

Design Verification and Test of Digital VLSI Designs
Dr. Santhosh Biswas
Dr. Jatindra Kumar Deka
Indian Institute of Technology, Guwahati

Module - 2
Scheduling, Allocation and Binding
Lecture - 2
Scheduling Algorithms

Welcome to module 2, let us check 2 and 3, this is a joint lecture, I mean the combined lecture. So, what is the last lecture you have seen, the last lecture you have seen that high level synthesis is basically comprises three steps scheduling allocation and binding. So, in that step we have seen what happens in scheduling what happens in allocation and what happens in binding. Then, actually we did not give any kind of algorithm to do these steps, but actually we have told that the main emphasis of this course is that we will be learning automatic algorithms or automatic tools which will help during the design process.

So, in the last lecture we have seen some examples like $a + b + c$, how you can go for scheduling allocation and binding of this specification. We have seen different options everywhere these will have their requirements where there and so forth, but we have not given any kind of automated algorithm which can automatically scheduled allocated and buying these steps given the specification. So, in these two lecture series what we will try to do with this or in module two on three, four more lecture, I mean two, three more lectures on this.

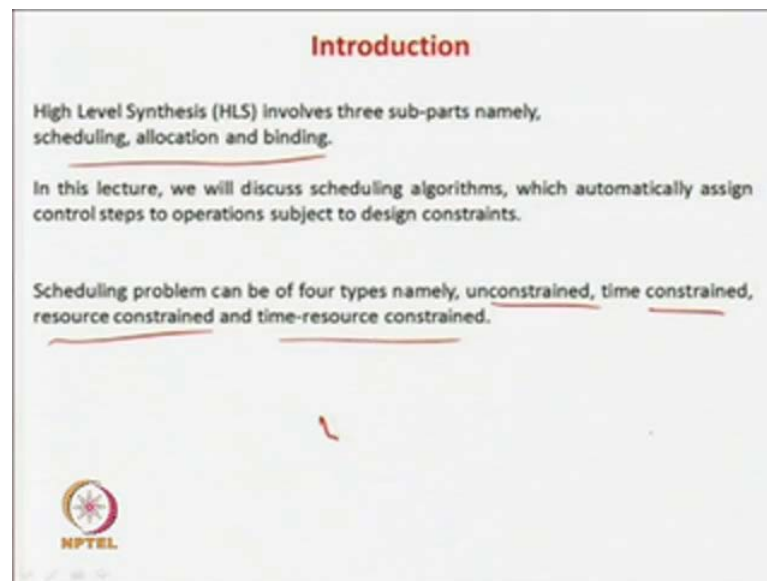
So, in this case what we will do, we will learn automatic algorithms which will do the scheduling allocation and binding for your automatic staffs given the specifications automatic, it will be done for you. So, in lecture two and three, first we are going to see scheduling algorithms, some automatic algorithms where you have to give the specification like $a + b + c + d$.

Standard example is which you are using on any other specification and then if you are going for a resource constrained, if you are going for unconstrained, if you are going for time constrain, if you are going for time resource constructs. You have to give the constructs and then automatically we will get a scheduled output, if the schedule is

possible, but if the schedule is not possible we will say that it is the algorithm, we will say that it is not possible.

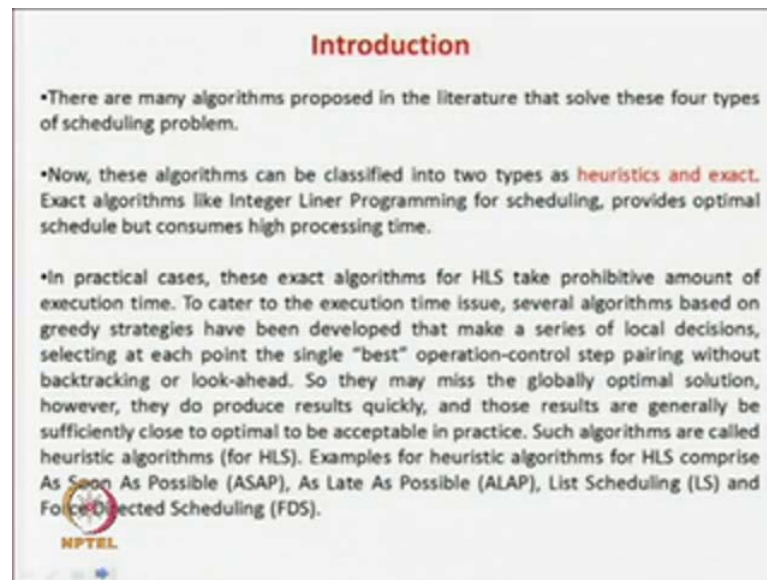
So, you have to I mean some of the constructs, so over here what you are going to learn given a specification and the type of scheduling we want to do outputs. If the scheduling was possible and if not it was their problem, you have to do that, so we will see different algorithms which can do it for you.

(Refer Slide Time: 02:14)



So, I have already told you high level synthesis, this one in this lecture we are going to see, I mean the scheduling algorithms which automatically do the steps for us. So, scheduling for unconstrained time constrained time resource constrained and time resource constrained. So, they are the four problems already we discussed, now we will see algorithms how which we can solve the problem now before going to this algorithm, so in in VLSI, as we prove in the scheduling procedure in the heart problem.

(Refer Slide Time: 05:40)



Introduction

- There are many algorithms proposed in the literature that solve these four types of scheduling problem.
- Now, these algorithms can be classified into two types as **heuristics and exact**. Exact algorithms like Integer Linear Programming for scheduling, provides optimal schedule but consumes high processing time.
- In practical cases, these exact algorithms for HLS take prohibitive amount of execution time. To cater to the execution time issue, several algorithms based on greedy strategies have been developed that make a series of local decisions, selecting at each point the single "best" operation-control step pairing without backtracking or look-ahead. So they may miss the globally optimal solution, however, they do produce results quickly, and those results are generally be sufficiently close to optimal to be acceptable in practice. Such algorithms are called heuristic algorithms (for HLS). Examples for heuristic algorithms for HLS comprise As Soon As Possible (ASAP), As Late As Possible (ALAP), List Scheduling (LS) and Force Directed Scheduling (FDS).

NPTL

If people's exponential time complexity, so if there are same operations or if the end time slips for time, I mean the best file of the best possible schedule the complexity will be in the order of 2^3 power a. So, you can assume that your specification if they are having hundreds of variables or your steps may be say hundred of steps you required to do a practical schedule. So, your complete, it will be 2 to the power of 100, so are they actually very infeasible to do this, so what do we do is that these are actually called exact algorithm.

So, what do you mean by an exact algorithm in exact algorithm, so given a problem, so even if you take exponential amount of time on or double exponential amount of time, but we will get the exact minimal. That is exact optimal solution that we actually tell you that this is the best solution possible for a given constrained, if the constraints can be satisfied, if it is not satisfied, you will not get the solution. It will say that and change some of the constraint or relax some of the constraints, but in the constraints if the constraints can be satisfied.

So, the algorithm will tell you that this is the best possible solution, but it may take horrible amount of time because all the problems. As we are seeing there to start with caviling problem is a mp or you can call a difficult problem or exponential amount of problem. So, I mean because of this, so we can prove that actually this scheduling problem can be mapped to a individual linear program ILP problem which is known to

be an complete problem. So, the time taken will be exponential so that we can theoretically show it in the end, so you require exponential number of steps if you want to find out the exact base solution. So, that is infeasible many of the time because of the time limit sometimes you go for what do you called as heuristic.

Now, what is the heuristic, so heuristic may not give you a best optimal solution, but the time taken is very less compare to the exponential time. Actually, if the numbers of steps are end, so you may get the solution in n number of steps, but you may not get the best solution, but your solution will be very near to the optimal solution. So, you call that near optimal solution on near optimal solutions, you will get, so that is actually called heuristics, so heuristics will be trying different ideas. So, you can think about that to try to get the best solution, you have to try all possible possibility, so this will take a huge number of times to do that, but we can say that in other in broad constraint of heuristics.

So, here you will not try all the all the possible solutions here, we will try some of the which we are tend to think that these are very good solutions and we are trying to put into it by which means you are trying to remove that all the solutions. These are all the possible solutions where we will not try to file out all the possible solutions, we get the time, but we may know that this is this is not of good solution. You may not try, this is also may not a good solution not try this, but we will try we will try some of the solutions which we may feel that are good solutions.

So, that is actually a heuristic, but in that process we save time, but sometimes we tend to find out that we have missed the best or the optimal solution, but in majority cases we will get a very near optimal solution and the time saving will be a huge. So, we will be using we generally use heuristic algorithms for solving this, so for scheduling first, we will see lot of heuristics algorithm. Then, we will see that what is the exact algorithm like the individual program which can give you the best solution or the most optimal solution, even the constrains if they are satisfiable and then we can also show that this actually which will take a huge number of time.

Actually, it takes a huge time to find out the solution to this one, so it is a difficult problem and so is working with heuristics and slowly we are trying to improve upon the heuristic. So, with less time they can get or a reasonable amount of competition time, you can get what you call reasonably near optimal or solutions near the optimal. So,

scheduling. So, you say that the status is non fusible and if it is possible, now what do we do each operation o_c , so you take each operation, one at a time you have to do.

So, you have to find out o_c has no intermediate immediate predecessors, so to first find out all the operations which do not have any predecessor that is they can be computed from inputs itself like our previous example $a + b + c + d$. So, $a + b$ and $c + d$ can be done in time step one because they do not have any predecessor, so these are the actually o_1 these as o_2 so $o_1 + o_2$ can be scheduled in time step one because I mean o_1 and $a + b$ and $c + d$ are taken from inputs. So, do not require any predecessor, but multiplication you require tenth 3 that is actually $o_1 + a + b + c + d$.

That has been multiplied by c , multiplied with e so I mean you cannot schedule the multiplication operation in the first time step because operation has some predecessors. So, what is the predecessor, it is the answer of $a + b + c + d$, so in the first time step what we do we allocate control step one to all the operations which can be confident for the input.

