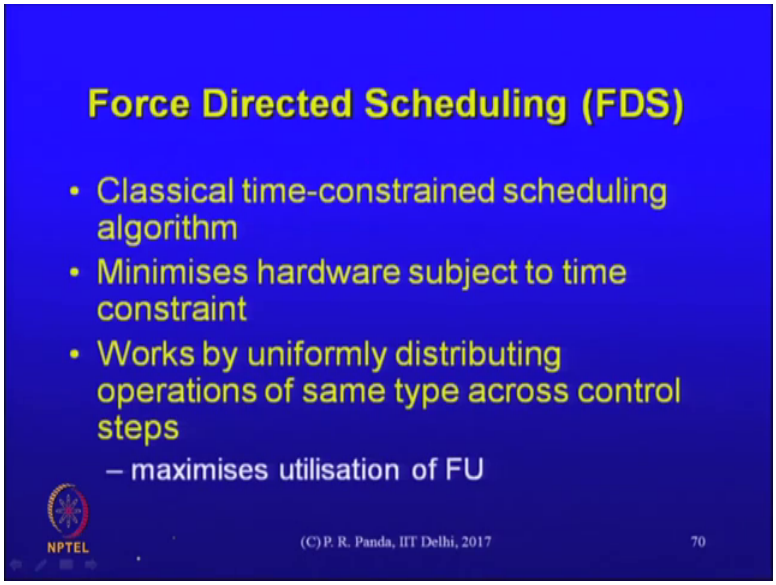
Synthesis of Digital Systems
Dr. Preeti Ranjan Panda
Department of Computer Science & Engineering
Indian Institute of Technology, Delhi

Lecture – 14
Force Directed Scheduling & Register Allocation

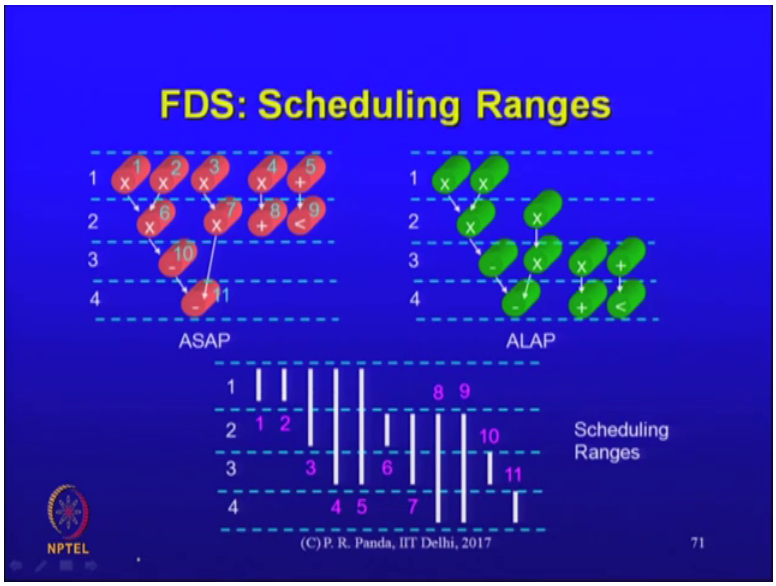
We introduced force directed scheduling and the main idea was that you would like to uniformly distribute the operations across the clock cycles, wherever you have the possibility of doing so.

(Refer Slide Time: 00:37)



Of course, at times there are dependencies because of which we do not really have a choice we have to respect their dependencies, but within that when we do have a choice the idea of time constraint scheduling is that if we try to uniformly distribute all the operations then we would end up with minimal resource count.

(Refer Slide Time: 01:03)

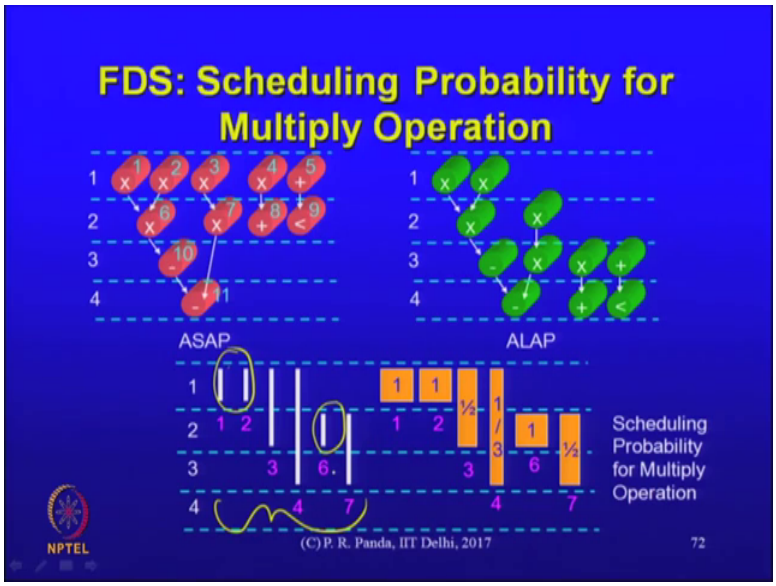


Let us look at the details of this force directed scheduling algorithm. This 2 is a strategy with a number of variations what we will introduces is the basic idea you can see there are opportunities for improving in a number of ways. Get back to the ASAP and the a lab as a way to influence some of the decisions here the ideas are similar to what we had seen in this scheduling, but they are executed in a different way. let us go ahead and perform the ASAP which gives us a minimum delay schedule assuming infinite number of resources we also perform an ALAP which gives us a schedule.

Taking into account the deadline that was specified this is a time constraint scheduling. A deadline has been specified. In our example here, the deadline was 4, but it need not be 4, but we come up with an appropriate ALAP schedule whatever the deadline is. Having come up with these 2 let us identify the scheduling ranges as we had done earlier. Which is for these operations 1 and 1 and 6 and 10 and 11 you do not really have any flexibility. You have to schedule them into their respective clock cycles remember that if the deadline was different from 4 then all the operations would have some scheduling range. that is what we have for nodes like 3 that one.

we have some flexibility because that could go into the first cycle or into the second cycle. These bars here represent the range of cycles to which each of those individual operations could be scheduled.

(Refer Slide Time: 03:16)

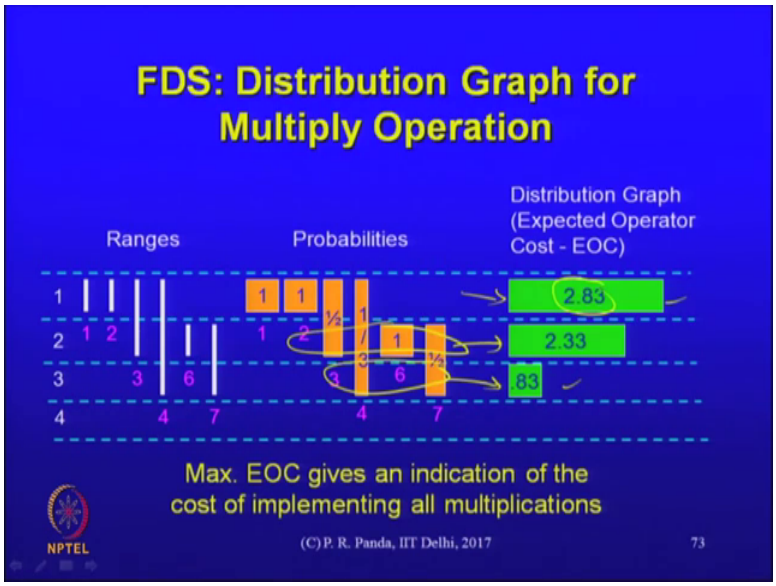


Let us take a subset of the operations here we have multiplication and addition and subtraction comparison is a several kinds of operation. let us just take the multiplication this has to be separately done for all the different resource types. These are the multiply operations at all of these 4 and these are our multiply operations and look at their scheduling ranges we are identify these some of these like one 2 and 6 we do not really have a choice, but the others 3 4 and 7 we do have a choice. let us identify the schedules for these that is essentially what the scheduling strategy would be.

Now, just for the multiplier operations I have to repeat that process for the other operations also problem of course, is that I do not have the flexibility to exhaustively consider all the operations being scheduled to all the cycles to which they could be scheduled because that could be a large number. This strategy first determines a scheduling probability for every multiply operations. For these nodes like 1 2 and 6 we do not have a range. The respective probabilities are one the others like 3 it could be here or here. let us say that that probability of going there is half and probability of going there is also half. Same thing I do for number 4 it could be into any of those 3. I have with a one third probability it could go into this or this or this.

That is the construction of a probability this is done just. That it helps us simplify the algorithm. That again we do not revisit decisions that are made, but hopefully the one-time decision that we are making is a good one for every operation.

