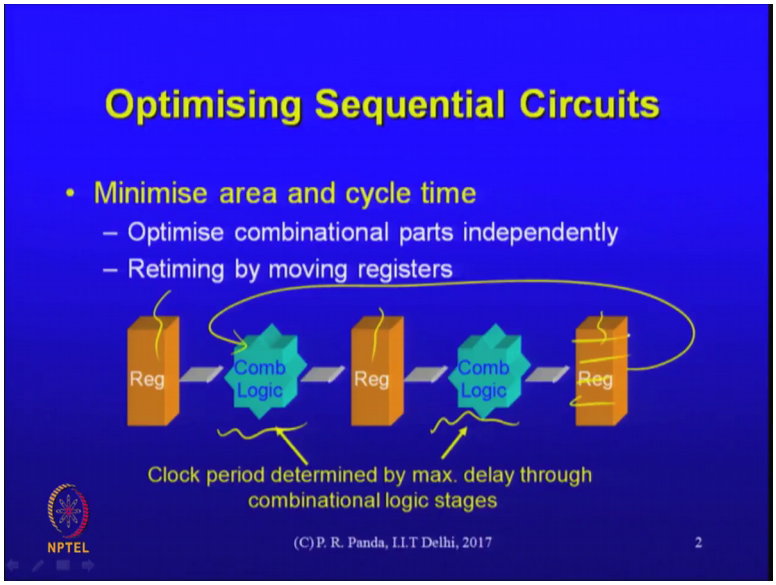
Synthesis of Digital Systems
Dr. Preeti Ranjan Panda
Department of Computer Science & Engineering
Indian Institute of Technology, Delhi

Lecture – 18
The Retiming Problem

Ok so, we talked about a new problem today called retiming. Although it is introduced at this stage of the synthesis, as we look at the problem you will realize that it is actually a lot more general with respect to its applicability.

(Refer Slide Time: 00:36)

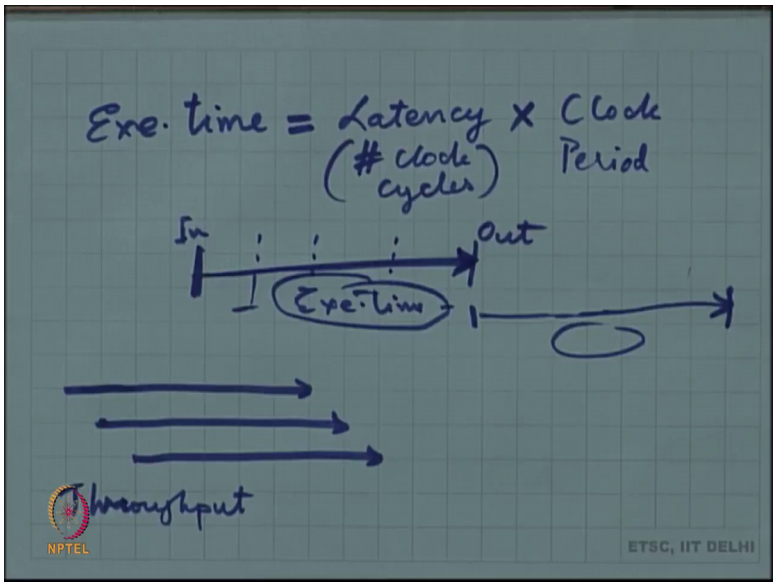


The context here is that you would like to optimize sequential circuits. Optimizing combinational circuits is something that comes later, but a sequential circuit has a combinational component and a flip flop and register component and one way of optimizing the sequential circuits is you just minimize the area of the combinational components or delays of the combinational components independently.

There are these two aspects one is an area and the other is the time. How do you define the time of a sequential circuit our circuit overall may look like this there are blocks of combinational logic that are separated by these register stages this is a fairly generic way of characterizing any sequential circuit. It is not necessarily unidirectional like this you may have cycles all of that is fine not within the combinational logic, but certainly from here you could have outputs that are feeding back to some other combinational logic.

But, one can view the overall sequential circuit as separate pieces of combinational logic and partitioned by stages of registers that is one view. So, if I were to optimize this what kind of optimization we are talking about of course, if it is an area optimization then we try to just do a logic optimization of all of those combinational pieces separately, that does reduce the total area. There are the registers here translating to a bunch of flip flops there, we can try to reduce the number of the flip flop and that also reduces area.

(Refer Slide Time: 02:43)



What if I want to optimize timing? What do we mean by timing? There are two kinds of timing that are involved execution time can be represented as the product of two things one is the latency which is the number of clock cycles times what?

Student: Number of (Refer Time: 03:08) states.

Number of states, the total run time of an application would be consisting of two components; one is how many clock cycles it took, other is?

Student: Unit.

Unit of?

Student: Unit of clock units.

The clock period.

Student: Clock period.

So, of course, it says it is just total time that we are talking about. So, that is the clock period those are the two components that determine the total execution time. Total execution time is not the only timing related metric, you could also have other metrics like throughput. The throughput would be once and how many cycles are you able to put out a new result?

So, these are in some ways orthogonal to each other these are different matrix one is how much time does the total execution take data is given here and you take that much time and that is our execution time right. So, that is one metric of time. The other metric is if data is streaming in.

Student: Streaming.

Right.

Student: (Refer Time: 04:33).

So then there is a different metric you may have that much execution time, but when are you able to accept the next piece of data and correspondingly when are you able to produce the next output. So, if this is where the input is given and that is where the output is ready the simplest system is one where you would accept the next input here right and you would take that much more time that much execution time for the next input and so on.

But, of course, our systems need not be designed in that way. It is possible that that is my execution time and even though the first one has not completed I am able to accept the next input and maybe so, if I have designed my system in that way then maybe the next output is produced after a relatively small amount of time, ok. And, similarly the third input could be processed here and so on.

So, such a system requires a slightly different metric than merely execution time, which is the processing time for one packet of input data or something. So, that I would characterize there is a throughput related metric which is at what rate am I able to accept input or equivalently at what rate am I able to produce my output ok. Execution time nevertheless is an important metric.

So, what determines the execution time in our sequential circuit?

Student: Sir, just one question.

Yeah.

Student: Would be a tradeoff possible between execution time and throughput, for a sample?

Yeah, there could be, yeah.

Student: (Refer Time: 06:30).

Certainly, that can be tradeoff. Well, it shows up in the design style there could be an area implication of a higher throughput.

Student: Right (Refer Time: 06:42).

That you.

Student: (Refer Time: 06:43).

You can see that standard way to do it is by doing a pipelined design. A pipeline design means that if there is combinational circuit that produces that output you break it at appropriate times. We will see this same problem in more detail right now, but you break how do you break it you put registers there. Putting registers increases the area of the circuit. So, there is some implication there.

That we will study in more detail, how to manage that implication, but execution time is determined by two things. in this picture what are those two metrics? What determines latency? What determines clock period? What determines latency here in this circuit?

Student: (Refer Time: 07:31) combinational sequential.

No. Let us see the way we define in terms of the number of clock cycles it is an integer it is a number, that number should be clear from this picture what is that number?

Student: 3.

Yeah just the number of flip flop stages through which the data is going. Does not matter how wide or narrow each stage is, right that is the other metric which is the clock period. So, what determines the clock period of this circuit?

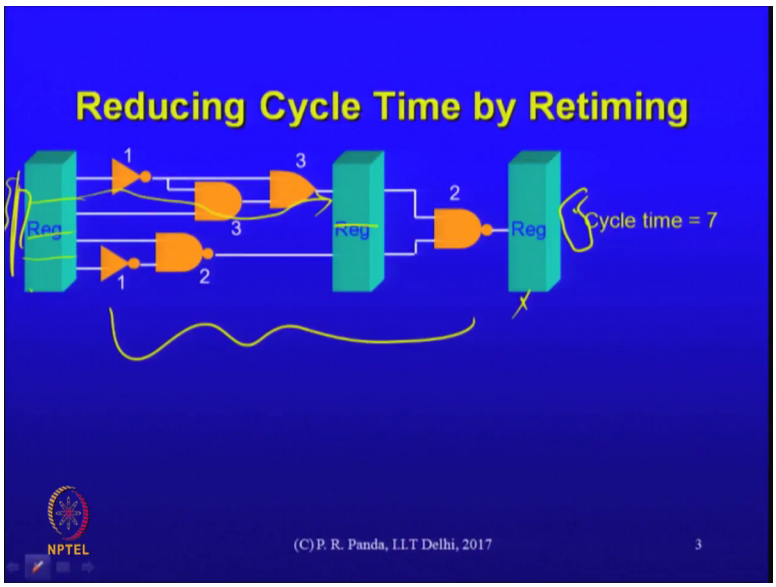
Student: Maximum delay.

Yeah, I have to.

Student: (Refer Time: 08:00).

