

Synthesis of Digital Systems
Dr. Preeti Ranjan Panda
Department of Computer Science & Engineering
Indian Institute of Technology, Delhi

Lecture – 13
Scheduling in High Level Synthesis: List Scheduling & Time-constrained Scheduling

Let us continue with the scheduling discussion, we studied the ASAP and ALAP schedules, and even though these apply to idealistic situations where you have infinite number of resources, between them they have some valuable information that could be used in a more realistic scheduling scenario.

(Refer Slide Time: 00:43)

ASAP vs. ALAP Schedule

- Nodes in **critical path** have identical schedule
 - schedule is fixed for these nodes
- For other nodes, the range of possible values is: $\sigma_{ASAP}(t) \leq \sigma(t) \leq \sigma_{ALAP}(t)$

ASAP

ALAP

$$\begin{aligned} \sigma(u) &= 1 \\ \sigma(v) &= 1 \\ \sigma(y) &= 2 \\ \sigma(z) &= 3 \\ 1 \leq \sigma(w) &\leq 2 \\ 1 \leq \sigma(s) &\leq 2 \\ 2 \leq \sigma(x) &\leq 3 \end{aligned}$$

(C) P. R. Panda, IIT Delhi, 2017
58

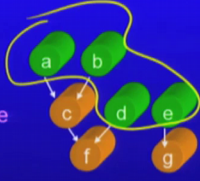
So, the information that we have here is that, if you make the ALAP deadline the same as the ASAP critical path in terms of number of cycles, then the scheduling ranges that you come up with scheduling range being defined as all the cycles all the states between the ASAP in the ALAP where that particular operation can fit, give us an idea about the flexibility that the scheduler has with respect to that operation.

So, we have these 4 nodes uv y and z having fixed schedules, which means that you do not really have any flexibility with respect to where to schedule them, for the others there is some range and there is some associated flexibility that is what comes out of the initial ASAP and ALAP schedules.

(Refer Slide Time: 01:44)

Resource Constrained Scheduling: List Scheduling

- Assign priorities
 - selection criteria among operations competing for resources
- At each control step (clock cycle)
 - Generate ready-list
 - set of operations whose predecessors are already scheduled
 - If size (ready-list) > #resources, schedule operations with highest priority



NPTEL

(C) P. R. Panda, IIT Delhi, 2017

59

This could be valuable input to some heuristic for performing scheduling, list scheduling is a standard heuristic for the resource constraints scheduling problem.

So, let us study this algorithm in some detail. So, the idea is you assign some priorities to the nodes, those priorities form a selection criterion among the different operations that are competing for the resources. So, the overall idea is very simple, you identify for each clock cycle or control step these are used interchangeably control step is a little more general mechanism.

But for now, let us just treat them as the same, they become different when you have conditionals and there are branches in the finite state machine. So, you generate for each control step a ready list that consists of the set of operations at the set of nodes, that are ready to be scheduled when does a node become ready to be scheduled when all its predecessors have already been scheduled, either it has no predecessor or all its predecessors have been scheduled. So, that is when it is ready. So, you generate a ready list if the size of the ready list is less than or equal to the number of resources that you have at your disposal, then there is no issue at all you just schedule all of the elements in the ready list to that clock cycle or that control step.

But if it is greater, then there is an issue we have to decide which subset of the operations to schedule, that is what is difficult in the most general case, although this is a local decision that is being taken the impact is more global, the impact of the choice of the

subset here. So, that is where this priority comes into the picture, and we try to schedule the operations with the highest priority. So, somehow the idea of a priority is associated with all the nodes, and you sort the nodes in order of their priorities and pick up the highest priority nodes for scheduling in that control step or clock cycle.

(Refer Slide Time: 04:14).

List Scheduling

- Algorithm leaves Priority Function undefined
 - can be customised
 - can be simple or sophisticated
- Popular priority function: MOBILITY

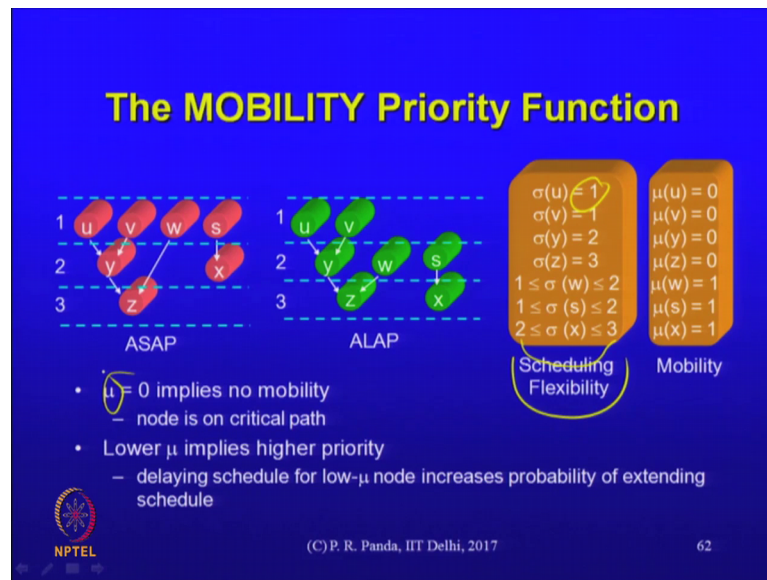
NPTEL (C) P. R. Panda, IIT Delhi, 2017 60

In fact, that is what the idea of list scheduling is of course, you can see that the point of that algorithm really is the definition of that priority function, the algorithm is sort of a meta algorithm where the overall strategy is given, but the quality of the results that you expect is of course, dependent on how good the priority function yes, you could customize it based on what you expect if there is any knowledge from the application or the particular graph or and so on whatever is being scheduled, then that can be incorporated in some way it can be very simple or it can be very sophisticated.

The time that the scheduling algorithm will take would be dependent on what kind of priority function you use, if it is a very local priority function it does not take into account too much of the dependent nodes and so on, then it could be very simple and efficient in terms of run time, but if it is a more complex one then it would take more time, you can expect a correlation between how sophisticated this priority function is, and the run time of the schedule.

So, let us take a look at a popular priority function called mobility, that mobility is just.

(Refer Slide Time: 05:36)

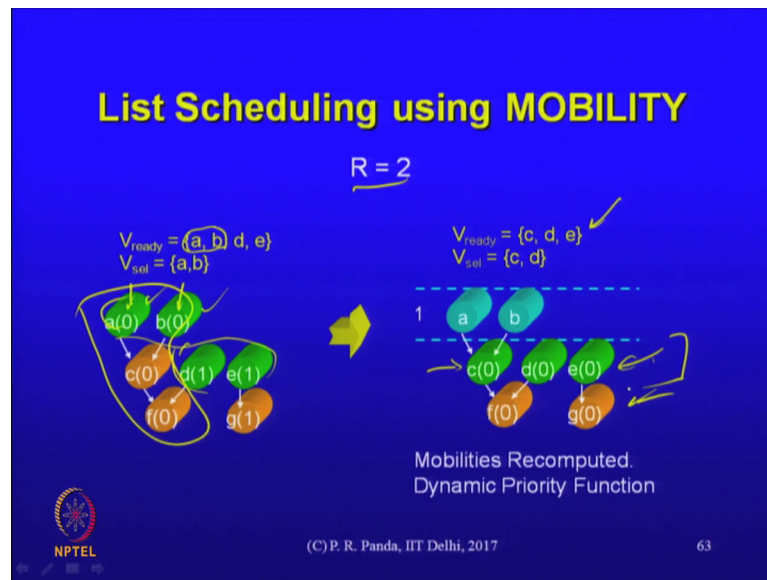


Defined for every node as the difference between the ALAP schedule and the ASAP scheduling, it is a very intuitive priority function for a node like s you see that the ASAP schedule puts s into clock cycle one, ALAP schedule puts it into the clock cycle 2, that is the range that we already identified this node has in terms of where it would be scheduled.