So, it saying that if o_i have no immediate predecessors, then you do that, otherwise what you do otherwise you assign control step say element or say another operation. Actually, operation has a predecessor, so you have to first schedule the predecessor, so it says that for a , o_i , we will have the predecessor. So, you say that control step o_i is equal to maximum of control step maximum of control step of g , sorry control step plus one where o_j is the immediate processor of o_i .

(Refer Slide Time: 09:06)

As Soon As Possible Scheduling

Algorithm 1: As Soon As possible

Input: Operations O , Maximum number of control steps M .

Output: Control step for each operations, Status of scheduling

Steps

for each operation $o_i \in O$

DO

if o_i has no immediate predecessors (i.e., computation from inputs)

$control_step(o_i) = 1$ /* $control_step(o_i)$ indicates control step into which operation o_i is scheduled */

else

$control_step(o_i) = \max_{o_j \in \{o \mid o \text{ is immediate predecessor of } o_i\}} (control_step(o_j)) + 1$, where

$o_j \in \{o \mid o \text{ is immediate predecessor of } o_i\}$.

END

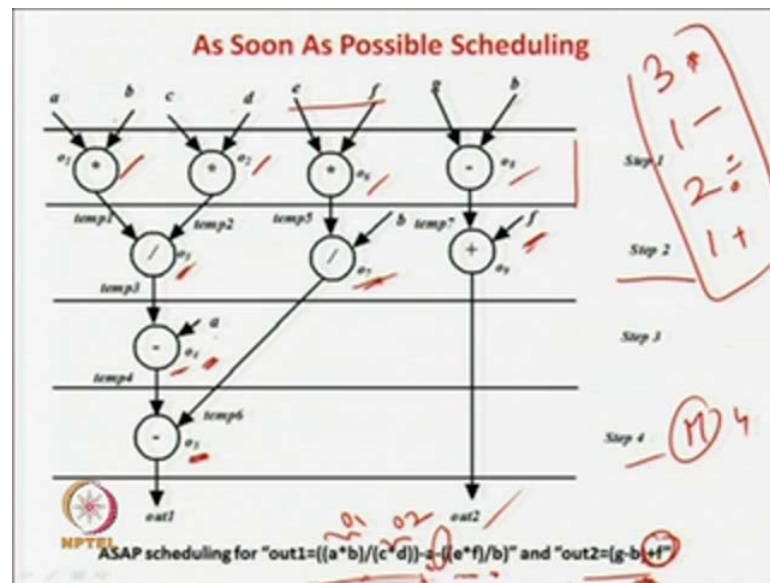
If value of $control_step(o_i) > M, \forall o_i \in O$ then Status of scheduling is Successful.

So, it says that this o_i and so there will be an immediate predecessor predecessors o_j , so this is a capital o_j . This can be also o_j it is the different procedure like o_j belongs to immediate predecessors of a , so I mean what you have to do you have to find out which is the maximum one. So, it may depend on this it may depend on this, so it may be in time step two it may in time step two, so it may be in time step three, so what you have to do? So, obviously you cannot schedule it in time step three because it is dependent on this element which is nothing, but time step which is nothing time step three.

So, you cannot do it in this one because o_i actually depends on the value of this note, so what will be the time step here it will be 3 plus 1 here we will say. So, we have to find out the maximum of the controlled step for all the predecessor of o_i and you have to add 1. Finally, you have to repeat this for all the elements, so what you will do for schedule for all the elements for time step one which can be directly computed from the input. Then, we will find out all those operations whose immediate predecessors have already been scheduled.

Then, you have to find out which is the maximum amount, then you have to add 1, 2, and then that element is also scheduled. You have to keep on doing it till all the operations are over and if in the end you find out that you could do everything within m . So, the maximum one step of $o_j + 1$ that is the last element is less than equal to a m , then it is scheduling

(Refer Slide Time: 11:14)



So, we will take this example and do this, so these are the different example which you have considered in the last lecture. So, it is what a star b by c star d minus a minus e star f by b this is one output b expression and this is single output. So, we are taking bit complex one because you have to illustrate the cases, so in this case you will see a star b c star d. So, and we have taken say m equal to M equal to 4, so we can see that it cannot be done anything less than 4, so it is four, now we have to do what now you have to schedule as soon as possible.

So, you have to schedule elements which are actually, so we can say that this one you can say as o 1 c plus d we can say as o 2. Then, what we can say let us go through see that little more clear, so a plus b is o 1 if c star d is actually is o 2, then divide by this actually o 3 then what is o 4. This whole staff actually subtracted becomes o 4 and then actually e star f is o 6, this divided by d is o 7. Then, finally, you subtract this whole star, it becomes o 5, so actually these are some of the names, we have given for the out 2 g plus b is actually o 8 and finally, the addition of this o 9.

So, this expression we have given the numbers in the slide itself, now you have to find out as soon as possible and n is equal to 4. So, you have to find out which we can schedule immediately at step one it is possible only when the variables or the operations who do not have any predecessors. So, a plus a star b can be done because they can be computed from the input c star d can be done. They can be computed from the input, but

this one $a \star b \text{ by } c \star d$ minus this subtraction operation cannot be immediately because it depends on the output of this two.

So, this cannot be done immediately at stage one, now also $e \star f$ can be immediately, so they put the $\star f$ o 6 to be in time step one because they know you will get predecessor, but this by b also you cannot do it. Now, because it depends on the output of $e \star f$, now for this other expression you know that $g \text{ plus } b$ can be done in time step one because the variables are directly as input and o 8 can be scheduled in time step one. So, these are the variables operations like o 1 o 2 o 6 and o 8, so we can schedule them in time step one because they do not have any predecessor their values are directly from the input.

Now, you will see, now we will find out that $a \star b \text{ c } \star d$ is division operation can be now scheduled in time step two because it requires $a \star b$ and $c \star d$ which is already been done. So, these are of o 3 o 1 and o 2 which is already been scheduled in time step one, so just add 1 to this one so that the time step of o 3 will be 2. Similarly, for o 7, the predecessor is tenth five which is the output of o 6 which is already been scheduled in time step one. So, you can easily schedule b in time step two and time step will be time step of o 6 plus 1 o 7.

Similarly, for this case like for this addition operation you know that $g \text{ plus } b \text{ g minus } b$ already scheduled in time step one and o 9. That is this addition operation plus f can be now scheduled in time step two because it depends on o 9 depends on o a as it is already scheduled in time step one. So, you add one to time step one, you get time step two, so o 9 is scheduled over here, so actually we are pushing everything as hardly as possible. Now, this thing is done, now if you look at the output is generator, now the path is over that you know that o 4.

You have the observed that o 4 that is subtraction operation with this one tins can be done in time steps to because predecessor of o 4 is o 3 because o 3 has to be done before o 4. Now, as already o 3 has been schedule in time step two, so in time steps three we can schedule o 4. Similarly, the last stuff that is the subtraction operation o 5, this can already be done in time step five that is sorry four that is 3 plus 1 because o 5. We have two predecessor one o 7, one is actually your o 4, so you have to take the maximum one

because o_7 is already scheduled in time step two, but still another dependency of o_5 is o_4 which is done in time step three.

You have to take the maximum one, you have to add 1 to it, so 3 plus 1 is 4 and o_5 is scheduled in this one, so in this way we get as soon as possible schedule. Now you have to compute how many resources you require, so you see that 1, 2, 3, so in this parallel 3 multipliers are required. So, 3 multiplied would be required over there, once subtracted is required over here, so here two dividers are required over here.

So, two dividers required over here one adder is required over here, so these two subtractions can be used from here. So, resource you can multiply one subtracted to dividers and one adder because whatever is in one step you have to do it in parallel. If you look at I mean register you are not consider here, but a number of registers will be 1, 2, 3, 4, 5, 6, 7, 8, eight register are there. They can be all reused in the other time steps, so now we go to another schedule as already said.


(Refer Slide Time: 16:51)

As Soon As Possible Scheduling

In this case, it may be noted that operations o_1, o_2, o_4, o_5 do not have any direct predecessors, i.e., they depend on input values. So these operations have the control step as 1 ($\text{control_step}(o_i)=1, i=1,2,6,8$). Operation o_3 has o_1, o_2 as predecessors, so, $\text{control_step}(o_3) = \text{maximum}(\text{control_step}(o_1), \text{control_step}(o_2)) + 1 = 2$. Similarly, control step assignment for all operations can be explained.

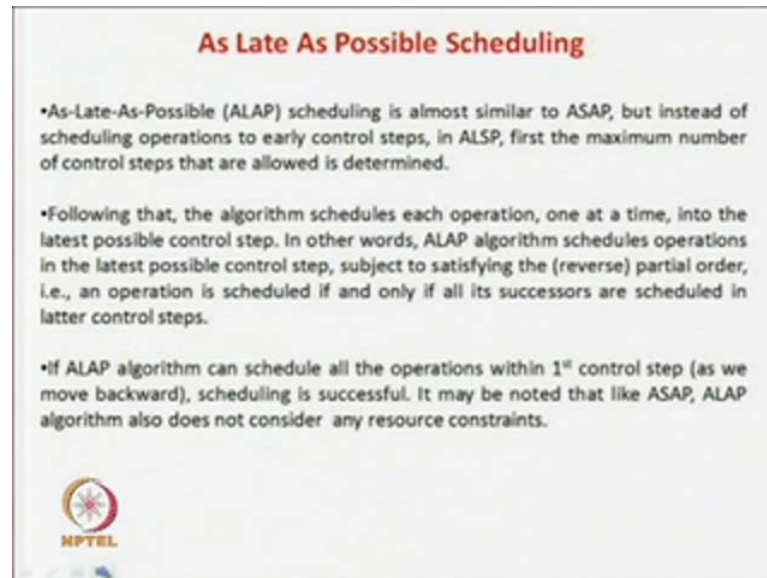
This schedule is complete within 4 steps, thereby making it successful. The resource requirements are—

- Step1: 3 Multipliers + 1 Subtractor
- Step2: 2 Dividers + 1 Adder
- Step3: NIL (subtractor from Step1 can be used)
- Step4: NIL (subtractor from Step1 can be used)




So, what are the requirements over here, so requirements over here three multipliers and one subtraction over here in first time step two is second step two divider one adder and third step addition. They can reuse, so at in all require is the resource, so whatever we have already told is actually listed out in this line you can go through it.