Somehow look at the delay through each of those combinational stages and the maximum out of them tells me the rate at which I can clock that circuit the fastest rate at which I can clock that circuit or to respect the maximum combinational delay of any path in any of those pieces of combinational logic ok.

(Refer Slide Time: 08:24)



So, in this context let us see what flexibility we may have. Consider a gate level design of this nature. These registers are just an abstraction of four different flip flops that are there here right each bit corresponds to the q output of a flip flop that we are taking we are just calling it as a register stage, but there are actually four flip flops there. Here, similarly there are two.

Timing of this circuit is determined by what for this given circuit anyway the number of stages is fixed right the latency in terms of the number of clock cycles is fixed. Clock

period is what would determine the timing and that clock period would be what our cycle time is the same thing. We said that we should look for the longest path through any of those combinational stages and the longest path here is if you were to go through that and then this and then this right; that is what makes the required cycle time at least 7 units.

There are a few other components like we had pointed out when we analyzed timing for the behavioral synthesis. There is a set up time for the for that next stage of the flip flop there is.

Student: (Refer Time: 09:48).

A propagation delay those are of course, there we are just ignoring that knowing that finally, I need to adjust these delays that I get by just looking at the combinational logic to those are the other things that are constant for this treatment. So, they are not explicitly being handled here ok. So, that is the cycle time.

Can I do something about it? Do you see there is some opportunity for improving the total execution time of this circuit? Total execution time you do not have that much flexibility anyway we said it is the product of latency and clock period. Latency in terms of number of clock cycles can I change for this circuit?

Student: (Refer Time: 10:45).

It is, it is fixed. If I if I remove one of those registers say I remove one of these registers then somehow I am changing the behavior of the circuit, right. So, behavior means what is the input that is what is visible externally, right and similarly, the output is what is visible externally it is assumed that these internal what we are doing inside the circuit that is not the visible external. I can do whatever I want within the circuit, but external behavior I do not change that is my specification for this class of optimizations.

So, then what can I do about it? I.

Student: Logic logical (Refer Time: 11:18).

But.

Student: (Refer Time: 11:18) circuits.

As a first step of course, I should minimize the respective combinational logic in every stage I should go ahead and we have not gone through the logic synthesis stage, but we know what you what logic minimization involves and we cannot just minimize, but minimize it for delay which may be a different minimization than minimizing logic for area, right.

If running this circuit at the fastest clock is my objective then I should be optimizing it for delay. Fine, I can do that is this such an opportunity in the circuit? We may not have seen the formal way yet of optimizing for timing, but it do not obvious, there is an inverter then an AND gate and then that path can it be brought down further can it be reduced the length of the path.

How about if I consider both the stages together? let us assume that we have the optimized logic. So, the gates I am not going to change, that is optimized gates. Still, there may be an opportunity to make the circuit faster, where faster is defined as the total execution time.

Student: (Refer Time: 12:31).

There may be an opportunity to minimize that logic like I said, let us treat that this is the output of a minimized logic, right. Of course, this is illustrating an orthogonal concept, that is why it has brought up. But, one thing you ought to notice, there is an imbalance between the two stages.

Student: Right.

Right? First stage the delay is longer than the second. Could we do something about it.

Student: We can moves or all logic.

Right.

Student: But that depends what kind of register methodology we are using.

They are flip flops.

Student: Ok.

They are flip just flip flops.

Student: Because latches allow you to borrow or (Refer Time: 13:20).

Defined. What do you mean by borrow?

Student: Not equal to.

Structurally you could do something about it. Remember, nobody cares what you have inside those two stages. This first stage of register, these we are not supposed to touch these also we are not supposed to touch anything in between we can.

Student: Sir, middle register can be.

Restructure if we want to. Middle register can be?

Student: Eliminated if the formal direct transform the register (Refer Time: 13:55) middle should be logic (Refer Time: 13:58).

Middle register can be eliminated. Eliminating the middle register.

Student: (Refer Time: 14:05).

Changes the behavior of this circuit you might actually get the result faster that is a different thing, but it changes the behavior in what way?

Student: More often architectural change.

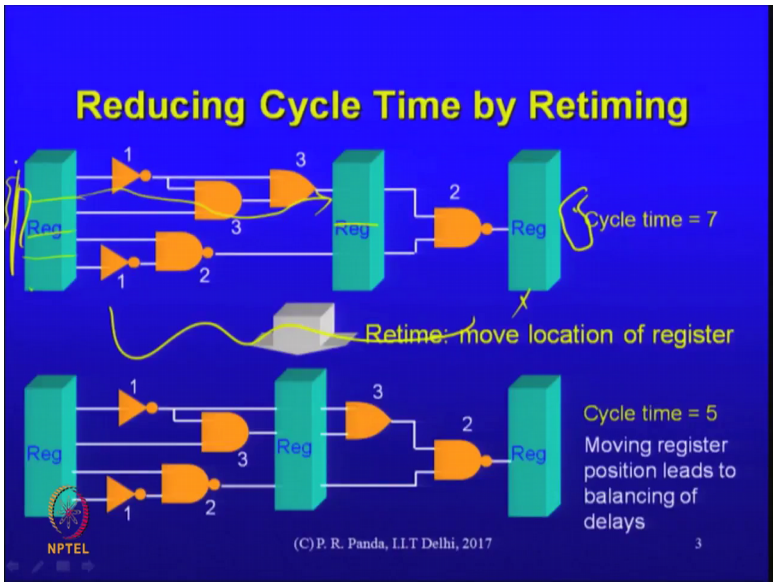
But what is the behavior that it is changing?

Student: Some states will not be there (Refer Time: 14:22) will be more if I.

If you change in the cycle time might be moved, but you have one clock cycle as opposed to two clock cycles.

That is the reason for eliminating it right there is only one clock cycle, but that does change the timing behavior of the circuit in ways that we do not want to. We said that the cycle to cycle behavior of the circuit as viewed by the external components whoever is watching that output earlier when you made a change here at the input the output was ready two cycles later.

(Refer Slide Time: 14:50)



That behavior we want to preserve. If I remove the internal register then you get the output one cycle later right, that is a different optimization we could try that out, but that is not the objects I am changing the IO behavior of that circuit with respect to the clock edge..

So that let us not change that for it is allowed or not allowed it is not clear, right? We said in a behavioral synthesis tool that may be allowed actually.

Student: Yes.

It is a you are allowed to put in or subtract put in a new cycles new states or subtract some states. when you get down to the next level, RTL level, for instance you do not play around will you do not assume that this flexibility is there.

Logic level and so on. This flexibility, but some other flexibility may still be there.

Student: (Refer Time: 15:47) they should work from logic (Refer Time: 15:48) stage 1 to stage 2.

We could shift some logic from stage 1 to stage 2 in a way that the external cycle behavior.

Student: (Refer Time: 16:00).

Clock edge behavior does not change, but it changes what? How does it help to move something from?

Student: Value of the register.

Reduces the maximum cycle time of the first, the first stage is the critical one right for this. So, the first stage inc decreases the second stage it increases, but that does not matter because the bottleneck is in the first stage, right. So, what is actually happening? So, what do I select? Which gate I select for moving possibly to the next stage?

Now, this much combinational logic is there, but where I place that register is up to me, because nobody is able to watch it externally. So, I can do this, I can just move that gate outside, right. This gate can be moved to the second stage, what is the consequence?

Now, let us recompute the critical path, how much delay is there in both the stages. This one has increased. So, these are delays that these numbers that that I have annotated are delays. So, this has increased from 2 to 5, but this one has decreased from 7 to.

Student: 4.

4.

Right so, the maximum delay of any combinational logic path in the overall circuit has decreased. Latency has not changed in terms of number of clock cycles it has not changed, but the clock period which is the fastest rate at which I can clock this circuit has improved. That is what is the classical optimization called retiming of a circuit. Circuit is given to us let us assume it is optimized for area or whatever other matrix. So, logic is otherwise minimized, right.

But, I have the ability to shift around what have I shifted here these are the location of the flip flops I have shifted in this case from here I moved it back. In fact, moved it back means that actually one flip flop became two flip flops, right. That is what happened.

There are three flip flops there in that register stage now, nevertheless my timing has improved, right.

Student: Sir, I have a question.

Yeah.

Student: (Refer Time: 18:20) sir, now like we have 3 input register and 3 output whereas, earlier we had 2. So, this will impact our resource constraint also?

This impacts the number of flip flops, through doing this you are increasing the number of flip flops. So, gates you are not changing.

Student: Yeah.

But flip flops you are increasing. So, the total area increases as a result of this all right. That is the tradeoff we have always been pointing to a delay versus area tradeoff, this is one of them.

Student: Sir.

Yeah.

Student: Are we still doing the behavioral synthesis steps or are we in logic optimization?