So, the mobility is just computed as the difference. So, if the difference is one it means that there is some mobility, if the difference is 0 then there is no mobility, that is the cost function and like we had seen earlier we had defined these we are determined these scheduling flexibilities, and based on that you can compute the mobility it is the difference between these 2 numbers if you have an associated range.

If you do not have a range, then that difference is 0 telling us that these are in some ways the highest operations. So, the node is on the critical path if the mobility is 0, and a lower mobility generally implies a higher priority. So, how is it useful in a practical scheduling algorithm, there is this information that is captured. So, which is if you delay a node which is on the critical path or which has low mobility, then that increases the probability of extending the scheduling.

(Refer Slide Time: 07:14)



So, let us just go through that algorithm with mobility as the priority function, this is the graph that we started off with R equals 2 just tells me that there are 2 resources let us assume that all the nodes are of the same type, that is why I left out what they were actually doing, but each takes one cycle and the 2 resources have to be used for determining a schedule for this graph, these numbers with the nodes are the mobility numbers, this is the priority function that is used by the scheduler, 0 means that all of these you do not really have a choice mobility is 0, this remember comes about from the assumption that the ALAP schedule deadline is the same as the critical path that you obtain in the ASAP scheduler.

If you give a different deadline, then you will have a different mobility number remember, the ALAP schedule tends to get all shifted downwards and therefore, difference would be obtained between the ASAP and the ALAP schedule, this is a function of the deadlines that you specify, but even though there is no deadline that is specified for the problem remember this is a resource constrained scheduling problem, means that the resources are specified deadline is not specified the objective is to just arrive at the schedule that minimizes the critical path, minimizes the length of the scheduling.

Still we could use the deadline to compute an ALAP schedule in that obvious way, you just determine how many cycles the ASAP schedule do and you give the same as the

input to the ALAP. So, my ready list consists of these 4 these 2 end, and these 2 the 4 that are highlighted in green and out of that I need to choose 2, 2 because R equals 2, I have 2 resources of that time.

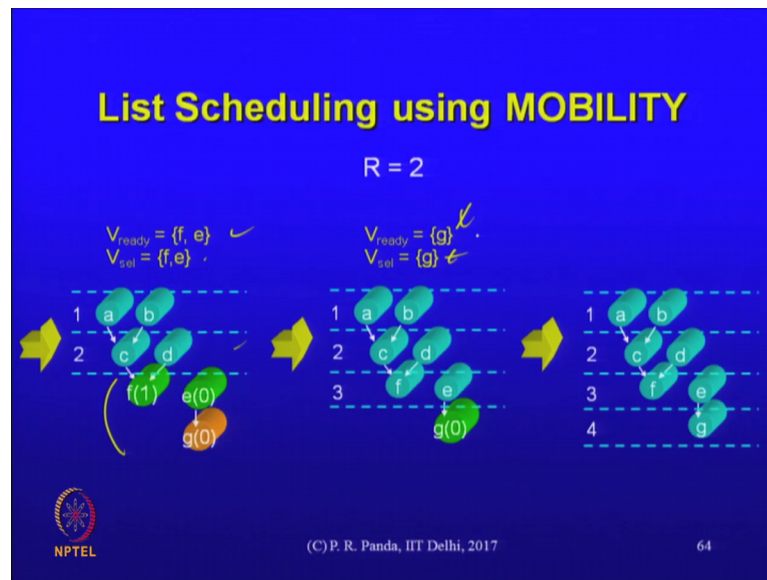
So, this is where the priority function is of some use, because the number of radial elements is greater than the number of resources, I have to decide which 2 to take, and the choice here is that I will pick up a and b because they have the highest priority. So, a and b are picked up because the mobilities are 0 whereas, the mobilities are one for the other 2 nodes that are ready, that leads to me dropping a b from here. So, the ready list now consists of just c d and e c got free just now. So, we add that into the ready list d and e were already there in the ready list.

Which ones do I select now, I can recompute the mobilities, now that the decisions have already been taken for a and b, for the rest of the graph ideally I would recompute the mobility, how would you recompute the mobilities you have to run the ASAP and ALAP again, now when you run you can see that all the nodes are on the critical path in this simple example, there is actually no flexibility if you run ASAP you would assign all 3 of them to the first clock cycle to the second, you would assign this ALAP is no different.

So, either way you need 2 cycles no matter how many resources you had, because of the nature of the dependencies here in between the nodes, you can see that all the nodes are on some critical path or the other of length 2, that is why all the mobility computations result in the same number. So, all 3 of these are of the same priority for a mobility computation. So, this simple priority function does not distinguish further, what happens if you get the same mobility for a number of nodes that is greater than the number of resources.

Let us randomly pick you could make it more sophisticated this is where there is some sub optimality, that is introduced in this simple priority function-based algorithm.

(Refer Slide Time: 11:48)



So, let us just pick up c and d for no particular reason, it is just a random subset out of the 3. So, if you schedule them into clock cycle 2 then that is what is left, and my ready list now consists of e and f and what would I select from there.

For the third cycle.

Student: (Refer Time: 12:11).

And you do not have much choice even though there is some flexibility for f , it seems as though g is higher priority than f , but because of the dependency on e , you are restricted to choosing f , do not really have a choice here we pick up f and e , that leaves g in the ready list and you would select them g this time you have the number of elements in the ready list being smaller than the number of resources. So, there is no choice to be made that is list scheduling.

Using mobility as a priority function, you can see that you could replace mobility by something else, and you would still have what is called list scheduling, list scheduling refers to this overall scheduling mechanism, where you do not revisit any decision once it is made, you make the schedule in decisions clock cycle by a clock cycle, but you do not go back and change anything, this is very simple and efficient from a runtime point of view, but relying on the mobility cost function or mobility priority function, means that we aren't really guaranteeing anything about the quality of the result.

This process will result in a fast relatively fast scheduling, but says nothing about the optimality, you can not prove optimality of the resulting schedule, where does the optimality question come in here you can see that there is one possible sub up to optimality in in this whole process, which is what is the sub set of nodes to choose when you have a choice, that is a totally non trivial strategy you could come up with an example yourself where this random choice well it wasn't random, but it was random when the priority numbers were equal.

But even that priority there is nothing sacrosanct about the definition of the priority, it seems intuitive it makes practical sense, but that does not lead to a theoretical guarantee of the best schedule, it is a good idea to think of counter examples that would defeat this strategy, for this simple example it worked out in an obvious way, but sometimes the choice would actually depend on what is happening later.

Here we didn't look at successors of the node as we were taking this decision of which subset to choose, of course the successors were indirectly accounted for in some way by the priority function itself, the mobility computation which resulted from ASAP and ALAP schedules, did take into account what is happening in the successive nodes, but this does not really make any guarantee and the way to understand is if we can construct some examples where you do not necessarily get a good result, for this particular example you can easily show that the result is optimal, what is the proof?

Student: why did we not take c d at the same time (Refer Time: 15:40) c d in the list (Refer Time: 15:42) same time like the 0 ones.

Why did we not take?

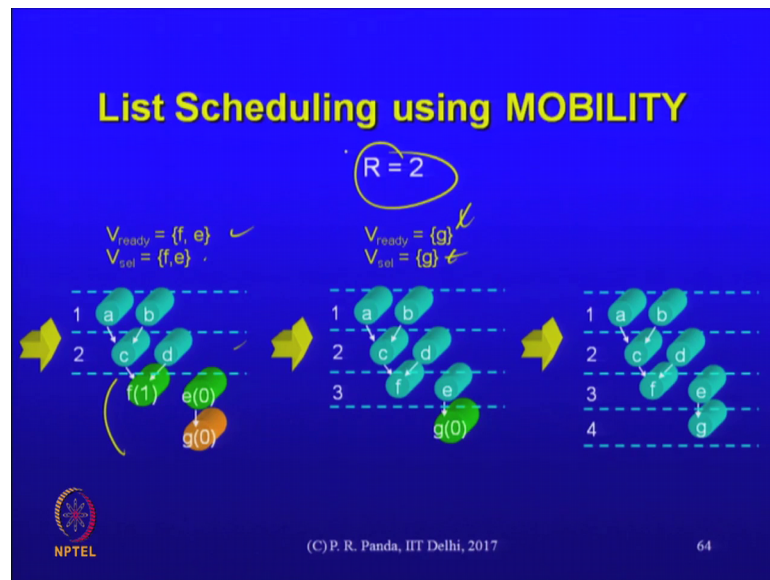
Student: cd

So, we had c d and e as all being free all being available in the ready list, your question is?