(Refer Slide Time: 16:56)



As Late As Possible Scheduling

- As-Late-As-Possible (ALAP) scheduling is almost similar to ASAP, but instead of scheduling operations to early control steps, in ALS, first the maximum number of control steps that are allowed is determined.
- Following that, the algorithm schedules each operation, one at a time, into the latest possible control step. In other words, ALAP algorithm schedules operations in the latest possible control step, subject to satisfying the (reverse) partial order, i.e., an operation is scheduled if and only if all its successors are scheduled in latter control steps.
- If ALAP algorithm can schedule all the operations within 1st control step (as we move backward), scheduling is successful. It may be noted that like ASAP, ALAP algorithm also does not consider any resource constraints.

 NPTEL

Now, we go to another extreme mode of schedule, so that is actually called as later as possible, now in this case what we have done is as soon as possible what our idea was in as soon as possible. We have tried to, I mean what do have done, we have tried to schedule as soon as possible, and we have taken m equal to 4. We have try to push everything as possible or in the first time step, but you have to observe here that if you say that our time step would have been 3 over here.

So, it would have resulted in a infeasible situation because this o 5 cannot be schedule over here because it depended o 4 which again depended on o 3. So, o 3 cannot be done in one, it is done into of our o 4 cannot be done to 3, so if you make that a or I mean time requirement this 3. So, this would have led to a infeasible situation, but now we have taking m equal to 4, so we get a perfectly correct schedule and in case of as late as possible what we have going it is very similar way. You have to get as soon as possible, but what we do again, what we have said that we actually take the maximum number of control steps.

Then, actually what we do we tried to one at a time in the as late as possible in this case what to do, try to push, earlier case we are trying to push as early as possible over here. In this case what we are going to do, we will try to take things as less as low as possible, we will take this aim as what you call the maximum one.

They will be try to keep the elements over here as slowly grow over here that is we try to push the elements in this direction. This is the lower, now again one when you consider it will successful, you will find out that all operations has been scheduled in within the first control step because we are going reverse, then it is done, other wise this not done, so we will see it again.

(Refer Slide Time: 18:20)

As Late As Possible Scheduling

Algorithm 2: As Late As possible

Input: Operations O , Maximum number of control steps M .

Output: Control step for each operations, Status of scheduling.


Steps

```

for each operation  $o_i \in O$ 
DO
    if  $o_i$  has no immediate successors (i.e., computation generates outputs)
        control_step( $o_i$ ) =  $M$  /* control_step( $o_i$ ) is assigned the
                                last control step */
    else
        control_step( $o_i$ ) = control_step( $o_j$ ) - 1,  $o_j$  is immediate successor of  $o_i$ .
END

```

all $o_i \in O$ are scheduled within control step 1
then Status of scheduling is Successful



Look at the formal algorithm and then we will see with an example, same thing the operations same thing as the algorithm operations will be given maximum number control steps will be given and output is the control steps for each operation and status.

So, again like the earlier case for all operations what you have you do now you can have to we are going as late as possible, so we have to just find out the all the operations o_i which do not have any immediate successor in this case. So, if you look at this, we which do not have any immediate successor o_9 , so o_9 is actually an output o_5 is actual output. So, they do not have any kind of a successor, nobody depends on them, so what we can do, we can schedule them as late as possible, so that is what they are saying.

So, you can see a if o_i have no immediate successor that is computation gene that is the computation generated outputs you schedule at the last step. In the previous case, what we were doing we were trying to find out all those operations which do not have any predecessors. We are trying to put it in the first time step, here we are doing this thing reverse, we try to find out all the operations which have not any successor and we have

putting in the end next. What are you trying to do, then you are trying to find out, I mean all the elements and you say this is o_j has been scheduled over here.

That is last may be last step or some step, you need to find out all other elements, actually whose successor is o_j , so o_i actually is o_j is the immediate successor of o_i , so all o_i 's will find out whose immediate successor is o_j . Then, we try to schedule it just one step will before it, so its control step $o_j - 1$ is the immediate predecessor of o_j . So, o_j is scheduled, now find out all o_j is scheduled in the onetime step, now you will find out all o_i 's, sorry these are not o_j 's, all o_i 's who actually need immediate predecessor of o_j .

You just schedule it one step behind this before this, so it is o_1 minus you have to do this for o_j , then if all o_j are scheduled within the first control step if everything in first control step. Then, this schedule is successful, else it is not successful is very similar, so you will just see this and then will take same example. So, again same example is like this, so in this case you reverse way, so you know that o_9 that is this addition operation and this subtraction operation o_5 .

They are just that generate the output, so you not need to worry at all about anything, so what you can do you can schedule at number 4. We already we have seen that you should not consider time step 3 because for this scheduling problem three in result with infeasible state, so in last step allocating 5, 6. Now, what do you have to do; now you know that already this has been scheduled, so we are putting it as late possible.

So, we have try to push in down, now you see that as this has already been done, now g minus b that is o_a . So, immediate predecessor of nine is o_8 , so you can schedule it at the time step three that is $4 - 1$ is just you put it over here. Now, in time step, this case you see what is the immediate predecessor of o_5 , it is actually o_7 , so o_7 is what o_7 is this actually this division operation.

That is e minus f by that division, this division operation is actually o_7 , and now what happens, so one, just one step which before this o_6 , you just schedule at time step three. Similarly, o_4 this o_4 this o_4 is what this o_4 is actually this subtraction operation is whole stuff minus that is also the predecessor of o_5 . So, just schedule it one step before this that will be time step, so in time step three you are scheduling o_4 and o_7 . Now, if you go by this, similarly if you go so what is o_7 , o_7 is actually this division by b 1 step.

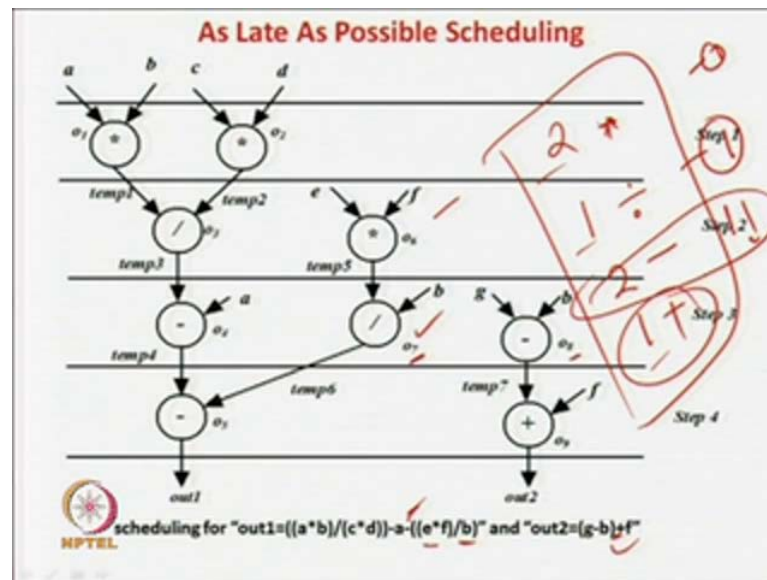
Before this, actually $e \star f$, so this is actually $e \star f$ that is o_6 , so the one step predecessor of o_7 is o_6 , so you can you just schedule just one step before that scheduling time step two you have scheduled. Similarly, for o_3 and o_4 , o_3 is immediate predecessor of o_4 , so just schedule it one step ahead, so it be an o_3 will be here. Similarly, o_1 and o_2 are the immediate predecessor of o_3 , so you have to schedule them over here.

Now, you just look at this situation, so this algorithm is very simple, just the reverse of as soon as possible we have done. So, let us see how many requirements are there, so we require two multipliers immediate here, same multiplier can be reused. So, you require one divider over here, you require two subtracts, these divisors can be reused, here you are requiring one adder, and so this is what your requirement is. Now, you see an interest thing so in the old case we require three multipliers, but now we require one multiplier, two multipliers. So, we have saved one multiplier is very important, but on the other hand we have actually lost one subtraction.

So, if we just look at it, so we required one what you required we required three multipliers from subtractor two dividers, one adder. Now, in this case we have done a great deal of saving, so in this case what you have to save we have saved one divider we have save one multiplier. We have saved and one subtraction we have lost, but you already know that subtractor area is much divider hardware or multiplier hardware.

So, we have we have achieved a great deal of optimization by going for as late as possible schedule, so in this example, you have scheduled everything below the end. So, we have got a very good schedule where you have saved some element, now what you have lost, so you have lost one subtracted. So, let us see one we can do to actually save of subtractions, so if you remember that that these case the two subtraction case, we got one subtraction when our solution was as soon as possible. This, this and this at this things adder, we multiply those, this multiplier and divider is saved when you are going for as late as possible.

(Refer Slide Time: 22:45)



So, let us try a bit different heuristics so actually what are you doing there are heuristics algorithms nobody will guarantee that you are always going to guess this. If you going to use as late as possible for all examples, you are going to get the best solution and for as soon as possible. For all cases, we are going to get the best solution heuristic means for some cases solution a will be put algorithm will give better result than algorithm b and situation will change in different example.

