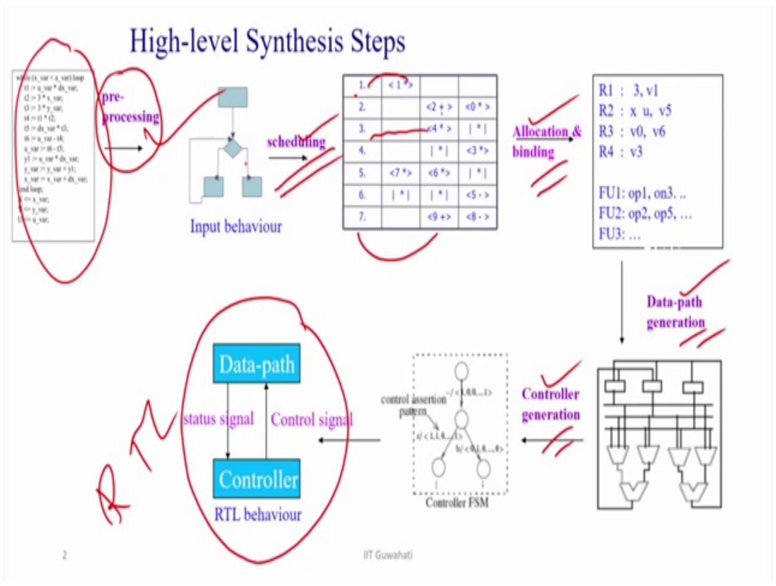
Optimization Techniques for Digital VLSI Design

Dr. Chandan Karfa
Dr. Santosh Biswas
Department of Computer Science & Engineering
Indian Institute of Technology, Guwahati

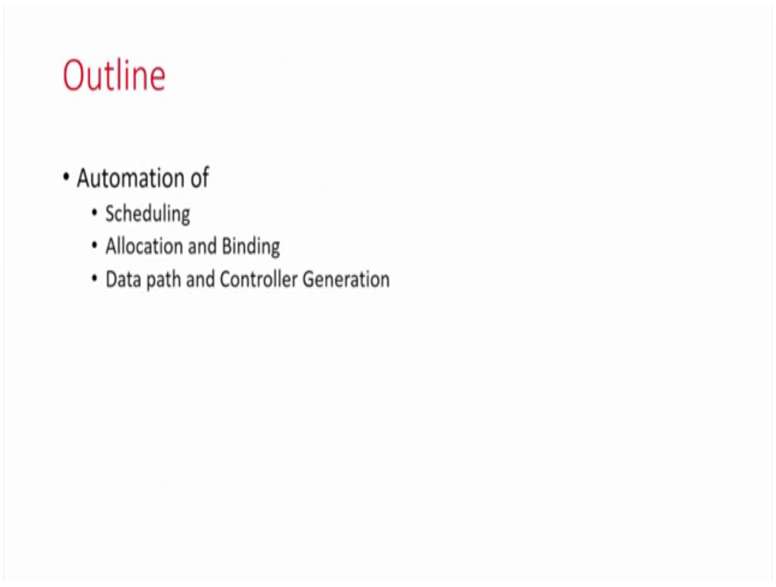
Lecture – 03
Automation of High-level Synthesis Steps

Welcome everyone. So, we are discussing on high level synthesis and in the last class we have discussed about this the steps of the high level synthesis using ah an example we just understand what will be the sub steps of high level synthesis.

(Refer Slide Time: 00:36)



(Refer Slide Time: 00:45)



Outline

- Automation of
  - Scheduling
  - Allocation and Binding
  - Data path and Controller Generation

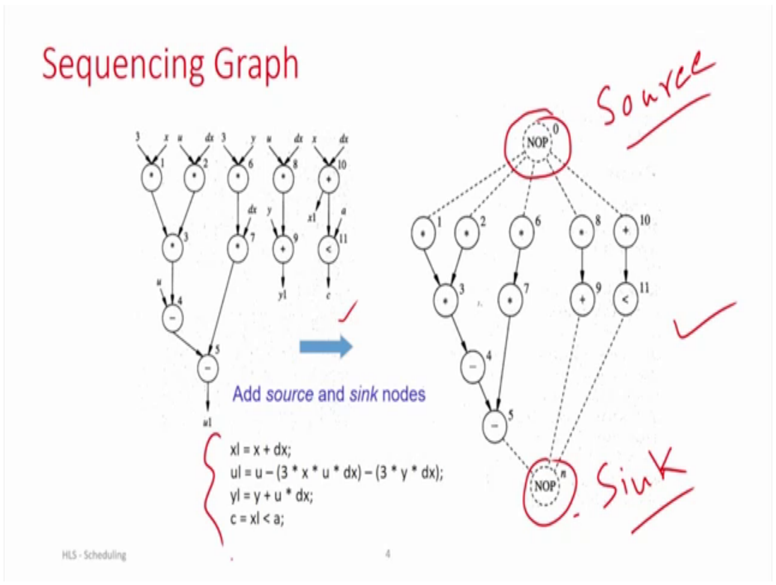
And today's outline is something how can automate those sub steps like specifically scheduling, allocation binding, and data path and controller generation right.

So, if you just look back into the high level synthesis steps we have seen in the last class that starting from initial behaviour in a written in C or something it go through pre processing scheduling, allocation binding, data path controller generation, data path generation and controller generation and finally, we will get this RTL right. So, we get this RTL out of that C code. And so what we have discussed in the last class is not how to automate these steps these steps we just talked about what will the logical next step right so that we have just try to understand through an example today we are going to discuss about those steps in a in the automation of those steps ok.

So, pre processing is also part of that and how do we as I discussed in the last class we apply we extract and data and control flow graph out of this behaviour and also do the data dependence analysis. Those things are very conventional compiler transformations we can assume keep those things is already known to the audience, but we also discuss about that this in pre processing steps we apply a lot of compiler optimization techniques that improves the since synthesis results that will be a separate class ok we will discuss in a separate class about that.

So, today we are mostly we will discuss about the automation of scheduling, allocation binding, and data path and controller generation ok.

(Refer Slide Time: 02:11)



Sequencing Graph

Add *source* and *sink* nodes

```
xl = x + dx;
ul = u – (3 * x * u * dx) – (3 * y * dx);
yl = y + u * dx;
c = xl < a;
```

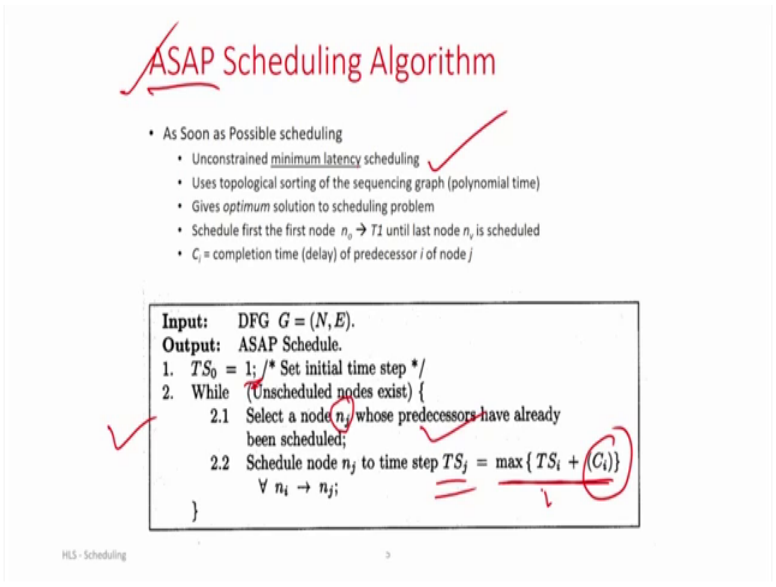
HLS - Scheduling

4

So, we move on so people moving on we know about this grub this is the data dependence graph how do we have the dependence you have discussed in the last class how the dependence you can extract. And just to convert this into in the sequencing graph what we just do we just add source node or and a sink node and we have ah connect all the input variable to this node and all the output variable to this node sink node that this is source node and this is sink this is just to make the scheduling process a little bit smooth, this is nothing much this is just a operation no operation kind of thing you know means no operations.

So, we are going to talk about this scheduling in this sequence graph which it is nothing, but the daily reminder scrub right yeah. And we know to talk about that take the same example and to discuss all the algorithm that we are going to understand learn today ok.

(Refer Slide Time: 03:03)



ASAP Scheduling Algorithm

- As Soon as Possible scheduling
  - Unconstrained minimum latency scheduling
  - Uses topological sorting of the sequencing graph (polynomial time)
  - Gives *optimum* solution to scheduling problem
  - Schedule first the first node $n_0 \rightarrow T1$ until last node n_v is scheduled
  - C_i = completion time (delay) of predecessor *i* of node *j*

```
Input:   DFG $G = (N, E)$.
Output:  ASAP Schedule.
1. $TS_0 = 1$; /* Set initial time step */
2. While (Unscheduled nodes exist) {
       2.1 Select a node $n_j$ whose predecessors have already
           been scheduled;
       2.2 Schedule node $n_j$ to time step $TS_j = \max\{TS_i + C_i\}$
               $\forall n_i \rightarrow n_j$;
   }
```

HLS - Scheduling

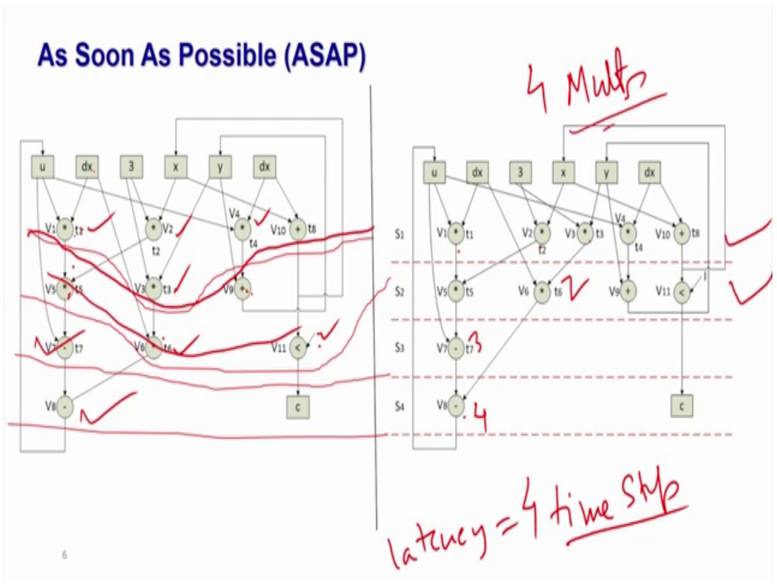
So, the first and ah the easiest of scheduling algorithm is ASAP. What is that? Is as soon as possible ok so, but as the name suggests is something we try to schedule the operation as soon as possible right.

If you just remember in scheduling what we do is just try to assign time step to the operations read the initial the operations are un timed we assign time step to each operations right. So for example, this operation is scheduled in time step 1, these operations scheduled in time step 7 these operation, scheduled in time step 3 and so on right.

So, you have an assigned time step to the operations without violating the data dependence that is our target. So, to do that we first the easiest algorithm is ASAP which is nothing, but as soon as possible as I mentioned whenever some operation is available you schedule it.

What does it mean available? Available in the sense either this operations the operation on which it depends is already executed or it depends on some variable which is available right so so that is what is called as soon as possible.

(Refer Slide Time: 04:05)



So, let us look at the same example that the differential equation solver example that we have taken the discussed in the last lecture that initially you can see this operation depends on the variable so so this is available. So, this is basically available, this is also depends on variable this is also available, this also depends on variable. So, this is also available this is also depends on the input variable so this is also available, this is also available. So, I am going to schedule all these 5 operation in the time stable form so that you what I have done right. So, these operations are scheduled in time step 1 right. So, this is time step 1.

Similarly, once this is done this operation is now available because the dependent operation is already executed this t 1 is now ready, this t 2 is also ready. Similarly this is also available because the procedures operations are done similar this. So, in time step 2 I am going to schedule all this 3 right so this is what I have done. All this 4 in the sense because this 1, 2, 3 and 4 because this is also done, because this is also done.

In the next step this can be scheduled because this produces or a person is now executed and then 4 step I am going to execute this. So, effectively I am able to execute all this operation 4 times step right, 4 time step so this is what is called as soon as possible. So, the ah this overview of the algorithm what is just whenever there is a so I start from the time step 1, whenever there is un scheduled node it will node exist. So, I am just going to select those predecessors already selected.

So, I am going to select one of the unscheduled node whose all the predecessors are already scheduled. And then I am just going to assign a time step to this based on the maximum of the all the predecessors right. So, for example, if you just look into this hat I am going to schedule this operation.

So this predecessor stands scheduled in third time step this is in time to time steps 2. So, the time step I am going to assign is 4 maximum of these 2, so this is what I am going doing here for all this node you just assign this max of this. So, what is that C i is the number of time it takes to execute that particular operation i.

So, I we will discuss on more so, so far assume that all operation will be single cycle operation; that means, operation can be scheduled in 1 cycle only, but there may be some operation which can be scheduled in will need more than 1 clock step we will discuss later.

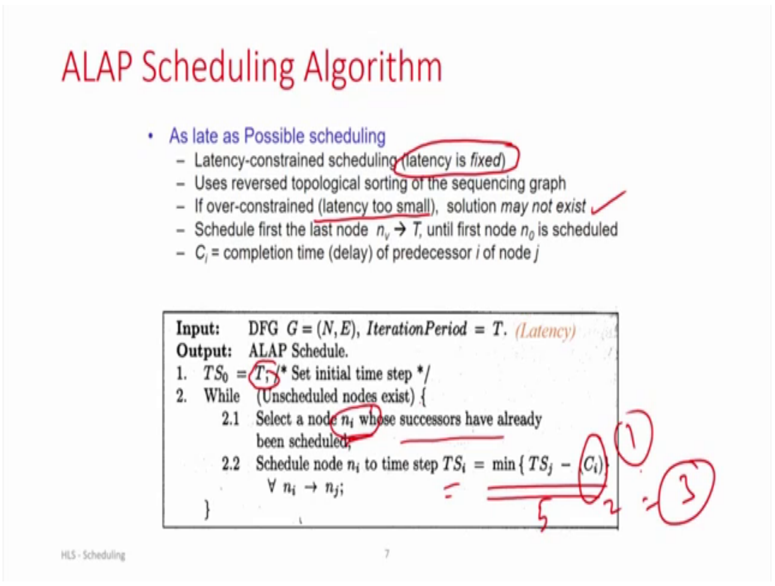
So, what is the advantage of as soon as possible algorithm this always give you this on a minimum latency right. So, the number of time step that will be given by this is minimum, so latency is nothing, but the number of time step ok latency is nothing, but time steps ok.

So, the advantage of as soon as possible is that it always give you the minimum latency; that means, there are minimum number of times step is required to schedule that particular set of operations ok. And this is kind of an optimal solution because we cannot have a layer over time 7 schedule which is kind of less time step than this whatever is given by this as soon as possible.

But this is unconstrained that means, the number of resource required here is there is no control over this right it always give you the maximum number of resource possible. For example, if you remember in the last class this particular example you actually able to schedule in 4 times step using 2 multiplier.

Now, you can see here there are 4 multiplier scheduled in time step 1. So, I need at least 4 month to execute this operation in hardware right. So, so; that means, it does not give you the optimum resource, but it gives the optimum latency or the number of time step. So, this is kind of unconstrained schedule, but if you are I ask you to use only 2 multi flat then this particular algorithm will not work right. So, that is something there is as soon as possible ok.

(Refer Slide Time: 07:41)



ALAP Scheduling Algorithm

- As late as Possible scheduling
  - Latency-constrained scheduling (latency is *fixed*)
  - Uses reversed topological sorting of the sequencing graph
  - If over-constrained (latency too small), solution *may not exist*
  - Schedule first the last node $n_v \rightarrow T$, until first node n_0 is scheduled
  - C_i = completion time (delay) of predecessor i of node j

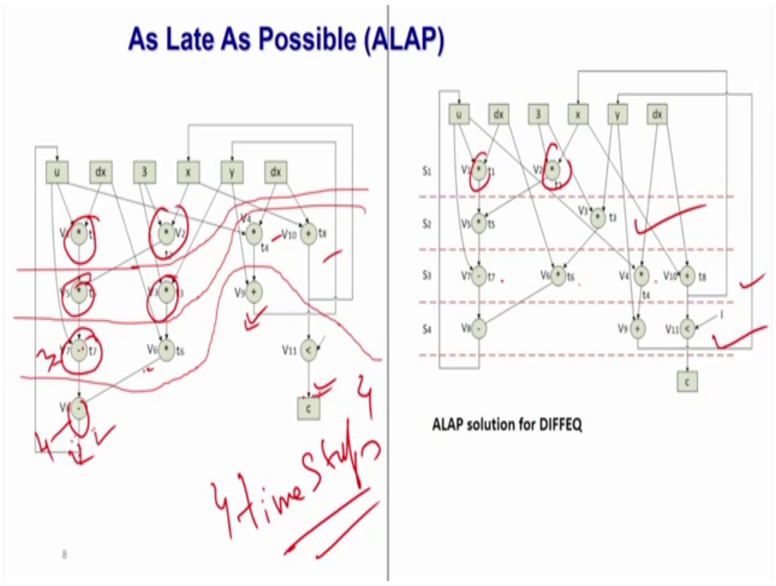
```
Input:   DFG $G = (N, E)$, $IterationPeriod = T$. (Latency)
Output:  ALAP Schedule.
1.  $TS_0 = T$; /* Set initial time step */
2.  While (Unscheduled nodes exist) {
        2.1  Select a node $n_i$ whose successors have already
             been scheduled;
        2.2  Schedule node $n_i$ to time step $TS_i = \min\{TS_j - C_i\}$
               $\forall\ n_i \rightarrow n_j$;
    }
```

HLS - Scheduling 7

So, similarly we have this as late as possible is just the opposite person of this s and as soon as possible algorithm over here we are going to try to delay the operation as late as possible right. So, here we will have a given a latency bound; that means, I want to schedule this particular thing in say k time step, or say 4 time step and whether this is a feasible or not. So, if the latency is given too small then they exist solution that we are going to get may not feasible right.

For example if I give the for this example as I understand at least 4 times step is required, but if I ask you to schedule this in 3 time step this is not feasible right. So, assuming that all multiplier is a one cycle operation so that is something is not feasible. So, if you just apply this as late as possible algorithm here is the solution may not exist because if you give a very less latency bound right.

(Refer Slide Time: 08:41)



As Late As Possible (ALAP)

ALAP solution for DIFFEQ

So, here the algorithm is I mean similar to that ASAP algorithm what it does it try to schedule the operation as late as possible. So, if you go into this we try start from the output so those operations are generating output right this, this, these operation. So, they are generating so they has to schedule in the last step right after that you do not schedule in the last step you need one more time step it.

So, we have to schedule all the operations that are producing output right. So, we have to schedule these operations in the last time step. So, here you start with a latency bound so latency bound this is so we T. So, we will start from the last time step so you say for example, say I ask you to schedule this behaviour in 4 time step. So, latency bound is given 4 right.

So, if you given this 4 time steps and then I will find out all the variables that are dependent that are producing output we are going to schedule them in the last time step of 4 times step so I am going to schedule these operations in 4 times step V 8, V 9, V 11, right.

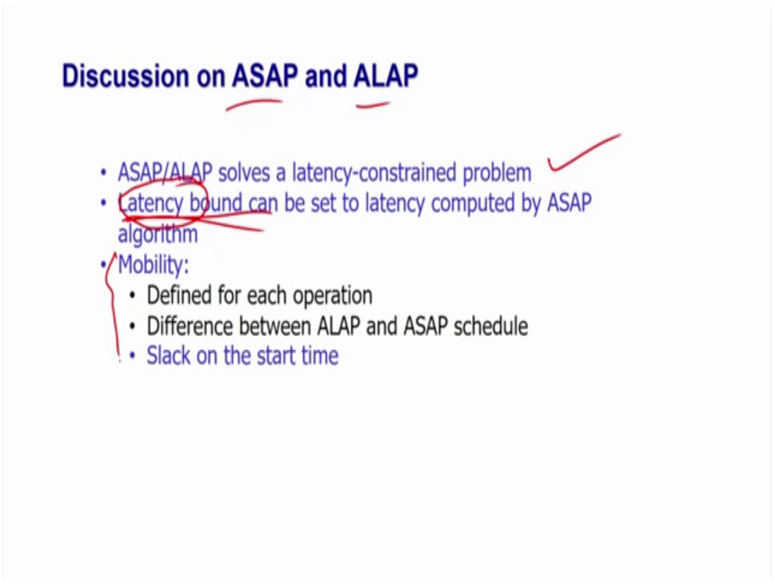
So, once they are done the operation that are actually depend on the this operation on which they are dependent they are now ready to be scheduled if you do not in this schedule here we did one more time step right. So, we have to so we are bound to schedule all the operations that the result of that operation is used by these operations right.