Student: So, why did we not (Refer Time: 15:55) at the same time c d like again we go to the e and z in the next one.

We scheduled c and d at the same time.

(Refer Slide Time: 16:04)



Student: and for the next time.

For the next time.

Student: So, (Refer Time: 16:07).

What is your question.

Student: (Refer Time: 16:09) equal the next time at the same time you can (Refer Time: 16:14) e could be in the second cycle.

You could have selected c and e instead.

Student: C d and e.

You cannot because R equals 2.

Student: R equals 2.

You have only 2 resources. So, you can only schedule 2 nodes per cycle.

Student: (Refer Time: 16:29).

Of course, there is no intelligence that went into the choice of c and d, this was a random subset I could have chosen c and e instead or d and e. So, I didn't a good priority

function might actually anticipate the consequences of a one subset or the other that is chosen, but what is the proof that you cannot do any better than this.

Student: (Refer Time: 16:50) exactly the total number of subsets available (Refer Time: 16:55).

Right

Student: depth of graph. So, if depth of graph is source side let say 4 that is the minimum number of cycles I would need.

What is the depth of this graph?

Student: (Refer Time: 17:08) a 4.

Depth is same as.

Student: (Refer Time: 17:10)

Height, what is the depth of this graph? Go back to the starting graph this graph, what is the depth?

Student: 3

3 and schedule is giving you 4 cycles, but it is not hard to prove that you can not do any better than this, you should be able to derive the proof immediately, for this particular example in the general case it is not clear, but for this you can show that you can do any better.

Student: So, we have 7 nodes to process and we have 2 resources.

Yes, you have 7 nodes to process, and there are only 2 resources how many cycles will you take.

Student: (Refer Time: 17:49).

4 is the minimum number of cycles. So, yes this is not a big proof, the dependency structures didn't really matter too much because this proof is simple.

But nevertheless, go back and come up with an example where this strategy is not good.

Student: if we have unlimited number of resources then that depth argument would (Refer Time: 18:16). So, if depth of my graph.

That is what the ASAP schedule we came up, with that is a guaranteed optimal you can easily show there if you find a path that is of length n , then you must require at least n cycles because of our assumption that each node requires one cycle of course, but the takeaway from here is that all the intelligence in list scheduling is in the construction of that priority function, this is a simple priority function that makes intuitive sense it does not guarantee your optimality.

Here too interestingly there is some further engineering efficiency that is involved, do you see any opportunity to improve this further, while keeping the same logic.

Student: sir whatever typical path would given by the a sub schedule.

Student: you should not alter the priority in that path only alter priority of the other nodes or the tree in the graph, when you recompute at second or subsequent alteration because if we would have if we say.

Mobilities are recomputed in our second iteration. So, what is your.

Student: the critical path.

So, we had c d e a originally in the critical path.

Student: (Refer Time: 19:51).

Sorry yeah now we had c c f.

Student: (Refer Time: 20:01).

Where they are in the critical path you are saying that we should not change the priorities, that is not clear see once you have taken this is a constructive algorithm this goes in the forward direction only. So, once you have you have schedule a and b, there is no further decision we are not going to reschedule it although that is just a property of this algorithm.

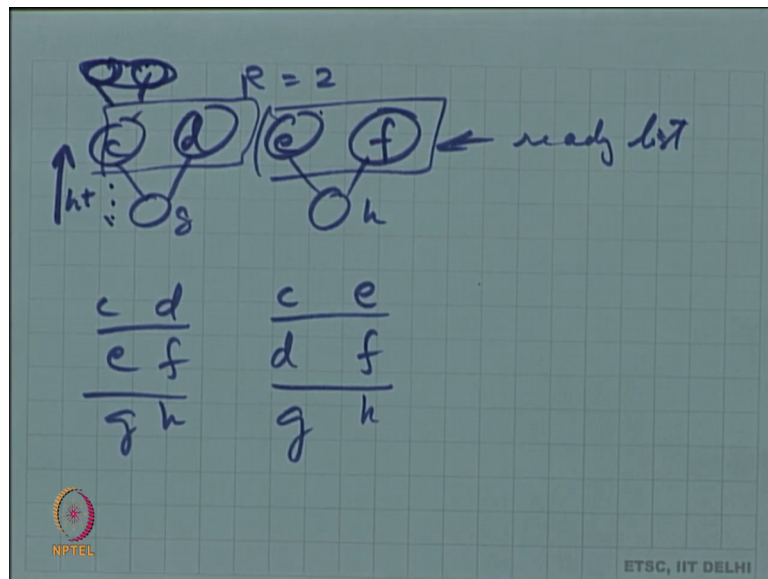
In fact, if you try to do re scheduling later on you might find a better solution, but anyway this strategy in the interest of time, and does not revisit the decision for specific

node, but under that constraint, you see that once you have taken the decision for some nodes, you might as well forget about those 2 nodes and drop them from the graph, and because of that everything would change.

Now, when you do ASAP and ALAP a different set of nodes would fall in the critical path.

Student: sir while selecting the nodes for computation that time (Refer Time: 21:07) take that into account, that it should result into a computation. So, if I have 4 nodes c, d, e, f, and I have to perform 2 addition and if I c, a, d, a, e, and f if I choose c and e it will not result into anything. So, whatever the 2 nodes I am choosing it should yield into a competition that kind.

(Refer Slide Time: 21:39)



It should yield into a computation means. So, I have c, d, e, f.

Student: (Refer Time: 21:37).

And the c and d are involved in a computation in the next node.

Student: So.

So, these are my choices that is my ready list, and you are saying there should be that kind of a.

Student: (Refer Time: 22:04) with e and f.

If this is the case then what should the choice be.

Student: Then either I should choose c or d or e or f I should not choose c and.

Considering my R equals 2 it is better to choose either these 2.

Student: Or (Refer Time: 22:16).

Or these 2, but it is not worth to choosing one from here and one from there, why?

Student: sir it will not yield into any computation. So, if I choose a and e, it will perform one operation in that cycle, but if I choose c and e, it is not performing any time. So, I have the (Refer Time: 22:31).

So?

Student: (Refer Time: 22:31).

R equals 2 let us shall we go and schedule this.

Student: (Refer Time: 22:35) it should cover this (Refer Time: 22:38).

There is some merit in the argument unfortunately it does not show up in this example, because you have c, d, you have e, f and you have g, h, right, that is my schedule if I had chosen c, e as the first cycle.

Student: (Refer Time: 22:58).

D f in the next.

Student: Then g a (Refer Time: 23:01).

You still get the same result.

Student: If you have taken c and f in the first cycle.

If we have taken c and f.

Student: (Refer Time: 23:11) in the first cycle.

Right you took c in the first cycle, d f in the next, gh here it does not show up, but this does bring up an important argument that I had alluded to earlier.

Student: (Refer Time: 23:24) nodes are (Refer Time: 23:28).

Which is.

Student: (Refer Time: 23:29).

All are not equal they are equal according to some metric that priority metric, but. In fact, with respect to their impact on later operations, they may be different and as you can see if you were to do a scheduling from here, if you had a lot of time you could try all the different schedules you could actually possibly choose the one that is best, but in general even though it seems as though we ought to take the successors into account, a simple heuristic might not be there, what is often used is just this thing of height of the nodes going from the bottom here we didn't.