(Refer Slide Time: 23:18)

As Late As Possible Scheduling

In this case, it may be noted that operations o_1, o_9 do not have any direct successors, i.e., they generate output values. So these operations have the control step as $M = 4$. Operation o_3 is the immediate successor of o_1, o_2 , so, $control_step(o_3) = control_step(o_1, o_2) - 1 = 3$. Similarly, control step assignment for all operations can be explained. This schedule is complete within the 1st control step, thereby making it successful. The resource requirements are—

- Step1: 2 Multipliers
- Step2: 1 Dividers + (multiplier from Step1 can be used)
- Step3: 2 subtractors + (divider from Step2 can be used)
- Step4: 1 adder + (subtractor from Step3 can be used)

NPTEL

So, in this example, let us divide it into two, we will see that, so in this case what we have achieved that is written that two multiplier one divider and two subtracts and one adder. So, this is these are the great saving, we have only one subtracted has been required, one more subtracted. Now, we will see what we can achieve more because you should not have the feeling that as soon as possible is always very bad and as late as possible will always give you a better result.

Had it been the case, we should not have lost one subtracted over here which should always have been the gate. So, in different example, one will give a better solution, then the other and so we will try to make a high bid of these two and see we can get anything better. So, heuristics like this, you are not exact algorithms, so always at for different examples you may get different solutions that different solution in terms of quality.


(Refer Slide Time: 23:57)

ASAP versus ALAP

If ALAP is compared with ASAP, it may be noted that we have achieved the following.

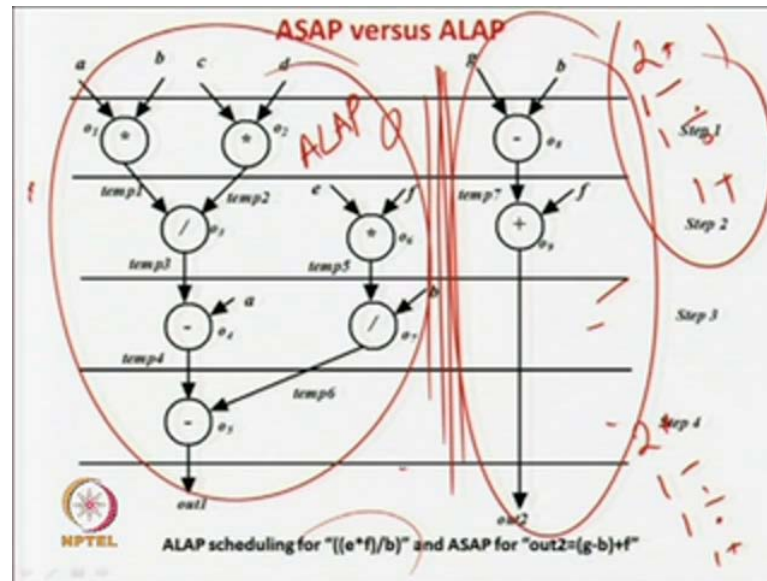
- Saved 1 Multiplier by delaying o_4 from step1 to step2.
- Saved 1 Divider by delaying o_5 from step2 to step3.
- Increased 1 subtractor by delaying o_3 from step1 to step3.

So, it may be observed that for the subpart of the expression, " $(e*f)b$ ", ALSP is better compared to ASAP. However, for the expression, " $out2=(g-b)+f$ " ASAP is better compared to ALAP. As already mentioned, ALSP and ASAP are heuristics and may not generate an optimal solution. By applying the scheme ALAP for " $(e*f)b$ " and ASAP for " $out2=(g-b)+f$ ", the schedule we obtain for $M = 4$ is shown next.



So, if you are comparing as soon as possible as late as possible, so we save from one multiplier and save one divider that increased one subtracted, this is what has been happened, now what we are going to do, so whatever is written over here, let us try.

(Refer Slide Time: 24:10)



So, let us try to separate this, so you see that multipliers and dividers are actually already in curve for this part or this problem like if you see that two sub parts as out 1 and out 2. So, you can see here that out one actually is involved with in adder and sorry multipliers and dividers, but out 2 is involved with subtracts and adders. You saw that the adding subtracted requirement was increased when we gone for as late as possible and divider and divider and multiplier. We got better solutions when you gone for as late as solution as late as solution possible algorithm.

So, let us partition this into two and let us try to apply as late as possible so when here as soon as possible, over here this is one another type of heuristics. We will see what did happen if you do that, so in this case, let us see this as late as possible, so obviously this structure will be similar. So, these two we could have put it over here and here that is as late as possible, so you are bringing in towards the down and if you observe that as late as possible, so we have the adder over here and the subtraction over here.

Now, we are going for as soon as possible, so we moving in towards the towards the upper end of it, now we require the first requirement what happens. So, you require two multipliers, one subtracted in the first case and then one divider over here, you can use multiplier, you can use one adder, you require one adder, you require in this case and here the multiplier subtracted have been reused. This can be reused over here, the divider can be reused over here and subtracted can be requirement. Now, this is your

requirement resource requirement s, these two multiplier one subtracted one divider and one adder, so this is the most optimum solution we have got.

So, you see that as soon, as late as possible if you apply over here and as soon as possible if you apply over here you are getting the best solution. So, most optimal solution for this example, so what is basically you are essentially you want to show you are that as soon as possible as late as possible are actually heuristics algorithms. They are not exact algorithms, so for some example one will give a better solution and from, some give other will be giving you a better solution.

So, next we try to design a heuristics which will actually take high bred of as soon as possible and as late as possible. So, I mean it is very difficult for us to first try as soon as possible in one part of the circuit and then try as late as possible in the other part of the sub circuit. Then, find out which one of them be give a better result like in this case we first tried as soon as possible, then we tried as late as possible, then we found out for this part of the circuit apply as late as possible. We apply as soon as possible is a very difficult way of solving a problem, so next we will see another heuristics.

(Refer Slide Time: 26:54)


ASAP versus ALAP

It may be noted that the resource consumption in this case is as follows.

- Step1: 2 Multipliers + 1 subtractor
- Step2: 1 Divider + (multiplier from Step1 can be used) + 1 adder
- Step3: 1 subtractors + (subtractor from Step1 can be used) + (divider from Step2 can be used)
- Step4: (subtractor from Step3 can be used)

So it may be noted that a schedule which is a "mix of ALAP and ASAP" provides better solution than by the individual algorithms.

Now we will see FDS scheduling algorithm which is motivated from above fact of combining ALAP and ASAP.



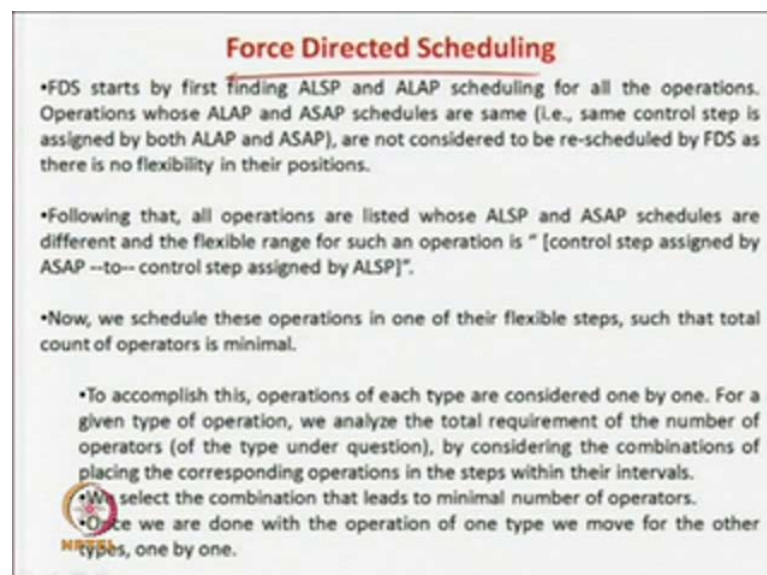
This is actually called first step of scheduling, this is actually high bred of as soon as possible and as late as possible. So, what it will do as we will see in the heuristic, so it takes ideas from both of them and final solution will be automatically wherever possible, it will be good as late as possible will be applied and wherever it is bad. So, I mean

appropriately things will be applied, so what is the resulted, I mean effect of the heuristics. So, which part of the circuit if in a part of circuit, it as soon as possible is better than as late as possible.

So, automatically as soon as possible will be applied and vice versa for the other case, so that will be happening automatically, here we have that to as soon as possible that you are gone for as late as possible. Then, you have tried the high bred of both, we are one part we applied, so it is very bad way of doing algorithm, now we will see the algorithm in which case that automatically taken care by the high bred nature of the algorithm. So, automatically for which part of the circuit it is required that heuristic in between as soon as possible and as late as possible would be selected.

So, this is what our discussion was about as soon as possible and what is as late as possible, so finally we achieve this one which is the best solution one adder one subtracted two multiplier, then one subtraction. So, multipliers 1, 1 subtracted, so these are which was the requirements are actually reused, these are all reused. So, that is what we have said; now we will be combining as soon as possible and as late as possible so that the algorithm is called forced directed scheduling.

(Refer Slide Time: 28:13)



Force Directed Scheduling

- FDS starts by first finding ALSP and ALAP scheduling for all the operations. Operations whose ALAP and ASAP schedules are same (i.e., same control step is assigned by both ALAP and ASAP), are not considered to be re-scheduled by FDS as there is no flexibility in their positions.
- Following that, all operations are listed whose ALSP and ASAP schedules are different and the flexible range for such an operation is "[control step assigned by ASAP --to-- control step assigned by ALSP]".
- Now, we schedule these operations in one of their flexible steps, such that total count of operators is minimal.
- To accomplish this, operations of each type are considered one by one. For a given type of operation, we analyze the total requirement of the number of operators (of the type under question), by considering the combinations of placing the corresponding operations in the steps within their intervals.
- select the combination that leads to minimal number of operators.
- Once we are done with the operation of one type we move for the other types, one by one.

So, what is that, so what do what do you do in forced directed schedule is in forced directing scheduling. So, what will do so first take will be actually go for as soon as possible schedule once and as late as possible schedule once? So, we will in large, also

you have been see in last case also what we have done what we have gone for as soon as possible. Then, we have gone for as late as possible, then you are trying to find out that which part of circuit which is better. That is actually good, I mean what you call a fussy way of stating something may be find out which part of circuit or which sub part of the circuit of the better.