We have transitioned to the next stage so, but what I do want to point out is even though this illustration is with gates you could well replace those gates with bigger units and all of these analysis is still applicable, right.

Student: Sir that is why I am (Refer Time: 19:18).

In fact, behavioral synthesis tool do this.

Student: They have to.

At that level, but these are RTL components not merely gate level components but, RTL components and they do this they are expected to do this.

Student: Sir, typical clauses we follow is after logic synthesis or any stage when we restructure the net list is some memory.

Yeah.

Student: We go through a logic equivalence checking.

Yeah.

Student: Wherever the RTL is compared against the gate level.

Yeah.

Student: Circuit for the functionality. I do not know the full details of it, but would not this break the LEC Logical equivalence? If I remove the register this way?

You are changing the logic of the individual combinational blocks?

Student: Yes.

Right? Moving the gate means what? You are of course, changing the logic. You can still do the equivalent, but remember that this is what was done and maybe as far as functional equivalence is concerned you can consider dropping those register the intermediate registers, but awareness has to be there of doing a.

Student: (Refer Time: 20:20).

Structure.

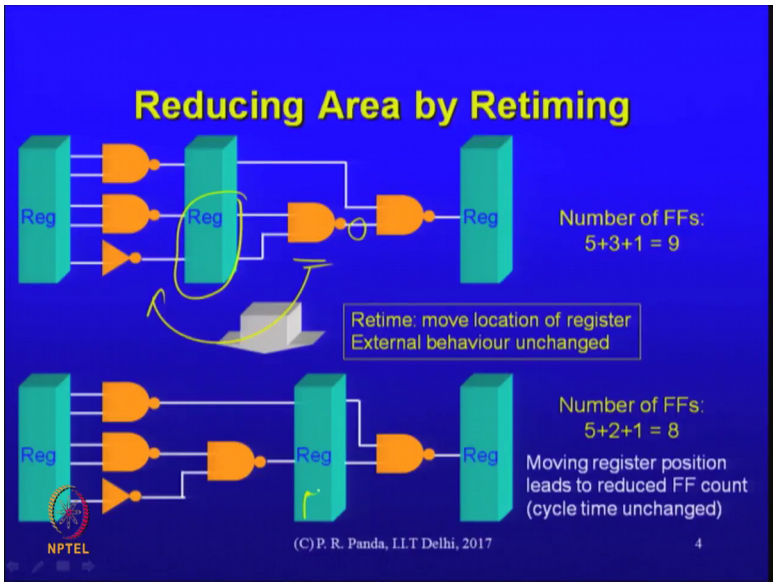
Student: The first one and second one.

Yeah.

Student: Would they decide as functionally equivalent when LEC is done?

We have to tamper with the meaning of equivalence. The overall circuit is still functionally equivalent, but it would be mismatch if we were to compare the individual stages, right.

(Refer Slide Time: 20:41)



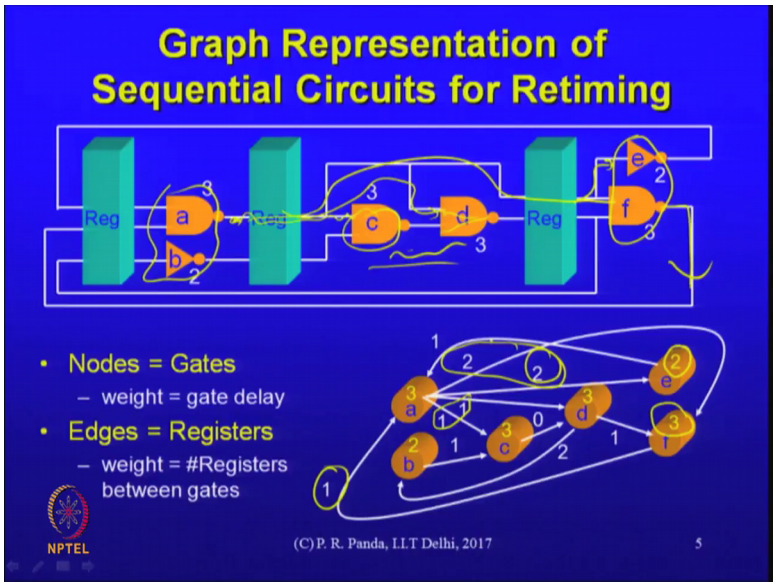
Equivalently we can of course, consider this as an area optimization. How? The flip flop count is this, right there are 5 flip flops here, 3 flip flops here and 1 over here.

As a result of any retiming that you do here it is the other way around that register here. These two flip flops will move to the output that is what happened essentially by moving that gate to the previous stage, it is the reverse of what we did earlier and what do you see? The number of flip flops has reduced because this you know middle stage has gone from three flip flops to two flip flops this is a different optimal it is an area minimization optimization, but it is compromising on timing here.

So, you could formulate this as a timing optimization or you could formulate this as an area optimization still retiming ah, but the targets can be different, but of course, the popularity of this is in the timing context area we can assume that we are optimizing through other means and as regards number of registers a number of flip flops and so on.

Remember, those decisions are involved in many other optimizations too; like the FSM state assignment problem and coding problem there also the choice of the sequential area was involved, how many bits do you use for the state register that was used?

(Refer Slide Time: 22:10)



Ok that is the problem definition. The problem is that a circuit is given to us and we would like to restructure it by way of choosing the placement of the registers in a way that the maximum delay through any of those stages is minimized, so that our clock can be the fastest? Is that clear as the just the problem statement, right

Student: Sir, every problem can be boils down to the graph (Refer Time: 22:41).

This is a favorite tool of synthesis researchers who know only graphs. So, whatever your problem is I understand it in terms of graphs. There are many more graphs to come. So, just hold on.

So, where is the intellectual input over here, since solving this graph problems are some other communities business. We are not smart enough to solve those graph coloring or ah

Student: Um.

Ah The other problem that we embedding problem.

Student: Embedding problem.

That we talked about the other day here new problems will also come up, but there is some contribution here that contribution is right there. Modeling of your problem whatever it is in terms of a graph a. So, graft is a sort of a simple mathematical formalism once you capture it then you could ask questions on it. Perhaps those

questions are questions that have been asked before in a different context of course, you can use all of that knowledge.

But, this tends to be an important maybe I would say the most important contribution of synthesis and EDA research in general that is you see a problem that seems as though it is very electrical in nature, what you want to optimize delay anyway what determines delay it is it is an equation that has capacitances and so many other terms, but while you could solve that if the circuit were very small. If the circuit is very large there is the need to model it in terms of some abstraction for which the solution mechanism scales well. Right, the reason for this kind of formalism is that.

Actually as we solve the retiming problem we will see that we do borrow other mathematical techniques from different domains. Graph is just the initial representation ah, but we would like to of course, use all the mathematical tools that are available to solve our electrical problems and this graph gives a sort of a nice way to capture some of this, right. The choice is not obvious if I have represented it in this way then it of course, the reason it was represented this way is that the solution becomes clean or nice or elegant solution somehow presents itself once you represent something in that way, right.

In fact, this is an unconventional graph even though structurally it looks like just a graph I know everything about it. Even within that it is not obvious if I have posed this problem to you the net list problem and if I also give you the hint that you can use the graph that is also not enough hint for you to first represent the problem and solve it.

So, once we discuss a solution it becomes only then obvious why we are choosing this structure. Graph is obvious net list looks like a graph right if you change the shape of those gates to all circles then that becomes a graph. So, in that sense it seems as though there is a very natural mapping yeah.

Student: This is not a dag. So, we (Refer Time: 25:42).

This is not a dag, the cycles are there.

Student: (Refer Time: 25:45).

Right, this.

Student: We know longer have dags.

Yeah, we are moving into more complicated graphs this is just not a.

Student: (Refer Time: 25:50).

Directed a cyclic graph. So, retiming is a problem that works if you have cycles too.

Student: Yes.

They are not combinational cycles they are broke every cycle is broken by some register

Student: (Refer Time: 26:02).

But, that is all right that is that does not affect this formulation, right. So, all of those other solutions that you had that worked only on dags they do not work on this, ok.

So, we build a graph that is the easy thing to say ah, but ah, what is there in this graph fortunately consists of only two kinds of things one is a node.

Student: (Refer Time: 26:23).

Other is an is an edge. So, node can represent a gate that is just a change in shape of these guys.

Student: (Refer Time: 26:30).

That is very easy. What about the flip flops?

Student: Representing (Refer Time: 26:35).

Yeah, it is not obvious that it should be done that way, right. The from the picture you can infer that that is what we have done, but why should we do it in this way? Why do not I make the flip flop also separate node?

So, that is where there is some intelligence in this whole process nodes are gates. So, edges represent registers. We attach some weights to both nodes and to edges. Why we do it in this way will become clearer when we do the analysis, but these node weights correspond to the gate delays remember even though we are talking about gates and such things the retiming is a much more general problem you could have adders there and you

could still pose the retiming problem you could have other kinds of much more complex units, you could have processors there and you can still ask the same problem.

