 4 -> 3. So, I must have an edge from 1 -> 3 and which is there. So, I have an edge from 1 -> 2, I have an edge from 2 -> 3 and I have an edge from 1 -> 3. So, this is holding the transitivity property. So, that means, this is a comparability graph.

But if you give the direction like this, so it is not holding the transitivity because you see here I have an edge from; so I have an edge from this and this and so, I have an edge here. So, I have an edge from this yeah. So, I have an edge from this and then, I should have an I mean this edge not holding the property right because then I have an edge here, I took, so I must have an edge here which is not there.

So, this is not the correct orientation, but this is a correct orientation. So, given a chordal graph, I can try to give the direction and if this particular direction holds the transitive property, then I will say this is a comparability graph ok.

(Refer Slide Time: 34:59)



Gilmore Theorem

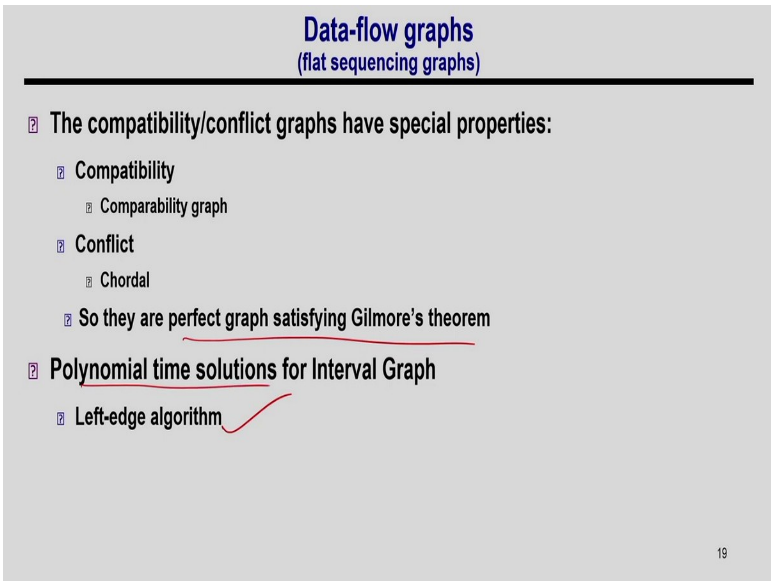
- An important theorem, by Gilmore and Hoffman, relates comparability to interval graphs
- Theorem: An undirected graph is an interval graph if and only if it is chordal and its complement is a comparability graph

18

And why does it help? This particular theorem tells us why this helps identify the interval graph ok. So, I will say an undirected graph is an interval graph if and only if it is chordal; that means, it is a chordal graph and its complement is a comparability graph. So, this theorem says that if I identify a graph which is a chordal graph and I take the complement of the graph and I identify that is a comparable graph; that means, in that particular graph, I can give this.

When I can make this is a directed graph that holds the transitive property, then my graph is an interval graph ok. So, this theorem gives us the solution that if try to identify a chordal graph and then, it takes the complement and checks whether that graph is a comparability graph or not. If it is that, then I can ensure that this particular graph is an interval graph and if I try to solve this problem for this, this graph coloring problem for this, I can solve it efficiently; that means, in polynomial time.

(Refer Slide Time: 35:59)



Data-flow graphs
(flat sequencing graphs)

- The compatibility/conflict graphs have special properties:
  - Compatibility
    - Comparability graph
  - Conflict
    - Chordal
  - So they are perfect graph satisfying Gilmore's theorem
- Polynomial time solutions for Interval Graph
  - Left-edge algorithm

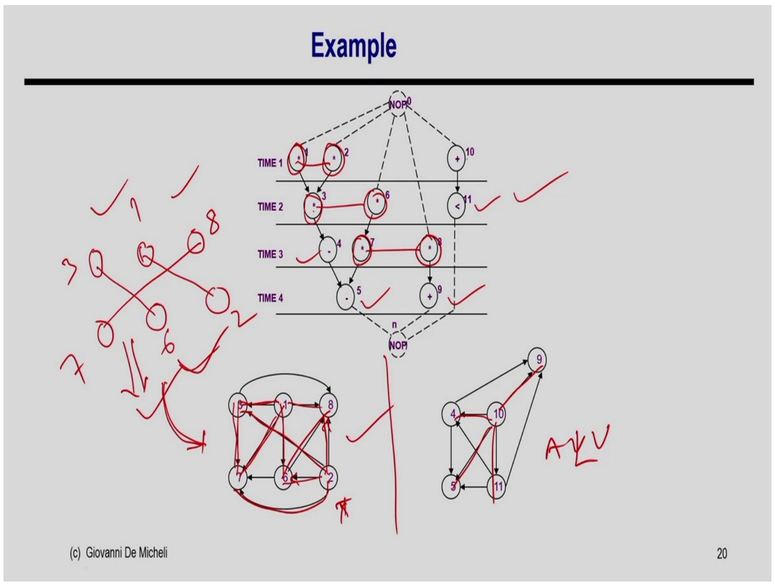
19

So, now, let us go back to this data flow graph. So, if you remember for the scheduling, we have seen that we have taken one basic block at a time and then, we try to schedule the behavior. And but in general, now if you try to solve this allocation and binding problem for the particular basic block, we will show that the particular graph is nothing but an interval graph ok.

But in general, whenever you have a function call and if-else and loops and all, that problem is not polynomial-time ok, and the conflict graph that we are going to construct will not satisfy the interval graph property. But only for a basic block or a single block, the conflict graph will be is interval graph ok. So, that is where this particular property holds.