So, here actually you have to going the way very clips steps of the algorithms, so what is a idea or idea we are basically trying to find out which in which the part of the circuit which is the better algorithm. So, obviously what you have to do you have to apply first as soon as possible and as late as possible. Then, try to choose some high bred amount, but the high bred nature what we are trying the last example was not good that is manually we try to find out that this part of the circuit. This is good that part of circuit that is good that is not a very good of good way of doing that, so what will doing force directing schedule we start by as soon as possible and as late as possible.

So, we do that, then what to do, then we find out some flexible operations like what do you mean by flexible operation like if you look at this if you look at this this thing. So, if you these are the only operations where there is flexibility that is you can move this two steps up or two steps down. Similarly, this can be move on the here or the here, but this part of this that is what you call schedule is almost as soon as possible and as late as possible. So, there is no flexibility over here, so either you apply as soon as possible or either you apply as late as possible with this sub part nothing is going to change because you cannot move any of the operation.

So, your aim is given to that is as show, so this with this aim is equal to 4, you cannot move this operation to this operation, so we this is actually either like as soon as possible and actually you can either as soon possible or as late as possible and fix it. So, that is what that is actually saving sum of a computation effort, so if you think that in the whole circuit first I will try as soon as possible. Then, I will go for as late as possible, then I will go for some high bred in the whole part of circuit that is will actually computation time they will take more computation time.

So, what do you have found what is the first step of force direct f d s schedule if you first as soon as possible that you gone for as late as possible. Then, you will trying to find out which part of the schedules is fixed that is there is no movement, so in as soon as

possible or as late as possible nothing could be move. I mean some operations, it will be same that means what means there is no change in the schedule if we apply the as soon as possible or as late as possible. So, both them were give equivalent results and in other words they are non flexible you cannot move they cannot move from their position.

So, you better fix them and try their heuristics with these type of notes or this type of what do what do I say these type of operations where there is flexibility. Now, what is the first step, so you say that we give a actually a range, so what is the range the range is actually control step assigned by as soon as possible to control step by as late as possible like for example, in this case this guy can be scheduled over here. This guy can be scheduled over here, so it is range 1 and 2 and 3, so this guy can be scheduled over here or this guy can be scheduled over here to its range 2 and 3.

So, this subtraction operation can be scheduled here or here, so its range is between 1 and 3, similarly this one can be scheduling 2, 3 and 4, so ranges 2, 3 and 4, so you just find out the range first. Then, what do you do, then we take each type of operation this one at the time like addition operations multiplication, then subtraction, then division something like that and then tried to freeze one at a time because you see what is the basic modification here. So, actually adder cannot do a subtracted that is the assumption so in a very I mean advance praline adders and also do subtractions subtraction can be there.

Actually, same elements and also you can think that multipliers are multipliers can also be treated as dividers and divider can also be treated as multipliers; sometimes we have general purpose blocks. So, for the simplicity, you have been assuming that adders can do all the addition and subtracters can only do. So, what you are trying to do, so if you are parallel scheduling one addition one multiplication one subtraction and one division, we are not have any problem because you require one block each, but if you subtract in one time step you are parallel assigning.

If you are parallel assigning two multiplies or three multipliers in one, then you are had a problem because then only you are going to require more number of other harder element like in time step one. If you scheduling three multipliers, then you require three multiplier block, so I mean you can understand that the heuristics f d s things in this way. We take only multiplier operations in a time optimism, then go for as subtraction, then

optimizing then go for addition, subtraction and optimization. So, for every broad sets, I mean you can think that the optimization of the dividers will not have much impact on the optimization of the adders, but obviously there will be some impact that we will see in the end.

If you fix one operation in one step, then actually flexibility of the other operations will get change because of that there can be some problems so that will be looking at the questions and answer session, but broadly. You can think in this way that if you are talking about optimization of adders just think about adders, if you are going as multiply let us think about multipliers. So, for that we have to think that of because if we consuming three hardware multipliers the one time set, then you require actually three hardware three multipliers, but even a single step you add schedule one adder one multiplier one subtraction, one divider.

Then, that one is not of a concern, so to do this operations of each type are considered one by one for each given type of operation the total requirements of the operation under question at the minimized. We select the combination that lead to minimum numbers of operations and when you are done with one type of operation we for another. So, what basically says that we first take multipliers and then we try to make a schedule such that multipliers minimum number of multipliers is required. Then, once the multipliers are done then you go for adders, but actually I mean this will not always require optimal solution because sometimes when you are thinking about the multipliers.


We schedule some multiplication operation, but based on data dependency the flexibility of the adders or the flexibility of the subtraction also may get chased change as of sometimes you may not get the most optimal result. So, that we will be very much clear when you will take the example in case of the question and answer session, but for the time being just think of this heuristic idea is something like this.

(Refer Slide Time: 34:17)

Force Directed Scheduling

Before providing the algorithm for FDS scheduling certain notations are introduced.

- $ASAP_i$: Control step scheduled by ASAP algorithm to operation o_i
- $ALAP_i$: Control step scheduled by ALAP algorithm to operation o_i
- $INTERVAL_i: [ASAP_i \text{ to } ALAP_i]$
- $RANGE_i: ALAP_i - ASAP_i + 1$
- $PROB_{i,j}$: Probability of scheduling an operation o_i in control step j ,
 $j \in INTERVAL_i; PROB_{i,j} = (RANGE_i)^{-1}$
- $LIST_{k,j}$: Set of all operations of type k in step j , i.e., set comprising all operations o_i of type k such that $j \in INTERVAL_i$.
- $COST_{k,j}$: Number of operators of type k required in step j .



$$COST_{k,j} = \sum_{o_i \in LIST_{k,j}} PROB_{i,j}$$

Then, go for the other and before we go to the exact algorithm and the example I mean examples. Let us define some formally you have to define some variables like as soon as possible that is the actually control step to operation by as soon as possible algorithm ALAP i. So, what does it mean control step assigned to operation i by ALAP algorithm what is interval i that is as soon as possible to as late as possible because say as soon as possible to operation one is one and as late as possible to operation 1 is say 5.

So, interval i is equal to 1 to 5 kind, so what is range as late as possible i minus as soon as possible i and 1. So, if it is actually as late as possible is 5 this is 4, so what is range this sorry this is 1, so 5 minus 4 plus 1. So, range is actually 2, sorry 5 minus 1 plus 1, so range is actually 5 so that means range says that within what is the distance or what is the boundary in which an operation can be flexibly scheduled. Then, probability i j is actually this is the probability of schedule operation i in step j in step j operation i in step j and what is step j.

So, step j should be in internal i that means what actually so if there is this operation o i, now we says that the time step three, now it is probability i j says that what is the probability of scheduling operation two which is this operation two in time step three. So, that is what actually, then I mean then what should happen that time the operation two intervals should be within 3 should be within a interval of o 2 because 3 should o 2 should be feasible. We schedule in time step three so that it says and then what is

probability of this one is actually one by range, so in this case we assume that as soon as possible schedule was 1 and this was actually 5.

So, range is actually 5 minus 1 plus 1, so to 5, so what is the probability that say this operation is schedule in a time step between 1 to 5 is actually 1 by the range that is 1 by 5. So, it is actually 1, 2, 3, 4, 5, so 1, 2, 3, 4 and one step is 5, so it can be scheduled from here and here. Anywhere it can be scheduled, so what is the range is actually 1 to 5, so that is range is 5, so what is probability that will be scheduled in any time step between 1 to 5 is actually 1 by 5 that is what the probability it is saying. Now, what is about the probability of i, j , then finally we have another thing is call the list k, j .

So, what is the list k, j is say that set of all operations of type k in step j that is actually say in this step. So, you can schedule say one multiplier operation that o, i is multiplier operation, and then you can say that o, j is actually addition operation. So, it says that in time step j set of all operation types k that is it can be multiplier adder, subtracted, divider whatever is possible in that time step. So, that will actually call the list j and obviously I mean that that says that that is set comprising all operations of type k such that j belongs to interval i .

That is what I have told you that in formal was that it says that list i in a time step we say what the different types of operations are which is possible to be scheduled over there. Finally, the last is actually its cost i, k , so that is number of operations of types k required in step j that is say for a example in one step you require say two multipliers. So, I mean cost will be actually two of type multiplier, so that is actually because and this is actually the formula of calculating this one that you have to submission all the probability values. So, I mean just take this expression whenever example, this mathematics will become very clear.

(Refer Slide Time: 37:44)

Force Directed Scheduling

Algorithm 3: Force Directed Scheduling

Input: ALSP and ASAP Scheduling.

Output: Control step for each operations, Status of scheduling .

Steps

From ASAP and ALAP scheduling, for all operations (i.e., $o_i \in O$) compute $INTERVAL_i$, $RANGE_i$, $PROB_i$ (for all i), $LIST_i$ (for all i), $COST_i$ (for all i).

for each type of operation $k \in K$

DO

BEGIN /*loop finds best steps for all operations of type k */

$\Delta_{k(max)} = \infty$;


$best_step = 0$;

for each operation o_i of type k whose $RANGE_i \geq 2$.