Abstraction changes and correspondingly some of the meaning changes, but nevertheless it is good to think in terms of a concrete example like this circuit to understand because retiming is something that is popularly done certainly at this lower levels of synthesis abstraction ok. So, these are the delays. Of course, through this process we have abstracted away some things, we have made the assumption that the gate can be characterized by only one delay number. We already saw earlier that that is an approximation because every path would have a different delay right this path has a different delay that path has a different delay. High to low transition has a different delay, low to high.

Student: (Refer Time: 28:16).

Transition has a different delay, standard condition, minimal, max condition process corners have different delay. So, the delay is in general a much more complicated beast, but this is just abstracted out in terms of one number, that number is the max out of all of those path delays ok. So, that is the node weight.

What is the edge weight? First of all, when do we have an edge? When there is a connection that goes from the output of one gate. Eventually, to the input of another gate.

Student: Single.

Possibly through some flip flops.

Right, through some register stages. That is when I will include an edge. First of all it is not a fully connected graph. All nodes do not have edges to all other nodes.

So, certainly these two nodes do not have edges going from one to the other these two nodes would have, right. These two nodes do not that does not mean there is not a path from one node to the other. There actually could be a path eventual path you come down this way, you go through this you can reach that other node, if there is a cycle in the graph you can reach it. But, that is not what we are capturing here the edge here is an immediate connection from output of one gate to input of another gate through some flip flops possibly ok, but not through other gates. So, that is what we have captured here.

And, that number there is how many flip flops do you cross in going from one to the other. So, things like this c to d.

Student: 0.

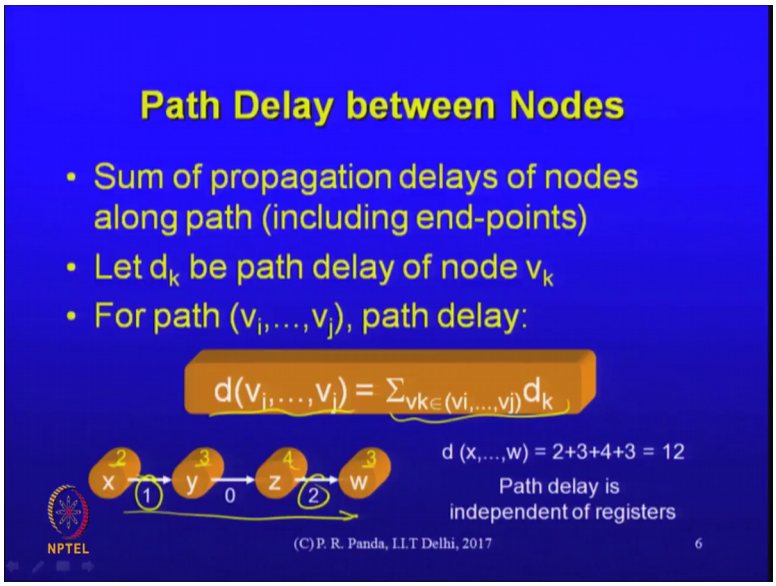
That number is 0, because it is a combinational path, right. You do not go through any flip flops, but most others you are counting how many flip flops would go through. So, from f you expect edges to a. In fact, there is only one outgoing edge from f to a with one being the edge weight because you are going through one flip flop. from some others like from a you have path to c and to right to e and to f with different edge weights, but you have three different paths.

So, that is why is there more c, d, e, f ok, to actually to d also there is a path right. So, c and d it is able to reach with one flip flop stage.

And, e and f it is able to reach with two flip flop stages. So, that is my graph construction. Any questions on this?

We should be absolutely clear on the way the graph is extracted from the net list.

(Refer Slide Time: 31:05)



Ok, there are some definitions here note them as we make them. Define a path delay between two nodes ok. So, the path delay is defined as the sum of the propagation delays of all the nodes along the path including the endpoints. So, if I have a path like this in my

graph from x to w. So, this path can now consist of multiple nodes along the path what are the delays. So, these are the delays.

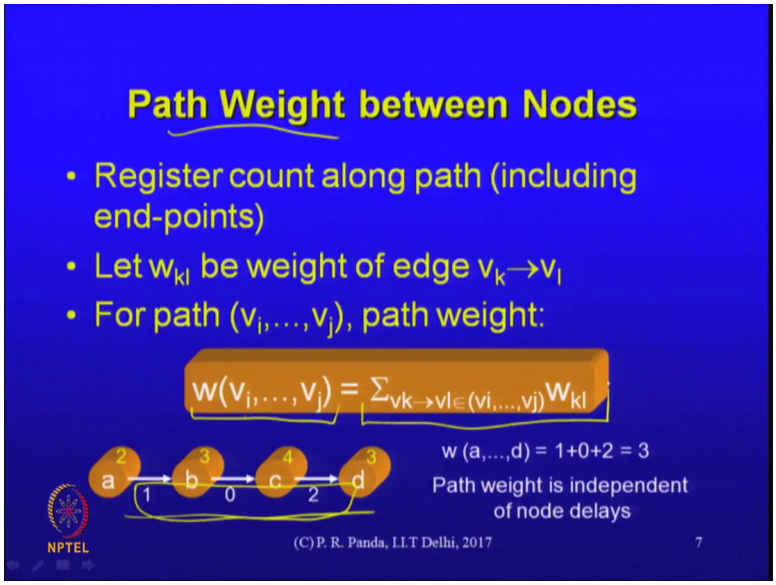
Student: 2, 3.

And, we say that that path delay from v i to v j is the sum of the individual delays of the nodes that are encountered along the path, ok, including the first node and the last node. This is independent of how many registers you are going through that you are going through 1, 2, 3 registers that is not counted. This is essentially what we are counting here is the combinational delay from the first one to the last one, if there were no registers in between it is it is it is a independent.

Student: The path delay is without.

Without considering registers. For good reason, right because we might want to move those registers, right. If we move those registers then we need to respect that path delay. So, anyway this is defined independent of any registers that are occurring in the path. So, this is not the technically what is the critical path the way we define when we analyze a combinational logic, but this is just for the purpose of this optimization that is the path delay [FL].

(Refer Slide Time: 32:42)



And, to complicate matters a little bit we also talked about a path weight that weight is an orthogonal thing it is the register count along that path.

Student: Right.

So, for the same path the weight would be the sum of these numbers, right. How many registers do I cross as I go from one node to the next one. So, that we represented like this we say the path weight from i to j is the sum of all the edge weights that occurred along the path.

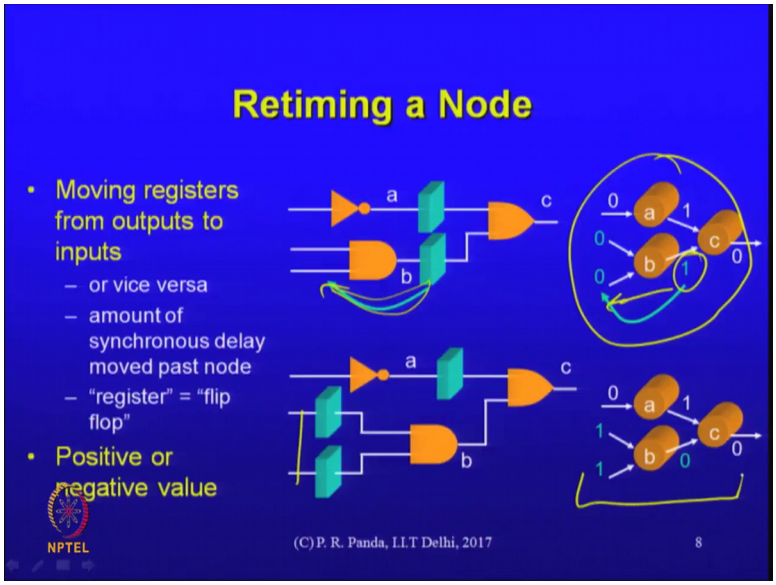
Student: It is the representative of the number of registers.

Correct.

Student: (Refer Time: 33:24).

Number of registers as you go from output of one to the to the input of the next. So, this number is defined in a way that is independent of the delays of the nodes that it is encountering. You are only counting flip flops not counting how long it takes to go from one node to the other, ok. So, those are definitions.