So, if you try to solve this allocation and binding problem for a specific block or a basic block, I can solve it efficiently. So, let us see how. So, I can apply this left edge algorithm which is a polynomial-time algorithm for this, and solve this problem ok.

(Refer to Slide Time: 37:15)

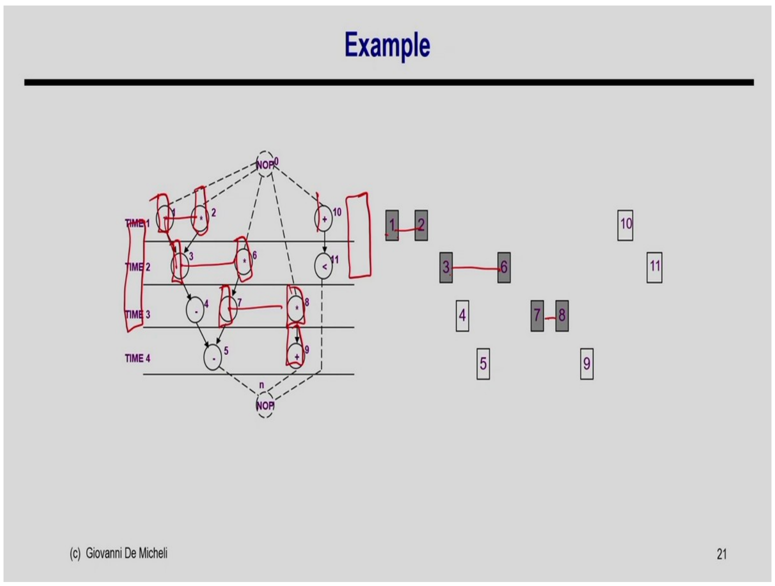


So, let us take an example. So, I have this schedule which we have taken in earlier classes as well. So, now, you see here, I am going to construct this is for multiplier and this is for ALU. So, here is what are the multiplication operation? 1, 2, 3, 6, 7, 8 right. So, I have these nodes here, and what is the ALU operation? 10, 11, 4, 5, and 9 right. So, these are the operations for the ALU. So, these two are different problems. So, let us consider this one first and I can actually, so this is my compatibility graph right.

So, you see here. So, 1 will be compatible with 3, 6. So, these are my multiplication operations right. So, this 1 is compatible with all operations which are scheduled in the different timestamps. So, 3, 6, 7, 8. So, 1 has edges 2, 3, 6, 7, and 8.

Similarly, 2 have an edge over 2, 3, 6, 7, and 8. You can check it here and this way, I can construct the compatibility graph for this right and similarly, for the ALU. This 10 is compatible with 11, 9, 5, and 4 right. So, 10 is compatible with 9, 4, 11, 5, and so on.

(Refer to Slide Time: 38:29)



Example

(c) Giovanni De Micheli

21

So, I will construct this and now, you see the conflict graph right. So, you can take the complement of this graph. So, I will construct the conflict graph here. So, it has 1, 6, 8, 2, 3, and 7. So, what is the conflict graph here? 1 and 2 are conflicting, so 1 with 2; 3 and 6 are conflicting, and 7 and 8 are conflicting.

So, this is my conflict graph ok. So, this graph is I am telling that this is interval graph and because its complement graph is this which is compatibility graph and this is comparable graph because I have given the orientation of the edges, you can no node say here. So, you consider any such transit. So, I have a 2 to 6, I have 6 to 8. So, I have an edge from 2 to 8.

So, I have an edge from 1 to 3, I have an edge from 3 to 7; so, I have an edge from 1 to 7. So, this you can cross-check. So, for example, I have an edge from 2 to 3, I have an edge from 3 to 7, and I have an edge from 2 to 7 as well. So, the transitive poverty hold in this particular graph ok. So, I will say this is a chordal graph because there are no 4 cycles and there is no problem with the chord.

So, this is a chordal graph and the implement, where this complement graph is a comparable graph and hence, this is an interval graph and we will argue that I can map this node into some interval such that whenever there is an edge between them, their interval will overlap and they when there is no edge, their interval will be non-overlapping right.

And how I can do that? That actually can be done easily. So, this is my schedule and what I am going to do is wherever the operation schedule is, I will construct the interval for this. So, for the interval of these nodes is this right. So, I am only considering the multiplier now. So, these are the intervals. So, if one operation is a multi-cycle, I have an interval of more than 1 cycle right. So, some operation is 3 cycles, and the interval will be 3 cycles; but the schedule that I have considered here is only a single cycle. So, this is how I have constructed the interval.

And now, see this 1 and 2 are overlapping, so their interval is overlapping. 1 and 3 is not overlapping, they are in different time step, so their interval does not overlap. 3 and 6 are overlapping, so their interval is overlapping. 7 and 8 are overlapping. So, these are the only three overlaps and there are no other overlaps right. So, these shaded nodes are for multiplier and the others are for ALU operation.

So, let us keep that because these 4, 5, 9, and 10 are for operation ALU and I will consider them separately right. So, this is my interval graph right. So, these are my intervals and which satisfy the edges requirement of the interval graph right. So, this is how I argued here that once I considered a graph only for this a single basic block, my conflict graph is an interval graph, and the corresponding interval I can construct this way.

Because I can construct the interval easily from the scheduled behavior and its complement graph is nothing but a comparable graph that I have already shown. So, I can efficiently solve this graph coloring problem for this conflict graph in polynomial time and for that, there is a very well-known algorithm called the left-edge algorithm and I am going to cover that algorithm in the next class ok.

Thank you.