So, now if you look into this V 11 it is very clear at that down V 7, V 6, V 4, and V 8 has to V 10 has to be scheduled now. Because they are now ready let us say I am going to schedule this V 7, V 6, V 4, and V 8 in times step 3 then I reduce the time step by 1. And then I will find out these 2 operations has to be scheduled because they are the next so I schedule it here and then finally I have scheduled these 2 operations in the first time step right. So, this is how what is that a ALAP operations given right.

If you go look into the algorithm I will so, I will again the same process that I unscheduled do not exist I will found out those node whose successor has already been scheduled right. So, here I try to find out this node the successor is already scheduled for this their output is already scheduled that means, they are producing output so their output is already scheduled.

So, I am going to find out those number of operations and then I am going to assign their time step, mean of their predecessors I mean successors sorry. So, for example, here if you try to find out the time step of this it is successor is this time step s n is 4, so I am going to make it 3 right. Assuming this C is 1 here right. So, if you have C say 2 or 3 then if it is say 5 then it will have be this is 2, then it will be a 3 right. So, this multi cycle operation I will come to come later so this is how this algorithm works so this is also very simple algorithm.

(Refer Slide Time: 11:12)



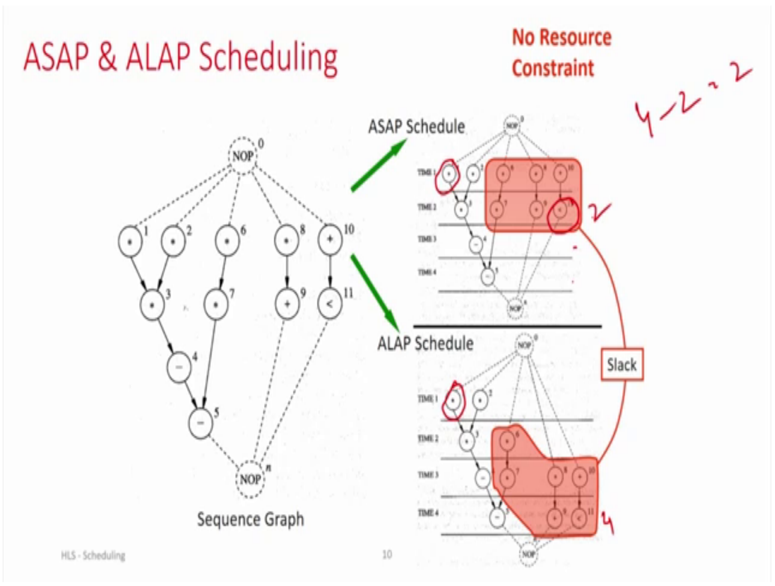
Discussion on ASAP and ALAP

- ASAP/ALAP solves a latency-constrained problem
- Latency bound can be set to latency computed by ASAP algorithm
- Mobility:
  - Defined for each operation
  - Difference between ALAP and ASAP schedule
  - Slack on the start time

So, discussion on this ASAP and ALAP i can understand say these are mostly we will not consider the resource bound. So, it always generate a meeting of minimum latency synthesis, but there is no control over this source it the kind of synthesis result we are going to get will be not resource optimized right so that is something.

But the good advantage is that is you give a latency bound. So, the ASAP solution or ALAP solution whatever that bound will get that will be the minimum. So, specifically ASAP will give the minimum latency bound, and that will be the minimum number of latency that you can obtain from this ASAP solution right. So, there is another important aspect we can utilize this ASAP, and ALAP algorithm to find out the mobility of operation what is that?

(Refer Slide Time: 11:57)

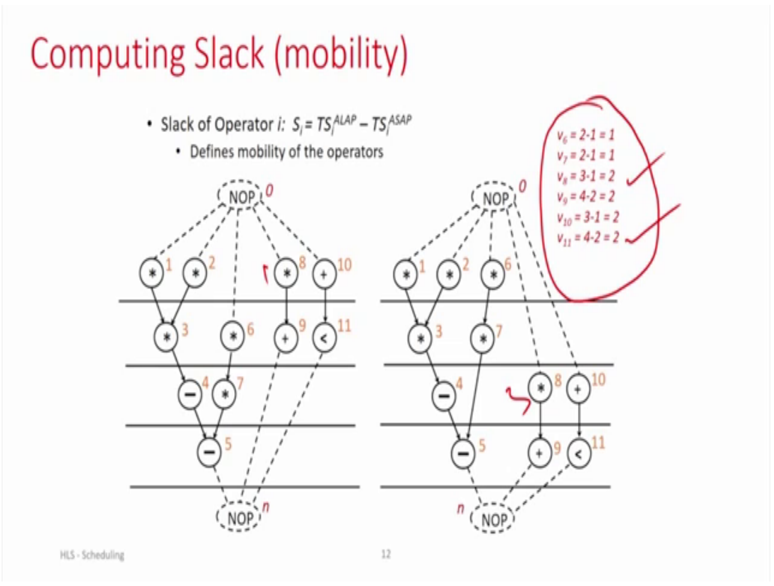


ASAP & ALAP Scheduling

So, mobility is something if you just go into this here you can see that this is my initial graph, and this is my ASAP schedule, this is ALAP schedule right. Here these operations scheduled in time step 1 in ASAP as well as in ALAP. So, these operation has to be executed in time step 1 there is no other alternative, but if you look into this operation determine V 11 this is scheduled in time step 2 here it is scheduled in 4 right. So, you can either schedule it 2, 3, or four.

So, there are mobility of 4 minus 2 which is 2 right the mobility of this is something the number the range of time step where you can actually schedule this particular operations. For example, here V 11 can be scheduled here and here right this is the length range from time step 2 times step 4. So, from this ASAP and ALAP we just define I mean when from that you can actually calculate the this mobility of each operations right.

(Refer Slide Time: 13:00)



Computing Slack (mobility)

- Slack of Operator *i*: $S_i = TS_i^{ALAP} - TS_i^{ASAP}$
    - Defines mobility of the operators

HLS - Scheduling

12

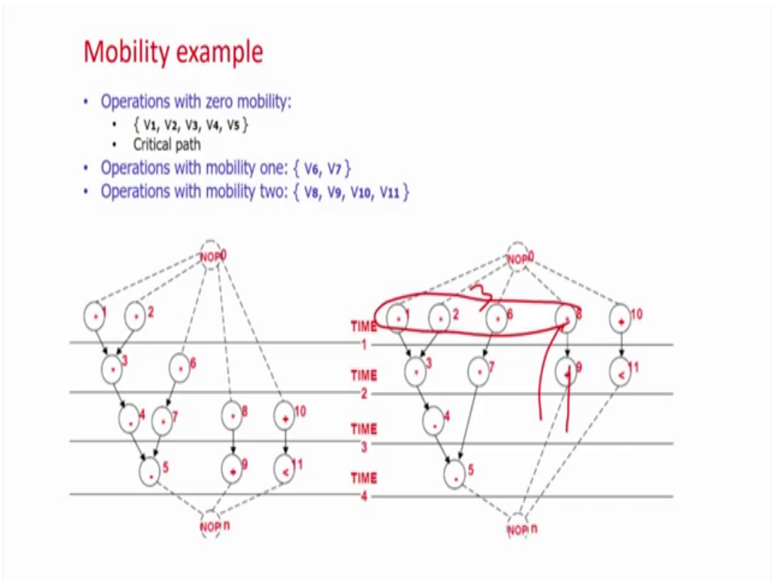
So, if you just see here these are the operations. So, how do they actually have a different time step you just calculate them you can actually get this mobility of each operators right. So, these operations for example, V 11 I just discuss it is 2, if we just take V 8 V 8 is here is time step 1 here. time step 3 so 3 minus 1 is 2.

So, similarly you can actually find out this mobility of each operations. So, what is the importance of this mobility? The mobility importance of the mobility is basically you can actually have a relaxation on where you can actually schedule this particular operations right.

For example when you have a resource constraint right so how do you have that constant that you cannot execute this operation now with more than 3 multiplier or something and if you have a mobility and if it is causing a problem you can actually delay that operation to the next time step so that you can actually improve that your resource utilization.

For example, here so here it there are 4 multiplier, but if you just move this to this and this to this because they have the mobility is more than 1, you can actually use 3 multiplier right so this is something we can utilize.

(Refer Slide Time: 13:55)

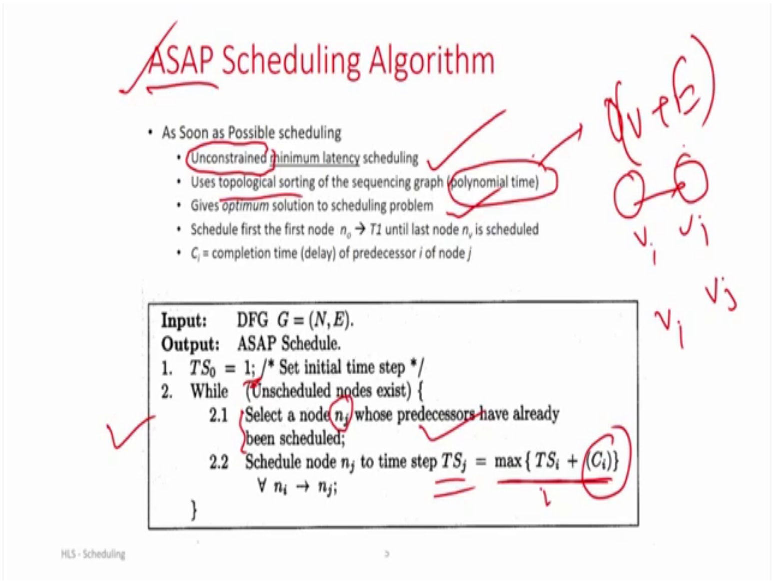


Mobility example

- Operations with zero mobility:
  - { v_1, v_2, v_3, v_4, v_5 }
  - Critical path
- Operations with mobility one: { v_6, v_7 }
- Operations with mobility two: { v_8, v_9, v_{10}, v_{11} }

So, this particular relaxation we can utilize when you have a resource constant try to generate a minimum resource we can utilize this mobility factor of which operations so that we can reduce the number of resource right. So, this is something also useful features of this ASAP and ALAP. And another important factor of this ASAP and ALAP is they are basically polynomial time algorithms right.

(Refer Slide Time: 14:25)



ASAP Scheduling Algorithm

- As Soon as Possible scheduling
  - Unconstrained minimum latency scheduling
  - Uses topological sorting of the sequencing graph (polynomial time)
  - Gives *optimum* solution to scheduling problem
  - Schedule first the first node $n_0 \rightarrow T1$ until last node n_n is scheduled
  - C_i = completion time (delay) of predecessor i of node j

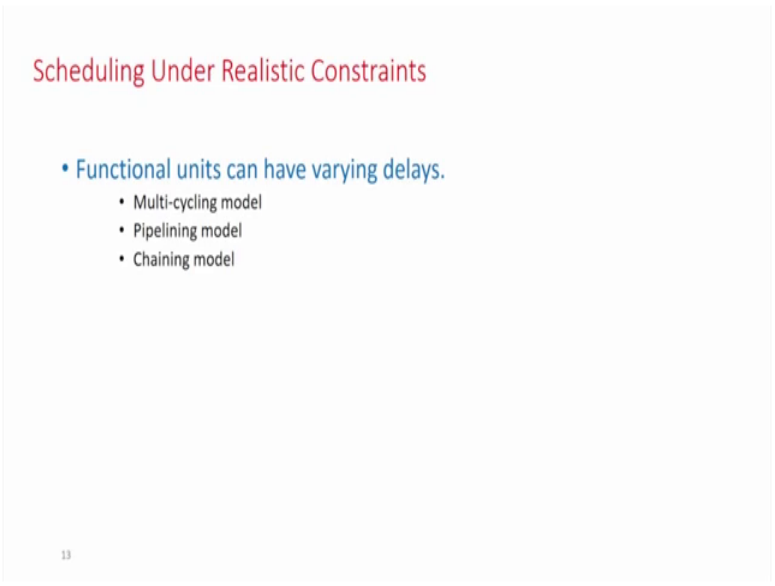
```
Input:   DFG $G = (N,E)$.
Output:  ASAP Schedule.
1.  $TS_0 = 1$; /* Set initial time step */
2.  While (Unscheduled nodes exist) {
        2.1  Select a node $n_j$ whose predecessors have already
             been scheduled;
        2.2  Schedule node $n_j$ to time step $TS_j = \max_i \{TS_i + C_i\}$
                 $\forall\, n_i \rightarrow n_j$;
    }
```

So, because you can actually use topological sorting topological sorting is something where if you have a is a sorting of the nodes of a graph if you have a edge from a V i to

V j then the sorting order V i you will always come before V j right and this is true for all possible such edges right. So, then this is coming to a topological sorting and those know that they are in the start we can actually schedule them faster right.

So, you can actually use topological sort to do these suppressor efficiently and this topological sort is use nothing, but graph this graph DFS, depth first search algorithm which is complexity order of V plus E, right V is the number of vertices and E is the number of edges. So, these algorithms are polynomial type, and they have given minimum latency bound, but without any resource constant right, so this is something the very simplest one of the algorithm ok.

(Refer Slide Time: 15:25)



Scheduling Under Realistic Constraints

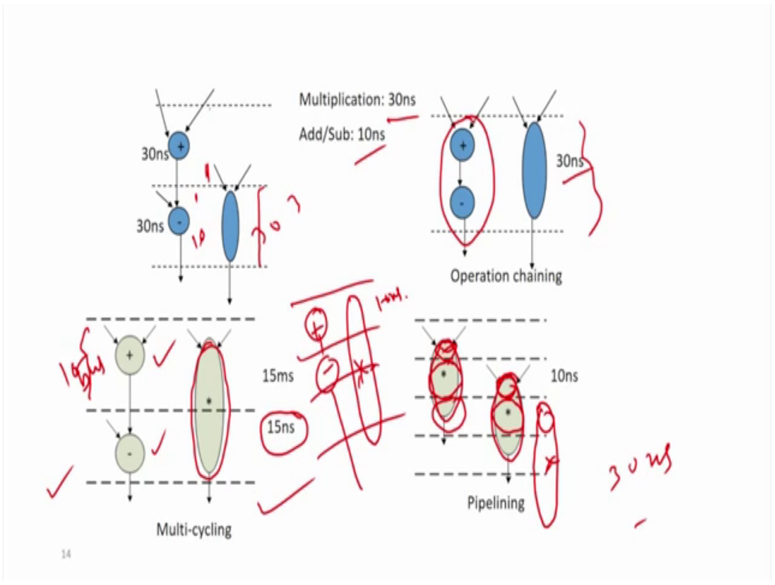
- Functional units can have varying delays.
    - Multi-cycling model
    - Pipelining model
    - Chaining model

13

Now, we will if we just move on as I discuss your algorithm in your circuit may have different kind of complexities. First of all you might have resource constraints if you have resource constraint; that means, you cannot use as much as possible resource you have to schedule everything with a resource bound right.

So, if you have that then ASAP, ALAP to work. You here as I mentioned earlier also if you have a operations can be of different time it can be multi cycle, it can be operation chaining, it can be pipeline, this example this example clarify this. So, for example, suppose this is the schedule that is generated by some algorithm say let me or something.

(Refer Slide Time: 15:58)



And your multiplier take 30 nano second and this adder is taking 10 nano second right. So, if you have this what will your clock pure here it has to be 30 nano second, right because your clock either have 10 or 30 you can have both. So, you have to decide that your clock period is something of 30 nano seconds right.

So; that means, in this particular schedule you can see that 20 nano second of the time this view of this clock period is even idle because we are not doing anything. Similarly here, but only because I have to execute the multiplier in 30 nano second I need that clock period. What will be the alternative the first alternative is multi second right what we can do we can define that my clock is obtain or say that 50 nanosecond and I can execute this in this in 1 clock this also in 1 clock, but I try to decide that I execute the multiplier 2 2 cycle.

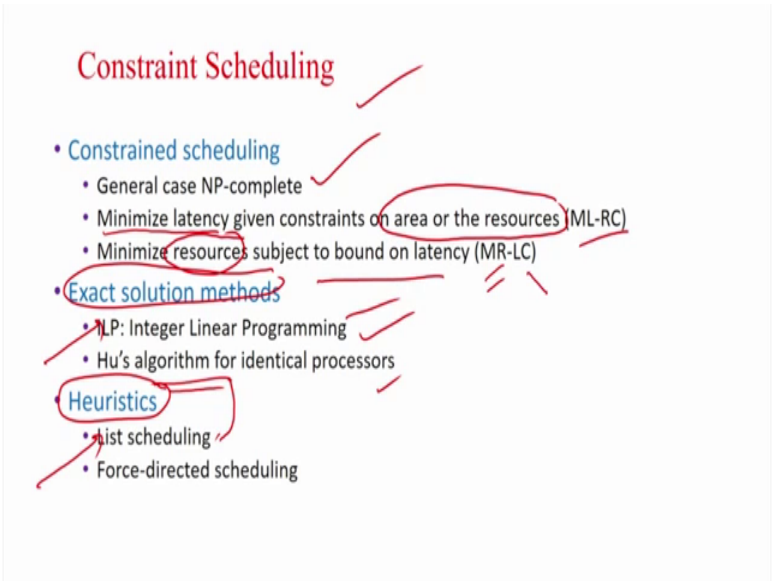
So, that now from 30 nano second clock I I go to 50 nano second even you can do it in 3 in 10 nano second period right. So, you can do adder here, multiplier subtraction here, nothing here and you can execute the multiplier in 3 clocks right. So, this is also you can achieve 10 nanosecond also you see multi second. So, this is the another radius and so you cannot always the case that your multiplier or any operations is always in a single cycle to get a high clock speed sometimes we decide to use a multi cycle for the bigger operations we use a multiple cycle.

A multiple clock cycle to execute that particular operations, but as the smaller operation can be executing first so this is one option or other way you can actually do it you can keep your clock your 30 nano second, but you try to execute both of them in the same clock right then also you can achieve 30 nanosecond right. So, when 30 nano second clock period right so that is also possible and the fourth third possibility is the pipeline.

So, you execute this in parallel you know the pipelining concept yeah so this is started after first step first phase of this I will start the second multiplier, and then he wants the so in the second clock the second part of the multiplier will executed whereas, the first part of the second multiplier will executed in the third clock the last part of the multiplier will be executed the second part of the second multiplier and the first part of the third multiplier will be executed right. So, this is how the multiplier works. So, after 30 nano second first clock faser output will come, but after every 10 nano second we will get another result so, that is also another possibility.

So, now the question is if you have all these complexities like your resource constant you have this multi cycle operation, pipeline operations, or say operations chaining is also possibility how hard will your good scheduling techniques right so that is something you are going to discuss. So, that is what is called constants scheduling right you have a constant scheduling.

(Refer Slide Time: 18:45)



Constraint Scheduling

- Constrained scheduling
  - General case NP-complete
  - Minimize latency given constraints on area or the resources (ML-RC)
  - Minimize resources subject to bound on latency (MR-LC)
- Exact solution methods
  - ILP: Integer Linear Programming
  - Hu's algorithm for identical processors
- Heuristics
  - List scheduling
  - Force-directed scheduling

So, what is constant scheduling as I mentioned it can have different kind of constant and you can have 2 different version. You try to minimize the latency given a resource constant or you try to minimize resource given a you latency constant.