(Refer Slide Time: 33:46)



Retiming of a node is defined as follows. We say that an individual node is being retimed in the graph which is to say that you are moving some registers from the output of a gate towards it is input the direction is just a convention it also works if you consider it from input to the output, but for this optimization we will consider that a positive value of a retiming number that is associated with a node means that there are some registers that

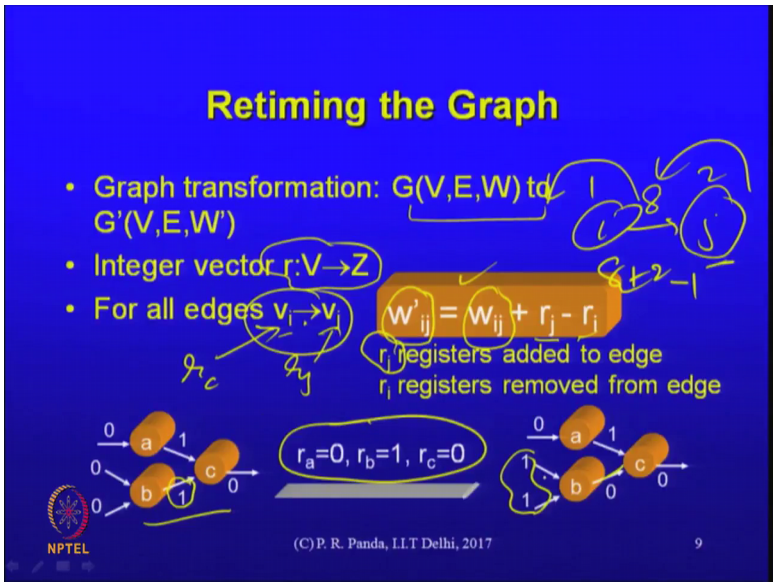
are moved from it is output towards the input. Remember, in the process you may be creating more registers right as we move the register from output towards the input we actually created two registers.

But, that is considered a retiming value of one essentially it is a number, but we say that amount of synchronous delay that is moved past a node that is the retiming value that is associated with a node.

So, that circuit had a graph that looked like this and a retiming of b equal to 1, means that you take that 1 and you move it towards the inputs you get this kind of a graph as a result of retiming of that particular node b with an integer value 1. This could be positive or negative; negative means that you are moving things from input towards the output; positive means you are moving it from output towards the input.

Student: Input.

(Refer Slide Time: 35:19)



And, what we just defined was for one node retiming of the entire graph means essentially you are transforming the graph, in what way? This graph consisted of a set of nodes a set of edges and a set of weights on the nodes and the edges. What happens as a result of retiming what kind of transformation it is?

Student: (Refer Time: 35:42) edge weights (Refer Time: 35:44).

Only edge weights are changing. The nodes are not changing, edges are not changing, so, structurally the graph is not changing, but the numbers on the edges are changing because you have associated a retiming value with every individual node. Those values some of those values can be 0, it means that you are not moving anything past that node, but some of them could be positive or negative meaning that some registers are flowing across it as a result of a retiming.

So, you think of that retiming of a graph as a vector r where for every node you are associating an integer for a there is an integer, for b there is an integer, for c there is an internal integer. So, the retiming of the graph is just represented as a vector of integers an array of integers one for every node. That is all.

The transformation we can represent like this for every edge v i to v j right there is a retiming number r j because j is the output node. If there is a retiming number r j for this if there is a retiming value r i for the node v i then w ij was my original edge weight.

From v i to v j. The new edge weight is given by the original number plus number of registers that you are adding to that edge because you move something across v j. So, some registers were added because of retiming v j some registers were subtracted because of retiming v i.

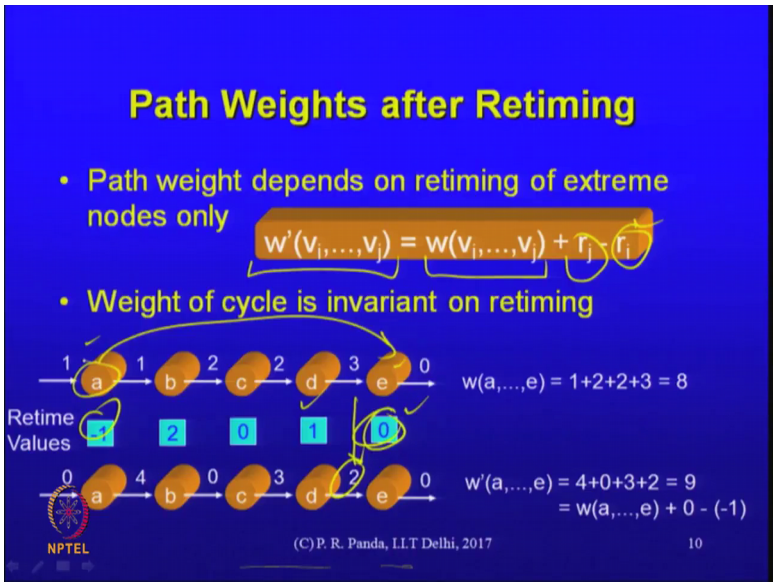
So, the new number would be w ij plus r j minus r i right I have i to j this already had some number 8, right. So, when I do retiming I associate a number 1 here and maybe a number 2 here. This 2 means two more registers are getting added there we will call them registers of flip flops and registers are used interchangeably. One number here means one register got subtracted from that edge. So, what is the retimed well after retiming what is the new edge weight? It is 8 plus 2 because you added to by retiming j right minus 1, because you subtracted one register from there. So, that is what this equation is telling us, pus as I mentioned, right.

So, in this example if I start with this and I use a retiming vector for the whole graph being 0 1 0 for a, b and c then the implication is that one register is taken from the output of b and that is moved towards this, right. So, the new edge weight is 1 minus 1 because that one is taken away. So, the edge bc there is plus 0 minus 1, right; c is not being retimed, nothing is coming from c, but b is being retime. So, one is being taken off. So, all the edges can does be recomputed as a result of one retiming vector. This is still only

a representation we are not solving anything yet ah, but the beauty of this algorithm is that in. In fact, everything is just representation at the end there is a trivial solution yeah or at least a known solution ok.

So, this was the implication on individual edge weights as a result of retiming.

(Refer Slide Time: 39:40)



We had defined those pathways to be the sum of all the edge weights as you go from one node to another node, right remember the path weight definition it was how many registers are you crossing as you go from one to the other.

Right. So, the path weight then as a result of retiming how would the path weight change?

Student: (Refer Time: 40:03).

That is my path, and let us assume that I do retiming. The retiming values for a b c d e are these for a it is minus 1, b it is retiming number is associated with a node.

Right, saying how many registers are moving past it. So, if these are the numbers then what are the new edge weights first. This one does not change well this is an output. So, we do not know what is happening beyond it, but as a result of e this one does not change this anyway there is nothing that is moved past e.

This edge weight is influenced by these two retiming numbers, right. So, from the outside nothing is coming, but because d is 1 you are taking one register away towards the input side. So, the new value is 2 right. So, I can do this whole thing for all the nodes. So, I get some set of retimed register values.

The question is what is the new value for the path weight? The original path weight from a to e was 1 plus 2 plus 2 plus 3, right it was the sum of these four. As a result of retiming what is happening to the path weight?

Student: (Refer Time: 41:18).

Can I characterize that in a succinct way given that I know all the retiming numbers what is the change in the path weight?

Student: it should increasing (Refer Time: 41:39). Sir, first and last we will make the difference in.

Only first and last we will make the difference. Is that clear?

Student: Yeah, rest one cancel out.

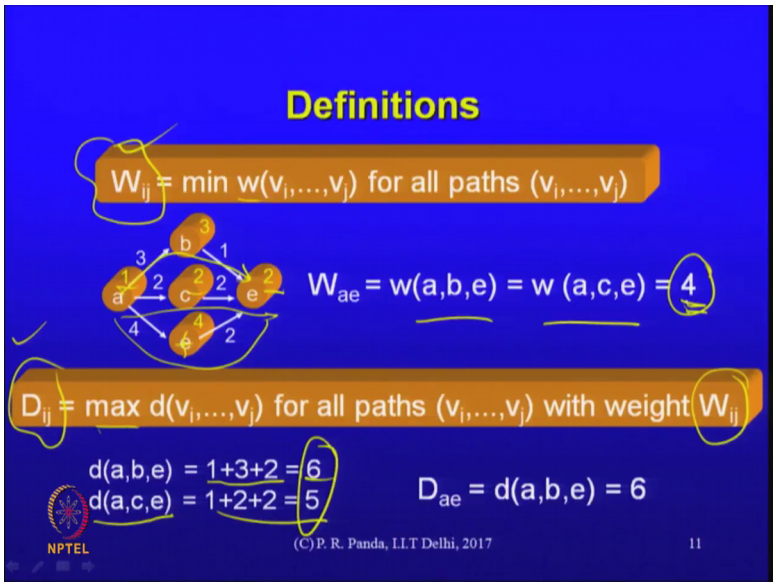
Yeah, it does not matter it is like all the intermediate registers you are just moving from someplace in the path to some other place in the path it still counts as one,

right. Number of registers may be changing, but the number of registers you need to cross are not changing if you just shift them along the path, right. So, the only thing that should influence is the extreme retiming values of that path; that number this number and this number should influence the path weights the other number should not influence the path weight.