(Refer Slide Time: 05:22)



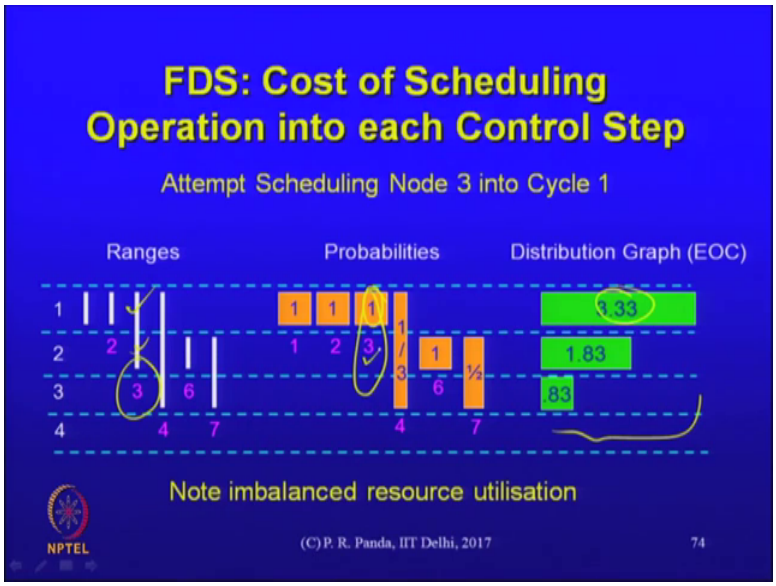
If you look at the distribution of the probabilities of all of these operations the picture, Actually gives us some idea about the kind of resources that may be needed in what way. let us say I add up the respective probabilities of all of those operations that have a probability of being shad non0 probability of being scheduled into the first cycle and I get a number like 2.8 if I add up all of these I get.

2.33 and on if I add up these I get this what does this tell us that distribution graph tells us something about the resource cost what does it tell us the 2.83 versus 0.83 means that you are likely to need more resources in the first cycle than in the third cycle in what way. That max expected operator cost that is the number that we are computing for each of the cycles there is an expected operator cost the maximum out of them tells us what? That number is capturing some information that is relevant to us what information is it?

Student: (refer time: 06: 56)

It tells us how many resources are needed there? Are this is not an integer number. It is not exact these are probabilities. They may or may not be respected the specific decision may or may not be respected, but out of these 3 numbers if I look at the max that tells us how many resources I might need that could be used in an algorithm that makes up the final decision, but this expected operator cost that just the distribution graph carries some meaning which could be exploited in a specific algorithm.

(Refer Slide Time: 07:36)



How do we exploit? it is not so hard to weave an algorithm around that idea? yeah.

Student: last picture. There is a algorithm also apply something like. We had this 4 part which is 1 third of probability across 3 cycles.

Yes

Student: now rather than considering it in first 2 cycles if I distribute to the.

Student: third cycle where the number is very low if

Yeah.

Student: I do an average distribution then probably the maximum number of resources which I will need may reduce.

Yeah, yeah, remember the intuition is that we try to uniformly spread out the resource usage across the clock cycle that of course, would be the objective. Somehow our objective in an alga we didn't get to an algorithm this is only giving us a picture of what is going on without taking any decisions, but the process of making those decisions can take this into account that we want to minimize the maximum expected operator cost.

Student: formalism of improving beyond if you want to this reduce the resource.

Yeah

Student: it is more of.

It is a bad from a resource well it could be bad from a resource point of view right. Because, first of all it is assuming there are infinite resources.

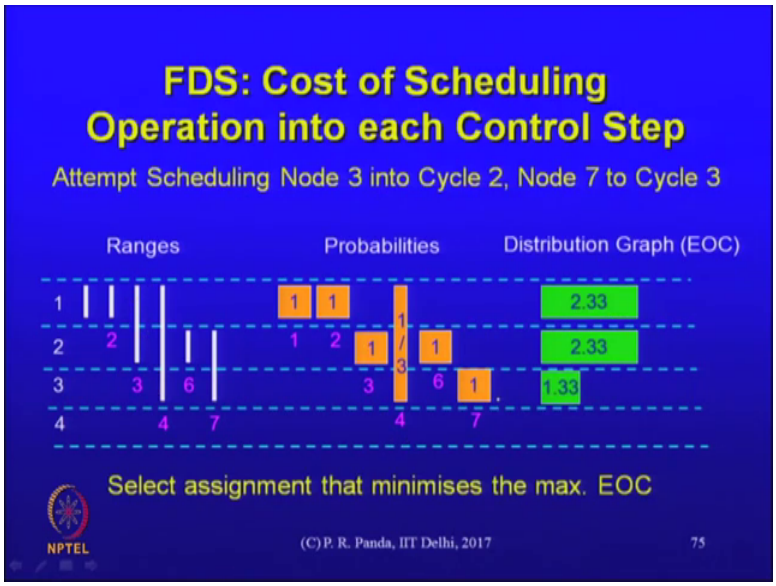
Student: (refer time: 09:00)

Resources any no constraint was of course, taken into account in coming up with ASAP. If you have a large number of operations for example, that do not have incoming edges then all of them will float up to the first cycle and it almost by definition for practical graphs it leads to a very uneven distribution of resources across their cycles there is nothing central to the algorithm itself , but commonly the kind of graphs that we have in practice may be such that a lot of these independent operations that are not dependent on any other operation. Occur early on and therefore, they may all get pushed up to the earlier cycles having identified that distribution graph you can come up with a simple strategy you take each of these operators that have some scheduling.

Flexibility where the scheduling range is more than one cycle and evaluate the impact of assigning it to the first cycle to the second cycle the third cycle and so on. We try to schedule it here see what happens we try to schedule it here and see what happens hopefully not too many choices are there, but at the end of the process we finalize the schedule for that particular operator 3. This is all that we are doing essentially what we are doing is we had this bar here we convert that probability into a 1 here which means that I have taken temporarily the decision to schedule that up operator 3 into clock cycle 1 then I look at the distribution graph and see what has happened to it what I am interested in is this number which is the maximum cost out of all the expected costs. That maximum cost is 3.33 I can also in an iterative manner.

Scheduled that one to this other clock cycle and correspondingly determine what the cost would be. This is still an expected cost because the decisions for these nodes have not been taken for 4 and 7 have not been taken yet. There is the question of in what order you will actually look at these operators because the decisions might actually be influenced by that, but this is an example.

(Refer Slide Time: 11:29)

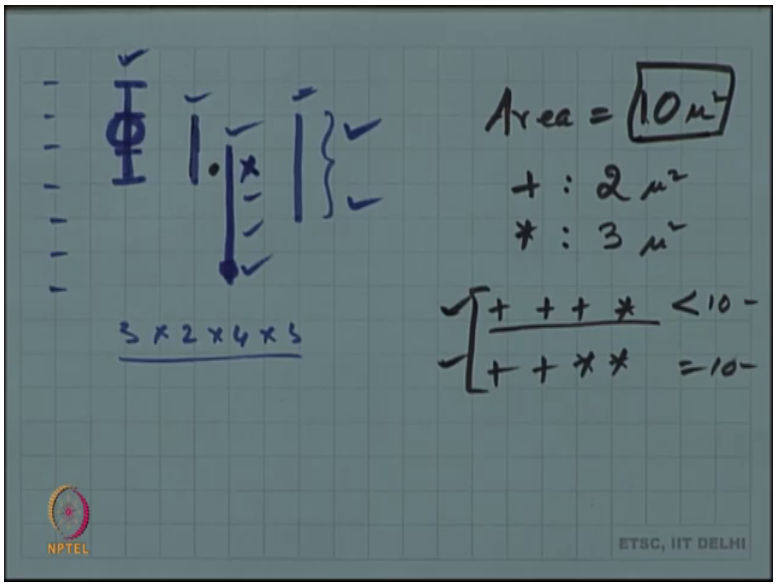


I go ahead and continue this process temporarily scheduled that operator into the second cycle and see what is the impact if I had this kind of an distribution where I had taken this decision and let us say I am trying to evaluate the impact of node 7 being scheduled into one cycle or this or the other cycle I can correspondingly keep track of these distributions and for each candidate that we are looking at.

We can look at the maximum expected operator cost. Temporarily for an individual operator we go through the different cycles try to assign it to each cycle in it is scheduling range. That compute the expected operator cost the max expected operator cost and finalize the decision for that operator. Then that probability becomes one for whichever clock cycle we assigned it to and then I just continue with the rest of the operations yeah.

Student: we are trying all permutation combinations of. If I could take this bar one of the cycles where it is possible to schedule that. These permutations and combinations could be very huge.

(Refer Slide Time: 12:57)



yeah, yeah, we are not trying out at least this algorithm is not doing. let us say. These are my cycles and I have some such distribution right each is 1 cycle. The real number of choices I have is 3 here times 2 here times 4 here times 3 that is how many choices I have one of them is the optimal, but the strategy that I just said is not evaluating all of them in some order of the operators. it is it is picking that I didn't say what the order would be let us pick up this one first it is evaluating the impact of scheduling this operator into 1 of the 3 cycles that it could go into once I made the decision that this is going here I fix it and then I proceed to the next one maybe this. The number of combinations I am evaluating is much smaller.