So, this priority computation was the difference between ASAP end and ALAP. You could take the height instead, this has some different implications although both are intuitively similar there is one issue that is there in this particular algorithm that we studied, which is after every cycle the mobilities are being recomputed, right that this strategy there is some disadvantage with that, what is the disadvantage?

Student: (Refer Time: 24:41) d and 0 e (Refer Time: 24:45) was in the critical path earlier (Refer Time: 24:49) more cycles

Well the algorithm is just you are again computing the mobilities. So, to perform that computation is not easy, it takes time right, even the ALAP schedule ASAP schedule you have to run ASAP again, and you have to run ALAP again, and then compute all the differences it seems as though it is an overkill to do this computation every cycle.

Student: (Refer Time: 25:17) use some mathematical operations.

You have to somehow do some while, this is faithful to the idea of the mobility, it is computationally too expensive you can do something about it, what can you do about it, one is you computed once and then forget about it, and just use those numbers as you proceed with the scheduling.

Student: (Refer Time: 25:41) I can not think of a counter example, where this will not work or this can yield in a bad scheduling what is (Refer Time: 25:48).

With?

Student: Choice mobility.

Mobility.

Student: Compute only once, I mean I am not able to think a counter example.

You are not able to think of a counter example online.

Student: Yes

But offline if you go back and think about it you may be able to the, the point is that we are making these random choices, every cycle when you make the random choice of the subset, you may actually end up with a situation where.

Student: when all the nodes are equal in the sense they perform the same thing, it will not be a problem, but if the nodes are unequal; that means, they perform different things, and they have different delays or though then it started making.

Scheduling is a difficult problem even under the simplest simplistic assumptions that we have made, in the general we will get to the generalizations remember there are lots of simplifications we have made in this formalism, what are they? One is that there is only one operation type for this example, but to what other assumptions we have made, everything is falling into what every delay is single clock cycle, also lots of such assumptions we have made the difficulty of the problem does not go away.

If you generalize it that is why we are creating this simplistic version with some seriousness, but you could do several things, one is you could do that that computation only once, and just proceed. So, it leads to inaccuracy is possibly later on depending on what intermediate choices the intermediate choices where, other is you could do that once in a few cycles, not just in the interest of practicality instead of every cycle you still can do the computation of the mobilities, but maybe once in 10 cycles in case that helps improve the computational complexity.

So, it what we have here is a fully dynamic priority function, but it need not be completely dynamic, but that other example of using the height of a node, height being defined as the critical part saying from the bottom right, this is actually an interesting alternative, it captures a similar property of the graph, but that is one that does not change as you take decisions about the schedules of the earlier nodes right, the height is a function of what happens later on in the dependent nodes, if you take some decisions about the predecessor node scheduling decisions about the predecessor nodes it makes no difference to the height, that is an alternative priority function that is popular with it is easy to see why it is essential, if the height is more it means that the priority should be higher or lower.

Student: (Refer Time: 28:52) right priority.

Priority should be higher.

Student: (Refer Time: 28:54)

Because there are more the lists of the chain of dependent nodes is longer if the height is more. So, it seems as though we are capturing information that is not very different from what we did earlier, but this property does not change as you take scheduling decisions from the top down, because this is only dependent on what happens later on right, that too does not guarantee optimality, but this is better from a computation point of view because you do not have to recompute.

Student: either of the 2 can be better in different cases.

Yes, you could also find other cases where height is not it a good enough measure right.

Student: (Refer Time: 29:40).

It is attractive from that point of view, as to which one is better it is not clear depends on the properties of the graph. So, it is usually with the experimental data that you say that this is performing better or the other.

Student: there are multiple discussions nodes (Refer Time: 29:56). So, that all the.

This is the maximum length from of the of any path from that node to that starts with that node all right.

(Refer Slide Time: 30:16)

**Refined Model:
Different Operation Types**

- Different operation types require different FUs
- How to handle in Scheduling algorithm?

NPTEL (C) P. R. Panda, IIT Delhi, 2017 65

Let us refine this model by getting rid of some of the assumptions that we have made first, of course different operation types required different function units, the assumption in that first problem was that we assumed there is only one type of resource.

So, all the nodes would be realized using that one resource type, but what if you had different of operation types you had additions, you also had multipliers, you still have the same graph as the input, but now you have different function units, how would you handle this in the scheduling algorithm.

Student: sir isn't it already handle in ASAP and ALAP schedule. So, can we create ASAP and ALAP.

The ASAP and ALAP schedule numbers are independent of what their function unit is doing indeed, but what about later on as we do the list scheduling. So, you compute the mobilities that is still common, that does not change that is just a property of the graph it is not dependent it is a property of the dependencies and it is a property of the latencies of the nodes.

But it is not a property or it not a function of the type of a node. So, that part remains the same, but what about the list scheduling.

Student: sir that part remains. So, almost everything is same unless we say get more more delay or 2 cycles kind of (Refer Time: 31:42).

We will relax that also, but as of now we just have more types.

Student: So, we have different class of resources now.

So, we have different classes of resources right

Student: Now I need to choose that these set belongs to (Refer Time: 31:53) then do similar scheduling as we would do.

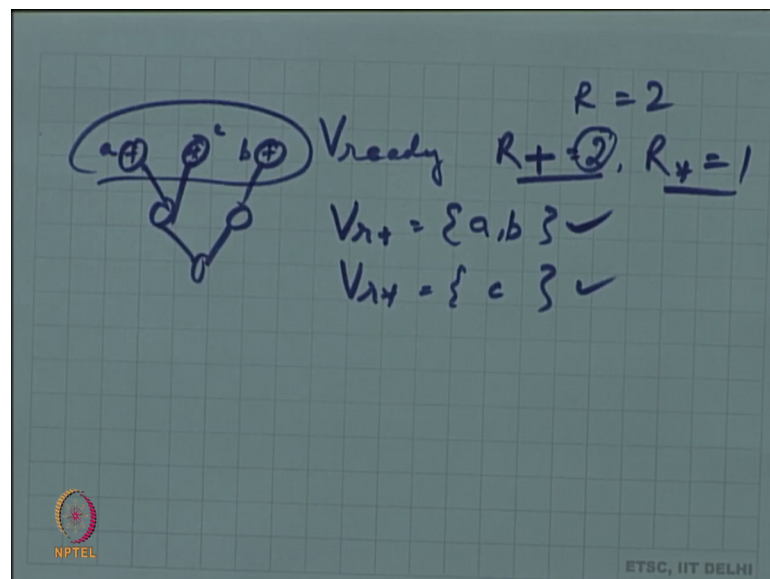
Right so.

Student: So, may be bounded by a (Refer Time: 31:59) function units that will be.

Student: A constant in a problem.

So, what changes you know. So, we had earlier.

(Refer Slide Time: 32:11)



Some such graph, where we identify that this was the ready list. So, there were only 2 things we are doing, identifying the ready list and from the ready list choose a subset to schedule, that is it, that is there in list scheduling nothing else. So, if this is plus these are different resource types, then how does this schedule change.

Student: So now, we will see we have (Refer Time: 32:48) components which are used and we have that (Refer Time: 32:51) resources.

Right. So, instead of saying R equals 2 earlier, I would have number of resources. So, plus. So, this equal to maybe 2 and R star equals one, or some such thing this is saying that there are 2 adders and one multiplier right. So, that changes in the input itself, but here how does a ready list processing change.

Student: So, ready list will have to, different nodes (Refer Time: 33:20).

You have so, the ready list has a bunch of nodes that are of different types, you could think of it as a ready list for every.

Student: Pperation.

Operation type. So, if this was a this is b this is c, then this would be a b, and that would be c, you have different ready lists and you have this different resource types and the associate the resource constraint is there on every individual resource type, and you perform the same the logic is the same, but it is applied to both the ready lists, for this ready list you are trying to select the best 2 nodes, for this ready list you are trying to select the best one node.