BEGIN /*loop finds best step for o_i */

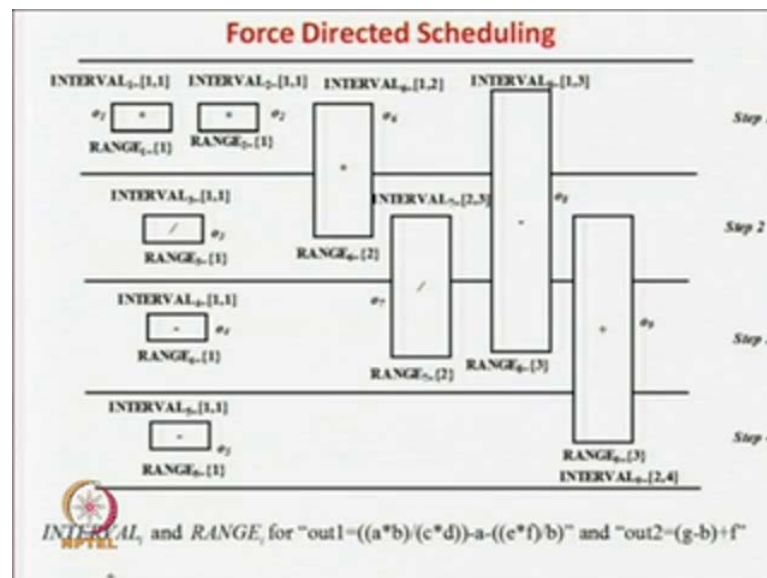
for each $j \in INTERVAL_i$

BEGIN



Now, what is actually force directing scheduling will do so what I feel that I will first go to the example, and then I will come to the algorithm.

(Refer Slide Time: 37:52)



Then, it will be algorithm and the steps to otherwise I mean the algorithm is being more mathematical, so first example explaining with steps is a very good idea of doing it. So, this is whole example we are going to take and so what is the thing we already saw that operation 1, 2, 3,4 and 5. So, these operations have no flexibility in as soon as possible or as late as possible, they are actually hardcoded in that. So, what range of one

and range two is one why because it is one and a 1 and 1 and a 1 for 0 1 and because they cannot be shifted.

So, in case of range two what is the interval, so interval as actually s, sorry the range is one in this you can write 2, 2, I am sorry this three, this is very this is 4, 4, sorry there are some issues over here. So, it is from one and a 1 and a 1, so what will be what will be the range, so range in this case will be 1 minus 1 plus 1 kind, so this is one, so in this case you have to 2 minus 2 plus is 1 is 1 whatever what do you mean by range equal to 1. So, range is equal to 1 means they are any kind of movement for this one interval.

So, as soon as possible type and as late as possible type, so in this case it is fixed, so this bothered about one because nothing can be done over here. Now, let us look at the o 6, so we saw that o 6 can be scheduled over here also o 6 can be scheduled over here. So, actually o 6 can be scheduled over these two places, so it as soon as possible range is one as late as possible value is true. So, what is the range 2 minus plus 1 is equal, so range is 2, similarly o 7 can be scheduled in the time step 2 or time step 3, so it is 2 or 3, so what is the range it is 3 minus 2 plus 1 that is again nothing but 2, so the range of this one is 2.

Now, if you look at this operation a, so it could be schedule here or here, so this as soon as possible is one as late as possible is a 8. So, if you make a difference, so it is 3 minus 1 plus 1, so it is range is 3, so range is 3 about it and similarly for the 9. So, this is your range, so this is actually different ranges we have computed over here, so that is actually the four combinations of the range one different time steps.

Next, we are going to see what is your list, so if you look at it so what will be list over here, so in in this case the list will be multiplier and subtracted, what will be the list over here. So, list over here will be multiplier, divider, subtraction, so this way you can find out your lists, now what will go we will go for optimization one step other time. So, this we have found out, now what we are going to do we are going to take one operation at a time, so what is the one operation time is say it has start with the multiplier operation.

So, what is the multiplier operation that all the mathematics will come back and will and tell you, so what is the multiplier operation, so the multiplier operation is actually this one. So, what is the probability that it will be scheduled over here, it is the half, so what is the probability that is formula, let us what is the formula for this, so what is formula. We have to see this formula probability of scheduling is here is actually 1 by 2 that is the

range. Again, this one will be range, so it will be actually 0.5 over here and 0.5 here is scheduling property.

That is the probability of scheduling the multiplier in this step is 0.5 and this is because and the divider is also 0.5 over here 0.5 over here the subtracted is 0.33. That is very obvious is 1 by 3, again this one is 0.33, probability of scheduling this over here as all have the probability, actually you should also count the probabilities will be 1. You do not have any flexibility, now we will take the case of multiplier and see where the multiplier is to be scheduled.

Then, you can forget about the all others, so you see the multipliers can be scheduled over this multiplier scheduled here and it can be scheduled here to here. I put the multiplier, so the probability is 0.5 this is the 1, so if I put the multiplier over here, so what will be the probability, then it will change from the 0.5 to 1 and the probability and the numbers of multiplier required will be 1 plus 2 plus 3. So, it will be 1 plus 2, 1 plus 1, so it is equal to 3, so initially it was 2.5, so 1 plus 1 plus 0.5, now it is become 3 over here, so it change from 2.5 to 3. If I schedule the multiplier over here, so if I schedule the multiplier, initially what is the probability here it is 1 plus 0.5.

So, initially it is 1 plus 0.5 is the probability, now if I schedule from this multiplier from here from 1 to plus 2, 2, so it will be 1 plus 1 over here. Then, it will be time step is actually equal to 2 and there is no other multiplier over here, so this actually stops of computation of multiplier. Now, you see if I schedule a multiplier here, then the probability cost I should tell about the cost because actually the summation of the probabilities. So, cost in this case is 1 plus 1 plus 0.5 which has changed to 1 plus 1 plus 1, so it is 3 and here if I schedule it here, so what will be the case it will be 1 plus 0.5.

It will be changed from 1 plus 1 plus 1.5 to 2, so you see in this case it is 2 and this case is 3. So, in this if I schedule this in time step one be maximum cost will be 3 and in this case cost is 2, so obviously I will go for scheduling this guy over here because the cost will be multiplier to the 2 over here. So, multiplier case is done and this is scheduled over here, now if I schedule the multiplier over here. Then, a very interesting thing will happen then this guide will be fixed over here. Then, the flexibility of this divider is no longer here, so it will be actually fixed over here and it is range will be fixed, you cannot do any kind of changes with it.

So, that is actually there, but in this case you could because you are schedule you what you call this multiplier over here. So, actually it is pushing down this one and you also saving on the divider range, but this constraining that you are scheduling this first is actually constraining this way of may not sometimes give you a advantage here to getting a very good advantage. This multiplier you are scheduling it over here, so divider is pushed over here, so you can you can reuse these dividers what had it been for some reason this is scheduled over here.

That may that may lead to other case that will see in the example in the question answer sessions, but what essentially has happened that this you have scheduled over here to reduce the probability. This is actually pushed over here, now fix this is range will be one, its range will be 3, 3 and it will be 1.

(Refer Slide Time: 44:23)

Force Directed Scheduling

Algorithm 3: Force Directed Scheduling

Input: ALSP and ASAP Scheduling.

Output: Control step for each operations, Status of scheduling .

Steps

From ASAP and ALAP scheduling, for all operations (i.e., $o_i \in O$) compute $INTERVAL_i$, $RANGE_i$, $PROB_i$ (for all j), $LIST_{k,j}$ (for all j), $COST_{k,j}$ (for all j).

for each type of operation $k \in K$

DO

BEGIN /*loop finds best steps for all operations of type k */

$\Delta_{k(best)} = \infty$;


$best_step = 0$;

for each operation o_i of type k whose $RANGE_i \geq 2$.

BEGIN /*loop finds best step for o_i */

for each $j \in INTERVAL_i$

BEGIN



Now, we can take another operation which is the sub divider operation and finally we will go for the adder operation that is the basic idea. So, we will go for the formal algorithm, now what we are going was the formula algorithm, so what are the inputs as soon as possible and as late as possible. Then, control step you have to give in the output and you have give the scheduling, now what you do using the as soon as possible scheduling for all operations. You compute into the range probability list and to cost, so whatever variables you have already done where to do that, now for each operation k.

Now, here the actually operation type is very important, we are going of a type of a operation like adders then multipliers then subtracted divider.

In other way, you go for a multiplier subtracted and divider and now for each operation you repeat, so what you do first you set that the best case is here, best value of the cost basis in infinite, say for example the cost of some resource you consider to the infinite. Then, you initialize the best step to be 0, so this this is the initialization step to initialization, we say that cost require is very high and the base step is 0, this is no step 0, this is the initializing operation.

Now, for each operation of type k whose ranges is greater than or equal to 2 because if range is 1, already we have seen that there is not flexibility in movement. So, you should forget about them, so wherever they scheduled their probability is 1 and there scheduled over there. So, you can just keep it over there, so you need not bother about this, so were where have you bother, so what where you have to bother in case of operation whose range is greater than equal to 2. Now, you begin, so this move actually find out the best step for operation o_i like we have already done in the multiplication operation already we have seen that that this o_6 the best step was here.

That is what will found out by this algorithm and in the first step we actually this each type of operation here with started form the multiplier example if you consider, you start for the multiplier for each operation which have to do in this. You have started with the multiplier operation, the example then for each inter for each j in the interval i , then what is what did you say to do we have taken this operation. Now, you have to see if each step which is in the actually interval of this operation in this case there is 1, 2, so in each that is what we say that e each in interval each j for each which is the interval of i .

(Refer Slide Time: 46:27)

Force Directed Scheduling

Temporarily schedule o_i in step j and compute the value of $COST_{k,j} (= \Delta_{k(i,step)})$ due to fixing the schedule of o_i and changes of schedule of other operations due to data dependency.


If $\Delta_{k(i,step)} < \Delta_{k(best_step)}$ then assign value of $\Delta_{k(i,step)}$ to $\Delta_{k(best_step)}$ and $best_step=j$.

END

Finally schedule o_i in step $best_step$ and other operators due to data dependency.

Update for all operations (i.e., $o_i \in O$) $INTERVAL_i$, $RANGE_i$, $PROB_{k,j}$ (for all j), $LIST_{k,j}$ (for all j), $COST_{k,j}$ (for all j).

END



You temporarily schedule o_i in step j and compute the value of cost due to fixing the schedule of o_i and the changes of schedule of other operations due to data dependency. So, what does it mean it means that you schedule, so you take this one take o_6 multiplier that is the first operation, like then you actually in both the ranges, so you schedule this guide over here. That is what been say that use temporarily schedule in the o_i in this and compute the value of cost due to fixing of o_j and changes of schedule of other of operation of data dependency.