Student: (refer time: 14:01)

Right it is.

Student: (Refer Time: 14:06)

it is the sum right

Student: but that that will not keep the most optimal.

Yeah in the process you might miss out the best solution indeed.

Student: sir how we arrived at this ranges.

Oh, ranges are arrived at by doing the ASAP and ALAP.

Student: ok.

ASAP tells you the earliest unity cycle a lap tells you the latest cycle and that is how you come up with the range. The heart of the strategy is this as you can see there are opportunities for improving the quality of this result by doing various things among other things what can we think of as variations of this basic strategy. I could still retain this as the heart of the evaluation strategy, but an important element that was missed out here is we didn't take the order.

Student: order.

Into consideration how could we take that into in some simple way.

Student: (Refer Time: 15:15)

Yeah, why? why should I start with this? maybe I could have started with this.

Student: pick the longest path and pick the smallest. smallest one is always correct.

Yeah between the longest and the shortest maybe we have the answer somewhere, but actually neither of them might be an optimal strategy this is we are only trying to hopefully improve upon the basic strategy. There which was a random choice of the operations. There could be some argument for picking up the node that has the shortest range. What is that?

Student: sir if the if I start with the shortest then by the time I will reach longest I will have more opportunities to get it adjust.

This argument we actually used in the list scheduling also the priority was determined by the mobility of the operation more mobility means lower priority same thing could also be used here which is those could be the this if the mobility is smaller the which means that the range is smaller than they could be the higher priority operators still not guaranteed, but it could be one way of sorting the order in which we evaluate the nodes. Anyway, it brings down the complexity significantly, does compromise on the on the quality of the result, but as opposed to the product of all these ranges which could be an exponential exponentially.

Large in the number of nodes that have non0 scheduling ranges.

Student: sir why are we started with ASAP graph first and not ALAP I mean in that sense.

We are constructing this range this range is determined by 2 numbers the ASAP number and the ALAP number. There is no relation between the 2 schedules. You do it independently.

Student: there is an assumption that we starting from.

I have to determine this number I have to determine this number.

Student: that is done

Student: if I start with ASAP; that means, I am starting with the ASAP and then going to the most relaxed. I am starting with the most resource required purposes to relaxed resource purposes which is.

But this is one way of looking at.

This kind of a strategy is sometimes actually you conclude that it is optimal for certain classes of problems here it is not necessarily optimal, but we are just saying that the nodes that have more flexibility. Those it is possible that you have good solutions even though you have made some other intermediate decisions. let us say this one has a range o F4 maybe by the time you reach there these clock cycles are already full of other operations. Because, I have already taken decisions for this this and this that clock cycle maybe I have already inserted too many operations, but because the range was larger I can take this I can take this I can. I could possibly schedule that operation into any one of the other cycles which are less busy if it was the other way around that is I started with this one.

Initially when I start off, perhaps there is nothing that is stopping me from assigning this to this cycle itself. Now in retrospect later on it might seem that when it comes to this guy later on. I really do not have a choice and I have to schedule that into one of the cycles that is already busy and I didn't anticipate that this other operations.

Student: I need to move the largest bar I need to pick things from the largest bar to the smallest bar. I can choose that as my strategy and I can look at the longest I think.

The your say there the remember it is.

Student: sir.

You see is an expectation, it is anticipating what you will do in the future, but these decisions that you are taken taking immediately are definite decisions. As a result of this, you know it is going to be one the others are still there. There is a back up against giving very bad solutions.

But,

Student: I can choose that 3.33 I know that this bar is longest or think that 1 from that and I move it to 1.33 I will start with a longest.

Yeah, yeah.

Student: (refer time: 20:14)

As what you are saying is another variation of the algorithm that. A number of variations are of course, possible. What we are just pointing out is a basic representation that will help us evaluate different choices. It isn't as simple as just the ordering there are a number of other things. Yes, intuitively the idea is we try to shuffle these in a way that we are able to compact.

Student: local versus global picture I was thinking more about the global picture by looking all e o c s in parallel. Basically.

You could. In fact, look at the distribution graph and your algorithm could be based on analyzing that distribution graph and from there proceed to the nodes that is the other way.

student: (Refer Time: 21:02)

We will look at example see, the point of showing this is to illustrate the representation that captures some of the tradeoffs that are involved and of course, we discuss here what are the simplest algorithms. There are a number of different ways, even the whole of the

high-level synthesis formalism that we are looking at is a very local optimization. Resource constrained scheduling is an example of a rather local way of formulating the problem, your real problem remember is that you have even in the context of resource constrained scheduling you have an area constraint.

If at all there is a area constraint may be derived from the cost of the chip or something like that right, but who is telling you that you should have 2 multipliers and 4 adders? Why should somebody give that information? That should be something you can argue it should come out as part of the exploration of the tool itself. Keeping that let us say the area equals 10 if you knew that plus was plus had to. Whatever those units are and the multiplier had 3 square microns. Some such areas were there this area o F10 could translate into multiple resource distributions. All of which add up to less than or equal to 10 right I can have 3 adders and one multiplier.

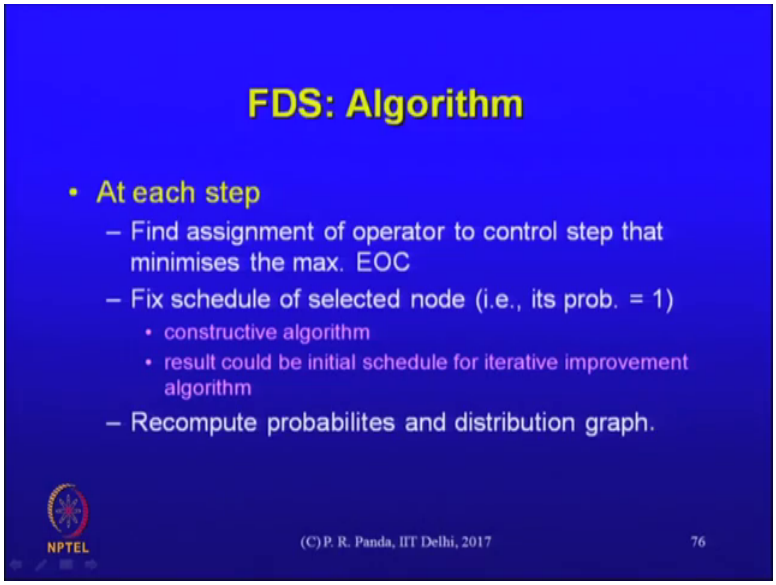
that is that is a total area that is less than 10 it could also be 2 multipliers and 2 adders. Right this is 10 more likely, that is the constraint that you are working with that in turn translates to this as you know the if you set the resource constraints as these then you get one set of schedules. If you set the resource constraint to this then you get a different schedule the way we formulated the resource constraint problem we are assuming that the number of resources of each type is given, but in reality, that might not be you can see that this is simplifying the practical problem. Practical problem is more likely to be that there is an area and within that area you want to get the best possible circuit in terms of delay, but while that is understood it is not hard to see a connection between reusing the same strategy resource constraint strategies that we looked at in this formalism where the constraint is not on the number of resources of each type, but on the total count how would you do it?

Student: similar kind of use.

I can have some strategy for generating different combinations that are promising. Let us say how I do it is a different matter, but as part of that generation process these 2-specific resource count oriented. Constraints could be generated and then I could use my list scheduling the way I had formulated. Yes, there is a bigger picture that can often be constructed using multiple iterations of the core algorithms that we are looking at and while through this course we may not be able to explore the entire space that is there the

focus is indeed to look at the core algorithms. that is why we are dealing with what seem to be little local versions of the optimization problems.