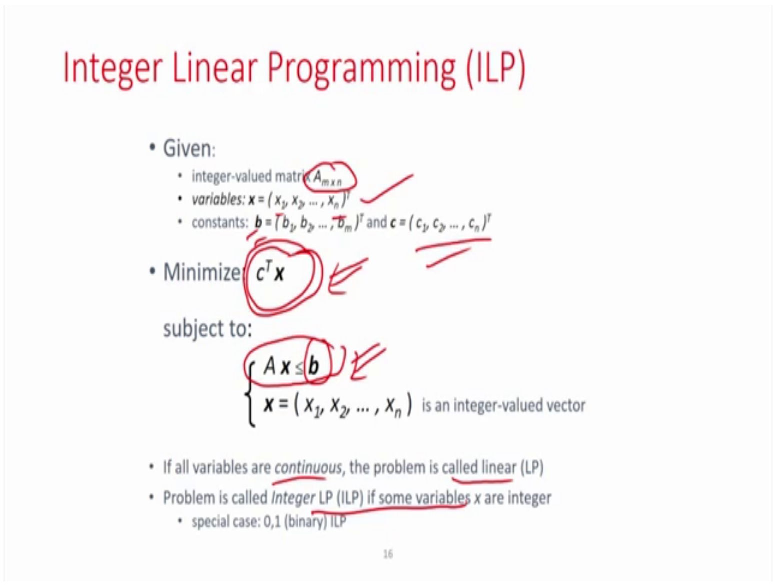
So, I have ML RC minimize latency under resource constant or minimize resource under latency constant. There are 2 variations of the scheduling algorithm right and in general both the problem are NP complete. What is NP complete? NP complete in the sense those algorithm those problem whose solution cannot have do not have a polynomial time deterministic solution right.

So, that means so whatever the algorithms are there is exponential. So, so in this kind of NP complete problems if you have a exponential algorithm I am saying 2 to the power k or 2 to the power k is very expensive and if you have a bigger algorithm it would not do alright.

If you just write it to write and program in C, say which is say 2 to the power n kind of algorithm. So, if your n is going beyond 40 or 50 your algorithm will never terminate right this is a lot of time so we try to find out it is a heuristic algorithm heuristic is what we try to find out the some logical consequence logical decisions which result in some good solution most of the cases instead of going for exact solution we tried to go for a heuristic solution right and exact solution also is there we can use integer linear programming algorithm or Hu's algorithm ok.

So, we will try to discuss one exact solution and one a heuristic solution in today's class and since this listen ruling based on the idea of this Hu's algorithm we will just briefly discuss what is Hu's algorithm, if we are going to this. So, the next 20 minutes are so we are going to discuss ILP with scheduling algorithm and list scheduling ok.

(Refer Slide Time: 20:40)



Integer Linear Programming (ILP)

- Given:
  - integer-valued matrix $A_{m \times n}$
  - *variables:* $\mathbf{x} = (x_1, x_2, \dots, x_n)^T$
  - constants: $\mathbf{b} = (b_1, b_2, \dots, b_m)^T$ and $\mathbf{c} = (c_1, c_2, \dots, c_n)^T$
- Minimize $c^T \mathbf{x}$

  subject to:

$$\begin{cases} A\mathbf{x} \leq \mathbf{b} \\ \mathbf{x} = (x_1, x_2, \dots, x_n) \text{ is an integer-valued vector} \end{cases}$$

- If all variables are *continuous*, the problem is called linear (LP)
- Problem is called *Integer* LP (ILP) if some variables x are integer
  - special case: 0,1 (binary) ILP

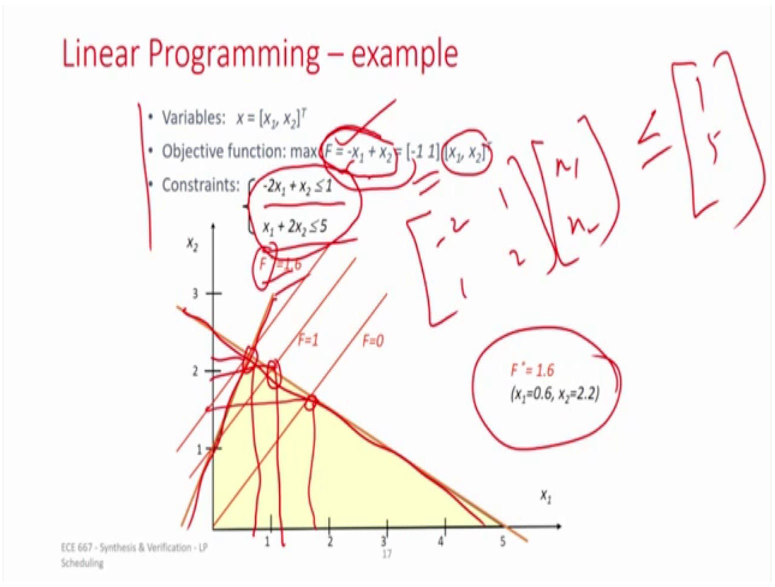
16

So, so before going into this modelling your ah scheduling algorithm instead of using n l p and it sorry a sorry ILP integer linear programming let us defined that what is into linear programming right. So, here you can see you have a minimize some factor where you have set of variables x 1 to x n and you have this constant this constants are there.

So, you try to minimize this value of this subject to satisfy all the variable value of this variable try to find out the value of this variable such that it satisfy all this constant and give the minimum value of this right where C is a constant vector, b is also a kind of a constant vector right so b is also a constant vector.

So, this is something that our formulation and if your variables are continuous then it say linear programming solve solution if your problem this variables x 1 to x n are integer then you say ILP ok.

(Refer Slide Time: 21:38)



Linear Programming – example

- Variables: $x = [x_1, x_2]^T$
- Objective function: max $F = -x_1 + x_2 = [-1\ 1](x_1, x_2)^T$
- Constraints: $\begin{cases} -2x_1 + x_2 \le 1 \\ x_1 + 2x_2 \le 5 \end{cases}$

$$\begin{bmatrix} -2 & 1 \\ 1 & 2 \end{bmatrix}\begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \le \begin{bmatrix} 1 \\ 5 \end{bmatrix}$$

ECE 667 - Synthesis & Verification - LP Scheduling

17

So, let us take an example for example, suppose I have the variable x 1 and x 2 and my objective function is this I want to maximize this value of this a minus x 1 plus x 2. So, if you have the constant this these are the 2 constant 2 x minus 2 x 1 plus x 2 is less than equal to 1 and x 1 plus 2 x 2 so the value of x 1 x 2 should satisfy both this constant and generate the maximum possible value of f right to model this your C is minus 1 and 1 because this is minus 1, this is 1 and this is your variable.

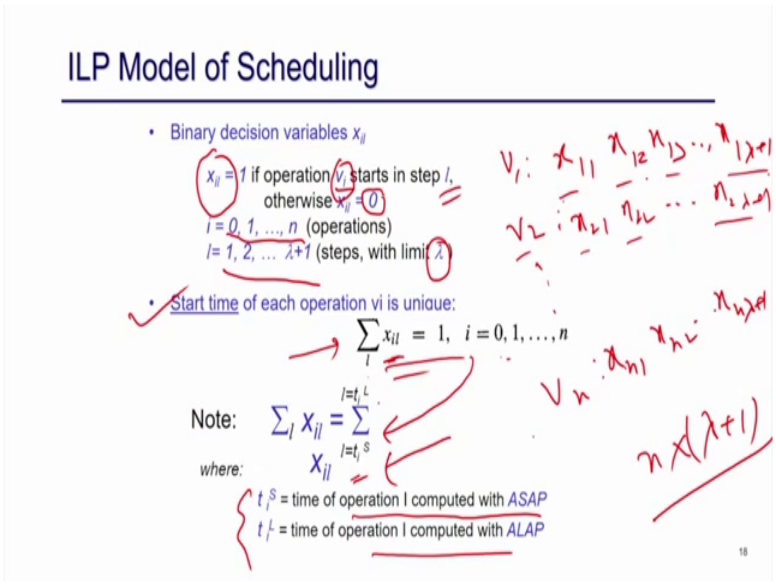
So, this is your this C t into x transpose of this and your constant is this a x minus b. So, your these constants are nothing, but this so you can see your a is nothing, but minus 2 1 1 and 2 and this is your x x 1 x 2 right so this is how this a x less than equal to b right and this is less than or equal to nothing, but 1 5 so this is how we model this problem.

So, the so if you just plot this variable you will get this line so this is the solution space like this yellow part is the solution space and we tried to find out the maximum value of these right. So, if we just put the value of this x and y this value then we get f equal to 0. So, if you just give the value of f 1 is this say something like a 1.8, and this 1.1, or something and getting equal to 1.

But if I give the value of x is 2.2, and 0.8 then you will get the maximum value right so that will result in my 1.6 so that is the maximum possible value of this s, this is how the whole thing is works. So, we try to find out solution space and then trying to find out the maximum value of this which will give you the maximum of this objective function.

So, now we are going to model our scheduling algorithm instead of using this. So, we have to find out what are the variables we need to model your scheduling problem to this and what would be your constant right so these 2 we have to find out let us discuss.

(Refer Slide Time: 23:43)



ILP Model of Scheduling

- Binary decision variables x_{il}
  - $x_{il} = 1$ if operation v_i starts in step l, otherwise $x_{il} = 0$
  - $i = 0, 1, \ldots, n$ (operations)
  - $l = 1, 2, \ldots \lambda+1$ (steps, with limit λ)
- Start time of each operation vi is unique:

$$\sum_l x_{il} = 1, \quad i = 0, 1, \ldots, n$$

Note: $\sum_l x_{il} = \sum_{l=t_i^S}^{l=t_i^L} x_{il}$

where:

t_i^S = time of operation I computed with ASAP

t_i^L = time of operation I computed with ALAP

18

$V_1 : x_{11}\ x_{12}\ x_{13} \ldots, x_{1\lambda+1}$
$V_2 : x_{21}\ x_{22} \ldots\ x_{2\lambda+1}$
$V_n : x_{n1}\ x_{n2}\ x_{n\lambda+1}$
$n \times (\lambda+1)$

So, for our ILP modelling of scheduling we are going to use a variable x i l ok. So, what does it mean x i l is one means variable operations V i start at time step l right. So, x i l is equal to 1, if that particular operations talk time of that particular operation V i is l otherwise it is 0 right. So, if we just assume you have a operation it is a V 1 to V n the operations are V 1 to o V n and say time limit is lambda then your lambda can be 1, 2 lambda plus 1.

So, how many variables are there so operation V 1 I have x 1 1 because it can s schedule this x 1 to x 1 3 into x 1 lambda plus 1; that means, operation if operation V 1 is scheduled in time step 1 then this will be 1 and if it is time if it is operation 1 scheduled in time step 2 then this will be 1 and so on.

So, I have the this number of variables for operation V 1 similarly for V 2 I will have a variable like x 21 x 2 2, to x 2 lambda plus 1 this way V n equal a x n 1, x n 2 to x n lambda plus 1 right. So, these are the possible variables I have right. So, these are the possible variables right n into lambda plus 1 these are the number of variables.

And we have defined this is 0 so, ah only that particular operations schedule on that right. So, you have to find out those values right so these are the variables the value of this variables has to be find out right.

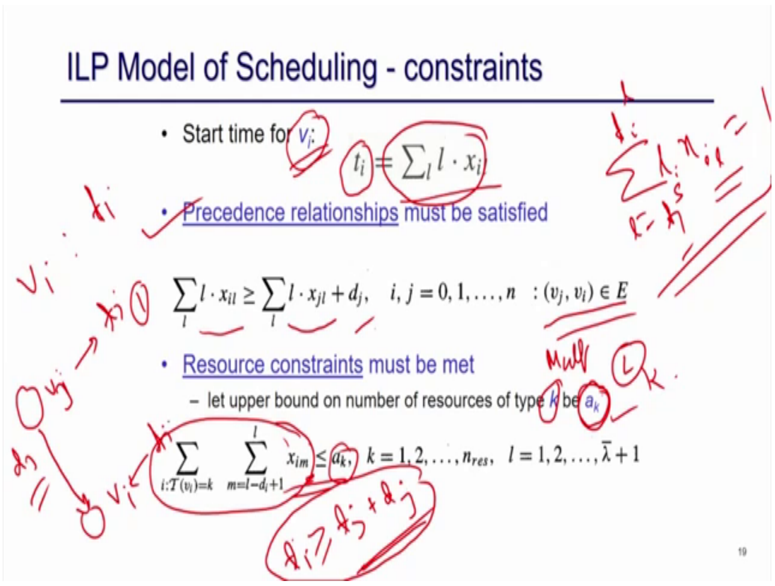
And now what are the constant I am going to use the start time up is variable is unique right. So, for us; that means, what you cannot start a operation in multiple time right it has to start in same time step right so; that means, only one of this in lambda plus some variable can be 1 plus V 1 similarly this for operation V 2 are exactly at most well exactly one of this lambda plus 1 variables will be 1 yeah.

So, I have mention the maximum of this n lambda 1 variable only one of them will be 1 right so that is that given by the unique start time that summation of this x 1 i l for i equal to 0 to n or will be 1 right and now if you just go back to your ASAP and ALAP solution in operation mean ASAP you will give you this minimum time step after that I mean it has to be exhibited by that time step it cannot go ah it cannot go beyond that because then the dependencies you violated.

And ALAP give will the last possible was scheduled right if you try to schedule it after that then your number of time schedule introduce the span that life span this mobility is something from AS the time step adds ASAP schedule and wide ALAP schedule right

So, now you can actually say I do not need to talk consider all are; obviously, the other variable will be 0 only this time variable that is between this time step t i s to t i n which is nothing, but the time operation sorry ASAP scheduled and this ALAP schedule around that variable only this this particular variable can be 1 right. So, that is so you can rewrite this expression in this summation from t i s to t i n the mobility spend right others other pair other times too it is; obviously, this is the other variable will be 0.

(Refer Slide Time: 27:12)



ILP Model of Scheduling - constraints

- Start time for v_i: $t_i = \sum_l l \cdot x_{il}$
- Precedence relationships must be satisfied

$$\sum_l l \cdot x_{il} \ge \sum_l l \cdot x_{jl} + d_j, \quad i, j = 0, 1, \ldots, n \quad : (v_j, v_i) \in E$$

- Resource constraints must be met
  - let upper bound on number of resources of type k be a_k

$$\sum_{i:T(v_i)=k} \sum_{m=l-d_i+1}^{l} x_{im} \le a_k, \quad k = 1, 2, \ldots, n_{res}, \quad l = 1, 2, \ldots, \bar{\lambda} + 1$$

19

So, we do not have to consider those. So, you just simplify this expression into this right. So, this is the first possible constant. What is the now that is also very important so even if I just as I mentioned that for variable this l equal to t i s to t i l from this ASAP to ALAP this x i l equal to 1 right. So, this is what I assumed so; that means, this operation will be must be scheduled at least one of the time step only this variable x i l would be 1, only one of the variable from this range will be 1 right.

And now suppose this is scheduled and said time step k right. So, what will be the how do you represent that your start time t just say per operation say V i start time say t i let us so how can I represent this t i is a l into t i agent. So, this is the summation in this summation I just multiply this l right because now if this particular person is scheduled in l what is the time step k then only that particular k will be 1 all other will be 0.

So, if I multiply that with this k value then we will be the k right. So, the talk start time of the particular can be represented by l into x i l right so, this is how I can find out the start time of this right. Because now I have to represent everything in terms of this variable because these are the only variables I have and I have to make the model using this variables only ok.

So, so now this is something the start time now first I just mention the first constant is the start time must be unique so which is given by this this constant, then I have to find out the prep residents relationship must be satisfied right. So, for example, suppose you have a h from V i to V j no sorry V j to V i then this and say this will take say ah d i d j time step right. So, the cycle of the number of times step require to execute this is d j so we did, so d j number of time step.

So, I if this is stars star to this t j and the start starts time step of this V i is t i then t i must be greater than equal to t j plus t j right because you cannot start unless this operation is executed because there is a dependency so if it is start at t j it will take t j t i this t i plus d j minus 1 basically those number of step and then only the next step I can only schedule this right.

So, this is the timing constant corresponding you have satisfying the precedence relationship and if I just represent this t i by this it will give you this l into x i and if I just represent t j it will give you this right t into i j l right this I l this is j l because this is nothing, but the operation g I and d j and this is d j. So, this is what I am going to get it for the precedence relation now this is for all possible edges I just discussed it for 1 edge and this is for all possible edges I will get this kind of constant.

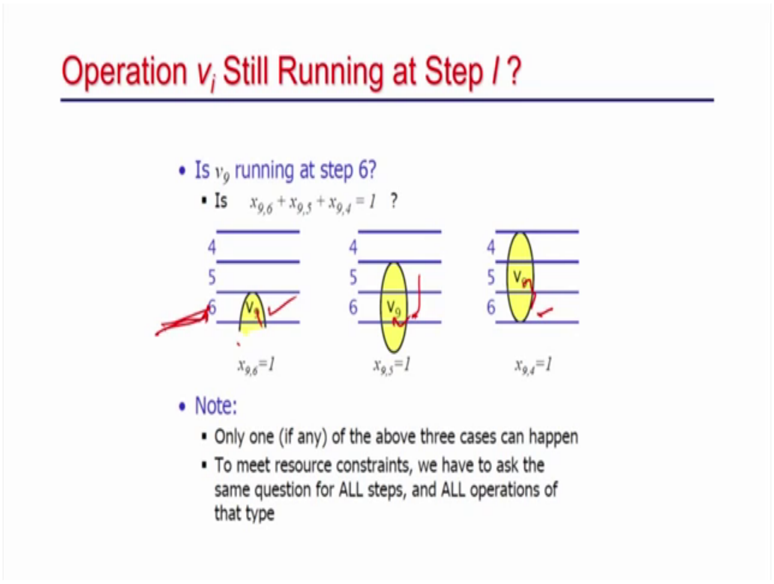
So, which is actually has to be satisfied because whatever the schedule you are going to get this has to satisfy this procedures relationship and this is given by this set of this set of with this setup constant right the equations and how many such equation will be there 1 equation for each edges. So, the number of equations such equation will be the number of edges in the dependent signal.

And the third set of constant is nothing, but that resource constant what is resource constant? So, if you just mentioned that I mean if you have some resource constant; that means, specifying the I want to execute this using 2 multiplier or say 3 multiplier how to put that particular factor into this constant solving solve solving scenario right.

So, suppose I assume that the operation operator type k has a k number of instances this is nothing, but say multiplier has two instances right. So, so now, since my operations are multi cycle at a particular time I can have only k number of multiplier active a k number of multiplier will be active at a particular time step and that is given by this constant that whatever this value I will discuss should be now this is this is define the number of multiplier required in any time steps say time step i, time step l should be sorry time step m sorry time step m should be less than up a k right

So, now what is this signifies since now the operation same multi cycle. So, if some operation started somewhere early that may be also continuing in this time step and this some operation will start here.

(Refer Slide Time: 31:47)



Operation v_i Still Running at Step l ?

- Is v_9 running at step 6?
  - Is $x_{9,6} + x_{9,5} + x_{9,4} = 1$?

4
5
6
V9
$x_{9,6}=1$
4
5
6
V9
$x_{9,5}=1$
4
5
6
V9
$x_{9,4}=1$

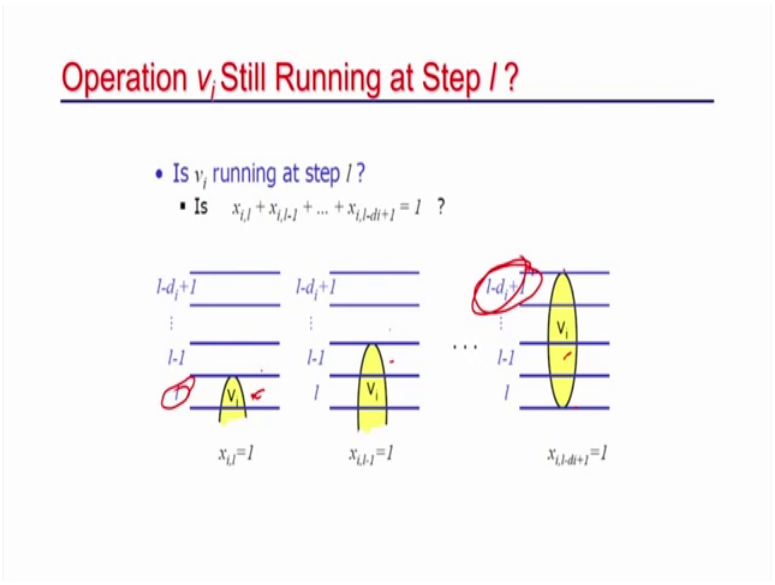
- Note:
  - Only one (if any) of the above three cases can happen
  - To meet resource constraints, we have to ask the same question for ALL steps, and ALL operations of that type

Let us see if we just go into this some operation if you just think about the time step 6 some operation starts here, some operation may continue here, and you started early and continuing in this time step or some operation started too earlier that is going to finish. So here see all these 3 are active in this particular time step all right.