So, that is what we have captured here, the new path weight as a result of the retiming vector r consisting of all the r i's would be the old path weight plus r j, where r j is this and minus r i where i is the first one. Any questions on this?

There no questions.

(Refer Slide Time: 42:49)



Continue with our definitions. So, so far we have defined small w meaning path weight from node i to node j.

So, here is what is defined; capital W ij is defined for a pair of nodes v i to v j, problem is there may be many paths.

Student: Yes.

Between two nodes a pair of nodes you may have multiple paths that is what is shown here. Each of them has a path weight right a, b, e has a path weight which is 3 plus 1, a, c, e has a path weight, a sorry this would be some f for something. So, there are three paths.

The capital W ij is the minimum of those path weights ok. So, what I have is 4 here because these two paths a, b, e and a, c, e paths have a path weight of 4, the other path has a weight of 6.

Student: 6.

So, we do not include that that is the minimum number the capital W ij. So, minimum of all the paths that are there in the graph.

Student: it is like the minimum term required for (Refer Time: 44:11) from a to e.

Yeah, minimum number of registers you need to cross as you go from.

Student: (Refer Time: 44:17).

a to e, right.

Student: Translate to time (Refer Time: 44:18).

Translates to time, but ignoring the delays, in ignoring the combinational delays, this is we are only counting the.

Student: Clock cycles.

Clock cycles yeah ok. So, that is my capital W ij. Correspondingly, let me define a capital D remember the small d was the path delay from one node to the other that was the path delay; capital D is defined like this. For all the paths that have weight W ij that we just computed capital W ij this is the max of all of those path delays. So, there are two paths under consideration in this example, right. One was the a, b, e and a, c, a f, e that we are taking out of consideration for now, but these two paths both have the same capital W ij.

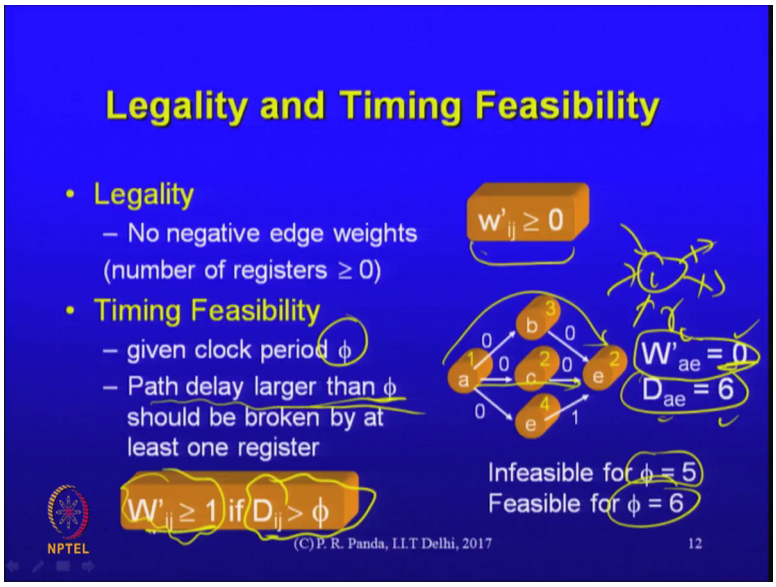
So, in that let us count the path delays this is the combinational delays right. So, for the a, b, e path right for that path the sum of the delays is 1 plus 3 plus 2 this node weights are being added.

To give me the path delay this we already defined actually. So, similarly for that a, c, e path the path delay is 1 plus 2 plus 2 that is my 5, ok. So, if I enumerate these for all those paths for which W ij is what we selected then the capital D ij is determined as the max value of all of those paths the at least the definition is clear why it is done in this way is not here yet, but this is what is the definition.

So, these two things are we identified one is the capital W ij the other is the capital D ij. D is defined in terms of W ij you have to first get the W ij's before you can get D ij because here it is the max of those paths that are selected that have a 4 for example, in in this case that problem is that there may be multiple paths with the same selected path to weight W. So, for those paths you take the max of all the path delays.

Remember, to separate out weight from delay in this discussion it is important they refer to different things; weight refers to the register count and delay here refers to the combinational delays, yeah.

(Refer Slide Time: 46:57)



Once you have those definitions as you do retiming you go from an original set of edge weights to a new set of edge weights. Also from an original set of path weights.

Student: Weights.

To a new set of paths. So, the capital W ij's a new set you get and correspondingly to that another set of D ij as you may you may be getting because the D ij's actually, the node delays are not changing through this process retiming is only changing the edge weight. But the D ij is defined in terms of W ij's therefore, it might actually change. Even though the individual node delays are not changing the path delay is defined in that way will change, because which paths you select are changing, ok.

So, once you do the retiming there are a couple of things you need to take care of which is legality. It should be a legal retiming for a given node is there a restriction on what the retiming number can be just one node.

Student: (Refer Time: 48:08).

a node has.

Student: Some number (Refer Time: 48:10).

Some.

Student: Greater than the current weight as (Refer Time: 48:14) if number is (Refer Time: 44:17).

But, my question was simpler than that. My question was is there any constraint on the retiming value for one node? One node may have some inputs and some outputs, right. This is node I and what constraint does are I have, but can it be anything.

Student: Actually the edge weights depends upon the number of registers in between the two.

Yeah.

Student: (Refer Time: 48:39) points.

As weight is capturing the number of registers.

Student: If there is no register then the variable will be 0. So, (Refer Time: 48:45).

I did not go to I just say.

Student: (Refer Time: 48:47).

I am just saying, given a node what what are the possible values of r i?

Student: Sir, r i can be maximum the weight of maximum weight of the output nodes that are going from that.

Ok.

Student: (Refer Time: 49:02).

So, first of all you understand r i is an integer.

Student: (Refer Time: 49:04).

We are looking for integral solutions to this. It can be 0, it can be positive it can be negative, that is pretty much what our our limitation is. You may have solutions for to the

system of equations that we will build with large or small values there could be interpretations that you come up with, but essentially you are looking for integer values positive or negative or 0.

What constraint would there be on the set of arrival, all sets of integers are not necessarily legal solutions.

Student: All (Refer Time: 49:40).

They are not legal.

Student: some of weights of the some of out edges weight of out edges (Refer Time: 49:45) not the some, the minimum weight of (Refer Time: 49:48).

Yeah, remember.

Student: Minimum out edges.

r i value of 1 means you take a register from every one of the outgoing edges and you move it to every one of the incoming edges that is a value of one. So, anyway the all combinations of r i's are not acceptable.

Student: (Refer Time: 50:05).

For the simple reason that at the end you need a valid graph.

Student: (Refer Time: 50:10).

That graph cannot have negative numbers on the edge weights. The r i's can have negative numbers and the meaning is clear what a negative number means, but the weight since edge weights since they represent the number of registers at the end of all this.

Student: (Refer Time: 50:25).

Ah They have to be 0 or more. So, that is a legality requirement whatever you do on the r i's, the new edge weights have to be 0 or moreover cannot be negative, ok. That is the easier requirement, but there is the other interesting requirement that it should be timing

feasible. The way you move things around given a clock cycle, you should not be able to find a path in the graph with all 0 edge weights, 0 edge weights means what?

Student: (Refer Time: 51:03).

No register, it is a combinational path in the graph. No combinational path should exceed your given clock period, ok. So, that introduces that clock period as a new input here to our strategy. So, if the clock period is given then a path delay that is larger than that clock period must be broken by at least one register.

Student: One register (Refer Time: 51:27).

This is of course, an intuitive requirement, otherwise it is violating the ah

Student: (Refer Time: 51:33).

Sequential behavior of the circuit. You do not have a properly working sequential circuit any path delay larger than phi is essentially captured by that D ij greater than phi um.

Student: Does it mean like ah?

Equation

Student: We have does it mean like after every cock cycle we need registers?

After every.

Student: After every clock.

Clock.

Student: Clock period is like (Refer Time: 52:02).

Outputs of a combination. Where do the outputs of a combinational circuit go? They go to registers. Right?

Student: Somewhere.

So, that that was our original picture. Remember, we consider this whole circuit as combinational blocks separated by.

Student: (Refer Time: 52:20).

Stages of registers. So, that is what we are saying that I should not find any paths that any combinational path delay. Remember, as the result of the retiming I am moving my registers around which will move those paths delays around I can capture this requirement quantitatively using just the simple condition which is if there is a D ij greater than phi D ij's are those remember the way we defined those.

Student: (Refer Time: 52:45).

Ah D ij is the max

Student: (Refer Time: 52:48).

Of all of those path delays for which the W ij is minimum. If the W ij was a larger number like 2 or 3 then I do not need to optimize that that is not a candidate that I need to worry about, but if it is small like if it is 0 for example, then I need to worry. I need to make sure that this path delay is enough. What we are saying is the relationship is as a result of the retiming I get a new set of the capital W ij's, right for every pair all of that has to be recomputed.