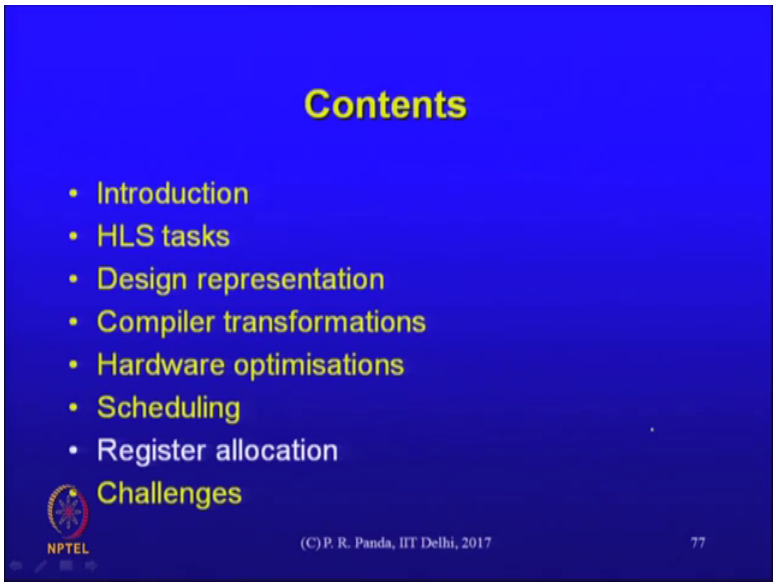
(Refer Slide Time: 25:03)



With that the strategy is actually very simple. Find the assignment of the operator to the control step that minimizes the maximum. You see you fix the schedule of that selected node fix the schedule translates to making it is probability 1 in the distribution graph for subsequent iterations or which see that selected schedule as showing up and influencing the probabilities of the graph. That this it is a constructive algorithm you do not revisit the scheduling decisions once they are taken; however, the result that you come up with could still be submitted for further improvement in some iterative improvement kind of strategy where you just take given schedule and maybe do some local or global variations in the schedules of the individual nodes to see whether there is an improvement.

That is always possible the schedule that you are given is the starting point that of course, respects the deadlines respects the time constraint and within that as long as that deadline is respected, you can try to move nodes to different cycles within it scheduling range in the hope that you might discover some solution that was missed out earlier.

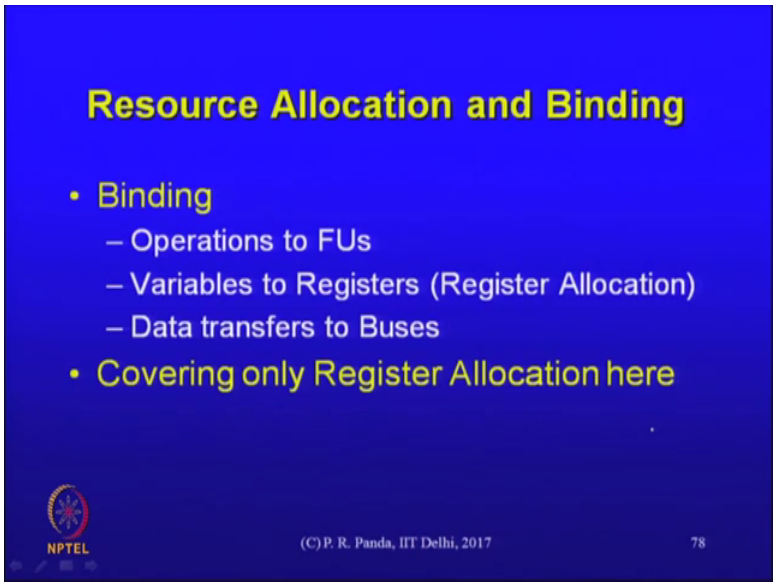
(Refer Slide Time: 26:26)



That is the 2 basic scheduling strategies one, was a resource constrained scheduling and the other was a time constraint scheduling this itself is a much more complex topic, but the idea here was that we get some basic idea about how to go about formulating the respective scheduling problems and how to come up with some simple heuristics that are practical in nature these are all of a practical nature.

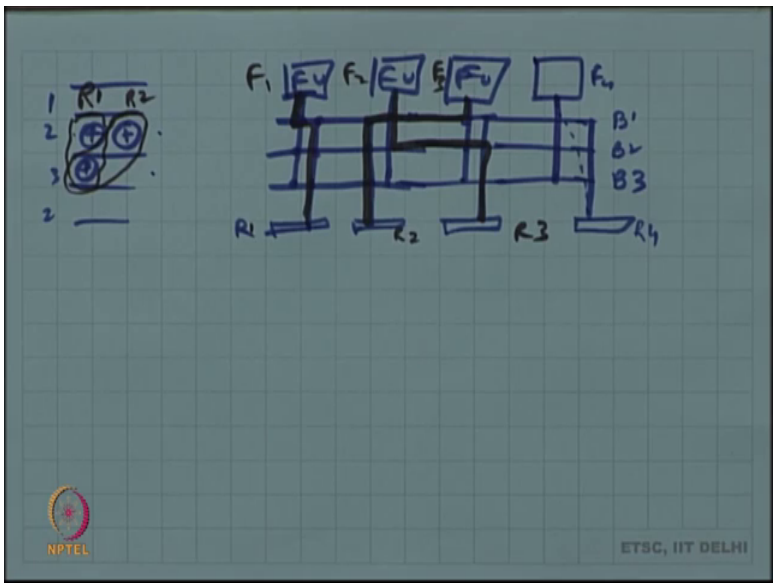
The these algorithms that we have studied with both the list scheduling and forced directed scheduling and so on. They are not very complex we didn't explicitly analyze the complexity, but you can see that at least the basic strategy is not complicated at all. You can try to make it more sophisticated, but there is usually a tradeoff between execution time and the equality of the synthesis algorithm.

(Refer Slide Time: 27:24)



The binding is a general problem that occurs in the synthesis. Where many different problems actually fit in to the overall context of binding resources are there. You have already taken the decision through scheduling or something that these resources are going to be used. Operations are their operations have already been mapped into specific clock cycles at the end of the scheduling.

(Refer Slide Time: 28:01)



We have taken the decision.

That 2 adders are there in clock cycle 2 and 1 adder is there in clock cycle 3 that is enough for the scheduler to guarantee that it is possible if you had 2 resources such as a schedule is feasible, but the information that is not yet there is which resource should perform which computation. You have the choice of sharing the resource between these 2 one resource between these 2 operations and the other resource doing that or it could also be that you do that kind of sharing. The binding problem refers to the assignment of individual operators that have already been scheduled in the case of function unit binding it refers to the problem of assigning individual operators.

That have already been scheduled to the specific function unit. That too is an interesting problem, but this is not the only kind of binding that is involved here we could also talk about binding variables to registers. That is also a binding problem I have some registers I have some variables those variables translate to what in the data flow graph.

Student: (refer time: 29: 25)

Those variables are captured in edges essentially the edges are the label and that label is just away. If that edges crossing a clock cycle boundary then we said that a register is inferred. But, there is this question of which register should contain which variable there are also other interesting problems that come up when you impose some restrictions about the target architecture binding data transfers to buses is one of them it didn't show up as of now in the formalism way of introduced. Far for high level synthesis, but there is one assumption we have made about the connectivity between function units and registers which is what, what is the assumption we made. Data can flow from any function unit to any register all of them could be active simultaneously. All the data transfers could be active simultaneously. If you have m function units and n registers you could have an architecture as a consequence of all the synthesis that we are doing in which all the m are actually connected to all the n. You could have right paths could be established from all the registers into function units and all and all the function units into the registers. It is a fully connected interconnection structure.

Which for various reasons could be inefficient it may become inefficient later on. Even though, at this logical level we are not able to see it. It hasn't been incorporated in our decisions in any way yet, it could be that it interferes with the good routing later o n because there is in that to universe therefore, you could think of imposing some

constraints perhaps on that target architecture do not allow the tool to just assume anything can be connected to anything else, but it could be that you specify a template for an architecture that looks like this. There are some buses around which your connectivity should take place and the function units that you have should all be connected to those buses and your registers should also be connected.

Just an example of a template. How they are connected? Is up to you, but the this could be connected to this, it could be connected to this. Each of them could be connected in this way you decide what those connections are at as a result of the synthesis. Similarly, the registers could be connected to possibly to all of these, but that is the extent of the flexibility you have in the connections let me add one more function unit. What this permits is that I am of course, able to connect any function unit to any other function. Unit that is because I can just establish all of these connections and it is fine right all of them could be connected. Paths are certainly there to any function unit to any register.

But what constraints does it impose.

Student: (refer time: 33: 04)

The number of buses is limited therefore, the data transfers that are simultaneous data transfers that are permitted in this structure are limited. This is similar to the memory port related argument you could access any pour any data element, but simultaneously there is a limit to how many data elements you can access. Here if I say that one or 2 or 3 if I say that this is F1 F2 and f 3 that F1 is going to transfer it is data toR1 F2 is going 2 transfer it is data toR3 in the same cycle then I can not use that connection I have to use some other connection like this.

And if I in that same cycle if F3 where to send it is data let us say toR2 then I would be limited let us say I would be using this connection now. This was for the 3 simultaneous transfers all of them going to each if I had a 4th one though, extends this this is another FU and I had a 4th register I. What I am limiting through this structure is the number of simultaneous data transfers I have to wait I may have this F4 being ready to send it is data to R4, but I do not have a mechanism for even though the connected structure is there. This is also connected to all of these and this too is connected to all of these, but in that particular clock cycle I do not have the resource.

You can think of these buses as also resources that need to be scheduled in to particular clock cycles.

Student: basically, for each mathematical operator or operator they are looking in the graph for example, we should also think of fetch and post as operations and then we can put buses as number of.