So, I need at least 3 multiplier to execute this these 3 operations so there are 3 more I if this does this is the scenario then I need 1 multiplier to say this is say V 1, this is say V 2, this is say V 3 I need to say 3 multiplier to execute this in which steps time step 3 right.

So, if you have so whatever the resource constant is given to you so I have to consider all these 3 scenarios and summation of this should be less than of the number of multiplier is specified right the opera operator is given.

(Refer Slide Time: 32:33)


Operation v_i Still Running at Step l ?

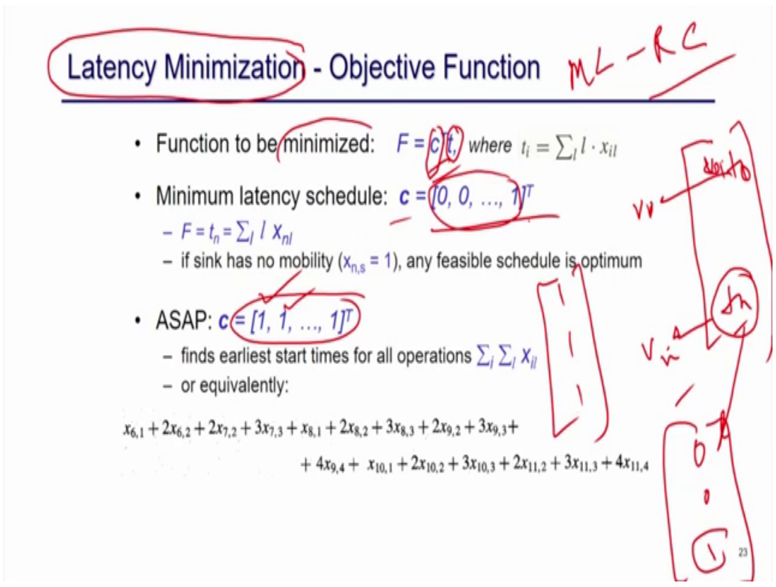
- Is v_i running at step l ?
  - Is $x_{i,l} + x_{i,l-1} + \ldots + x_{i,l-di+1} = 1$?

So, if you make it generic so if you are considering the time step l some operation start by here or start the next pervious state or pervious state or till d i l minus d i plus one right because this is a V i operation and it takes d number d i number of time step it might start here still this these are the all possible scenario right. And all these scenarios of summation of all these scenario should be less than equal to a k and this is what is given by this constant that this is for the multiplied specific type say this is operator type k, the towns time step.

So, so I have to consider all that operations that started from l minus d i plus 1 to l because those are active in time step l right because I am now considering our concern about the time step m m. So, in m all the operations that are started till this l minus d plus one or to this previous step all are active so for all of them I need a multiplier.

So, all these active operations should be less than equal to a k so these are the 3 possible constant one constant is this unique start time, next constant is to maintain the data dependency, and third constant you should maintain the resource constant right. So, this is the set of constant.

(Refer Slide Time: 33:49)



Latency Minimization - Objective Function

- Function to be minimized: $F = c^T t$, where $t_i = \sum_l l \cdot x_{il}$
- Minimum latency schedule: $c = [0, 0, \ldots, 1]^T$
  - $F = t_n = \sum_l l \, x_{nl}$
  - if sink has no mobility ($x_{n,s} = 1$), any feasible schedule is optimum
- ASAP: $c = [1, 1, \ldots, 1]^T$
  - finds earliest start times for all operations $\sum_i \sum_l x_{il}$
  - or equivalently:

$$x_{6,1} + 2x_{6,2} + 2x_{7,2} + 3x_{7,3} + x_{8,1} + 2x_{8,2} + 3x_{8,3} + 2x_{9,2} + 3x_{9,3} + \\ + 4x_{9,4} + x_{10,1} + 2x_{10,2} + 3x_{10,3} + 2x_{11,2} + 3x_{11,3} + 4x_{11,4}$$

23

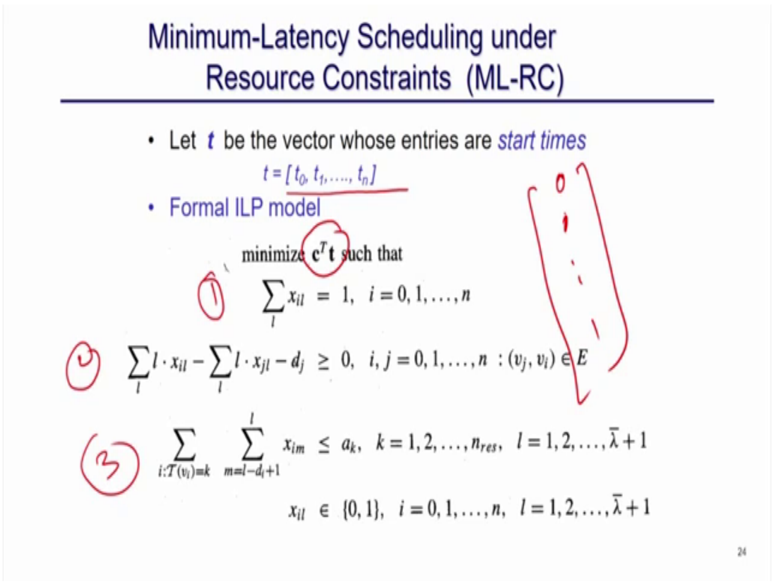
And now so what are problem. So, I have this. So, my now this is my so if you have the latency minimization this is minimized it latency under the resource constraint then my objective resource it minimize this factor which is C by t and t is nothing, but this V this start time of these operations right the start time of the operations which is nothing, but this say start time of these operations of all the operations.

So, if I just give you this value so this t would be of size n right. So, some t 1 to t n where the t n is the start time of the operation n and t sorry t 0 is the start time a operation V 0 right. So, that will give you the start time up V n right so this is how the whole thing works. And now this con constant is now we have to decide the what is the constant value right.

So, now, if you want to minimize the latency what will be the constant value I am going to consider this C n all 0 only in the last one is 1. Why because I want to get the minimum latency; that means, I want to schedule this t n as soon as possible right. So, that is the minimum value of this t n will give you the minimum latency right other value I am do not care right, but if you the ASAP solution I want to minimize that I mean I want to schedule all the operation as soon as possible then the or C will be nothing, but all one right.

So, all operation try to find out the minimum value start a minimum start value of that particular operations right. So, this is something for ASAP is C will be this for a minimum latency ML R C problem the C will be 0 0 right so 0 0 last bit is 1.

(Refer Slide Time: 35:36)



Minimum-Latency Scheduling under Resource Constraints (ML-RC)

- Let t be the vector whose entries are *start times*

  $t = [t_0, t_1, \ldots, t_n]$

- Formal ILP model

minimize $\mathbf{c}^T\mathbf{t}$ such that

$$\sum_l x_{il} = 1, \quad i = 0, 1, \ldots, n$$

$$\sum_l l \cdot x_{il} - \sum_l l \cdot x_{jl} - d_j \geq 0, \quad i, j = 0, 1, \ldots, n : (v_j, v_i) \in E$$

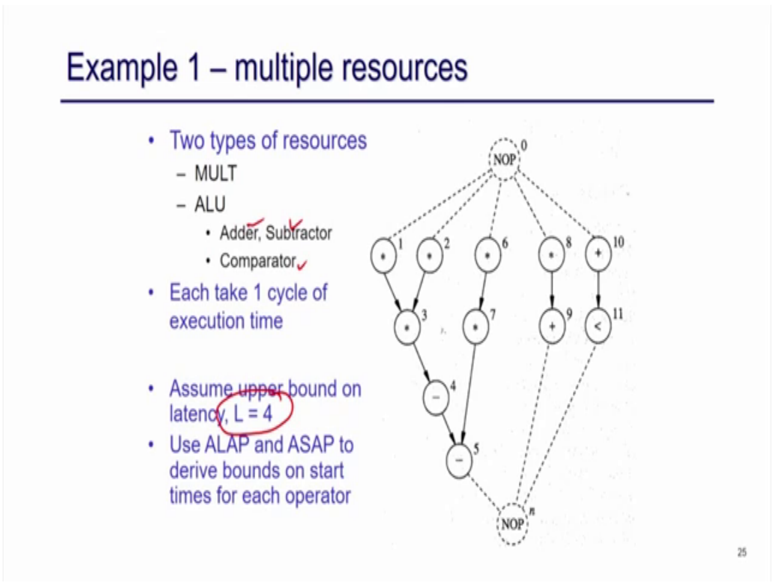
$$\sum_{i:\mathcal{T}(v_i)=k} \; \sum_{m=l-d_i+1}^{l} x_{im} \leq a_k, \quad k = 1, 2, \ldots, n_{res}, \quad l = 1, 2, \ldots, \bar{\lambda}+1$$

$$x_{il} \in \{0, 1\}, \quad i = 0, 1, \ldots, n, \quad l = 1, 2, \ldots, \bar{\lambda}+1$$

24

So here is a somebody I have the start time t 1 to t n I want to minimize this C t where this ah your and I have the constant 1, this is a unique start time, this is the data dependence, and this is the resource resources constant. So, this is subject to this I want to minimize this value where C is for ML RC this is nothing but 0 0 to 1 right. So, this is how the minimum latency is scheduling and a resource constant is modelled right.

(Refer Slide Time: 36:09)

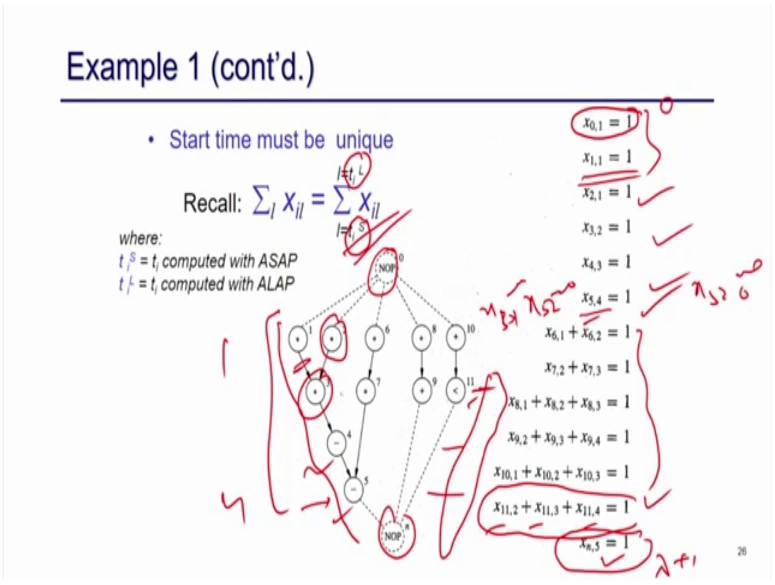


Example 1 – multiple resources

- Two types of resources
  - MULT
  - ALU
    - Adder, Subtractor
    - Comparator
- Each take 1 cycle of execution time
- Assume ~~upper~~ bound on latency, L = 4
- Use ALAP and ASAP to derive bounds on start times for each operator

Let us take an example because things little bit looks complicated. So, here suppose I want to have this same differential equation solver example and this is that sequencing graph and I have multiplier and ALU say ALU can execute addition subtraction compare all can be executed same block. And I say I assume for simplicity that I have all the operations are single cycle so that d i value for each hour V i is 1 right. If you have this and and say my latency bound is for I want to exude everything in lambda is 4 right so that is something I have to solve I see whether this is feasible not right.

(Refer Slide Time: 36:46)



Example 1 (cont'd.)

- Start time must be unique

Recall: $\sum_l x_{il} = \sum_{l=t_i^S}^{l=t_i^L} x_{il}$

where:
t_i^S = t_i computed with ASAP
t_i^L = t_i computed with ALAP

$x_{0,1} = 1$
$x_{1,1} = 1$
$x_{2,1} = 1$
$x_{3,2} = 1$
$x_{4,3} = 1$
$x_{5,4} = 1$
$x_{6,1} + x_{6,2} = 1$
$x_{7,2} + x_{7,3} = 1$
$x_{8,1} + x_{8,2} + x_{8,3} = 1$
$x_{9,2} + x_{9,3} + x_{9,4} = 1$
$x_{10,1} + x_{10,2} + x_{10,3} = 1$
$x_{11,2} + x_{11,3} + x_{11,4} = 1$
$x_{n,5} = 1$

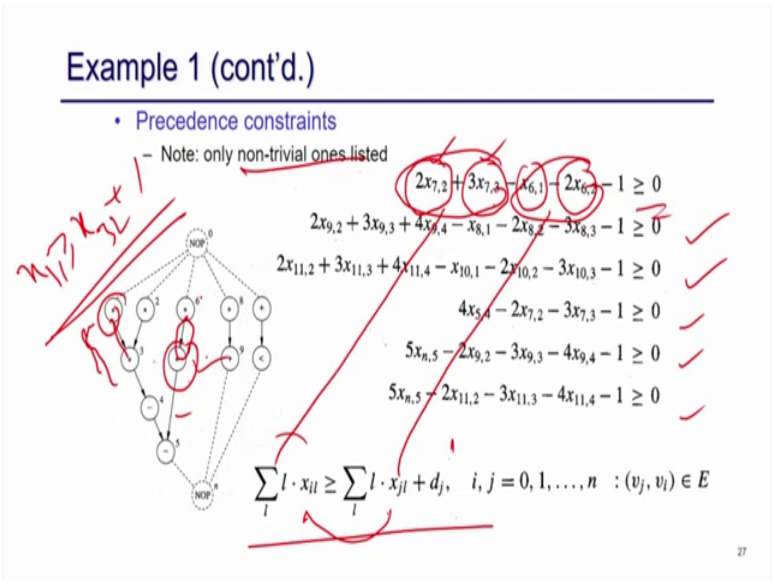
Now, if you find out that unique start time so unique start time is given by this now the mobility will come to in picture for this source and sink operations they are always scheduling this is the for source, and this is for sink these are always scheduled in the last time this is for lambda plus 1 right because 4 and this is 0 this is always there.

And I am going to schedule all these things from 1 to 4 right that is what my the way model that sequencing graph. Now this as I mentioned this is from the a l ASAP to ALAP is solution so now, if you consider this operation one here ALAP, and ASAP both give me one right because there is no mobility for this operation that we have discussed earlier.

So, x 1 1 must be 1; that means, what this operation viewer must start as time step 1; similarly for x 2 also this ah this x 3 also this because this is that critical path and there is no mobility here all are 1 x 4 and x 5 till this point. So, x 5 has to schedule a time step 4 so, that is x 5 4 is 1, but x 5 1, x 5 1, this x 5 2, x 5 3 all are 0 right. So, these are all 0 this is definitely giving this has to have scheduled in time step 4.

And now for the other operations for example, V 11 mobilities C r is two if you remember. So, this can schedule here, here or here 3 is time step this is the ASAP solution. So, V 11 to V 11 x 11 3 and x 11 4 somewhere some would be one because it can schedule anywhere any one of these 3 steps so this is what is given by this.

(Refer Slide Time: 38:32)



Example 1 (cont'd.)

- Precedence constraints
  - Note: only non-trivial ones listed

$$2x_{7,2} + 3x_{7,3} - x_{6,1} - 2x_{6,2} - 1 \geq 0$$
$$2x_{9,2} + 3x_{9,3} + 4x_{9,4} - x_{8,1} - 2x_{8,2} - 3x_{8,3} - 1 \geq 0$$
$$2x_{11,2} + 3x_{11,3} + 4x_{11,4} - x_{10,1} - 2x_{10,2} - 3x_{10,3} - 1 \geq 0$$
$$4x_{5,4} - 2x_{7,2} - 3x_{7,3} - 1 \geq 0$$
$$5x_{n,5} - 2x_{9,2} - 3x_{9,3} - 4x_{9,4} - 1 \geq 0$$
$$5x_{n,5} - 2x_{11,2} - 3x_{11,3} - 4x_{11,4} - 1 \geq 0$$

$$\sum_l l \cdot x_{il} \geq \sum_l l \cdot x_{jl} + d_j, \quad i,j = 0,1,\ldots,n \quad : (v_j, v_i) \in E$$

27

Similarly for I can find out for all variables right so this is my constant corresponding to the start time. Now, I will go to the presidents constant now if you just take about this example right 1, 2, 3. So, now, this 1, 2, 3 ah there is something just a minute so yeah this is not mention so I will let us say take about this x 7 so for x this because this is scheduled.

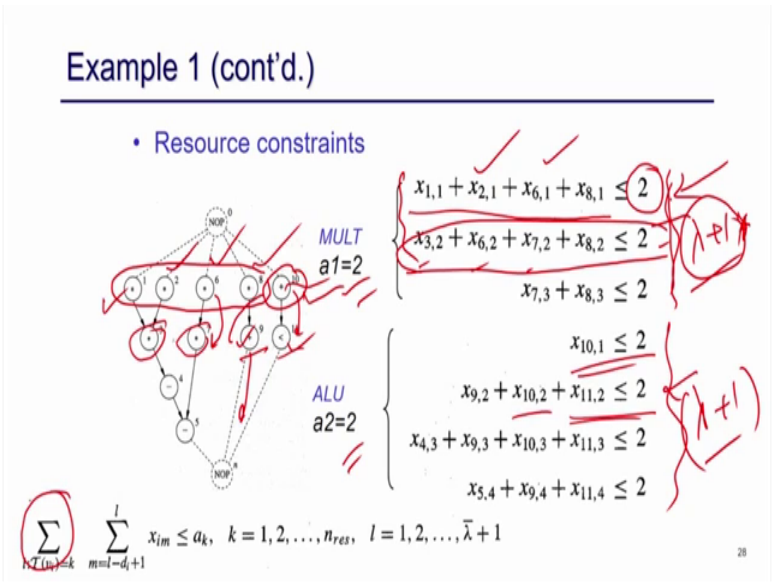
So, let us take and V 1 and this edge right so what will happen this schedule in 1 1 and this is scheduled in always this is 1 3 this is 2 right. So, this is x 3 2 because this is always scheduled in x 3 and then this is something what is given by these rights because this is always peak this is my x 1 1 this is x 3 3 and this is one second. So, this must be greater than equal to plus 1 right. So, this is constant I am going to get for this edge.

Now take a other edge right say let us take this edge V 6 to V 7. Now what are the possibilities of this V 7 sorry V V 7 it can schedule in 2 or 3 right. So, V 2 into 7 2 it will give you the time step or start time of this operation 7 or 3 into 7 3 either it will start in time step 2 or time step 3.

And for 6 it can start either in here or here because there is a mobility upon so 6 1 1 into 6 1 and 2 into 6 2. So, this is what this part is nothing, but this part is nothing, but this and since I have only one single time step so this is one so, I just put everything this side.

So, this is 2 into 2 x 7 2 plus 3 into x 7 3 minus 6 x 1 this is moved here minus 2 6 2 minus 1 is greater than equal to 0. So, this is for all edges similarly for all other edges also you can find out this is kind of constants right. So, these are listed on the non trivial one, trivial one like this is you can find out.

(Refer Slide Time: 40:26)



Example 1 (cont'd.)

- Resource constraints

MULT a1=2

$$x_{1,1}+x_{2,1}+x_{6,1}+x_{8,1} \le 2$$
$$x_{3,2}+x_{6,2}+x_{7,2}+x_{8,2} \le 2$$
$$x_{7,3}+x_{8,3} \le 2$$

ALU a2=2

$$x_{10,1} \le 2$$
$$x_{9,2}+x_{10,2}+x_{11,2} \le 2$$
$$x_{4,3}+x_{9,3}+x_{10,3}+x_{11,3} \le 2$$
$$x_{5,4}+x_{9,4}+x_{11,4} \le 2$$

$$\sum_{i:T(v_i)=k} \sum_{m=l-d_i+1}^{l} x_{im} \le a_k, \quad k=1,2,\ldots,n_{res}, \quad l=1,2,\ldots,\bar{\lambda}+1$$

28

NOP
NOP
λ+1
λ+1

So, this is the set of constant for precedence now, if I assume that I have multiplier two multiply on to ALU I have to put constant for multiplier and as also constant for this ALU right. So, and the constant is given by this so this is what the type. So, for multiple types since these are all single cycle I do not have that consider a multiple step I have just consider to present state.