Now, if it turns out that D ij is greater than phi, right; phi is my clock period then I require that the capital W ij is greater than or equal to 1, you see that it captures this informal requirement that if the path delay is greater than phi, then it should be broken by at least one register that is what we are capturing here.

Student: Cannot we have a case where, we W ij's are very large, but my D ij's are still within 5 or maybe less than 5, I mean maybe very less than 5 cannot we you know such a case?

Yeah, that is fine. This does not preclude that we are saying we are concerned about only those cases where D ij is larger than 5, if it is small then it then it is, ok. it is not even captured in that condition it. So, which means it is an acceptable.

Student: (Refer Time: 54:15) someone has putting.

Yeah it is an acceptable condition and this is may require a violation. So, we only check the voila there is a feasibility check, right. We are saying it is timing feasible if that

condition is met, otherwise also it may be physical, but infeasible conditions are captured by this precisely by this condition that is what we are saying, ok.

So, that example we had, where I have zeros here. So, that path has a weight zero, this path has a weight zero, this path has a weight 1. So, this is zero because that is the minimum of the path weights for those we enumerate the path delays is 1 plus 3 plus 2 is 1 and 5 is 1.

Student: 1.

So, D ae is 6. So, 1 plus 3 plus 2 is 6, right. So, 6 that is my number. So, what we are saying is that is 0, that is 6. So, if I had phi equal to 5, then this is an infeasible retiming if this was the result of some retiming then that is an infeasible retiming, but if it is some number greater than 6 then it is. Of course, if these numbers were greater than 0 some number here was greater than 0 resulting in the w prime itself being greater than 0, then it is all right then essentially it is this part of it. You need it to be.

Greater than 1 (Refer Time: 55:41).

Student: Sir, so, after all retiming after applying all retiming the retiming vector

Right.

Student: All the path should pass.

Yes, this is.

Student: (Refer Time: 55:47).

ij for every ij.

Student: Sir, when my design is not timing clean then it is possible that out of n pass that are that are failing between two nodes.

Right.

Student: I may at least try to fix and tell designer that n out of n n paths are fixable n minus m are not fixable, you need to go and change your RTL to do that.

Ok.

Student: But, if I drop that in this process if I am dropping the transformation, the vector, then I lose that information. I may turn out that it is not at all doable, but which are.

Right.

Student: Not doable it is not (Refer Time: 56:31).

I mean yeah I mean the reason there is a an iteration in the design process; you do a timing analysis and indicate that these are the situations, then you go back you ask the RTL design I have to fix this code.

Student: Fixed.

Rather than you do it yourself here you are assuming that that is my circuit and I want to optimize it for timing. This obviously, does not incorporate any feedback to be we given right away, it is it is an optimization problem.

Student: All i.

The.

Student: All I, all I was saying is let us say there were 10 vectors possible for retiming.

Yeah.

Student: And of all those 10 vectors when the algorithm tried it came out that at least one path was always violating. So, it will not apply any change and it will give the original circuit back.

Yes.

Student: So, the feedback is not quantified which was the most problematic situation.

Feedback can be quantified we.

Student: (Refer Time: 57:22).

Are not doing it.

Student: (Refer Time: 57:24).

But.

Student: (Refer Time: 57:25).

That is not hard to give for all of those stages you. In fact, those D ij values there could be the feedback.

Student: (Refer Time: 57:32) was little more than that. So, I can get the worst one or 2 or 10 or whatever.

Right.

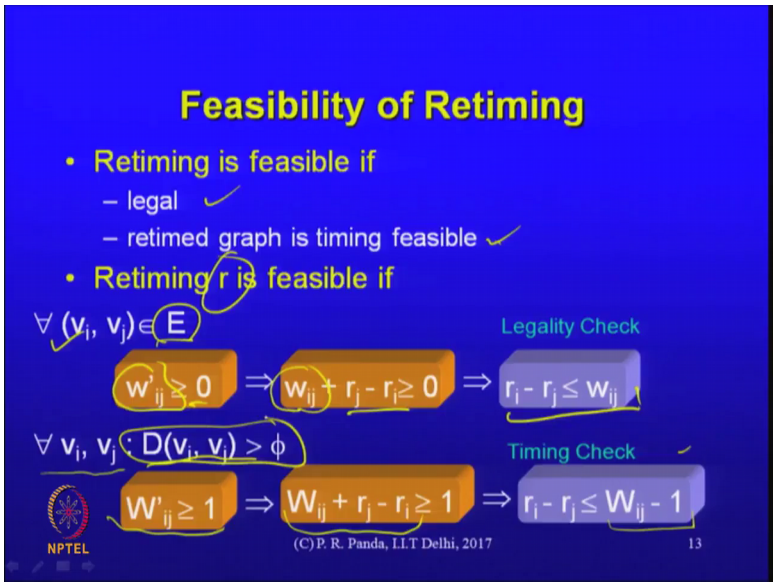
Student: But, if tool can tell that out of 10, 8 are fixable, 2 or not fixable maybe I need to just give a feedback about the 2, but rest all are still doable.

I think.

Student: More (Refer Time: 57:50).

This feedback is a separate analysis really it is not directly incorporated in every timing optimization. It is incorporated in an overall timing analysis loop in which you would do that analysis and give try to give some intelligent feedback not just that it violates, but which are the paths that violate and how to correlate it to what you wrote in the code, right. This is a separate analysis really here it is more restricted with respect to what assumptions it is making ok. So, that is the feasibility checker.

(Refer Slide Time: 58:21)



So, the feasibility of retiming we are talking about two things right one was the legality check that is very easy to see. It just needs to be the new numbers all need to be positive, there is a feasibility check. Every one of those paths the condition has to be respected and feasibility of an entire retiming vector is something that translates to just this a given retiming is feasible. If it is legal that condition we know and that retimed graph is feasible from a timing point of view we already defined those conditions. If those two conditions are met then that retiming vector is feasible.

That make sense? It requires a proof that I have given, but we will now discuss why it is so, but it turns out that if these two simple condi relatively simple conditions are met then you have a feasible retime.

Student: Feasible (Refer Time: 59:2).

Meaning of the graph.

Student: As of now just means that we can apply this transformation on our graph and we still get a valid graph.

Yeah, right.

Student: But, it does not guarantees that we will get a quicker run or a slower run or anything like that, it does.

No, no. We have not solved the problem yet. Now, I have to select the vector in a way that optimizes the timing.

Student: Up till now.

Until now this is only a feasibility check. As long as those two conditions are met you at least have a valid graph that that is all we have.

Student: Now, we will select those vectors which are feasible.

Right.

Student: And then we will see which one.

Right.

Student: Is the faster.

Yeah.

Student: Faster (Refer Time: 60:04).

Yeah.

Student: (Refer Time: 60:05) this course.

Yeah. So, after all this so, we are just saying that the retiming vector r is feasible if that is the case. First one is for all i, j belongs to e means for every edge that is there this edge weight for every edge now, that new edge weight should be greater than equal to 0 means that new edge weight was what anyway that W ij prime was the old edge weight plus r j minus r i right that is greater than or equal to 0 or if you rewrite it it this is my legality check. The r's I do not have yet, but whatever r's I choose legality checks means this has to be respected.

Second is timing feasibility. This is you do this for all edges, right. This you do for all paths for all i, j if it turns out that the d vi vj if that is greater than phi for the phi that is input to us then I require that the capital W ij prime that should be greater than equal to 1 what was this anyway that path weight was again this right the changes depended on the changes from the extreme are does not matter what is there within from r i to r j this is

not just one edge, but any number of edges the path new path weight is only a function of the extreme values. That needs to be greater than or equal to 1, if I just rewrite it r i minus r j is less than or equal to this that is my timing check.

As long as these are valid for my selected r i and r j then I say that this is, ok.

Student: (Refer Time: 61:42) from a feasibility.

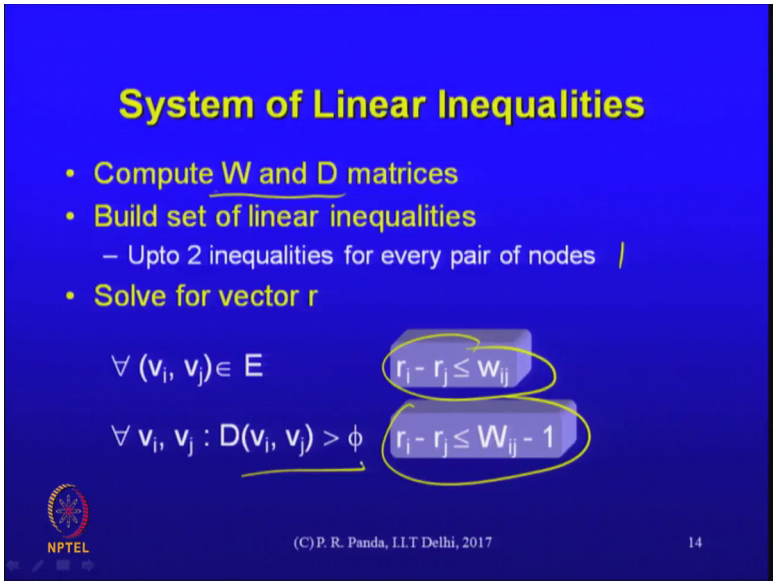
Student: (Refer Time: 61:43) just one condition for feasibility check that is even as I just should be less than 10 ij minus 1 because we need to we need to ensure both, but the conditions are.