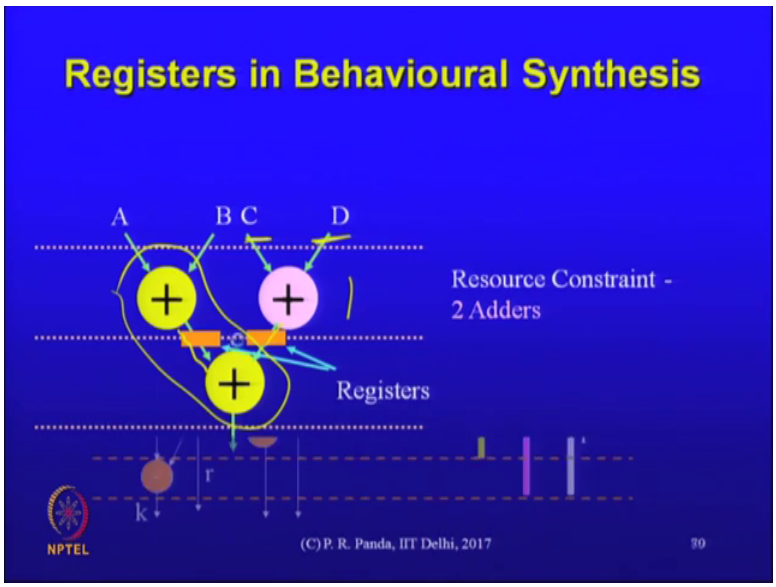
Right.

Student: (refer time: 35:30)

You can see that such a structure has it is advantages and it has it is disadvantages advantage is that it controls the layout and the routing a in the you know the placement and the routing you know in a more reasonable way. That would lead to a better architecture later on in the physical design step. Disadvantage of course, is that limitations are placed with respect to these data transfer that is another thing.

That should influence our scheduling decisions or in this case it is a binding decisions all of these data transfers need to be made and I need to bind each data transfer to one of these resources this buses are the resources to which the binding needs to be made. This binding problem shows up in different forms in different stages of high level synthesis, but the one that we will focus on to illustrate it is the register allocation problem. In fact, variations of that could be used to handle other binding problems also.

(Refer Slide Time: 36:43)

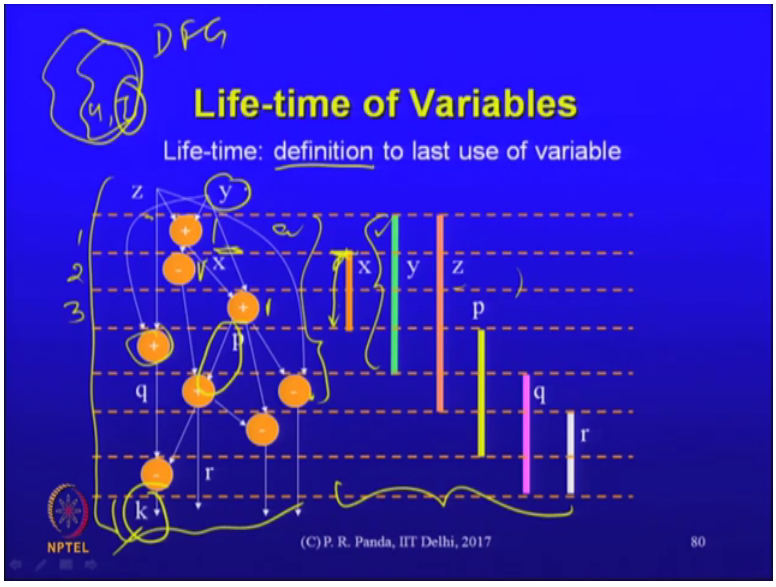


Registers in high level synthesis, the basic idea we had already seen these are inferred from scheduling decisions that we have already made. The inference is that if you have data being transferred across clock cycles then.

By default, we infer a register which stores the data at the end o F1 clock cycle and that data is available on that register to be used in a different clock cycle exceptions are there, but the general case is this when an edge graphically if an edge crosses one of those clock cycle boundaries then, there is the need for a register exceptions would be let us say there are 2 adders here right, one is this. The spring color is one and the yellow adder is different these 2 operations are shared on the same adder that adder does only one operation in this example. If that is the only thing that it is doing and let us say C and D were held stable in the first clock cycle and therefore, I have a result if they continue to be stable even in the second clock cycle then that.

Outputs value is not changing and therefore, there since that is not changing that could be used as input to the next operation. In such cases there is a possibility, but in the general case when an edge crosses a clock cycle boundary we infer a register.

(Refer Slide Time: 38:20)



Problem of course, is that for example, we want to minimize the number of registers here. A graph is scheduled in this way and the objective is to minimize the number of registers where. A register is inferred whenever any data edge any dependence edge like

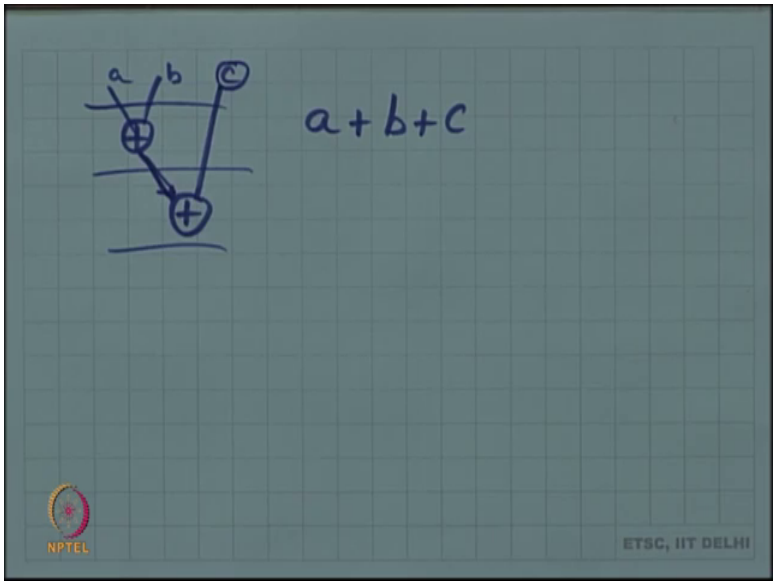
that is crossing a clock cycle boundary if an edge crosses multiple clock cycle boundaries then what does it mean as far as the register inference is concerned.

Student: (refer time: 38:55)

That register has to hold the value for multiple clock cycle.

This problem is formulated by introducing what is called a lifetime of variables. Remember variables are these right, on the edges you have labels those labels correspond to variables some of them may not appear explicitly, in your program. In your code you might not have that variable, but nevertheless the dependency might still be there because of the result of the schedule.

(Refer Slide Time: 39:38)



If you have a schedule like this, this could be the result of scheduling an expression A plus B plus C. Yes, that edge did not necessarily correspond to any variable that you declared in your code, but nevertheless it is.

Inferred you can assume that there is a temporary variable or something internally generated name of a variable that corresponds to that edge. Some of them may be labeled by names that you chose, but there are other edges that did not necessarily have a name that corresponds to anything you wrote, but we will still call it a variable. Anyway, for this purpose. I have a DFG that has been scheduled and let us define lifetime of a variable. In this way that variable x the lifetime for it begins after the first clock cycle.

Right this is when the data is ready. that is when I will store the output x in a register and I need the value of x actually I need it here in that subtraction and I also need it here for that addition.

But after this clock cycle I do not need x anymore. I will say that the lifetime of x is this starts with the first clock cycle where I need the value that is what is called a definition. Ends with the clock cycle beyond which I do not need the value. It is produced here it is produced at the end of the first clock cycle. it is clock cycle 2 onwards when I need it. that is where it is defined. Definition starts at 2. The lifetime starts at 2 the definition 2 the last use. The last use is in clock cycle 3 I do not need it beyond 3. that is the lifetime starts at 2 and ends at 3 for variable x y is an input to this graph perhaps it was produced by some other.

Node before this flow graph fine I am using y here that is the first that is the definition. I start the starts here and among all the dependent operations of y you see that this is the one that is scheduled latest. that is the lifetime for y the one for z starts here similarly I see that this operation will needed. that is the lifetime for z. Similarly, I identify the lifetimes of all the variables why are we doing this how does it help to identify the lifetimes.