Here the number of multiplier executive in time step is this 1. So, this x 1 1, x 2 1, x 6 1, and x 8 1 1 should be less than equal to so that so this actually ensure that all 4 will not be scheduled in time step 1 because your start time only out of this 4 operation only two of them can start at time step 2 because I have 2 multiplier.

So, this is what is this constant signifies similarly if you just go to time step 2 this 3, 2; 6 7, and since this has a mobility this also can executed here. So, 6, 2; 7, 2, x 3, 2 6, 2, 7, 2 and also this also can execute because this is there is a mobility. So, x 8 2 should be less than equal to t. So, this combining this to ensure that not all of them will executed here some of they will move back some operation will move down so that all these resource constraints and satisfied.

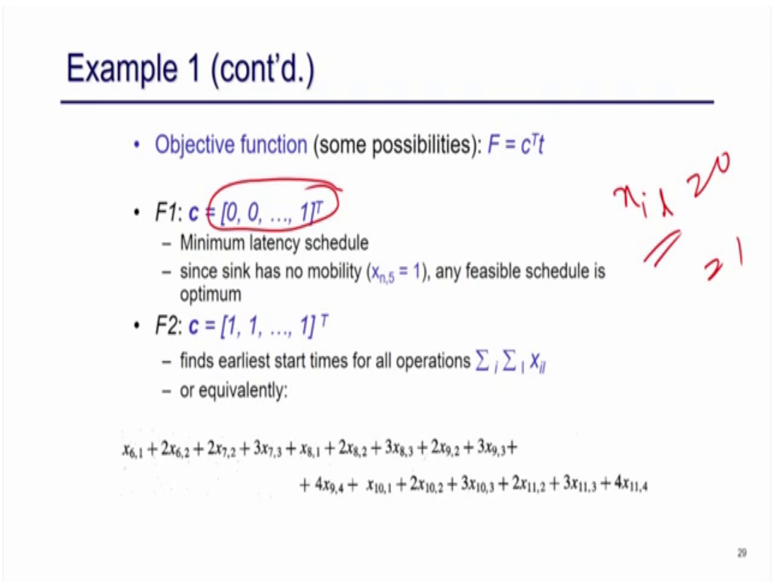
Similarly for ALU we can say that this operation has to be operation in this operation only this can execute. So, only this 10,1 less than equal to 2 and operation time step 2 this can execute this can execute these also can move down. So, this 3 can execute so x

9, 2, x 10, 2, and x 11, 2 should be less than equal to 2 and this way we can actually find out the constant for each ALU.

So, how many constant will be here for each multiplayer I have number for each time step I have a constant. So, there are lambda plus 1 number of time step into multiplier so this is our number of constant for multiplier and lambda plus one number of constant for adder. So, I this a number of resource constraint we will get from this resource constant.

So, you can understand now I found I have find this these are the set of constant, this is the dependency constant, this is my the precedence constants. And this is the resource constant.

(Refer Slide Time: 42:33)



Example 1 (cont'd.)

- Objective function (some possibilities): $F = c^T t$
- F1: $\mathbf{c} = [0, 0, \ldots, 1]^T$
  - Minimum latency schedule
  - since sink has no mobility ($x_{n,5} = 1$), any feasible schedule is optimum
- F2: $\mathbf{c} = [1, 1, \ldots, 1]^T$
  - finds earliest start times for all operations $\sum_i \sum_l x_{il}$
  - or equivalently:

$$x_{6,1} + 2x_{6,2} + 2x_{7,2} + 3x_{7,3} + x_{8,1} + 2x_{8,2} + 3x_{8,3} + 2x_{9,2} + 3x_{9,3} + \\ + 4x_{9,4} + x_{10,1} + 2x_{10,2} + 3x_{10,3} + 2x_{11,2} + 3x_{11,3} + 4x_{11,4}$$

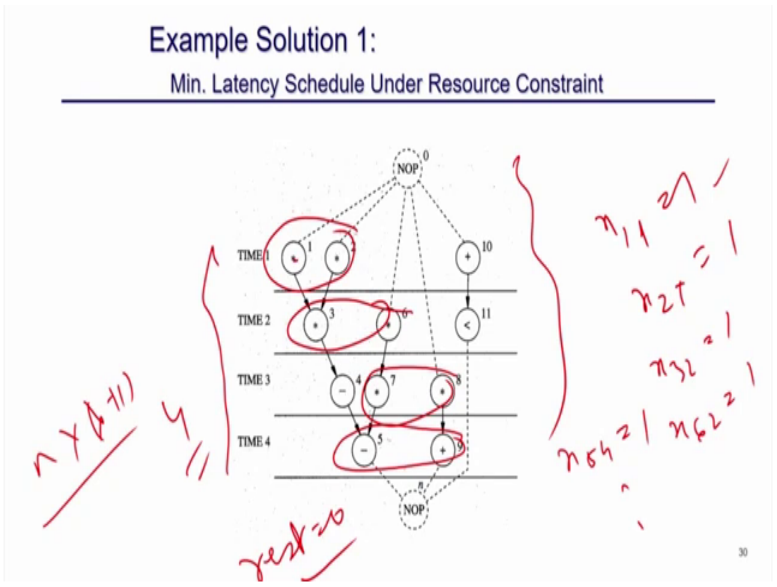
$x_{il} \geq 0$

$// \leq 1$

29

And if I solve this using this value the value of all this x i will get so some of this x i l value will be 0, and some will be 1. So, this x value will give you the starts time step of each operation which is nothing, but the scheduling of that operation right.

(Refer Slide Time: 42:49)

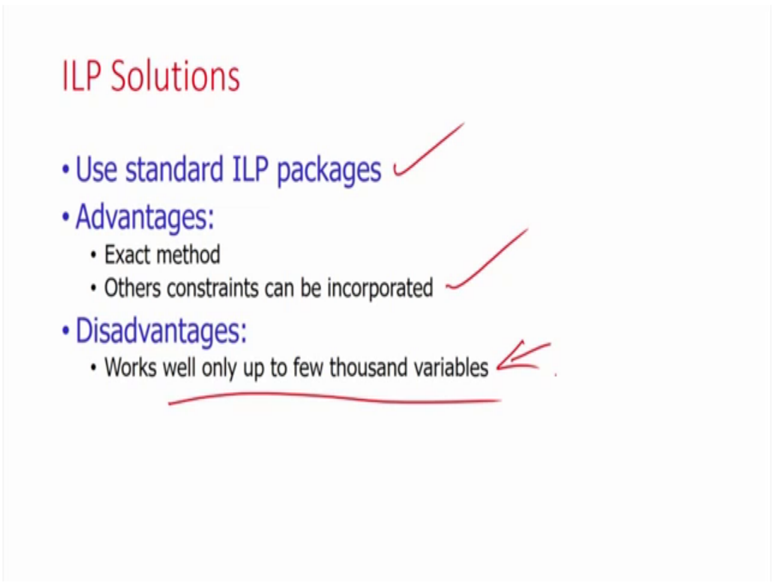

Example Solution 1:

Min. Latency Schedule Under Resource Constraint

So, if you just go into this is the solution I am going to get if you solve all this constant you can see here I am actually that optimum solution that we have discussed in the last class we are getting that because I have 2 multiplier, 2 multiplier here, 2 multiplier here. So, I am able to execute everything in 2 multiplier, and 2 adder still I am i am getting 4 time step which is the minimum possible value. So, this is if you just add these 3 constant and if you do to some solver you should give you this solution right.

So, for this example now, this x 1 1 is 1, this is x 2 1 is 1 for this operation x 3 2 is 1, for this x 6 2 is 1. So, for this operation say x 5 4 is 1 you can see I I defined that n into lambda number of variables out of this only this 11 for each operation only one of them will be going to rest are be 0 right. So, this way and rest are 0 right so, this is how the solution that solver will give you the solution these are the variable 1, and the rest of the variables are 0, right this is what will get you understand right.

(Refer Slide Time: 44:00)

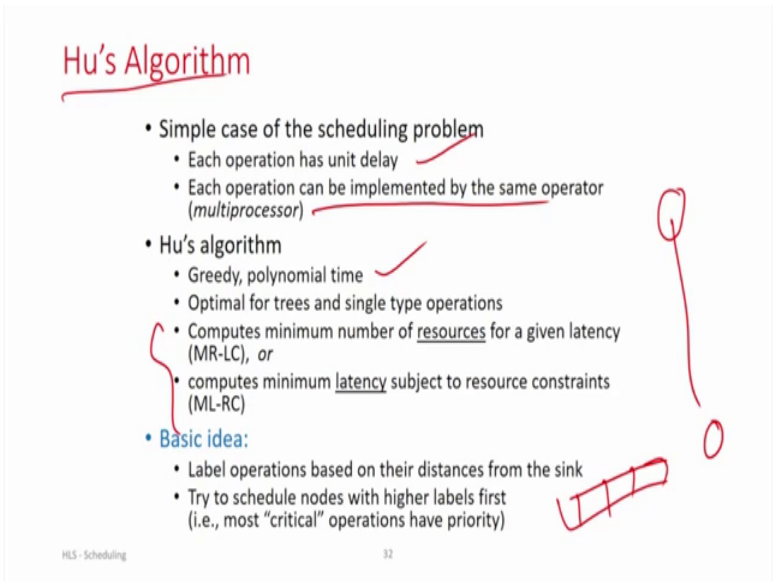


ILP Solutions

- Use standard ILP packages
- Advantages:
  - Exact method
  - Others constraints can be incorporated
- Disadvantages:
  - Works well only up to few thousand variables

So, now, if you just so, ILP solution advantage is that you can use their standard ILP solution packaged l p solvers or there is some solvers and we can get a exact solution right. So, we will get a exact solution, but the problem is and also another element is that for a resource we add some constant, but we can also add some other kind of constant into the framework and it will still solve the problem because this is a constant solver. But the problem is that it is not scalable if you have say very in a large behaviour this is may not homework because this is may not walk for a big example. So, right big test cases so that is the problem with the I ILP solution. So, I will now move on to this next category of algorithm which is heuristic base.

(Refer Slide Time: 44:38)



Hu's Algorithm

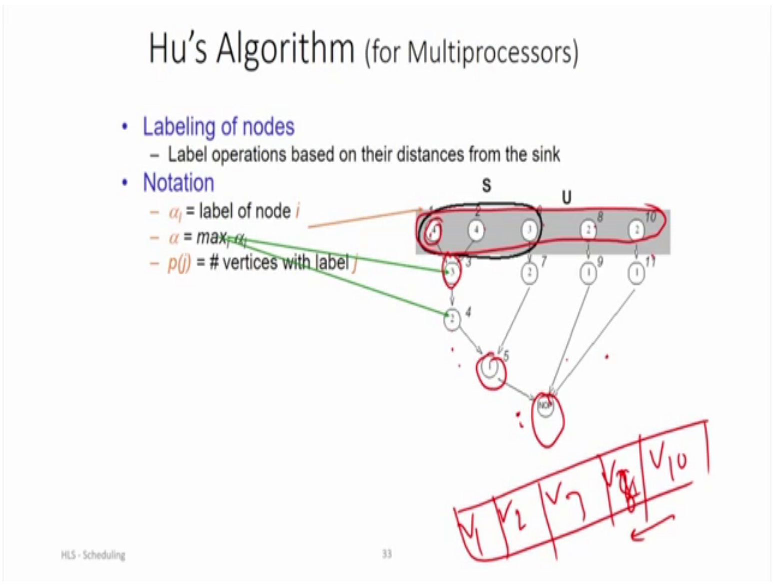
- Simple case of the scheduling problem
  - Each operation has unit delay
  - Each operation can be implemented by the same operator (*multiprocessor*)
- Hu's algorithm
  - Greedy, polynomial time
  - Optimal for trees and single type operations
  - Computes minimum number of resources for a given latency (MR-LC), *or*
  - computes minimum latency subject to resource constraints (ML-RC)
- Basic idea:
  - Label operations based on their distances from the sink
  - Try to schedule nodes with higher labels first (i.e., most "critical" operations have priority)

HLS - Scheduling 32

But so before that I am going to talk about this Hu's algorithm which is there are some simplification assumed here that each operations are you need delay; that means, what that each operation has to be executed in one cycle and all operations are same type. So, in practical preferences all will be multiplier, or all will be adder, but you can understand in practice not all operations are of same time right so; that means, it does not actually applicable to the scheduling time problem, but this is the ah the idea or the idea that is given by this algorithm will be taken in the heuristic solution like list scheduling.

So, you will just discuss here so this is a greedy, and polynomial time solution. So, it will give a polynomial time solutions and the both can be this ML R C and ML R C both can be solve using this algorithm and basic idea is that it maintain a value for each node the distance from the sink node right.

(Refer Slide Time: 45:38)



Hu's Algorithm (for Multiprocessors)

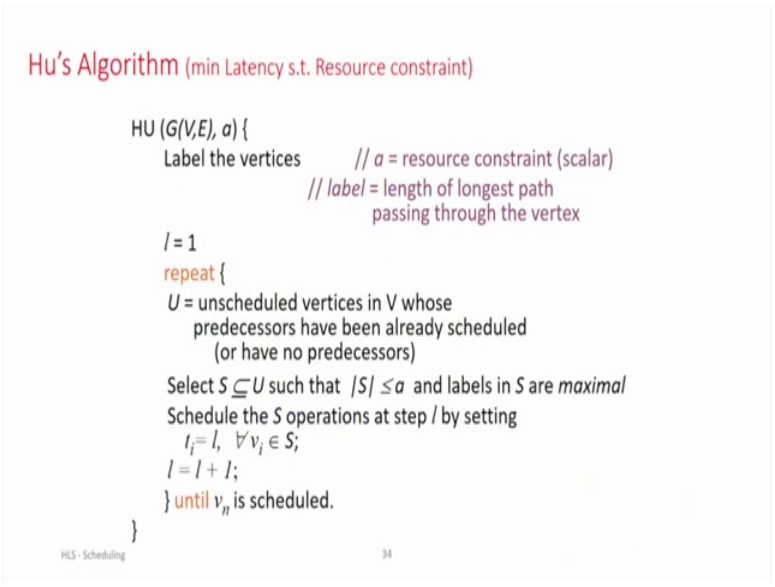
- Labeling of nodes
  - Label operations based on their distances from the sink
- Notation
  - α_i = label of node i
  - $\alpha = \max_i \alpha_i$
  - $p(j)$ = # vertices with label j

So, now I just consider same difference in because solve an example, but I assume that all operations are now, of same type so all are adder I now right. So, now, I just this so this is my sink node I am just trying to find out the distance which is nothing, but a critical path length right.

So, distance of this node is 1, distance node is 2, distance of this node is 3, distance of this node is 1, 2, 3, 4, for this node also 4, this is 3, because 1, 2, 3, 1, 2, this is 2, this is 1, 2, this is 2. So, I assume this values for each operations right and in maintain a priority list ok. It maintains a list of operation which is actually now available or can be scheduled and based on the resource operations it select the operations which has the highest priority and the priority value is this distance value this 4, 4, 3 and it has a max is so, the variable which has the more value which is more ah which has a higher priority and the other operations.

So, it maintain that list so, initially since all these operation depends on the input these are all available right. So, this all 5 operations are available and they are there they are put in priority list since this operation is 4. So, 1 this is 2, and this is this is 3, this is 8, and this is 10, the operation V 1, this is V 2, this is V 3 this is V 8, and this is V 10. So, this will be the priority list order because these are the highest priority. Now, if you assume I have only 3 resources so I am going to select the top 3 priority right. So, this is what I am going to do.

(Refer Slide Time: 47:13)



Hu's Algorithm (min Latency s.t. Resource constraint)

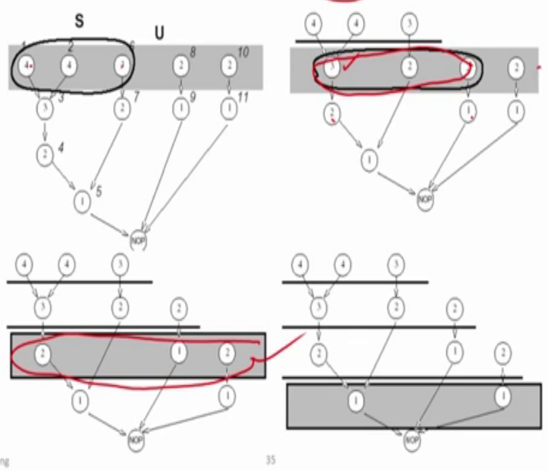
```
HU (G(V,E), a) {
    Label the vertices          // a = resource constraint (scalar)
                                // label = length of longest path
                                   passing through the vertex
    l = 1
    repeat {
     U = unscheduled vertices in V whose
         predecessors have been already scheduled
           (or have no predecessors)
     Select $S \subseteq U$ such that $|S| \leq a$ and labels in S are maximal
     Schedule the S operations at step l by setting
        $t_i = l, \ \forall v_i \in S$;
     $l = l + 1$;
    } until $v_n$ is scheduled.
}
```

HLS - Scheduling 34

So, this is the algorithm I am not going to detail.

(Refer Slide Time: 47:15)

Hu's Algorithm: Example (a=3)



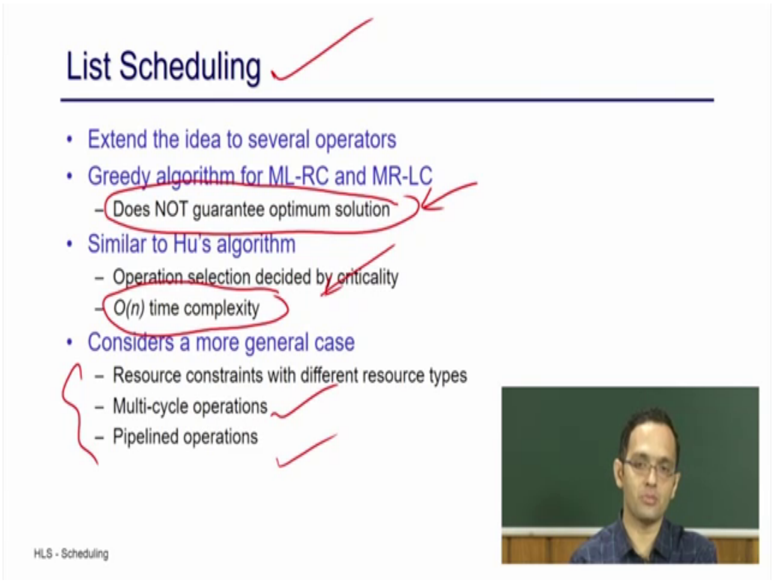
HLS - Scheduling 35

But if you just have a working of this algorithm here. So, initially all this in the priority list I and say assume I my resource is three. So, I am going to select these 3 because they have the highest priority compared to these 2 right. So, now if you just after that these 3 are done so my priority list will be updated with this so all the operations that are succeeding this operation will be now added and now the priority lists have this right so ordered will be this way right.

Now I am going to select these 3 because they are the highest priority I could have select this instead of this works fine, but this has to be selected. So, these are not after that then these two steps are done then the next set of priority is these this and this. So, this is what their and I have 3. So, I can schedule all of them and the next time. So, I can schedule of this 4 right. So, this is how the whole algorithm.

So, basic idea is that we calculate some priority value of each node and then we try to schedule the operation because the highest priority based on the ability of the resource right this is how the algorithm works.

(Refer Slide Time: 48:15)



List Scheduling

- Extend the idea to several operators
- Greedy algorithm for ML-RC and MR-LC
  - Does NOT guarantee optimum solution
- Similar to Hu's algorithm
  - Operation selection decided by criticality
  - O(n) time complexity
- Considers a more general case
  - Resource constraints with different resource types
  - Multi-cycle operations
  - Pipelined operations

HLS - Scheduling

So, now we will move into this list scheduling algorithm which is nothing, but then this that is another person of this Hu's algorithm. Here this is also this ah, but here you can have multiple type of resource instead of only multiplier or on only adder you can a multiplier adder subtract all kind of resource can be available and your operations can be pipeline, multi cycle, or other various is also possible so and we have some resource constant right.