Student: if both the ready list (Refer Time: 34:16) on same clock cycle yes (Refer Time: 34:19).

Of course, the ready list is independent of the clock period, ready list just says that the predecessors have been scheduled, now they are there in the list, which one you pick up now assigns those into a single clock cycle, but if you have different resource type, then in every clock cycle those resources can be used to do something and therefore, essentially you are it is like you are choosing from different ready lists, and it is different resource constraints would apply to the different lists, that is it otherwise the algorithm does not change.

(Refer Slide Time: 35:02)

Refined Model: Multi-function Units

- Multiple operations can be performed by same FU
 - ALU (+, -, AND, OR, etc.)
- May lead to smaller area compared to separate FUs



There could be multi-function units instead of assuming that every operation type, this is derived from the sdl right you wrote a plus or a minus or something like that, it translates to some resource type, now instead of assuming that each operation type requires a different resource type, we could have a more complex resource type that is capable of implementing more than one operation type.

So, that could be the case if you have alu sort of operations, where it is capable of doing plus and minus and logical operations and so on, why is this interesting it may lead to overall smaller area compared to if you had individual adders and subtractors and so on right. So, I am just pointing this out all of these generalizations we need not necessarily work out the details, but you can see that the algorithm does change, if you have alus right in what way would it change again the ready list is the same as earlier.

But now you could instead of saying that this is this ready list is specific to that operation type, earlier for the ready list the different ready list maybe you would have maintained for the subtractor operation, but now you may actually combine them into a single ready list, it does get more complex when you have the ability to instantiate either an individual adder or a subtractor, or an alu that is when this this simple strategy you can not merely merge the 2 because, you have a choice there that needs to be explored.

So, it is not a trivial extension, but let us just understand that this particular extension is there to be explored in general, but let us look at a few refinements with respect to timing. We are trying to generalize that situation where this assumption where we said

that all operations take a fixed amount of delay, and that fixed amount of delay is one clock cycle, that leads to an inefficiency of the following kind of course, the different operation type in general takes different amounts of time.

Student: Sir (Refer Time: 37:23) will also come into picture (Refer Time: 37:24) multiple operations together.

Precedence of the operations.

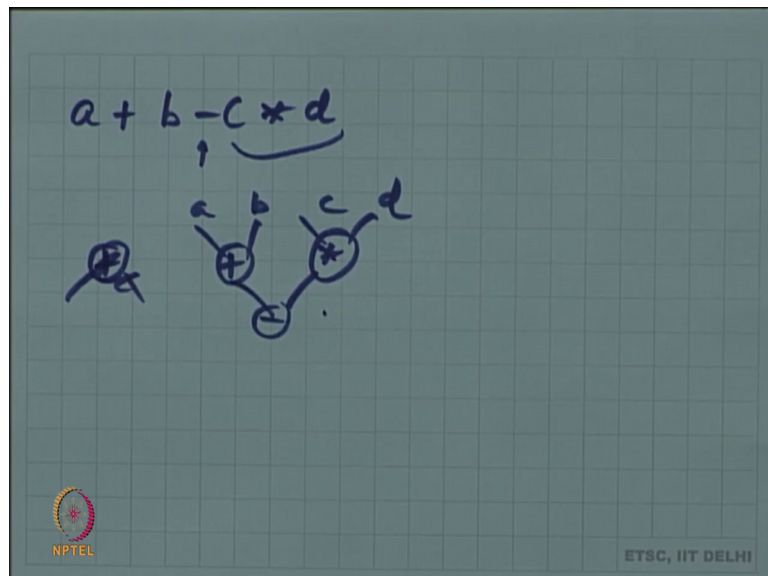
Student: Like for example, if it is a plus b and in parenthesis and then c. So now, we have to sure (Refer Time: 37:36), but that during graph formation we have already done right.

Precedence of the operation yeah actually the by the time you arrive at the graph that has been.

Student: (Refer Time: 37:54).

Taken care of if you had a plus b minus c star d.

(Refer Slide Time: 38:00)



what is a reasonable graph of course, it isn't a. So, there is only one graph that represents this, but the fact that this operation or to occur this has a higher precedence over this, is something that we did take into account when we form the graph, the grammar of the language has been decided in that way and therefore, the parse tree when it was constructed in the initial stages, does have this kind of a structure that you have a

put the ab there, this kind of a structure you are assured that you are seeing at this stage right, where this did come how did this happen, it happened through the grammar rules that we specified to the parser. So, of course, it has to recognize this in a way that is consistent with whatever our rules are.

So, do you see this inefficiency, if I insist that I will schedule one operation in every clock cycle, then my clock cycle has to be as long as the longest operation that I have right. So, if the multiplier takes 20 nanosecond, and the adder takes 10 nano second, then the clock in order for any schedule to work the clock period has to be at least 29 seconds, otherwise it does not work and as a consequence whenever you have an addition.

(Refer Slide Time: 39:39)

**Refined Model:
Multi-cycled Operations**

- **FUs with different Delays**
 - clock cycle \geq longest operation
 - low utilisation of faster FUs
- **Multicycled Operations**
 - an operation can take multiple cycles
cycles $\lceil \text{delay/clock period} \rceil$
 - inputs have to be held stable
- **How to extend scheduling algorithm?**

NPTEL (C) P. R. Panda, IIT Delhi, 2017 67

That is scheduled all of that time is wasted.

So, essentially this leads to an under utilization of those fuse that are simpler and faster. In fact, those might be more in number, those nodes might be more in number add nodes might be more in number, than the multiplier nodes and because of this inefficiency, you might end up wasting a large part of the schedule.

What can we do about this.

Student: (Refer Time: 40:09).

So, we have to relax that assumption, that all of these operations even if they were combinational we have to relax the assumption that they all finish and one clock cycle, let us define the delays to be independent of the clock period of course, who decided what the clock period was, for this algorithm, the clock period was an input who decided what the delays of these function units there.

Student: structure of the.

Adder delay and the multiplier delay who made the decision.

Student: that is may be designer.

The library designer designed cells and the all of these components, and this decision was an outcome of whatever the design steps were in designing those components, there is no relation between the to the library came from somewhere, and the application requirement might require us to run it to one gigahertz or something like that. So, there is no relationship between the 2.

So, in general let us just relax that assumption that the delays have nothing to do with the clock period. So, if they have nothing to do if the clock period were 20 nanoseconds are greater, than we are we nothing changes, but what if it was less, if it was 10 say then the add would finish in one clock cycle, but the multiplier would take multiple clock cycles, but in general the number of cycles for a function unit can be computed like this, whatever is the is the characterized delay of that that component divided by the clock period you have to take the ceiling of that to give you integral number of cycles, that is the treatment as far as the schedule is concerned, there is one implication with respect to the controller.

What is that implication? Scheduling it is very clear earlier we used to put every node in one clock cycle, now I have to stretch this schedule for a node to multiple clock cycles right. So, that part is quite clear multiple clock cycles, but there is one other thing that is implied from the designs implementation point of view, when you have a multi cycle operation, multi cycle combinational operation, what do you need to ensure?

Student: output form that every source (Refer Time: 42:38).

Indeed the meaning of the multi cycle operation is that, the output is not ready at the end of the first clock cycle, it is ready at the end of the second clock cycle, I should rely on it is value only after the second clock cycle, which is fine right because if there is a dependent operation to this multiplier coming later, then that would be scheduled into the third clock cycle onwards, that is all right that is not hard to employ there is just one other thing you have to make sure.

Student: if the output of in in this example output of adder. (Refer Time: 43:10) for a pipeline somewhere in between the, it should be carried on to the next some register inference or.

Actually

Student: Some pipeline must be.

The output of the adder can be used in the second clock cycle remember, if you wanted to you could use it if you do not use it, instead if you use it in the third cycle what is the and how do you realize that.