Yeah, we need to ensure both the matter.

Student: (Refer Time: 61:53) that second one is that first one will be went for this.

Right, but the second one is not always there. Second one is there if this condition is true. Remember, phi is small then it is not always there. So, let us look at it with an example.

(Refer Slide Time: 62:10)



So, the second one is not always there. If it is there then that takes over.

Student: (Refer Time: 62:14).

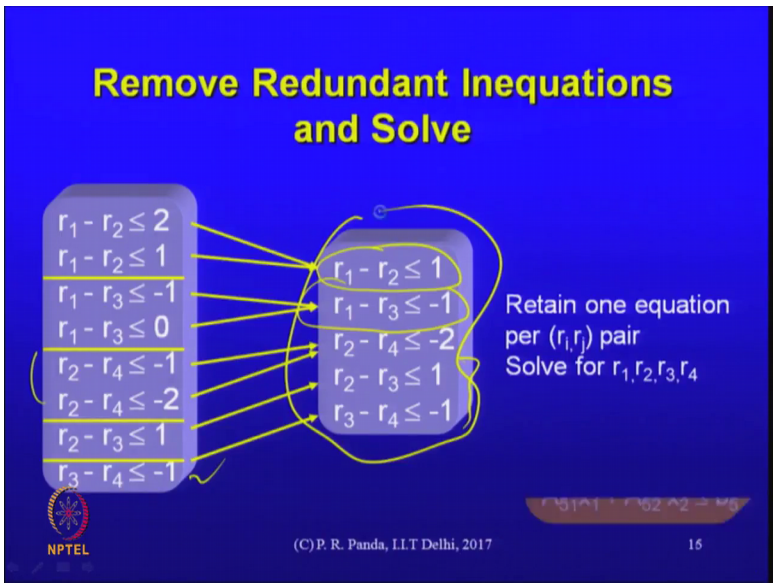
Right.

Student: (Refer Time: 62:15).

So, what you have is a system of linear inequalities where we as a result of a retiming let us say we I compute the W and the D matrices.

Now, you build a set of linear inequalities where you have up to two inequalities for every pair of nodes i, j that we identified. This is one condition, this is the other condition, but this shows up only when that is true.

(Refer Slide Time: 62:42)



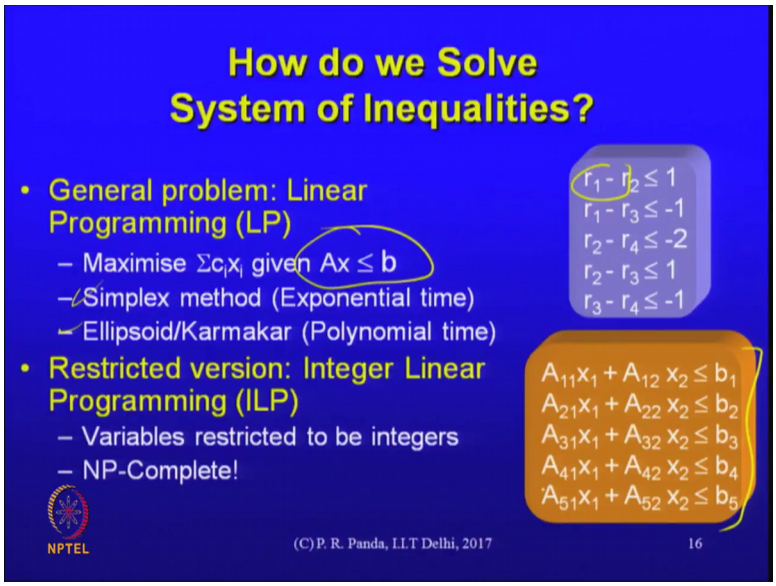
So as a first preprocessing step here is what you would do. For some pairs you may have only one inequality, for other pairs you may have two inequalities, but where you have two you can drop one of them.

Student: Right.

Right. So, for these two would translate to only one. These two would translate to only one the more restrictive condition, right and these ones will.

Just sure. So, this is what I have as my system of inequalities, then I just need to solve for r 1, r 2, r 3, r 4 all the r r i set I have, right. That is all I need to do how do I solve it.

(Refer Slide Time: 63:19)



Student: (Refer Time: 63:24) (Refer Time: 63:24) sir, it is a ILP (Refer Time: 63:29) problem.

How do I have you solved these kind of systems?

Student: ILP. No, I am not studied like that (Refer Time: 63:34).

First of all.

Student: (Refer Time: 63:36).

Say this is of course, a class of a linear system of inequalities actually we do have this linear programming class of problems, where that problem formulation looks similar, not quite the same, but it is similar where you have some such thing these are matrices is a is a matrix. x is a vector that is what we want to determine, b is a vector that is given to us there are several known techniques this is a solved problem of course, there is a popular technique called simplex which is exponential time in principle, but often works very well in practice.

There are other algorithms that are more polynomial time, but this is the general linear programming problem. Our example is not an instance of a general linear programming problem, why is it not?

Student: Integer.

These are integers r i and r j are restricted to be integers.

Student: (Refer Time: 64:30).

Unfortunately integer programming version of that same thing integer linear programming where you restrict the variables to be integers is a harder problem.

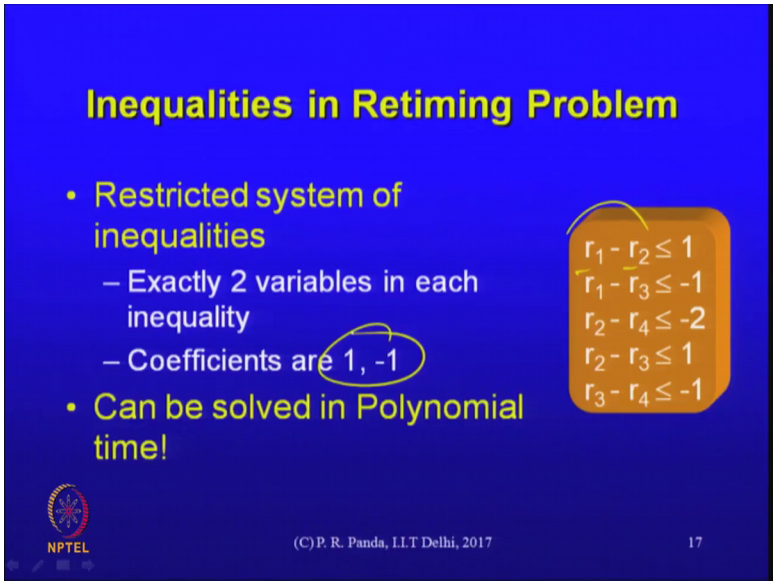
Student: Yes.

If they are real numbers then it is an easier problem this is the space um. So, the general ILP problem is difficult nothing new by now you know what to expect in this course.

Student: (Refer Time: 64:54) NPR (Refer Time: 64:55).

Yet this is an example of a specific instance of a problem that is not difficult, general ILP problem is difficult.

(Refer Slide Time: 65:02)



This is not a general ILP problem not only are these r i's restricted to be integers, but there is another condition you look at these set of equations there is something else this is a special case of an integer linear programming problem where these coefficients of the variables of the unknowns are all 1 and minus 1.

Student: (Refer Time: 65:26).

They are all of the kind r i minus r j less than or equal to some constant, right. That is my formalism, there is no getting around this the way we derived this these are the only the kind of equations inequalities that we will get. This though can be solved in polynomial time.

So, that saves us even though the general ILP problem all are like our scheduling allocation all of those things could be formulated as ILP problems we did not go through that, but you can easily as a homework and therefore, consequently has an exam question we could say that the problem you know you understand the scheduling problem, you could write an ILP formalism for many of the synthesis problems that we have already seen, does not make it any easier it only helps our understanding of what are the variables that are involved what are the constraints, but the ILP problem is hard, but this specific instance of the ILP problem where I have coefficients are all 1 and minus 1 which is to say that I have all of them of this form r i minus r j is the less than or equal to something that can be solved is an easy solution to this.

Think I do not have enough time to go through this, but we will resume it the next time.