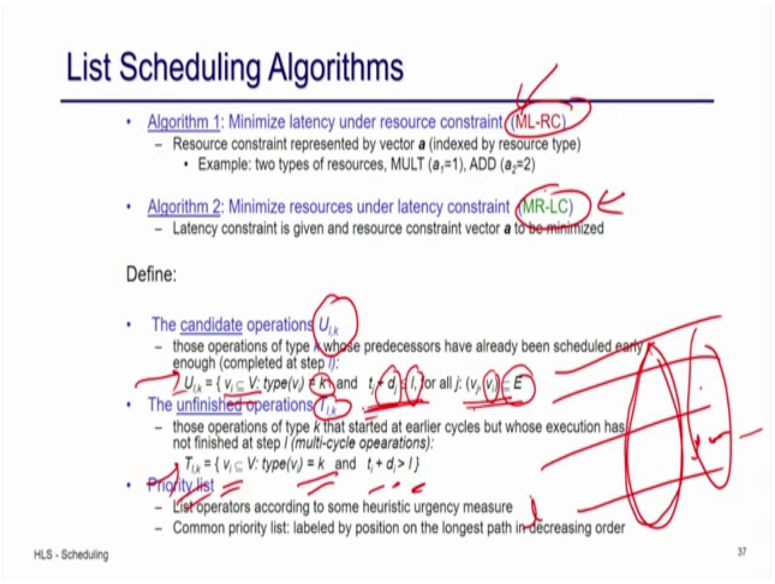
So, we can have both algorithm again this ML RC that it may be minimize resource r minimize latency, but you are going to discuss that minimize latency and the resource constant the other can be understood from this discussion ok. But here the object the algorithm is that this particular solution does not guarantee to optimum solution because this is a heuristics algorithm it does not guarantee that you will always get the optimum solution out of this.

But this is something very fast because this order of n time algorithm you understand from the complexity of exponential here I have no polynomial time algorithm linear algorithm. So, it will very fast even if my solution is degraded by 10 percent, or say 5 percent is fine because I am able to do this very first.

For example, say I have a very big very operations big a and a big behaviour that ILP give you solving 1 hour and this is can solve it say 2 2 seconds. So, I will always go for this is because this is something is giving very fast and in fact, industry most of the industrial tool actually use least scheduling to execute this will do the scheduling in their internal tool so this is something is very popular algorithm which is you widely used ok.

And as I mentioned it can actually work for different kind of resource multi cycle supports multi cycle pipelined operation all can be supported their only thing is that it does not give a optimal solution always, but give you a very good solvents most of the time. So, this is something which we will take it because that is something is useful because sometimes I have to finish everything on time. So, time is important design cycle has to be is minimized so list scheduling is widely used ok. So, we will discuss on that.

(Refer Slide Time: 50:20)



List Scheduling Algorithms

- Algorithm 1: Minimize latency under resource constraint (ML-RC)
  - Resource constraint represented by vector **a** (indexed by resource type)
    - Example: two types of resources, MULT (a_1=1), ADD (a_2=2)
- Algorithm 2: Minimize resources under latency constraint (MR-LC)
  - Latency constraint is given and resource constraint vector **a** to be minimized

Define:

- The candidate operations $U_{l,k}$
  - those operations of type k whose predecessors have already been scheduled early enough (completed at step l):
    $U_{l,k} = \{ v_i \subseteq V: type(v_i) = k$ and $t_j + d_j \leq l$, for all j: $(v_j, v_i) \subseteq E$
- The unfinished operations $T_{l,k}$
  - those operations of type k that started at earlier cycles but whose execution has not finished at step l *(multi-cycle opearations)*:
    $T_{l,k} = \{ v_i \subseteq V: type(v_i) = k$ and $t_i + d_i > l \}$
- Priority list
  - List operators according to some heuristic urgency measure
  - Common priority list: labeled by position on the longest path in decreasing order

HLS - Scheduling 37

We will mostly discuss that M L R C problem; that means, minimize leads to latency under resource constant and we will we can have the different versions like this minimize resource under latency constant, but we are not going to cover it in this discussion ok.

So, we have 2 kind of variables first kind of variable is U l k; that means, those operations those presidents are operations are already executed right so; that means, this is set of operations who is of type k because I have to now think about each type at a time multiplier, adder, differently we will not go to merge them. So, we are going to schedule multiplier we are going to early schedule multiplier when you are going to schedule or adder we are going to only schedule adder and so on.

So, we are going to consider set of operations V i which is stored in this u l k variable this set where all this predecessor operations are already execute. So V i to V j so these are the set of operations V i all these predecessor operations those are arrangement these are the predecessor operations they are already executed. So, for this d j, this d j is the number of time steps required so this is less than l.

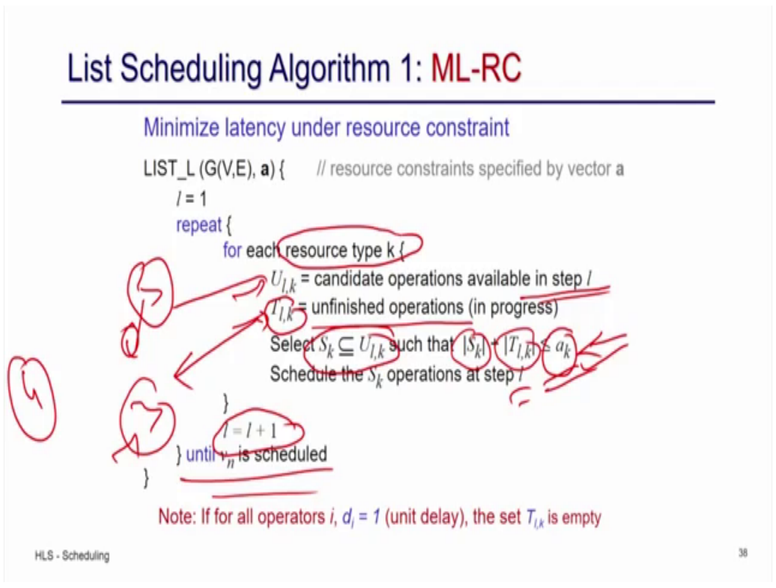
So, this is basically to say the number of operation whose predecessors even use a multi cycle or pipeline they are already executed and these operations set are ready can be scheduled now right. So, this is the priority list, the priority operations that can be executed.

Similarly I have a set operation a T l k; that means, unfinished operations so; that means, those the operation that are running; that means, they are started early, but they are still continuing in this l l-th time step. So, we just see that set of variable V the operations V of type k they are still this t i plus d i greater than n; that means, they are still they have started somewhere early. So, this is my say time stable they have started somewhere here or here, but they are still continuing here right so these are the set of operation that are running here.

And what is the important here is that I mentioned early also these operations also need resources in this time step l and these are the most important they have to be scheduled because they are already started and we cannot start another operation by passing this operation. Because they have started they has to complete then I then only I can start other operations and I can only start other operation see after I mean scheduling all these such operation that are actually active in this time step some other resources are available.

So, the idea is that in a particular time step these are the highest priority operation they have to be scheduled and if other resources available I will take it from here based on some priority value and priority value can be same as the Hu's algorithm the critical path from the sink node ok. So, this is the l list scheduling algorithm.

(Refer Slide Time: 53:00)



List Scheduling Algorithm 1: ML-RC

Minimize latency under resource constraint

```
LIST_L (G(V,E), a) {     // resource constraints specified by vector a
    l = 1
    repeat {
        for each resource type k {
            $U_{l,k}$ = candidate operations available in step l
            $T_{l,k}$ = unfinished operations (in progress)
            Select $S_k \subseteq U_{l,k}$ such that $|S_k| + |T_{l,k}| \le a_k$
            Schedule the $S_k$ operations at step l
        }
        l = l + 1
    } until $v_n$ is scheduled
}
```

Note: If for all operators i, $d_i = 1$ (unit delay), the set $T_{l,k}$ is empty

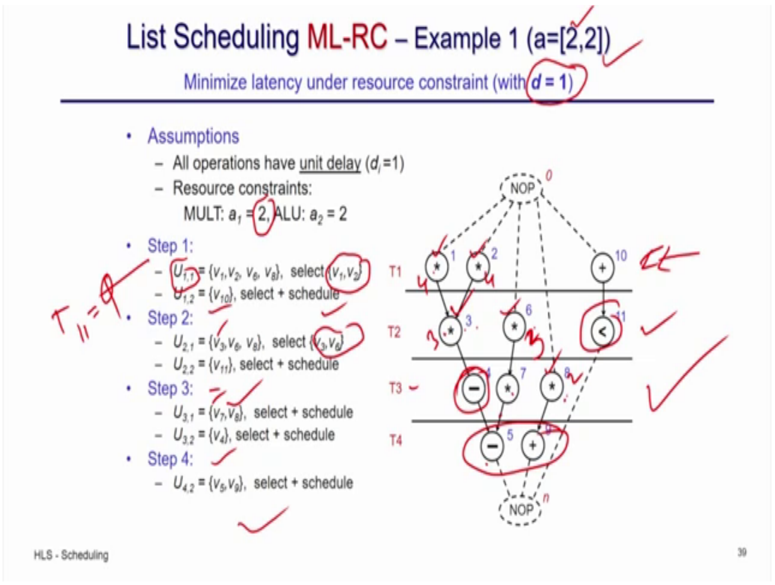
So, I am now going to do it for each resource type right source a multiplier adder and one and this is U k is the set of operations that are available right. So, the ready list; that means, these operations are ready their processors are done and T l k is the set of operations that are unfinished; that means, they are in progress in this particular time step. So, I have to schedule all this T l k plus some sub set of U l k based on the such that this T l k plus S l S k is less than a k; a k is the resource bound right.

So, I will just select so suppose my T l k has a 3 operations and U l k as a 5 operation, but I mind your multiplier is only 4. So, I will take these 3 and only one of them from here so that I can execute all these 3 plus one of them here right this is how the whole thing works and it ensures that this T l k is never greater than the a k because I i can I will not start any operation more than l l k any time step.

So, if I start from system step 1 I always ensure that with the number of running a operation in time step is more not more than that resource bound and then I am going to schedule all this operation time step and then I increase the time step by 1 right this is how the whole thing works until all operation are schedule ok.

So, this is something is called list scheduling as I mentioned again I will maintain this priority list of finished operation, in progress operations, I will also maintain a priority value which is nothing, but the distance from the sink node and I will schedule all the in progress operation first and if further resources are available I am going to schedule some other operations. So, let us take an example.

(Refer Slide Time: 54:35)



List Scheduling ML-RC – Example 1 (a=[2,2])

Minimize latency under resource constraint (with $d = 1$)

- Assumptions
  - All operations have unit delay (d_i=1)
  - Resource constraints:
    MULT: $a_1 = 2$, ALU: $a_2 = 2$
- Step 1:
  - $U_{1,1} = \{v_1, v_2, v_6, v_8\}$, select $\{v_1, v_2\}$
  - $U_{1,2} = \{v_{10}\}$, select + schedule
- Step 2:
  - $U_{2,1} = \{v_3, v_6, v_8\}$, select $\{v_3, v_6\}$
  - $U_{2,2} = \{v_{11}\}$, select + schedule
- Step 3:
  - $U_{3,1} = \{v_7, v_8\}$, select + schedule
  - $U_{3,2} = \{v_4\}$, select + schedule
- Step 4:
  - $U_{4,2} = \{v_5, v_9\}$, select + schedule

$T_{11} = \phi$

HLS - Scheduling

39

So, in this example you can see suppose this is my that differential equation solver again and I assume that my d 1 equal to 1 that is all are single cycle operations and I have 2 multiplier, and 2 adder so I have 2 multiplier, and 2 adder.

So, now what is U here at the start I start state because I start from one all these operation are available this V 1, V 2, and V 10 this sorry this V 1, V 2, V 6, V 8 and V 10 let us all are available ok. So, for sorry so this V 10 is something is a adder so I am going to only consider the operations or multiplier time first then the adder. So, I have only this 4 operation V 1, V 2, V 6, and V 8 are available they are actually U, they are in the U list because they are now I mean ready right.

So, their predecessors are single or which is scheduled and my T 1 1 is not because there is no operation is running now because I just start the operation thing right. So, I am going to select this operation because the distance of this node is 4, distance of this node is 4, this is 2, this is 3 sorry this is 3 and this is 2. So, I am going to since I have to multiplier I am going to select this and this so this is what I am going to select V 1 and V 2.

Because I have to select there is no T i j there is no running operation I am going to select all of them from this ah ready operations and I am going to select these two because they have the highest priority. So, once this is done by in the next step because this is a single cycle everything will be there the T j is not there because that is there is no running here this is t i i 1 2 1 2 2 everything will be 0 because there is no running a operation always if it is in the single clock.

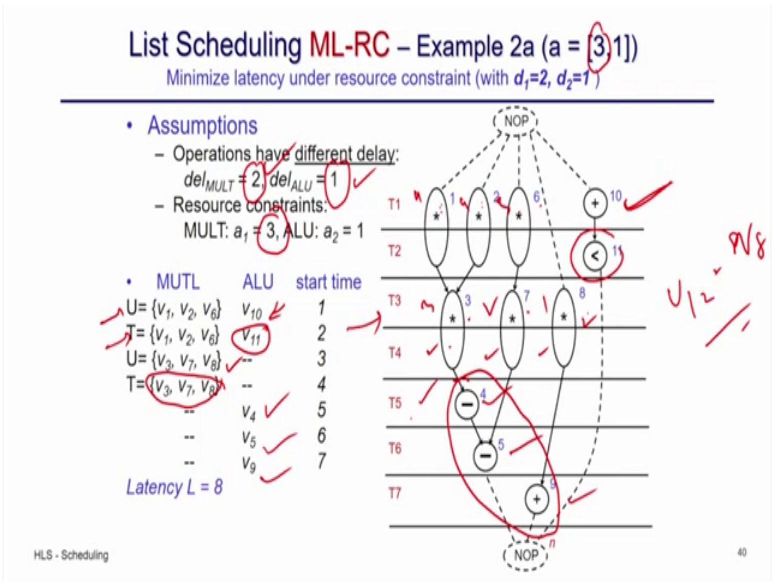
So now I have this once this is done this also available. So, my u will content V 3 V 6 and V 8 for this distance is 3. So, I am going to select this and this not this 1 because they have the highest priority so I am going to select 3 and 6 and after that in the next time so I am going to select 7 um, 7 and 8.

For adder here I am only one a operation is ready. So, I have V 10 and I am going to select it because there are 2 adder available and the next time step I have this only this is available not no other. So, I am only V 11, so I am going to schedule it in time step 2 and then V times step 3 I am now V 7 and V 8 because now V this this 3 and 7 is done so 7 and 8 it is now available and 2 resource.

So, I am going to select both of them and once this is done now I have this and this both are available let us say 4 and 9 4 and sorry sorry V 4 is only available this is not available yet because I am going to do it now.

I am going to do in time step 3 so, now I am going to be do before in the time step 3 and then finally, V 5 and V 9 because there are 2 adder, I was going to be scheduled in time step 4. So, this is what is going to give this list scheduling give you the solution and you can understand this is also optimal solution because I am adding 2 multiplier, and 2 adder which is also given by the this list scheduling.

(Refer Slide Time: 57:38)



List Scheduling ML-RC – Example 2a (a = [3,1])

Minimize latency under resource constraint (with d_1=2, d_2=1)

- Assumptions
  - Operations have different delay: del_{MULT} = 2, del_{ALU} = 1
  - Resource constraints: MULT: a_1 = 3, ALU: a_2 = 1

MUTL	ALU	start time
U= {v_1, v_2, v_6}	v_{10}	1
T= {v_1, v_2, v_6}	v_{11}	2
U= {v_3, v_7, v_8}	--	3
T= {v_3, v_7, v_8}	--	4
--	v_4	5
--	v_5	6
--	v_9	7

Latency L = 8

HLS - Scheduling

40

Now, I will just consider the different variation of this suppose I may have a multi cycle operations. So, all operations are 2 cycle and again I have only 2 multiplier and 1 ALU; not 2 ALU and now why I want to see how it can happen. So, initially as I mentioned this 1 3 so for multiplier this 1, 2, 3, 4, this V 1, V 2, V 6, and V 8 are available ok.

And I am going to schedule this because they have the highest priority this is distance 4, this is distance 4, this is 3, this is 2, I am going to select only this 2 V 1, and V 2, right. So, V 1 sorry I am sorry so I have to 3 multipliers. Now, so I have 3 multipliers and the delay of the multi cycle or the multiplier is the 2 cycle operations and ALU is a 1 cycle operation.

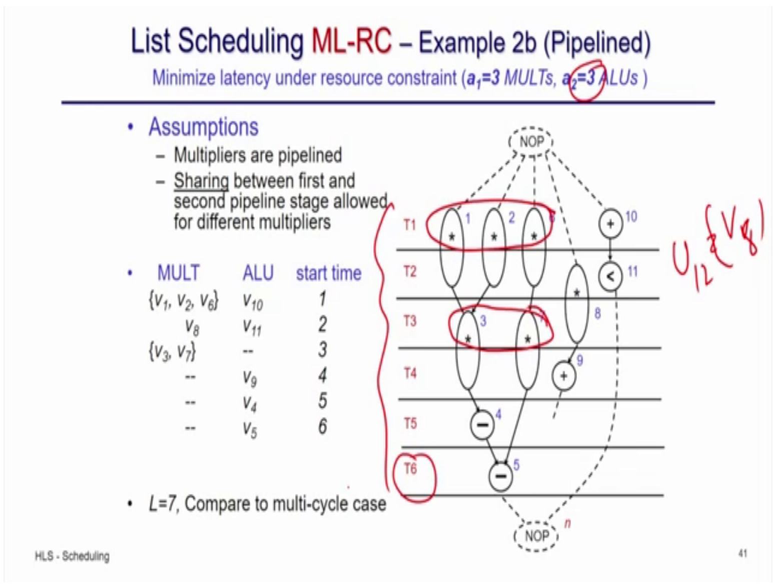
So, in this time step 1 I am going to select all this 1, 2, and 6. So, this will be progress and this step for adder I am going to select this because I have only 1 operations. So, in the next time step this T T now content, this V 1, V 2 this because they are all are running. So, even if my U content this V 8 that finished operation this operation type 1, time step 2, a V 8 I cannot schedule it because all are running here.

So, even if I have some ready operation I cannot schedule it because they all are running. So, in the time step 2 there is no operation executed only this operation 3 will continue the multiplier and since this is already done in time step 1 I can do this V 11 here right. So, this is what I have done.

So, in time step 3 and now, all these operation fields right. So, my T list is null, and once they are finished in my U list this V 3, V 7, and V 8 all out there and the distance this is 3, this is also this is 2, and this is 1, but I have 3 multiplier I can start all of them right so 3, 7 and 8 are starting in time step 3.

In time step 4 I have to carry out this because I cannot do anything else. So, in the time step 4 this 3 will be executed and no other operation addition operation will be there because these are all pending until this is done I cannot start then after that and V is 5 once the T 4 they are executed execution is complete in T 5 I can do this, T 6 I can do this and T 7 I can do this. So, I need 7 times step to execute these operations where my multiplier are 2 cycle operations and I have 3 multiplier and this is also the list scheduler will give you the solution.

(Refer Slide Time: 60:07)



List Scheduling ML-RC – Example 2b (Pipelined)

Minimize latency under resource constraint (a_1=3 MULTs, a_2=3 ALUs)

- Assumptions
  - Multipliers are pipelined
  - Sharing between first and second pipeline stage allowed for different multipliers

MULT	ALU	start time
$\{v_1, v_2, v_6\}$	v_{10}	1
v_8	v_{11}	2
$\{v_3, v_7\}$	--	3
--	v_9	4
--	v_4	5
--	v_5	6

- L=7, Compare to multi-cycle case

HLS - Scheduling

41

And if you know the pipeline one just briefly I will talk about so say multiply there are 3 multiplier, and there this multiplier is 2 stage by pipe lining. So, again now you can start this 3 because there are 3 multiplier available in the next time step I could have start another 3, but I have only one multiplier is ready let my V t this u list only content V 8 1 2 contain only V 8.

So, I can I can only start this as a pipelining with any of them right. So, after time step 3, so, after in time step 2 these all 3 are finished this is continuing. So, and there are 2 operations are now available so I can actually schedule all of them because I have 3 multiplier so I can just start all these 3 in the time step 2 and so on. So, this is how I can if I do this pipeline manner I will take 6 times to execute this where I have 3 multiplier available and this all multiplier are pipeline 2 stage pipeline algorithm ok.