So, if you schedule this over here, so this can be scheduled over here or here no changes here is also no fixing and no fixing. So, this one will be there that you have scheduled it over here or sometimes it may happen that if you schedule a operation in a time step other things get may get constrained. It may be a hardcoded and then we have to find out the cost, so in this case what is the cost over here it is 1 plus 1 plus 1 instead of 0.5. It is now become 1, so this cost is actually 3, so you have to record this cost, so we have record this cost in some variable that is k_{nu} , now if k_{nu} is less than k_a .

Then, assign the value of k_a to k_{mn} , so initially we have started with k_{best} as infinity, so k_{best} was infinity and now we have got the value of 3. So, obviously this is the k_{best} , so you have to apply k_{mu} which was 3 to k_{best} and time step as j . So, in you apply find out this is the and you schedule it over here, so this is the first step, now again you come back for each interval, now what you do now again you have to repeat because

this is also in the interval. Now, you forget about now you schedule it in this interval now what will cost is 1 plus 1, it will be actually equal to 2, so from 2.5 which have not become into initially it has 2.5.

Then, if you scheduled it becomes 3, now if you schedule guy in this second place in disposable in this rate, so it will become to and this one also become 1 plus 1. It will be 2, so again we are actually scheduling in this interval initially in the term going at stage 2. Now, in this case what is going to happen, sorry in this case you are temporarily scheduled o_i in step j . Now, it is 2 and computes the value of this one, so in this case the computation value and you have $k m u$ equal to 2, so $k m u$ is 2 and k best is 3. So, we have if this is the case sorry $k m u$ is 2 and k best was 3 in the earlier case, now what happens, so if this is the case assign the value of $k m u$ to 2 k .

Now, your k best is 2, actually now here you will find out then again will go back to this look then you have to find that the interval has only 1 and 2. So, your case is done, so you find now you what you do, now you finally schedule o_i in the best step or all operation will be data dependency s . Now, what you have what you actually have done you scheduled it here and found out the value to be 3 that was the best. Now, we have again there was another interval, it could sorry another step which could have procedure. Now, you schedule it here and then you find in the best value to be 2, so 2 is less than equal to 3.

So, you schedule the value of o_6 over time step 2 and you are done, so this is o_6 will be actually scheduled over here because that value is 2. So, that is done, now again it says that update of all operations everything all the values you have to update it and due to the data dependency and because something will be freeze. Now, you see already told you because if you said this yet now this guy cannot be this this was o_7 cannot be placed over here. This can be placed only over here, so all ranges that is extra everything will be changing, now this range will be one because this interval will 3, so it will be only one, so this range will change so all these computation.

It is like a cost excreta like interval and range probability list cost everything we read and again, you do this, again you come to this look and then you start doing for other operation may be for adder may be subtracted or dividers.

(Refer Slide Time: 50:13)

Force Directed Scheduling

Now we will illustrate the FDS algorithm with the running example of scheduling "out1=((a*b)(c*d))-a-((e*f)b)" and "out2=(g-b)+f".

Next Figure illustrates *INTERVAL_i* and *RANGE_i*, for all the operations. Here we have four types of operators; let $k=1$ represent multiplier, $k=2$ represent divider, $k=3$ represent subtractor and $k=4$ represent adder.

Now we will illustrate FDS for scheduling all operations of type $k=1$. Computation of $PROB_{i,j}$, for all the multiplication operations are as follows.

- $PROB_{1,1}=1; PROB_{2,1}=1; PROB_{6,1}=0.5;$
- $PROB_{1,2}=0; PROB_{2,2}=0; PROB_{6,2}=0.5;$

Further $LIST_{i,j}$ for the first two steps for multiplication operations are $LIST_{1,1}=\{o_1, o_2, o_3\}$ and $LIST_{1,2}=\{o_4\}$. So, $COST_{1,1}=1+1+0.5=2.5$ and $COST_{1,2}=0.5$.

So, these how it happens, so let us I mean just look at the example, so initially this was an example, so this range etcetera is already being given over here. So, in the example $k=1$ represents multiplier $k=2$ represents a divider $k=3$ represent subtraction and $k=4$ represents adder, so probability of 1, 1, 2, 1, first we are trying for the multiplication. So, probability of one already told one then probability of 1, 2, 1 is 1. So, to 1 is a actually 1 because this things can be hardcoded over there and then probability of 6, 1 is 0.5 that is 6 can be schedule over here is 0.5.

Then, again probability of 1, 2 that is probability of 1, 2 in this one, sorry one probability of one it this is one into be actually no longer there probability of 1, 2 is 0, why this is 0 because operation one cannot be scheduled in time step two.

We are actually concentrating only for two time steps over here you should actually see for all the other time step because the multipliers do not get any concern in time step 3 and 4 the multipliers are only mean limited two time step one and two. So, we are actually listing for time step one and two over here, so probability of 0 any that given operation one in one step is 1, 2 is 1. What does it say that is operation one probability in time step one is one operation two probability in one again operation one in time step 2. That is actually no not possible because 1, 1 cannot be scheduled over here that is what is written and similarly 2, 2 is also 0 because 2 cannot be scheduled over here.


This 1.5 and similarly 6, 2 is also 0.5 because 6, 2 can probe o 6 can be scheduled in time steps 2, so this which has been calculated and the list is actually a what is the list cost for time step one it is over o 2 and o 6. So, what are the list mean the list mean that o 1 o 2 and o 6 can be scheduled over here because we are already taking care of the multiplication. So, this is how we are calculate the then o list of 1, 2, so list of 1, 2 will be actually only 1, 6 will be there here. So, is only 6 over there because 6 can there is only one multiply that may be possible over here. Similarly, you can find out the cost is 2.5 and the cost here is 0.5, already we have shown.

(Refer Slide Time: 54:25)

Force Directed Scheduling

Among three multiplication operations, we have freedom only in scheduling o_6 ($RANGE_6 = 2$). If we schedule o_6 in step1 then

- $PROB_{1,1}=1; PROB_{2,1}=1; PROB_{6,1}=1;$
- $PROB_{1,2}=0; PROB_{2,2}=0; PROB_{6,2}=0;$
- $LIST_{1,1}=\{o_1, o_2, o_6\}$ and $LIST_{1,2}=\{ \}$.
- $COST_{1,1}=3$
- $\Delta_{1,1(new)}=3$; as $\Delta_{1,1(new)} < \Delta_{1,1(best)}, \Delta_{2,1(best)}$ is assigned 3 and $best_step=1$



I have told you we have already shown this how we generate the cost this is how everything has been done over here. Now, actually what we do now we have freedom of scheduling o 6 the in o 6 range 2 in if time step if we schedule in time step 1, then actually what happen the cost is 3 already we have told the cost is 3 than best step is one. That is the best is that is the first step when we have scheduling o 6 in time step one, then what we do, then we actually schedule an also schedule o 6 in time step 2.


(Refer Slide Time: 52:45)

Force Directed Scheduling

We can also schedule σ_6 in step2, which results in

- $PROB_{1,1}=1; PROB_{2,1}=1; PROB_{6,1}=0;$
- $PROB_{1,2}=0; PROB_{2,2}=0; PROB_{6,2}=1;$
- $LIST_{1,1}=\{\sigma_1, \sigma_2\}$ and $LIST_{1,2}=\{\sigma_6\}.$
- $COST_{1,2}=2$
- $\Delta_{1(new)}=2$; as $\Delta_{1(new)} < \Delta_{1(best)}, \Delta_{2(best)}$ is assigned 1 and $best_step=2.$

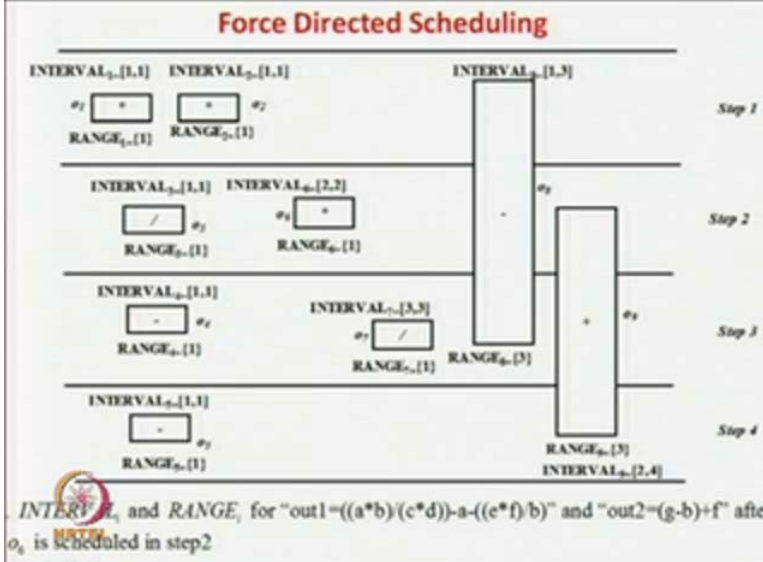
So we schedule σ_6 in step2, which results in fixing the schedule of σ_7 in step3 (due to data dependency); σ_7 loses its flexibility. This is shown in next figure .



If you do this, you have values of probability values the list values and the cost will become two as already we have seen. Now, we have seen that two is actually less than three so the best step is two and there is no more range where σ_6 can be scheduled it can be scheduled only one. So, we stopped over here and we actually schedule σ_6 in σ_2 , but now actually σ_7 will lose its flexibility.

(Refer Slide Time: 53:08)

Force Directed Scheduling



$INTERVAL_{\sigma_i}$ and $RANGE_{\sigma_i}$ for "out1=((a*b)*(c*d))-a-((e*f)/b)" and "out2=(g-b)+f" after σ_6 is scheduled in step2

So, this guy will be you have to be scheduled over here there is no only this thing over here, now again you are actually having these two which you can play with. So, if you again I mean this is this part is fix, now you have to think of the adder and the subtracted. So, we can find out that very easily we finding of that if we put the subtracted can be put over here put over here and put over here. We put the subtracted over here the cost will be higher because the probability of subtracted, then one is one over here and if you put it here, so it will be 1 plus 1 it will be 2, but here if you to put it will be one only because this is one.