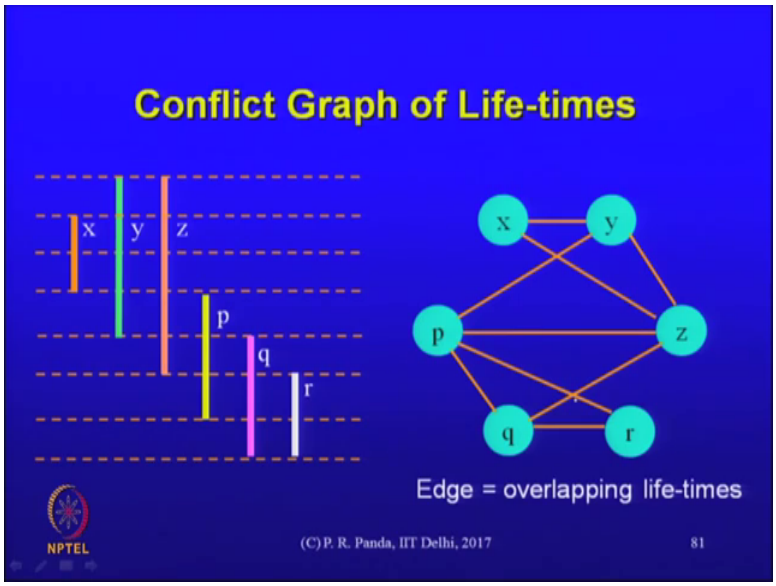
Student: (refer time: 42: 56)

Correct you these variables are going to be stored somewhere they are going to be stored in registers and the information that is being captured locally here is the register that is storing x needs to store it only for clock cycles 2 with 3 beyond that it does not need since it does not need that resources the register resource could be reused to store. Something else reduces as.

Student: (refer time: 43: 32)

Yeah, our objective we have to define what is the objective function given the scheduled data flow graph our objective function is to minimize the number of registers that store all the variables that need to be stored.

(Refer Slide Time: 43:47)



Here to we could come up with several different strategies let me illustrate here, a general strategy for solving the register allocation problem here is a way to represent this.

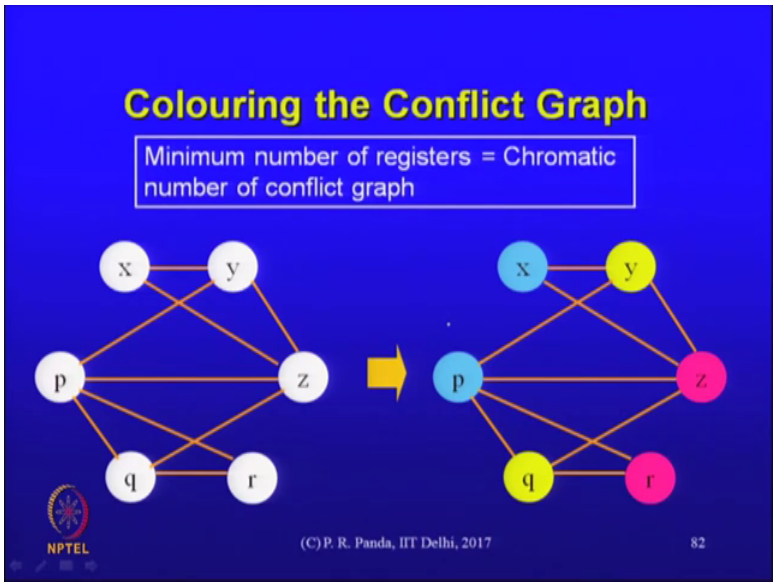
Let me have a node corresponding to every lifetime interval that is identified. For every variable I have a node in a graph and an edge between 2 nodes x and y is introduced in the graph if the lifetimes of x and y are overlapping. You see why it is called a conflict graph what is the conflict if there is an overlap.

Student: (Refer Time: 44:32)

They can not be since they both need to be alive, but we need to maintain both of their values at the same time. At least 1 clock cycle overlapping defined as there is at least one cycle in which we need both the pieces of data it is a conflict as far as their possible mapping into registers is concerned the same register cannot hold both the values. that is all. One node for every variable and an edge in the graph.

If the lifetimes are overlapping which implies a conflict as far as our register allocation is concerned then.

(Refer Slide Time: 45:10)

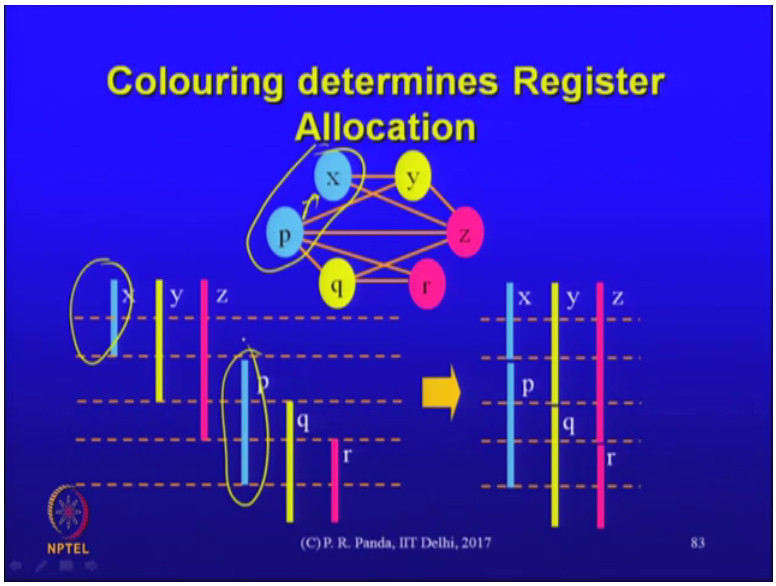


This register allocation problem could be the side effect of solving one of the classical problems of graph theory which is a graph coloring problem if you color the conflict graph we have to define what does the coloring mean. Coloring problem is defined as assigning color values to each of the nodes in the graph in a way that if there is an edge between 2 nodes then they do not get the same color. You can see that this example here the way we have colored these z and r have the same color y and q have the same color x and p have the same color this respects that constraint is that if 2 nodes are connected by an edge they can not have the same color. Objective of coloring the graph is you use the minimum number of colors to color a graph. While respecting the constraint that connected nodes connected by an that are immediately connected by an edge not that there is a path between them, but they are not connected by an edge. Then you if the 2 nodes are not connected by an edge you could assign the same color, but if they are connected by an edge then you are not allowed to use the same color. that is the graph coloring problem chromatic number refers to the minimum number of colors that you need for coloring a graph. We can go ahead and color this how does it help? Solve the register allocation problem.

Student: number of colors basically number of registers

Number of colors would be them the number of registers why.

(Refer Slide Time: 46:58)



Student: if you add all the different colors these are the main in an given cycle.

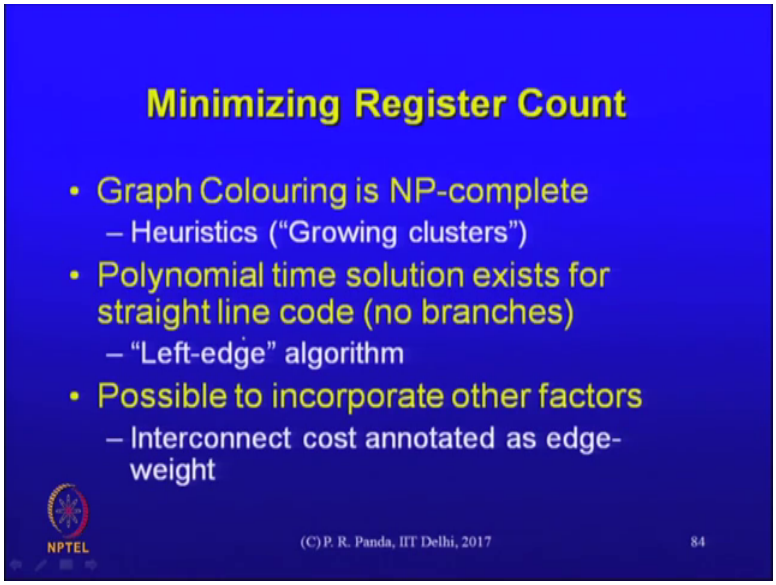
student: (refer time 46: 10)

Right what is that color tell us tell us if 2 nodes have been assigned the same color that didn't have an edge and therefore, they are candidates for sharing on a register. The number of colors we are using is an indication of the number of registers I need and each color. In fact, translates to a sharing decision that we have made for variables on a register. That color tells me that I have decided to share x and p on to the same register. In a straightforward way you can see that there is a mapping from a well-known graph problem into the specific problem of register allocation that we have this does not make it any easier it is just pointing out that there is an.

Association there is a parallel between the problem that we are solving here and a well-known problem what; that means, is that solutions that are there to that well-known problem could possibly be a by be used by us adopted in a straightforward way once we have composed this conflict graph we could use one of the standard well known graph coloring algorithms and solve our register. Allocation problem this kind of an analogy is very important we do not want to reinvent the wheel where there are known good solutions of course, we would like to adopt the good solutions for our problems. Then the problem just translates to the synthesis problem translates to the modeling of that

problem in terms of one of the known problems. That we can use the solution to that known problem.

(Refer Slide Time: 48:54)



This is a known problem, but the was not great is there.