Student: Store it in a register.

Store it yeah, we did say that the moment there is an edge, dependency edge that is crossing a clock cycle boundary you have to store it in a register, that we already know, but what is different now for multi cycle operations, the inputs to the operation they have to be held stable for multiple cycles.

Student: Ok.

Right this is an obvious requirement, since this is taking multiple cycles of course, the processing is incomplete at the end of the first cycle, you need multiple cycles. So, which means that if they came from a register, then that register cannot be reused for the duration of this operation.

Student: (Refer Time: 44:09).

If?

Student: If that only directly from another operation like.

If they came directly from some other operation, then you did this falls into the standard requirement, that I have an edge dependent incoming edge which crossed the clock cycle boundary right, this previous clock cycle boundary is what it is crossed therefore, you need a register for it.

Student: How do we capture this timing requirement in a graph, till now we were capturing everything in the dependencies in form of a graph right, but now I have the data 2 dependencies one is I should not use the output before I test process, I should keep my input stable till the output is there.

Student: Output is there right. So, this is the kind of timing dependency on to my operation.

Right.

Student: So, how do I capture this in graph (Refer Time: 45:06).

In the original graph you do not capture it original graph was capturing only the functionality, that does not change, this needs to be maintained through the later phase of the scheduling. So, this is an implication that comes up later right, now it is the scheduler only sees the nodes the resources have delays associated with it right so. In fact, an individual node which you say plus or a star, might actually be implemented in many different resources maybe it is an alu maybe it is an adder and so on.

Each of which would have a different delay. So, that delay is to be taken into consideration as you schedule the nodes, the scheduler does need to be aware of it, but more important later on these decisions have to be communicated somehow, when a register allocation is being done. So, that you do not rewrite the register at the wrong time. So, some information has to be handed over from one stage of the synthesis to the other, the input representation does not change it is still the same.

Student: (Refer Time: 46:08) understand quietly this information is annotated in the system it its.

Somewhere it some data structure that we are not talking about yet but.

Student: (Refer Time: 46:17) take care that before it is done than only schedule the next time on this (Refer Time: 46:21).

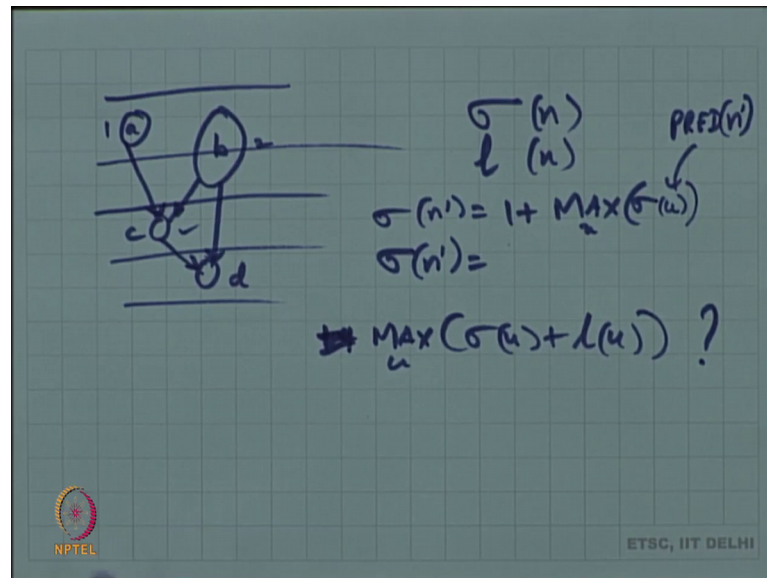
Schedulers responsible is only to make sure that a dependent node of a multi cycle operation does not get scheduled at the wrong time, does not get scheduled too early. So, anyway instead of me saying, why do not you suggest how to extend the scheduling algorithm, it is a formal question that is posed to you in the slide, how do I modify list scheduling I have to take care of this.

Student: while building the ready list at least one thing we can do is, we can not say the dependent nodes are ready for the multi cycle output, right, that must be my figure out automatically without much effort.

So, this is a change in the ready list computation. So, check not merely whether the predecessor has been scheduled, because it might have been scheduled and might not have completed. So, you have to check that enough cycles have passed, that all predecessors have completed their operation, that is all once you do that once you put it into the ready list the rest does not change this is.

What about the idea of the height and the mobility and so on, what happens to that when you have multiple multi cycle operations, you have let me assume that this delay is 2 and this delay is one, and do you have some search ASAP and ALAP computations have to change now.

(Refer Slide Time: 47:57)



Student: Yes.

In what way will the change. So, these are one in this direction what goes into the first cycle in the ASAP a, b and c, d, a and b clearly are scheduled into the first cycle. So, that is b has not completed yet and.

Student: (Refer Time: 48:31).

What is ready now? So, c cannot be scheduled yet right, even in the ASAP forget about resource constraint, but even in this ASAP it can not be scheduled because of the dependency on that multi cycle operation, after this I schedule c, and after this I scheduled. So, ASAP ALAP computations changed in this simple way, that your calculation will not merely do plus 1, earlier our algorithm was that the schedule for a particular node was computed by adding one to the max schedule of all it is predecessors, that was the equation, here I would change it in an obvious way I would not merely say max plus 1, but instead it would be.

Student: The size of the node max (Refer Time: 49:21).

Each has a latency. So, sigma of a node was when it is scheduled, and let me add a latency of a node in that way, and this schedule for a dependent node would be in terms of.

Student: (Refer Time: 49:42).

So, earlier we had 1 plus max schedule of u where u was in pred of n.

Student: (Refer Time: 50:03).

This is what it was instead it would be now what

Student: (Refer Time: 50:09).

It would be 1 plus.

Student: (Refer Time: 50:19).

So, u belongs to a. So, overall that max was overall the all the u.

Student: (Refer Time: 50:28) and then of max of that is sigma (Refer Time: 50:35).

Would I still have max of.

Student: l_n of max of.

Σu plus l_u .

Student: Σ (Refer Time: 50:49) minus 1 minus 1 why minus 1 because I am (Refer Time: 50:52) consuming max.

Let us define this is. So, let us let us remove this l is one if it is a single cycle operation. So, is this where, u is over all the predecessors are not. So, so it is a simple computation, this is still optimal the ASAP schedule is still guaranteed to be optimal ALAP schedule similarly you can.

Student: Now let us say my scheduler was considering height in it is algorithm, then height will be of height plus these steps. So, these.

It would be in terms of cycles.

Student: (Refer Time: 51:33).

It would be internal computer that is an easy computation, but it would be a computation in terms of cycles, which is the sum over all the individual cycles of multi cycle

operations if there are any. So, that is one generalization, not much changes as you can see in the overall strategy of the scheduling, height is still as an important priority function mobility is also a relevant priority function, is just that the way you would compute them would be different.

(Refer Slide Time: 52:11)

Refined Model: Chaining

- **FUs with small delays**
 - clock cycle \gg operation delay
 - low utilisation of faster FUs
- **Chained Operations**
 - multiple FUs are required
- **Extension to scheduling algorithm?**

(C) P. R. Panda, IIT Delhi, 2017

How about generalizing in the other direction, where clock cycle is given and the FU d lists are too small, not too large compared to the clock cycle, but too small compared to the clock cycle. So, small that. In fact, dependent operations could be scheduled into the same clock cycle, right this is an example of that, I have a dependent operations and the standard algorithm we have seen so far would actually put them in different clock cycles, just because they are dependent.

But actually, if you look at this you see that there is enough time, because to schedule both of them in the same clock cycle that would lead to better utilization of a resources it has some implications though. So, ideally, I would like this, which is why not since I have enough time I should schedule both of them, dependent operations also into the same clock cycle as long as enough time is there, this has one implication with respect to resources, what is that?

Student: The resources have to increase now.

You need 2 adders to do this this is a combinational operation.