So, with this I finished this discussion on scheduling as I mentioned all this scheduling algorithms are NP complete so exact solution is not scalable we can have ILP a solution. But smaller this always give you the optimal solution you do not have to worry about this, but we usually prefer the heuristic ways algorithm because that is something very first polynomial time under up n is equally and that also give a very good solution most of the time right so this is something in the scheduling part.

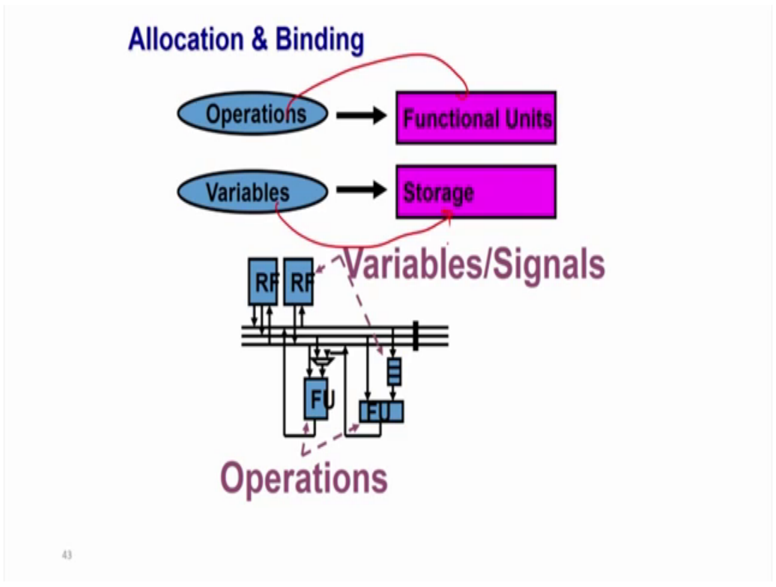
(Refer Slide Time: 61:30)

ALLOCATION and BINDING

42

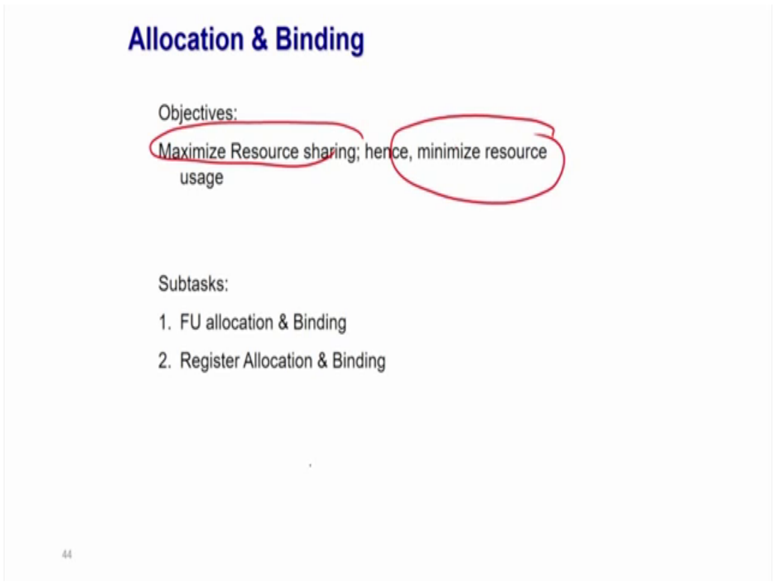
So we will now try to talk about this allocation and binding there are automations of allocation and binding. So, as you mentioned in this allocation and binding we try to map those operation into function unit and variables to the registers right so this is an try to optimize.

(Refer Slide Time: 61:39)



Allocation & Binding

(Refer Slide Time: 61:45)



Allocation & Binding

Objectives:

Maximize Resource sharing; hence, minimize resource usage

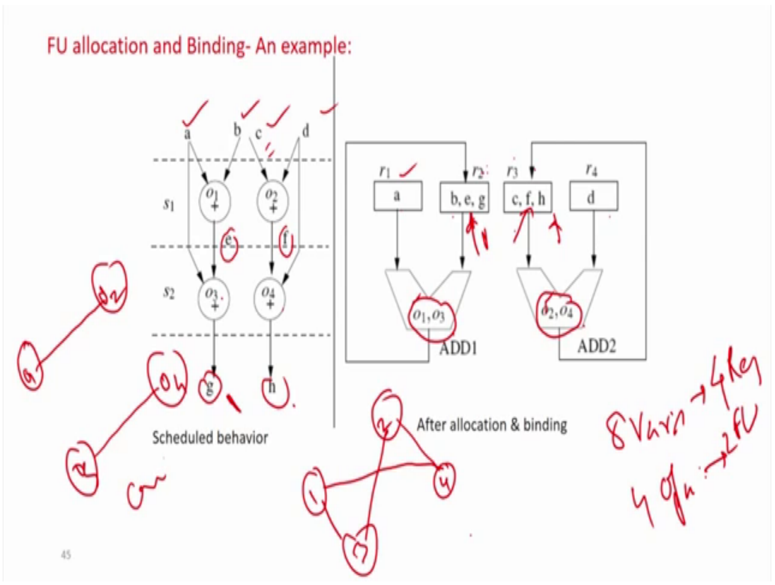
Subtasks:

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

44

And our objective is something try to use maximize the resource sharing; that means, use minimum number of resource right ; that means, minimum number of registers, minimum number of function units. So, that is our objective ok.

(Refer Slide Time: 61:56)



FU allocation and Binding- An example:

So, let us take an example so suppose this is my the schedule generated by some scheduling algorithm where operation 1 and 2 scheduled in time step 1 operation 3 and 4 scheduled in time step 2. So, what will the optimum hardware here I can you I have to use 2 and all are say type adder so I need at least 2 adder to do this because this 1 and 2 is running in time step 1.

So, I can schedule o 1 here, o 2 here, and now in the time step 2 I can do o 3 in this time here itself because this is now free and I can do this in this right. So, I can schedule o 1 and o 3 in the adder one and o 2 and o 4 into this I need 2 adder. So, this is a minimum number of resources possible for this.

Similarly for registers I need at least 4 registers for throwing this 4 variable so I need say r 1, r 2, r 3, and r 4 and a is store here in r 1, b is store r 2, c store in r 3, and d store is r 4. Now after time step 1 e is available, a still using so this is still we need a, but b is not required. So, I can store e in b, that I can store in the same register is r 2 b e right.

Similarly here d you are going to use here, but c is not required. So, I can store f in the same register at c that is why I store f here. So, after this again once this ready and this my g and h variable is required, but again this all e and f is dead now I do not need so I can store again g in this same register r 3 and h in this register.

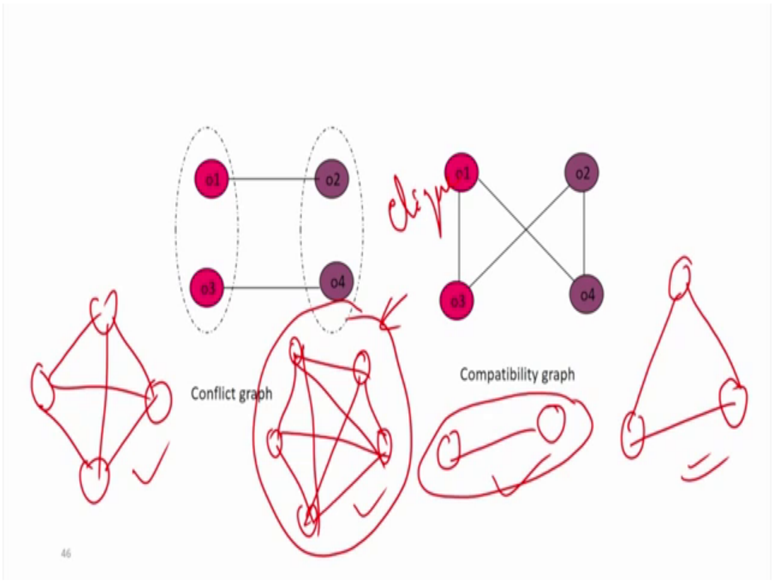
So, I need 4 register minimum to store this. So, I have 1, 2, 3, 4, 6, 7 sorry 4, 5, 6, 7, 8, 8 register, 8 variables, I can store them in 4 registers. I can store them in 4 registers, and I have 4 operations, I can execute it 2 f u right, so that is what is the minimum thing?

So, now the question is how can automate this particular process right. So, to do this we try to find out some conflict graph or and or compatibility graph let us find out that. So, what is conflict graph so I will do the same thing for a f u register is the same way we can do. So, so we will just talk about that a f u allocation and binding register in the same process which will follow the same kind of algorithm ok.

So, what is the conflict here so o 1 and o 2. So, for each operation I will draw a node ok. So, this is my operation 1 1 this is o 3, 2, this is o 3, and this is o 4 ok. And now 1 and o 2 has a conflict; conflict why because there are going to be a schedule in the same time step so, I need I cannot do that with the same adder right so I have a conflict between them sign an h.

Similarly o 3 and o 4 has a conflict, but there is no conflict between o 1, and o 2, and o 3 to o 2. Similarly o 4 to o 2 and o 4 to o 1 so I have this is the conflict graph ok. So, this is what is conflict graph ok.

(Refer Slide Time: 65:02)



Similarly I can construct a compatibility graph what does it means which operations are compatible to each other level they can be shared a same resource ok. So, to draw the compatibility graph out of it I can try that again so this is say operation 1, this is 2, this is 3, and this is 4.

So, operation 1 is compatible with 3, and 4, because they are in different cycles. So, 3 or 4, operation 2 is compatible in 3 and 4 right 3 and 4 and operation 3 is compatible to 1 and 2 and 4 is operation common so this is this graph I am going to get this is compatibility graph right so this is the compatibility graph I will get if I draw this right.

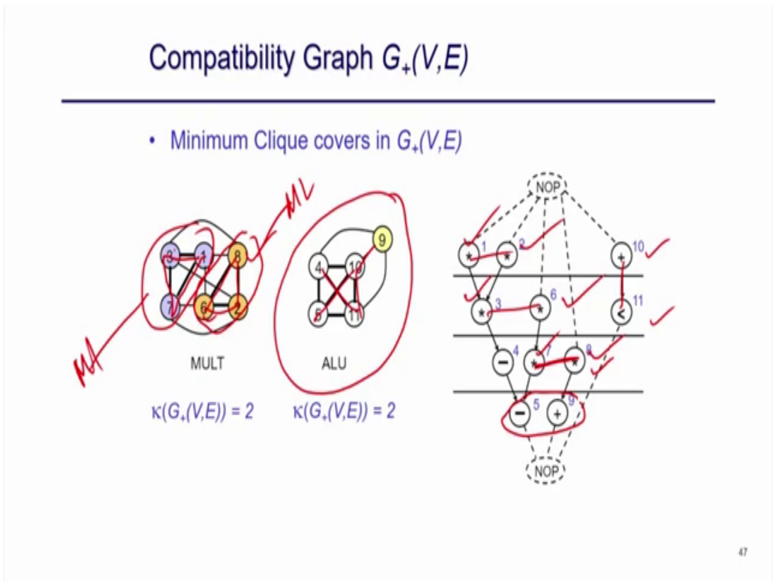
So, once so from this schedule behaviour you can either construct conflict graph or compatibility graph and then what does it help? How does it help? For a compare compatibility graph we try to. So, what does it mean if there is age means they are compatible they can be scheduled in one time step in the same sorry they can be scheduled using same adder or same resource right.

So, if I have this then they can be scheduled in this right. Similarly if there are 3 nodes and they are all compatible to each other they can be solved also so can be scheduled into the into the same resource they can be executing the same resource. But if there are 4 nodes and they have compatible to all each other then also this all 4 can be scheduled into the same they can be executed in the same resource.

Similarly if I have 5 node if I can find such node where all are basically compatible to each other this is nothing, but a complete graph. So, what I am drawing here is nothing, but compared complete graph of node 2, this is complete graph of node 3, this is node complete graph of node 4, this is complete graph of node 5. So, if you can find such component; that means, they are all compatible to each other; that means, they can share the same resource then the same that particular all these things can be may appear same resource.

So, in the compatibility graph we have and this is is a complete graph subset and this is called clique ok. Clique is one complete sub graph of a big graph so in that be the original rather may be say 100 nodes and out of that I find 4 nodes they are compatible to each other so that is a clique, maximum sub graph, complete sub graph of a big graph right.

(Refer Slide Time: 67:38)


Compatibility Graph $G_+(V,E)$

- Minimum Clique covers in $G_+(V,E)$

So, this is nothing, but once you have that come compatibility graph this is nothing, but you have to find the all cliques of this your compatible graph and your compatible this clique each clique will be mapped with same resource right. So, this is kind of a clique partitioning problem. So, our minimum clique cover problems so either click partitioning you will see you try to cover all your graph using some cliques so that all of your nodes are covered right.

So, for example in this for this graph if you comply to complete the so construct the compatibility graph for multiplier this will be the compatibility graph and if you compare construct the multiply compatibility graph for this adder this will be the compatibility graph ok.

I will just explain for this 3 operation 3 here it is not compatible in 6, but it is compatible to any other multiplier right. So, for 3 it should be from to 1 to 7 and 8 from 3 there is not from 1, 2 sorry 7 8 and 2 so 3 is compatible to all other operation except 6. Similarly for one this is not compatible to 2, but it is compatible to any other operation so this is how you are constructs this and for each type have to do this because I am not going to club multiplier adder in the same resource right.

So, similarly for this ALU, so these are all scheduled in different time step. So, all are scheduled only these are not compatible 5 and 9 so all other are compatible. So, this 10 is compatible to all others right. So, from 10 you have h 2 h to 11, h to 11 h to 5 and 9 from 10 to 5 and 9 and then I have also h to this 4 right so it is connected to these.

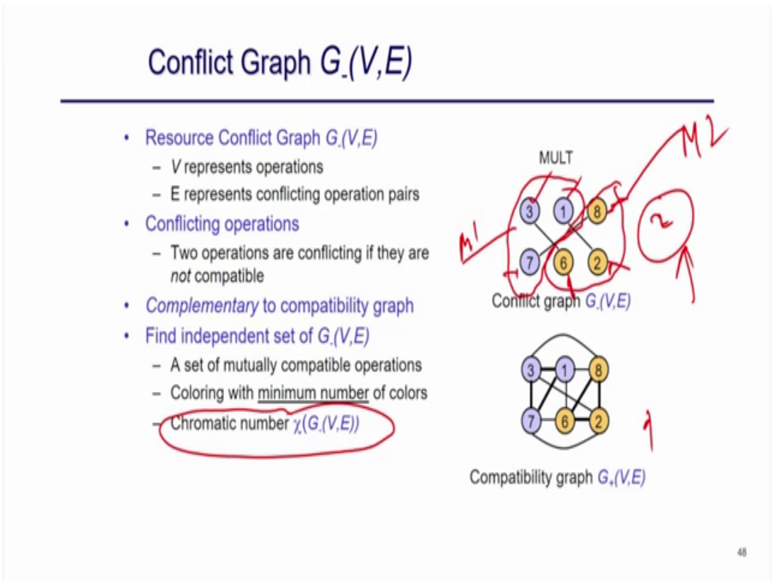
So, this way I can actually construct the compatibility graph and then you have to apply this clique, but it is a clique cover problem which is very well known graph graph problems. So, you can actually apply this to find out the minimum number of cliques that is will covered the whole graph and each clique is nothing but one resource right. For example, here you can find it here is so in this graph you can see that these 3 3 node because all are multiplied.

So, this is a this is a complete graph I am only a complete graph of size maximum size 3 there is no complete graph of size 4 here because if you add one of them if you add this to this node is not there right so I cannot have size 4 clique here. So, I have 2 clique, so, this is 1 clique, this is 1 clique. So, this will be multiplier 1, this is multiplier 2.

So, I need 2 multiplier to this, to execute and we can understand in multiplier 1 I am going to map 1, 3, and 7, 1, 3, and 7. In multiplier 2 I am going to map 8, 6, and 2, 2, 8 and 6, this 3 I am going to execute in 1 and this going to execute 1. So, this is how I am going to solve this problem so the one approach is that you construct the compatibility graph and you apply this clique covering problem to find out all the cliques and each clock clique is mapped to a single resource and all the operation will be can be executed in that particular resource ok.

This is something we can do in one way and conflict graph is something as I mentioned if these 2 operations are this is the reverse graph of this reverse graph is means what the you just remove all the edges here and you add all the edges that is missing here so that is will give you the ah that conflict graph right. So, here also I can construct from this behaviour I have a edge in the conflict graph 1 to 2 right, I have edge from 1 to 2 and that this 3 to 6 right. So, I have only 3 to 6 and 7 to 9 because they are all executed I am talking about the multiplier. So, I should have this a 3 8 should be there right so these 3 should be there.

(Refer Slide Time: 71:15)



Conflict Graph $G_-(V,E)$

- Resource Conflict Graph $G_-(V,E)$
  - V represents operations
  - E represents conflicting operation pairs
- Conflicting operations
  - Two operations are conflicting if they are *not* compatible
- *Complementary* to compatibility graph
- Find independent set of $G_-(V,E)$
  - A set of mutually compatible operations
  - Coloring with minimum number of colors
  - Chromatic number $\chi(G_-(V,E))$

And now once you have this what algorithm you can apply you can actually use this graph colouring problem this problem can be worked to graph colouring problem what. So, what is a graph colouring problem it tried to find out the minimum number of colour to colour all the nodes of the graph such that 2 adjacent node have a different colour so, that is called this chromatic number ok.

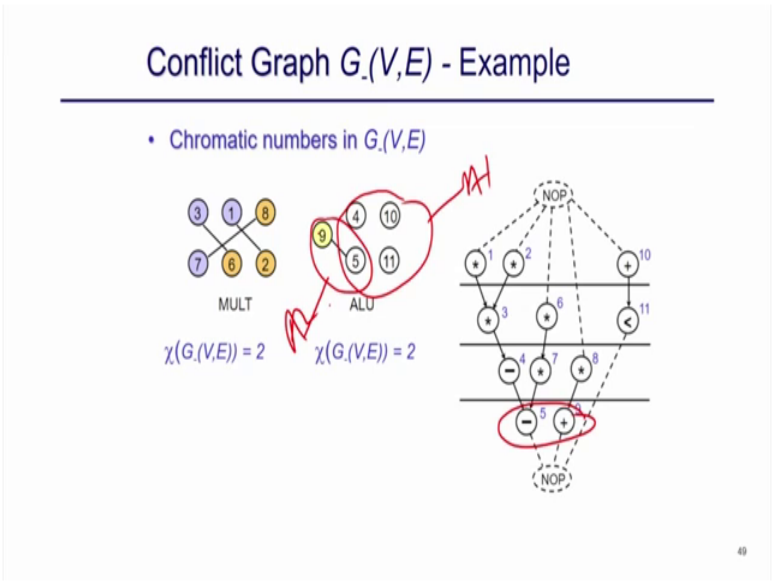
Chromatic number is the minimum value of a colours so which is required to colour all the nodes of their graph such that two adjacent node have a different colour. So, what does it mean? So, if you just do this so you can apply say blue here, but this a yellow because they are come where they have edge between this and then edge between so I can apply blue here, and yellow here.

Because there is edge this I can apply blue or yellow because there is no edge between this if I apply blue here it has to be yellow and if I apply blue here this has to be yellow. So, I need 2 here right so there are 2 2 colour is sufficient to do this right. So, now if all the node have the same colour is nothing, but they are they can map to the same resource.

So, you can see here this also give you the solution like 1 1, 3, and 7, 1, 3, and 7 is mapped to this the same resource here also 1, 3 and 7. So, both are same problem, but you have a different approach one is clique covering problem, other one is graph colouring problem and here also this is 2, 6, and 8, here also 2, 6, and 8, 2, 6, and 8 is the map to the 7 is same a multiplier right.

So, here also it give you since the chromatic number is 2 ; that means, you will say there are 2 multiplier is required and this 1, 3, and 7 is going to map in the multiplier 1 and 2, 6 and 8 going to map into multiplier 2. Similarly, for adder this is the compa this is the compatibility graph for this multiplier. So, you can see that the results is same right, but you can apply 2.