It is known to be a difficult problem we did identify something, but that does not take us any closer well it does take us closer because it is a well-studied problem. Good solutions are there out there that we do not have to reinvent, but they too do not guarantee anything. But we are in comfortable zone. There because all through in our discussion we haven't been guaranteeing anything anyway we are just pointing out that all of these problems are difficult and what we are actually pointing out is something that is practical. But, not necessarily making any theoretical guarantees. That does not mean that I stopped as usual since the problem is important we still need solutions.

You can see that it is not hard to come up with some solution for the graph coloring problem. Finding the optimal chromatic number may be difficult, but finding a solution that works can not be that hard that growing cluster that I referred to could just be a simple heuristic in which let us try to build that heuristic right now. You start with a random node and assign it some color does not matter it is the first node that is being colored. Assign it red then, my objective will be to find other nodes that can also be colored red in a locally optimal way I can just go ahead and find all the other nodes that could be given red color could that be hard problem it can not be what is the algorithm?

I just I trait through all the nodes and if there is an edge with what is already colored red I am not allowed. But, if I find a node like r which is not connected to z then is it safe? For I to color it red or should I look at it is safe as of now it is certainly safe.

We are in the first iteration right after that I still continue. let us say I found r and then I still continue the iteration over all the nodes right I find a new node q. What is my algorithm? Now I should check whether q has edges to any node that has been colored red. Not just z, but also r because I will already taken the decision for r right. In at the end of 1 iteration I have found all the nodes that should be colored red it is at least a consistent solution I didn't guarantee anything through this process, but I am guaranteeing that I will find some solution it might not be the minimum chromatic number that way, but after that I remove all those nodes from consideration that we have already colored. I am now left with a smaller graph on which I run the same algorithm. that is the drawing clusters heuristic it is a very simple one, but it could be part of a practical solution.

That is one thing the other thing is that an efficient solution does exist for straight line code. A straight line just refers to code without branches that is within the DFG that is important here what we scheduled here was a DFG right this was a DFG there is no control here, there is no branch here. The point being that from the DFG I generate these lifetimes there is the question of finding the optimal register allocation for those lifetimes. It is eminently possible I didn't talk about that algorithm, I just introduced the most general formalism of course, once you translate it that way into a graph coloring problem, graph coloring is indeed a difficult problem, but for this special case.

Where you are only within a DFG this can be very easily solved and solved optimally where it does become difficult is extending the register allocation problem to work on a controlled data flow graph. Then you can see that I this is one DFG, but before that I had some other DFG. Now, these x s y s and so on at least the y s and z are also occurring in this DFG. The register allocation problem has to be solved globally for the entire c DFG for example, that register that you chose for y has to be the same as the register that you chose for y in the other DFG. Now, this problem that we solved is a local problem within the DFG, but the register allocation is beyond that because there are some variables well there are some variables like p that seem that they are local to that DFG, but there are others.

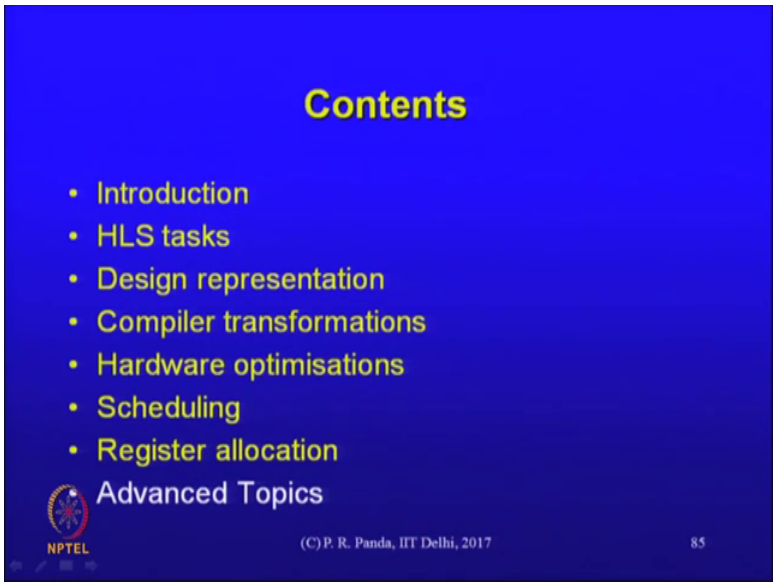
That are global the k that is going out is going into some other DFG. You could solve it optimally locally, but then you end up with a problem if y is assigned to register r1 in one DFG, but as part of the register allocation solution to the other DFG you assign a different register for y right then that is where the sub optimality is introduced within a DFG you do not really need to formulate it as a graph coloring problem. There are other ways of solving the same problem, but of course, a couple of reasons why we pointed out one is that the general problem is indeed just solved using a graph coloring. There are other equivalent ways of solving the that same problem without necessarily going into the graph coloring, but you can prove that it is an NP hard problem that register allocation problem is a is a difficult problem.

There is a different reason why you may want to go for that formalism for the graph coloring formalism for register allocation. It is possible to incorporate some other factors as part of the problem formulation itself. For example, a measure of an interconnect cost could be annotated on to the edge w8s. These edges that we had were simpler edges right either it was there or not there depending on whether there was an overlap or no overlap, but that is just one variation of the problem. Even in register allocation it could be that some solutions are better than the other because there is a downstream requirement on maxis and on right that we didn't take care into account. There may be multiple register allocation solutions that have the same number of registers, but they are not necessarily the same area because later on.

Extent of sharing is different the implication on maxis is different we will point that out later on, but as of now we didn't take anything else into account.

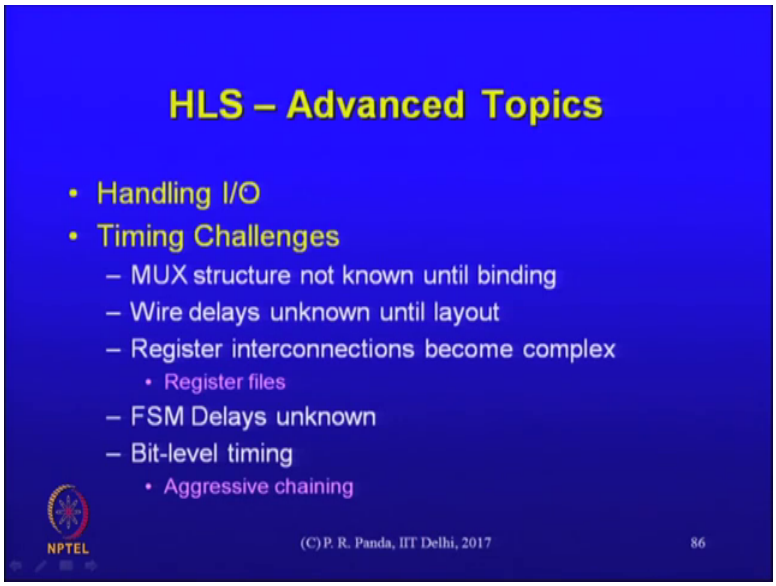
But, if we wanted to then the graph coloring formulation is good because we could annotate some interesting information on to those nodes onto the edges in the graph.

(Refer Slide Time: 56:26)



That was the discussion on the register allocation we did say that the particular manifestation of the binding problem that we studied is register allocation, but this kind of a strategy of building a conflict graph and asking some questions on the conflict graph is a general mechanism for solving many different binding problems here. It was variables being mapped on to registers, but there was this other problem of function units being mapped on to resources that too, we could solve using the same or a variation of the same strategy in all cases it could be that building a conflict graph and asking questions could be useful with respect to how resources will be shared for different elements different nodes of the graph yes. Conflict graph is still there, but it means different things based on what problem you have actually captured in that conflict graph.

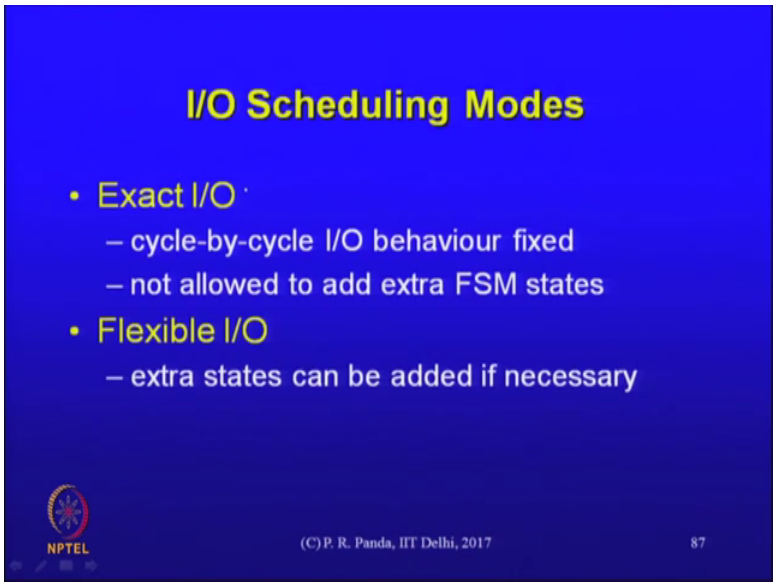
(Refer Slide Time: 57:33)



We can take a look at a few topics that are tentatively considered advanced. Here it is advanced only with respect to the basic algorithms that we looked at it is in reality is it is not. Advanced it is actually very necessary commercial tool that does high level synthesis most definitely needs to worry about such things. One advanced topic we already looked at which is an extension.