Student: (Refer Time: 53:19).

Still it is a combinational circuit. So, changing can be done if you actually have more resources, you could still have muxes between them in this path, in the sense that in some other cycle you could be reusing that adder to do something else, you could be using this adder to do something else, but still for this to work you could have muxes, but you should not have a register in between them. So, one path should be there from the output of one adder into the input of another adder. So, that is what is called chaining multiple FUs are required.

But if there is enough time then you should be able to accommodate dependent operations into the same cycle, how do you change the scheduler for this?

Student: So now, we have to look at the downstream nodes also for the dependency, we can not like with the current method as ease because we do not know if a register was needed in between.

So, but remember at the scheduling time we do not worry about registers, registers are inferred, but that happens at a later stage right, we do not have to worry about whether we have put a register already or not, this solution being that if we decide to change then we will create a path later on, between the output of one adder and the input whatever the mux was there at the input we will introduce that path. So, register decisions have not yet been taken, but that is already that that does not really affect this.

Student: (Refer Time: 54:54) counter example sir initially (Refer Time: 54:56) algorithm, we were when we had a (Refer Time: 54:59) we were giving at a value of 2, going by the several energy we can give this as a (Refer Time: 55:05) and the (Refer Time: 55:07).

If the height if it was defined in terms of number of cycles, then that is an integer and it does change.

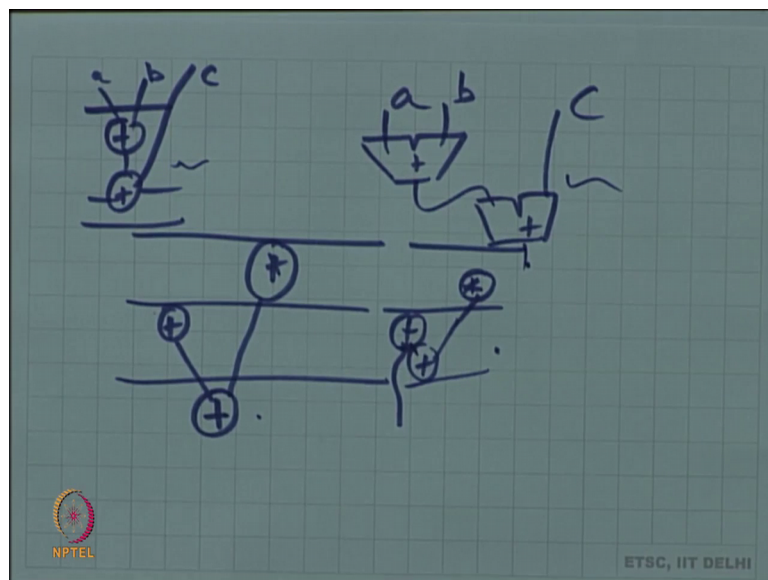
Student: So, (Refer Time: 55:18) give then these (Refer Time: 55:19) as the height of (Refer Time: 55:23), then when we compute the height (Refer Time: 55:28).

This height could be an absolute value as opposed to relative value.

Student: sir instead of an integer value we give them a real value. So, we allow a continuous domain.

So, of course, the delay is a real number it is it need not be an integer, delay need not translate to a fixed an integral number of a clock cycles, you could as part of the height computation just, you could define the height in terms of actual delays instead of a cycles, you could do that ultimately the scheduling decision does translate to integers, though what circuit does this translate to that schedule that we have.

(Refer Slide Time: 56:13)



Student: (Refer Time: 56:15).

I am sorry.

Student: Yes.

Right, this translates to what circuit then what.

Student: Then another adder.

Another adder.

Student: (Refer Time: 56:40) then register.

That is it as of now that is just that you have. So, when is this second adder looking at it is input.

Student: The second output only after (Refer Time: 56:55) adder that is why.

This is a combinational circuit it is looking at it is input all the time, but the point of the schedule is just to make sure that by the end of the clock cycle we have the right.

Student: (Refer Time: 57:08).

Value on the output it is giving us incorrect results somewhere in the middle of the clock cycle, but that is now handling this does require changes at multiple stages the priority function could change, but even before that there is a simple change that we have to do, that is not dependent on what is happening later, but when you have the ready list your you are already checking whether all it is predecessors have been scheduled first, later on we extended that to say whether all it is predecessors have been scheduled and they have completed.

The third check that you can make is whether it is possible to accommodate this considering where all it is predecessors have been scheduled, let us say I have a node here, it might have one predecessor here might have some other predecessors there, considering the different clock cycles in which this nodes predecessors have been scheduled already, could I move it to an earlier cycle than the current one, that is being considered right. So, that is all that is there, now you could refine the priority computation and so on of course.

Student: Now.

Now, I have to make this.

Student: Now in this case what will happen is we will go back in time and then add these second add operation there right, yes so, we are we are altering our. So, so far whatever scheduling you are doing you are not looking back in our previous cycles, but now in this case we will go back in our previous cycle and then do it.

So, let us speak tweet the since we are violating something let us tweak what it is that we were not supposed to violate, we will not revisit the scheduling decisions for a node,

looked at it that way we are still not revisiting anything, we are changing what is happening in each cycle because we change the decision for that clock cycle, but.

Student: (Refer Time: 59:22) become a single node in this.

No, it is still no.

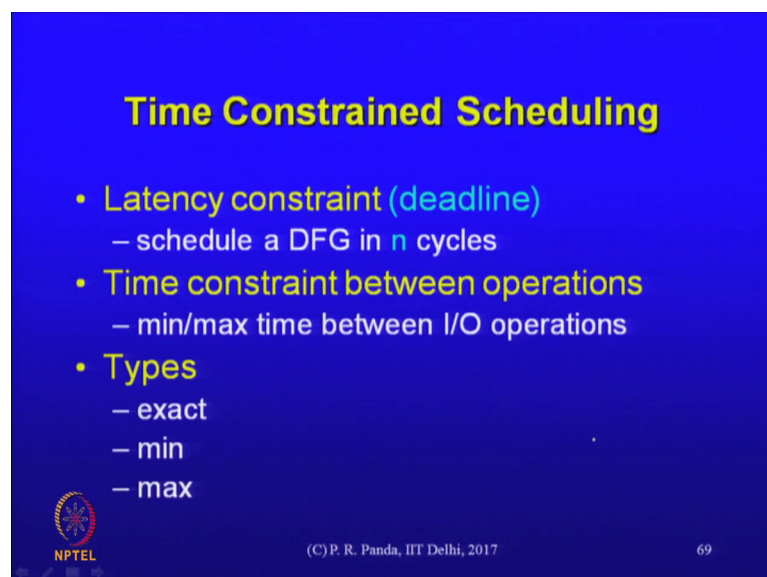
Student: (Refer Time: 59:28).

That that is still 2 different nodes.

Student: There is still a graph there still (Refer Time: 59:31).

It is still the graph. So, nothing has changed, but because you would have other dependencies perhaps right from here to others. So, that the graph structure is still maintained there is no merging at least this does not require the merging, we did do some merging in one of our transformations remember when we had that mac operation, we actually substituted multiple nodes into one node this isn't an example of that this is a proper chaining in which you are instancing both the resources, and are connecting up in a way that both are being scheduled into the same clock cycle.

(Refer Slide Time: 60:13)



Time Constrained Scheduling

- **Latency constraint (deadline)**
 - schedule a DFG in n cycles
- **Time constraint between operations**
 - min/max time between I/O operations
- **Types**
 - exact
 - min
 - max

NPTEL (C) P. R. Panda, IIT Delhi, 2017 69

But otherwise there is nothing complicated that is happening, let us look at the other problem definition we looked at resource constrained scheduling, resource constrained scheduling fixed the resources that you would take, and the optimization was to

minimize the latency, time constraint scheduling is the other way around you impose a latency constraint, that is a deadline that says that you must schedule the DFG within a fixed number of cycles.