(Refer Slide Time: 73:07)



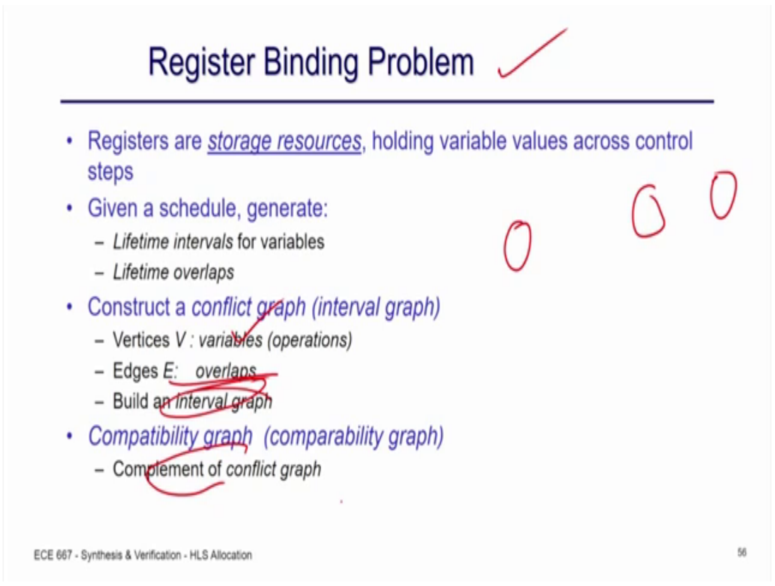
Conflict Graph $G_-(V,E)$ - Example

- Chromatic numbers in $G_-(V,E)$

For conflict graph for adder I have this is the only thing because there is only shading is here and I can use a white for all this 4 and only this is yellow. So, I need 2 adder and this 4, 10, 5, 9 can be executed in the same same adder and this is adder 1 and this is adder 2 right. So, here I just discussed how that ah finding minimum number of this a FU function unit can be mapped to a graph colouring problem or comparability graph problem and I apply this conventional graph colouring problem algorithm, or say heuristic, or say comparable clique covering, heuristic to solve this problem to get the solution ok. So, this is one approach.

And now the register binding problem is same right. So, either here I mean this node each node will be registers you can actually construct a conflict graph or a compa compatibility graph how if the life time of 2 register are overlapping then there is a conflict between them right and if they are not overlapping there is no conflict. So, I can construct based on that idea I can construct the conflict graph or compatibility graph from the register lifetime information.

(Refer Slide Time: 74:15)



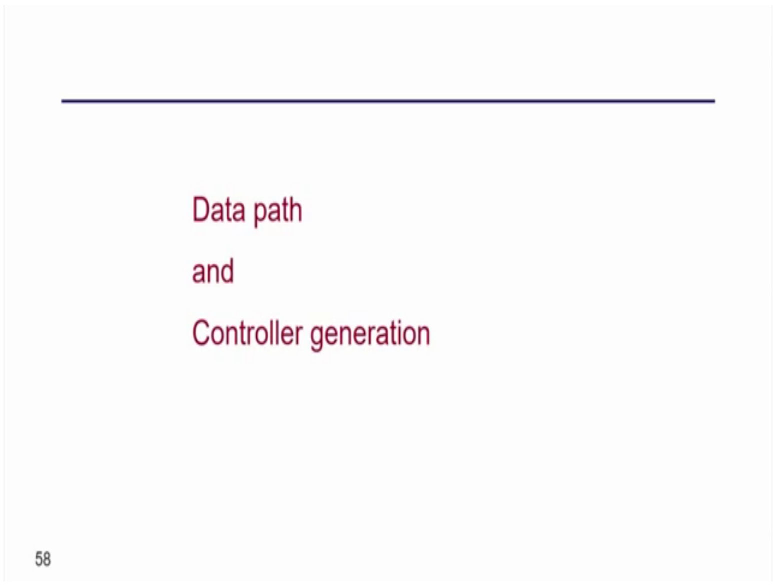
Register Binding Problem

- Registers are *storage resources*, holding variable values across control steps
- Given a schedule, generate:
  - *Lifetime intervals* for variables
  - *Lifetime overlaps*
- Construct a *conflict graph (interval graph)*
  - Vertices *V* : *variables (operations)*
  - Edges *E*: *overlaps*
  - Build an *interval graph*
- *Compatibility graph (comparability graph)*
  - Complement of *conflict graph*

ECE 667 - Synthesis & Verification - HLS Allocation 56

Similarly, I can also find the interval graph and I can apply each any of the technique that I discuss so far so to solve this register binding problem as well right so this is also same problem. Just here the each node will represent the registers the V is the set of variables, and E is the overlap if their lifetime overlap then I will add a edge and then I can consider my interval graph or complete graph or compatibility graph you can apply the same algorithm. So, both can be solved in this same approach.

(Refer Slide Time: 74:42)



Data path

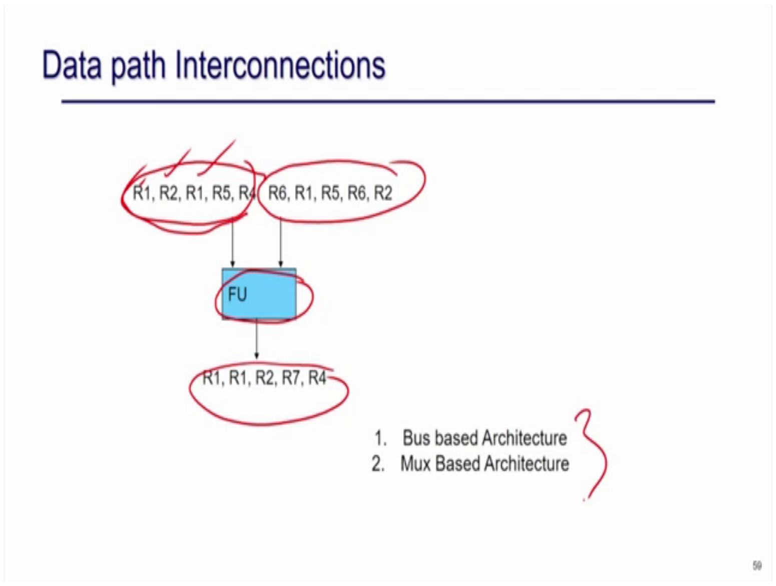
and

Controller generation

58

I just go to the last part that data path and controller generation it is very obvious there is no much compliancy here yes.

(Refer Slide Time: 74:48)



Data path Interconnections

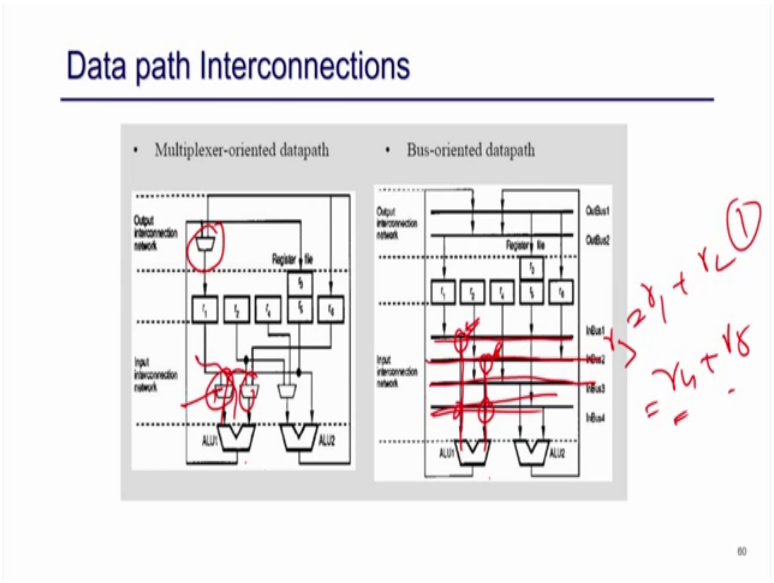
1. Bus based Architecture
2. Mux Based Architecture

59

So, here as I mentioned that after this register allocation binding and function you need reminding I know this operation is map to this and these are the set of inputs possible here, now these are the set of input possible here and I have to make some connection here interconnections so that this kind of operations can be executed.

And in some time step I need this, and sometime step I need this, sometimes step I need this, and so on. So, I need to add some data path interconnection you need like multiplexer, demultiplexer and so on. So, that I can execute this operation smoothly right and there are 2 possible operation bus based architectural, mux-mux based architecture.

(Refer Slide Time: 75:30)



Data path Interconnections

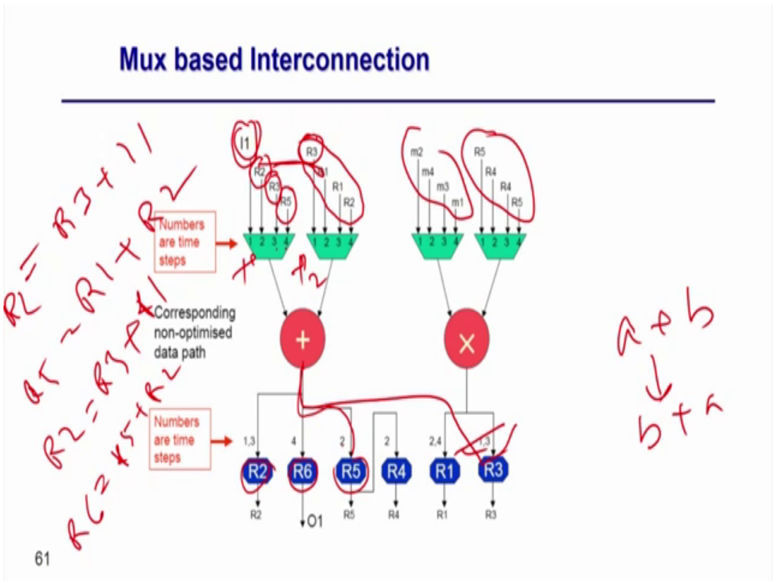
- Multiplexer-oriented datapath
- Bus-oriented datapath

So, in bus based architecture what happened for each register there is a output bus right for register 1 there is a bus, for register 2 there is a output bus, for register 3 is output bus and if this r 1 is connected to this I will just added connection here and added switch here.

Similarly, if say this ah r 4 is connected to this I will just add a line from here and I for r 2 and then I had a switch right. So, this switch will be controllable by a control signal. So, if in a time step 1 says I am doing say I am doing r 1 plus r 2 equal to r 3 then this switch and this switch will be activated in time step 1 right. And in other time step if say r 4 and r 5 is happening then the switch corresponding to r 4 line and the switch corresponding to r 6 line will be enable in that particular time step so that that the particular register value will come to the FU.

So, this is bus base data path and mux base which we have discussed in the last lecture that we have mux will put mux at the input of the fuse will part put mux at the input of the register which you have discussed. And then will add control signals so that I can control the data flow these versus this based on operation going to execute in the particular time step. So, both are possible and both are kind of same here mux here your only switch, switch is nothing, but 2 is to 1 mux. So, both are possibility possible and we can generate any of them ok.

(Refer Slide Time: 76:49)

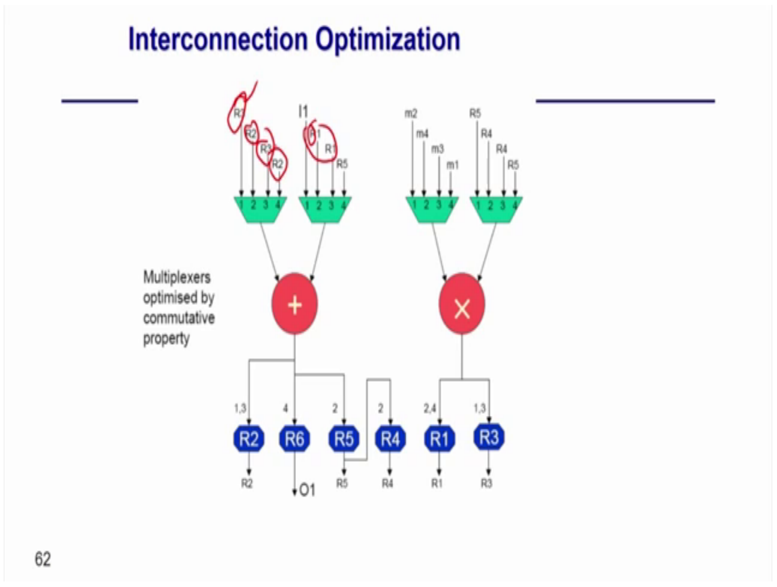


And here what kind of optimization is possible although actually that commutative operations can be applied here to improve that for this mux size. So, if you just see here so for example, say time step 1 this I 1 is coming time step 2 r 2 is coming, and time step 3 r 3 is coming, and time step 4 r 5 is coming and similarly here r 3, r 1, and r 2 right.

And similarly here also right and here also these value in time step 1 and 3 go to r 2, times step 4 it going to r 6, time step 2 is going to r 5, the output of this adder. So, we will add so conventionally what we are going to do I will just put a mux 4 is to 1 mux then control signal length will be 2 here also 4 is to 4 is to 1 mux control signal in control signals with 2 bit control line and so on.

So, this is the data path I am going to generate by if I generate the data path similarly here since there are 2 output is so they are all connected. So, there is no problem, but if there is a multiple so I adder this also coming here and then I have to put a mux here right. So, this is what I have generated here right so this is something is known. Now if we just see if I just do here I just swap this r 2 and r 1 and we will get this right. So, I will just swap this r 2 and r 1 ok.

(Refer Slide Time: 78:00)

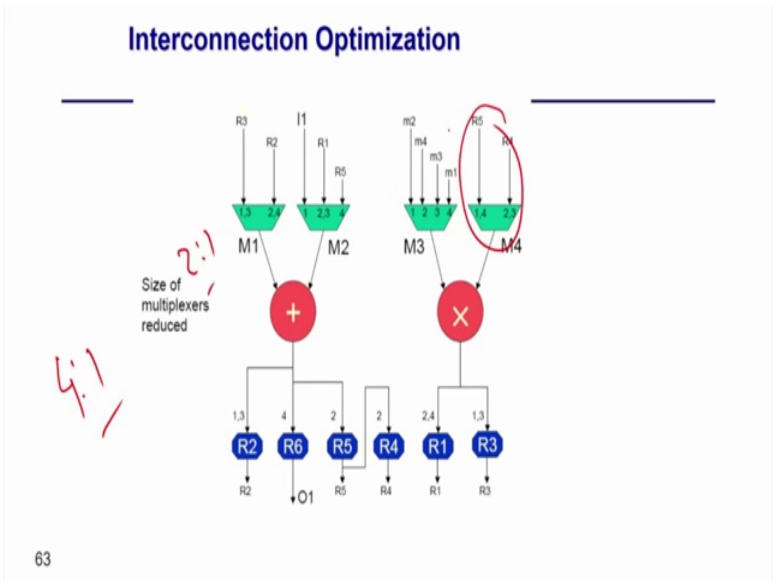


So, now you can see initially I have this r 1, r 2, r 3, r 5 and if I just move in this swap this r 2 and r 1 what will happen you can see r 2 comes here, and r 1 comes here, and now I have 2 r 1 right. So, I can select this r 1, and r 1, r 2, r 1 so, I can reduce the mux size, so I am only 2 possibly, 3 possibly input r so there r 2, r 3 also.

So, I and also I swap this I 1 and r 3. So, this also I swap because I am doing a plus b right a plus b and b plus a is both are same. So, I can just swap this a and b and b to a what is done here r 1. So, here you can see I actually am doing in time step 1 I am doing I 1 plus r 3 right equal to r 2 this is what I am doing here. And this time step 2 I am doing r 2 plus r 1 and it is storing into r 5 time step 3 I am doing r 3 plus r 1 and it is storing in time step 3 r 2. And time step 4 what I am doing this r 5 and r 2 and it is storing into r 6. So, this is the 4 operation I am executing here.

So, similarly can be done here and I am not going to that detail so now what I am just a doing I am just swapping this. So, instead of r 2 plus r 1 I am just doing r 1 plus r 2 they both are same r 1 plus r 2. Similarly here I am doing l 1 plus r 3 instead of doing I aM just doing r 3 plus I 1 right. So, as I swap these two computed this operation right. If we just do this what will happen you can see now I just do this so now, I have two r 3 and two r 2 so I can actually reduce the size to 4 to 4 is to 1 mux 2 is to 1 mux.

(Refer Slide Time: 79:53)

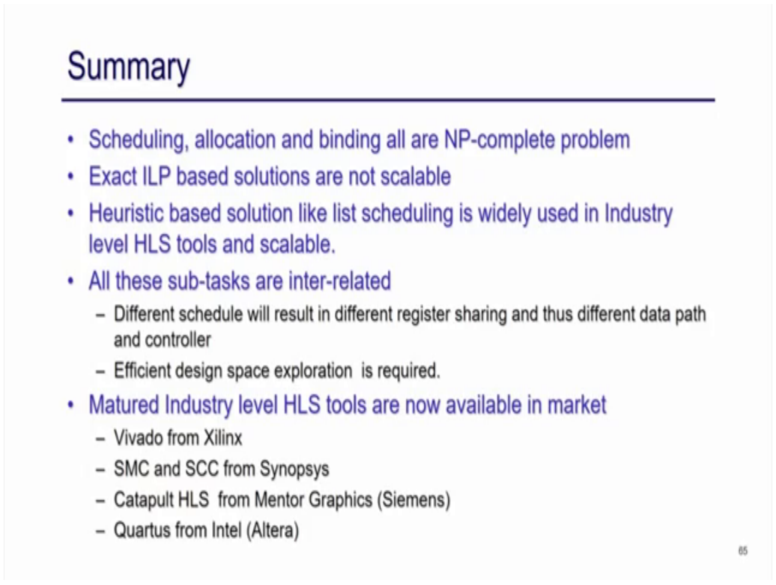


Because I am now actually to only two possible input here also I mean one of the input readings. So, similarly I can do the same thing for this this mux which I am not going to need to discuss detail and just reduce the size. So, this is the kind of optimization we can do in the data path generation using this commutative operations and all we can actually reduce the mux size which will also give you the what are area requirement of the whole design will be less.

And also the number of control signal required also less earlier, I need 4 bit I need 2 bits, I have now only need 1 bit. So, the controller also size also will reduce so this is the kind of inter connection optimization you can do and there finally, we just add those control signals to this all these mux size based on the size right.

So, this is so in the control the generate and data path generation steps the steps are kind of obvious you have to make add mux or the bus there and we have to put switches or muxes and then you can actually apply this kind of commutative operations or associatively and all those kind of operations just to reduce the interconnection size. So, these are the possibilities here ok.

(Refer Slide Time: 81:00)



Summary

- Scheduling, allocation and binding all are NP-complete problem
- Exact ILP based solutions are not scalable
- Heuristic based solution like list scheduling is widely used in Industry level HLS tools and scalable.
- All these sub-tasks are inter-related
  - Different schedule will result in different register sharing and thus different data path and controller
  - Efficient design space exploration is required.
- Matured Industry level HLS tools are now available in market
  - Vivado from Xilinx
  - SMC and SCC from Synopsys
  - Catapult HLS from Mentor Graphics (Siemens)
  - Quartus from Intel (Altera)

65

So, now we will come to a summary of these steps. So, we found that all these steps scheduling allocation binding all are kind of N P complete problems. So, exact solution is something is not is asked for most of the time. So, you apply usually a heuristic a solution like this is scheduling and I mean used to solve this problem.

And another important factor here is something all these subtasks are kind of inter related. If you have a deferent schedule your register allocation or binding the number of register is required or a function is required will be different if you have a another person other is scheduling right so all are kind of inter related. So, you need to explore the design space as much as possible to give you the best results.

And if you just seem to the industry there are various in the synthesis tools is available like Vivado from this Xilinx or say SMC and simple model compiler or SCC the C compiler both are from Synopsys. Catapult HLS from mentor graphics, or say Quartus from Intel which is basically altered.

So, all this tool armature and they are available and them in this kind of a widely used in the industry and so high levels synthesis very exciting area, but still there are much more thing can be done which you will discuss in subsequent classes. So, in tomorrow's class we are going to discuss about what is a good way of writing C as I mentioned in the first class also that not all C will going to give you the good synthesis results right. We have to write the loop in specific way or say write that array into specific way, access them in the specific way. So, those we are going to discuss what is a good way of writing your operations or the behaviour in C which will actually which will generate a good hardware, or good RTL through the synthesis tool so that will be the topic of discussion in the next class.

Thank you.