Of that resource constraint scheduling problem in reality it is likely to be an area constraint if at all not necessarily just count of the resource being constraint, but a few other things we have alluded to as we had these discussions we can point out in this section of the HLS discussion.

(Refer Slide Time: 58:24)

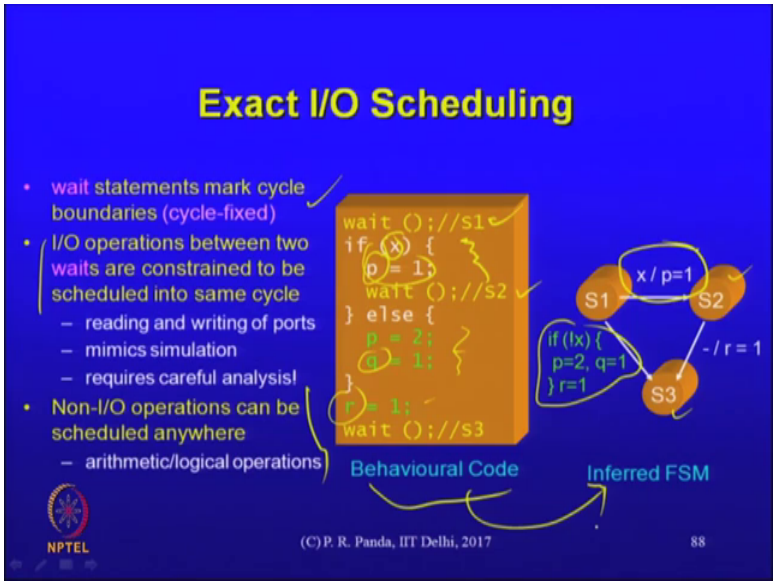


First is IO scheduling modes these are interesting variations that didn't occur in the basic formalisms of the scheduling that we looked at, but there could be protocol related requirements of the particular design that you are making, which may insist that a particular behavior be maintained with respect to timing it could be that cycle by cycle there is a specification of when what happens as far as the external inputs and outputs of a particular design are concerned. Internally, it does not matter what you do because nobody is able to see it, but from outside there could be a requirement that says I give you this data at this clock cycle you give me the result 2 cycles later. An exactly 2 cycles later this you can see is a variation of that deadlines problem time constraint scheduling problem. But, in fact, this could be extended to particular pair of nodes in the graph it does not have to be for the entire computation the variation that we looked at the entire computation a deadline was set, but instead it could also be that I take a pair of nodes and impose a timing constraint on them that is the class of this IO scheduling mode it could be that the cycle by cycle IO behavior is fixed other computations that are internal they are up to the scheduler.

But the IO behavior cannot be changed specifically what it means is if you fix that then you are not allowed to add extra FSM states. Remember the flexibility that we started off with which we said that the scheduler has is that DFG is given to us and we take the decisions of when what will happen? But if I fix the schedule of some of the nodes it means that I am giving up a part of that flexibility here these are IO nodes which means

that externally visible nodes the schedule relative schedule is fixed, but internal nodes you can still scheduler them whatever. This scheduler still has work to do even if it respects the cycle by cycle IO behavior. It isn't allowed to add extra states, but in a state, what happens it may have some flexibility on the externally visible nodes it has no flexibility, but for some of the other nodes it does have some flexibility.

(Refer Slide Time: 61:01)



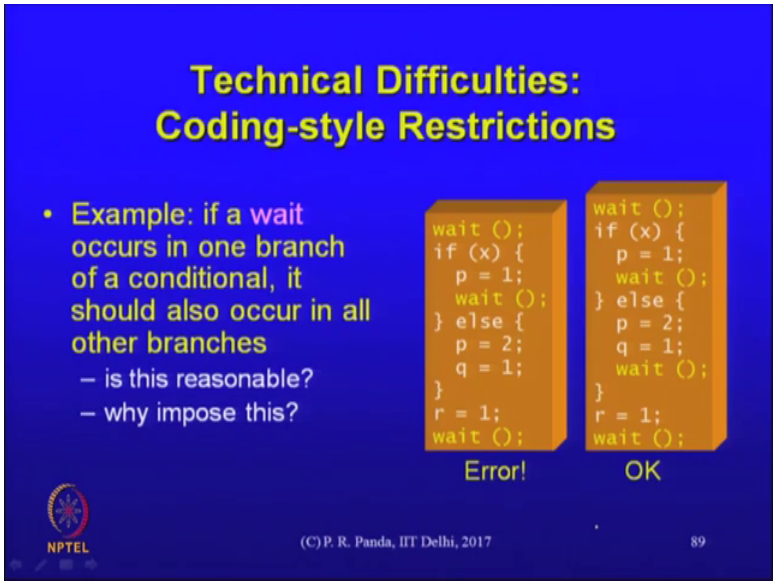
That is illustrated in an example here the language usage is system c, but that is already this w8 translates to our wait until clock event and clock equals one w8 on an edge. There is in general the w8s like we said they can be distributed anywhere in the code. Of course, but these w8s in general could mark cycle boundaries. That operations that you perform here if they were IO if p was an output and x was an input and q was an out. All of these if they were externally visible operations then there are certain requirements that are fixed what is the requirement. let us say that w8 corresponds to a state there is other w8s corresponds to another state here and that third w8 corresponds to this is right this is how that that.

that fsm would be inferred from the code that you have written normally, we could expand the path from one way to the other w8 into multiple states if necessary, but if I am fixing the schedule it means that you do not have a choice. If you do not have a choice then it means that this test for x and the assignment to p all of them has to happen in that transition from s one to s 2 here we are annotating the actions on to the transitions

in a mealy format. IO operations that are there between 2 w8s are constrained to be scheduled all into the same cycle as long as there is a path from one way to the other w8 all the IO operations there have to be scheduled into that one cycle. That there is a path from S1 to S3 directly because in that other branch here you do not have a w8. P equals 2 q equals 1.

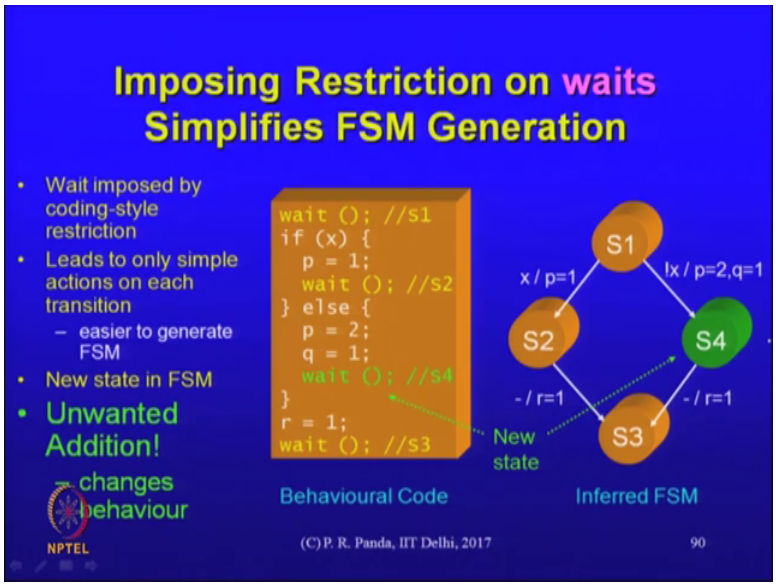
Followed by r equals 1 all of them actually have to happen in the same cycle that is the requirement of an exact IO scheduling non-IO operations can be scheduled anywhere if there are arithmetic and logical operations that others are not able to see, but IO operations need to be scheduled in to specific cycle. From behavioral code like this you see that you need to infer an FSM that looks like.

(Refer Slide Time: 63:29)



That and sometimes that might actually show up as coding style restrictions that a tool might impose. Maybe they will say that this is not allowed, but if you have a w8 in one branch, then you must have a w8 in a different branch. That restriction may be imposed is this acceptable? That depends on your design if the protocol did not allow then it is not acceptable it is.

(Refer Slide Time: 63:57)



From an FSM generation point of view, you know exact what to pick up and put where.

But it is an unwanted addition of a state if I insist that you must have a w8 there that translates to a new state that is there in the FSM now whether it is or not it depends on the situation if this were to be strictly IO fixed scheduling then then that is an unwanted addition. It is expected that to be able to handle such situations and handle IO in a way that is correct with respect to the specification. Let me stop here and we will continue with some of the timing approximations that we are making and what difference they would make to the synthesis strategy.