But within that you have the flexibility to scheduled operations wherever you can, and in the process you optimize the resources right. So, in general this is a time constraint for the entire DFG, but you could actually extend that to also mean time constrain between particular nodes of the DFG, when that happens we will get to, but in any case you have a time constraints, those could be of different types it could be a max time constraint that is what we normally understand by a deadline.

But it could also be that there is a min time constraint, it could also be that there is an exact time constraint, when you say 5 cycles it should take exactly 5 cycles, such if extensions one can also think of those are also practical in some scenarios. So, that is the constraint, but what we optimize then are the resources it is an area optimization timing is fixed, and you have to perform area optimization and; that means, in terms of resources that you try to minimize the number of resources that are required to implement the DFG.

(Refer Slide Time: 62:00)

Force Directed Scheduling (FDS)

- Classical time-constrained scheduling algorithm
- Minimises hardware subject to time constraint
- Works by uniformly distributing operations of same type across control steps

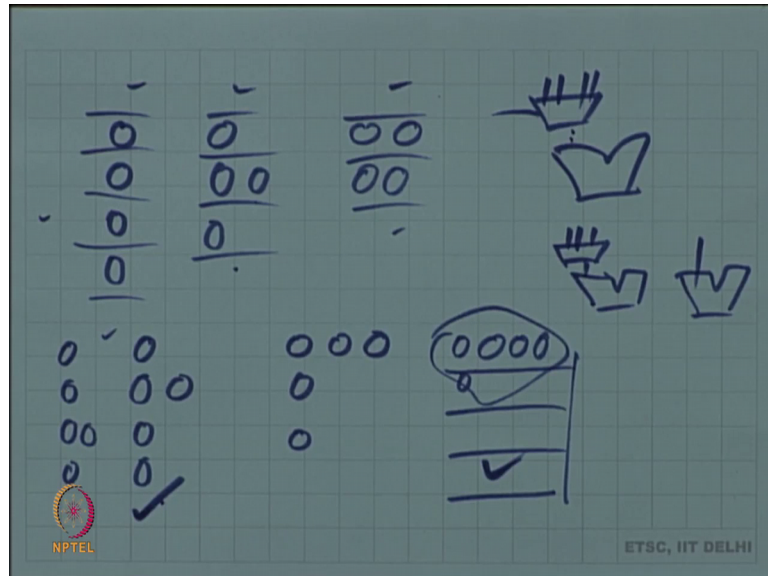
– maximises utilisation of FU

NPTEL (C) P. R. Panda, IIT Delhi, 2017 70

The classical time constraint scheduling algorithm in high level synthesis is called force director scheduling, this too is of the kind that the basic idea is simple, but that could be extended in various directions, now hardware is minimized subject to a time constraint

the idea is you try to uniformly distribute operations of the same type across control steps. So, the intuition being that suppose you had 4 operations to be performed right.

(Refer Slide Time: 62:35)



So, those could be distributed like this, could also be distributed like that which one would be preferred.

Student: Third one (Refer Time: 62:57)

Third one would be preferred, why?

Student: minimum clock cycle.

Minimum clock cycle, but that is not our problem here problem is a deadline is given to us, and for that deadline we have to arrive at a minimum resource schedule.

Student: (Refer Time: 63:17)

The answer is you I it is an incomplete question, because I didn't tell you what the deadline is if, I do not tell you the deadline how can you give me the minimum resource. So, if the deadline is.

Student: (Refer Time: 63:39).

Let us say let us say it is 4 then.

Student: minimum deadline is 4.

No deadlines if I did not say anything else then let us take that it is a max deadline, you must complete the operation within so many cycles.

Student: (Refer Time: 63:47) exact also.

So, let us say it is 4, it could be exact also remember that exact and minimum you could actually translate one solution to the other. So, in any case suppose I had 5 operations of course, you could add a dummy state and nothing changes, if it has to be exact you could still add one more clock cycle and nothing changes. So, if I said 4 was the deadline, then you would go for this you would not go for this, why?

Student: (Refer Time: 64:23) only one (Refer Time: 64:26).

Because of a single resource if I said 4 was the deadline, but I had that kind of a let us say I had 5 operations, then would I go for. So, I have to do something like that or I also maybe can do something like that, I also have 5 operations pardon me.

Student: (Refer Time: 64:53) we have to 2 or 3.

You have to tell me how many resources you have to optimize the number of resources remember the resources, are not fixed I am only saying that 4 is a deadline this one would be chosen because.

Student: we have.

Because you require fewer resources. So, what about this.

Student: (Refer Time: 65:22) in the first one why (Refer Time: 65:24).

My deadline was 4. So, that.

Student: 4 is the maximum length, rather schedule the last one with third I have 2 resources (Refer Time: 65:34).

No.

Student: (Refer Time: 65:40).

You can, but it does not help to complete faster the point of resource of a time constraint scheduling, is that the time is given if you do not use that much time then it is your problem, you are not saying anything by.

Student: (Refer Time: 66:01)

So, I already said it is max.

Student: there is some dependencies also you can not do (Refer Time: 66:06) like one one. So, in the second we can do like 2 adders at the same time (Refer Time: 66:09), 2 adders (Refer Time: 66:11). So, there going to some dependencies also.

There is some dependencies might be there this is just if I drawing it this way I mean that there are no dependencies, but of course, your solution is based on what dependencies are there.

Student: it is an actually if I choose all sequential like let us take the first question all yeah one one resource schedule in one cycle, that has an impact on the size of control fsm size now.

Fsm size.

Student: So, is it is it always trivial in practical life that the decision on the size of fsm is not always critical it does not (Refer Time: 66:57).

Different size can make difference here. So, let us say I have one resource right in on all these 3 cases I have a single resource each. So, which means that that is my resource, and there is a mux here for which the select input is coming from the fsm, well actually for this solution you would have 2 different adders, with muxes maybe in the general case, actually the second one does not require a mux, but the first one does require a mux. So, I think your point is that this one might have a 4-input mux.

Student: it was the register that I need to.

This one might have 3 input mux, because the other one there is no select that is an there is there, that is an interesting complication which sometimes we do not really address properly at the scheduling stage, because we are not at that level of abstraction yet, but

we could the extent of sharing that is implied by one solution versus another solution, could actually be incorporated in some way in the scheduling algorithm itself.

So, in the interest of simplification we are doing these steps in sequence, you are doing shuttling first and then we are doing resource binding and then we are doing register allocation, but what these are pointing out, is that these are interdependent the decisions are interdependent what decision you take here might actually make a difference later on right so.

Student: So, do these decisions also take a (Refer Time: 68:32) into account for example, let us say I have deadline (Refer Time: 68:35) I can take 4 or 5 cycles, but I need not to power up my another (Refer Time:68:37) though it is available, but say one power by just (Refer Time:68:40).

As of now these formalisms we have not taken power into consideration, but indeed power would be a reason to prefer this kind of a solution to this kind of a solution at least peak power would be better in this solution, that in this solution of course, area point of view this one requires more resources and than this then therefore, you would choose this.

Student: (Refer Time: 69:10) average power will be better in first one.

See our if our deadline was 4 then you could say that the average power does not change, because that is the energy of 5 operations has divided over that much time.

Student: (Refer Time: 69:25).

If nothing else is happening, if you have no op states not no other operation is scheduled then indeed the total energy is the same therefore, it is average energy is the same, but from a peak power point of view you would choose that.

But the intuition that maybe we can stop at for handling time constraint scheduling is that, you try to uniformly spread out the resources over the clock cycle, the more uniform it is the more likely it is that you have fewer resources, it is it is better utilization of the resources right.

If you if it is the user is skewed in this way or in this way, then you need more resources power also presents an interesting application of the same idea, peak power at least does you have a better solution if you uniformly spread out rather than a few bunch operations together, and the force directed scheduling formalizes this idea in some way, let us stop here and introduce the actual force director scheduling in the next lecture.