This can be one and again if you put the subtracted over here, the probability will be one and the cost will be one over here the cost will be one because there is no subtracted here no subtracted here, you put the subtracted in time step three. So, it will be 1 plus 1, it will be 3, so I mean if your f d s will put the subtracted over here, so if you put the subtracted over here, then again this guy this adder will have a flexibility of here and here now you will be easily able to find out that.

Again, you will find out that in this case mean there is no other adder over here, so you can either if you placed here and here because everywhere the cost will be over.

(Refer Slide Time: 54:33)

Force Directed Scheduling


Similarly, computation of $PROB_{i,j}$, for all the subtraction operations are as follows.

- $PROB_{i,3}=1$.
- $PROB_{i,1}=0.33$; $PROB_{i,2}=0.33$; $PROB_{i,3}=0.33$;

Further, $LIST_{i,j}$ for the first three steps for subtraction operations are $LIST_{i,1}=\{o_1\}$ and $LIST_{i,2}=\{o_1\}$ and $LIST_{i,3}=\{o_1, o_2\}$. So, $COST_{i,1}=0.33$, $COST_{i,2}=0.33$ and $COST_{i,3}=1.33$.

Among two subtraction operations we have freedom only in scheduling o_1 ($RANGE_i=3$). If we schedule o_1 in step1 then

- $PROB_{i,3}=1$.
- $PROB_{i,1}=1$; $PROB_{i,2}=0$; $PROB_{i,3}=0$;
- $LIST_{i,1}=\{o_1\}$, $LIST_{i,2}=\{\}$ and $LIST_{i,3}=\{o_1\}$.
- $COST_{i,1}=1$

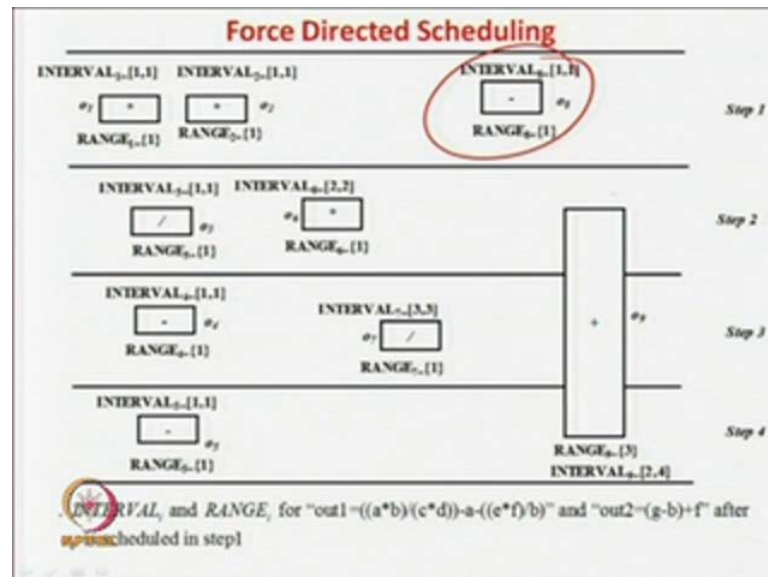
 **MPTEL**

- $\Delta_{i(new)}=1$; as $\Delta_{i(new)} < \Delta_{i(best)}$, $\Delta_{i(best)}$ is assigned 1 and $best_step=1$

So, your adder will be over here, so everything will be actually done, so these are all the values of I mean all calculations which we have done before this operation power 3. So, you can see that that subtracted is probably 0.33, it is showing, so if you it shows that

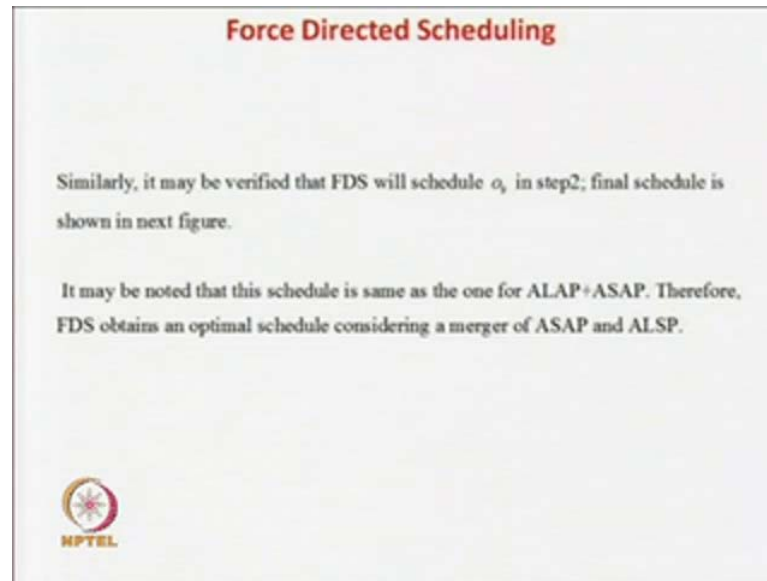
among the operation I mean if you schedule o i in time step one, then this will be the value. So, if you schedule o 2 in time step to the cost is one if you schedule o time step one cost is the one, if you schedule o 8 in time step 1, 2 the cost is 1, but if you put o 3 in time step 3.

(Refer Slide Time: 55:28)



The cost becomes 2, so that is not so good, so you will schedule o 8 in either time step one or time step two, so in this case let us see schedule this one in time step one, now it is the left to the adder. Similarly, we will able to find out that there is no more adder, so the cost will be one wherever you put in time step in 2, 3 or 4.

(Refer Slide Time: 55:48)



So, anywhere you can put it, so the final step o 9 is actually placed over here that can easily visualize and finally you may be note that whatever we got over this one adder over subs will be here. Whatever we got is actually same as the optimal solution, we got by merging as soon as possible and as late as possible, so manually what we have done we can say that this part of the circuit, we apply as soon as possible, sorry as late as possible. This is apply as late as sorry as soon as possible and this is as late as possible you individually do that you do that to get the optimal solution what here we not do anything like that.

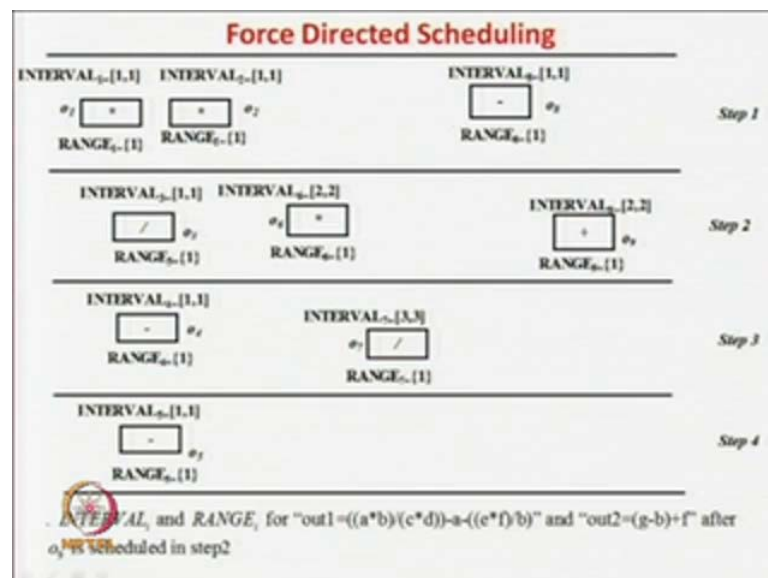
Automatically, the algorithm actually settled out in, a in I mean what you call a settled out in a schedule like this. So, whatever what do we find out that force directed scheduling actually it takes modification from the high bred nature of as soon as possible and as late as possible. It will automatically find out cases when which have to be apply and it actually does it for you.

That is what is a force directing schedule, so this another heuristic though which actually takes care of your I mean high bred I mean sometime as soon as possible is greater. Sometime as late as possible is greater for a more general class of circuit force directing scheduling is better, but you still you can find out the example where focus directing scheduling is also will not give you almost optimal solution. All of them are heuristics

and we have trying to gain our type by finding out a near optimal solution rather than going for very complex algorithm.

All the three algorithm discuss today basically time constrain, so you have to fixed up the time we are we are not thing about the number of resources based on the time step 3 or 4. Among that, we are trying to the number of resources for example, like if we say that time step have taking time step four, but I take three multipliers to the job and somebody takes I also time step to do the job. I have tale two multiplier to do the job, obviously we go for the second person that is what is being tried by this three algorithms. You have studied the time is speaks based on the time we have trying to reduce the resources as much as possible.

(Refer Slide Time: 57:03)



So, in the tomorrow's lecture what do have to do, sorry this one is your last schedule I mean, so this is the final I mean output of this thing.

(Refer Slide Time: 57:13)

List Scheduling

- All the scheduling algorithms we discussed till now were heuristics based on time constants, in terms of number of control steps. Now we discuss another heuristic scheduling algorithm which is resource constrained—List Scheduling.
- Unlike ASAP, ALAP or FDS scheduling, which process operations individually in a fixed order, list scheduling handles each control step individually (in increasing order).
- List scheduling works by trying to schedule “maximum” number of operations in the control step, subject to resource constraints and data dependency.
- During the scheduling process, list scheduling uses a ready list (hence the name) to keep track of data-ready operations subject to data dependency.
- The ready list in a control step comprises those unscheduled operations that can be scheduled into the current control step without violating the data dependency
- As long as there are operations in the ready list that meet the resource constraints, operations are chosen from that list and scheduled into the current control step.

So, the thing that we are going to do next class we are going to proper list scheduling which case it will be more about a resource constrained scheduling. So, the resources will be fixed and depending on that you will have to go for whatever time in whatever minimum time you can achieve for them.

Thank you.