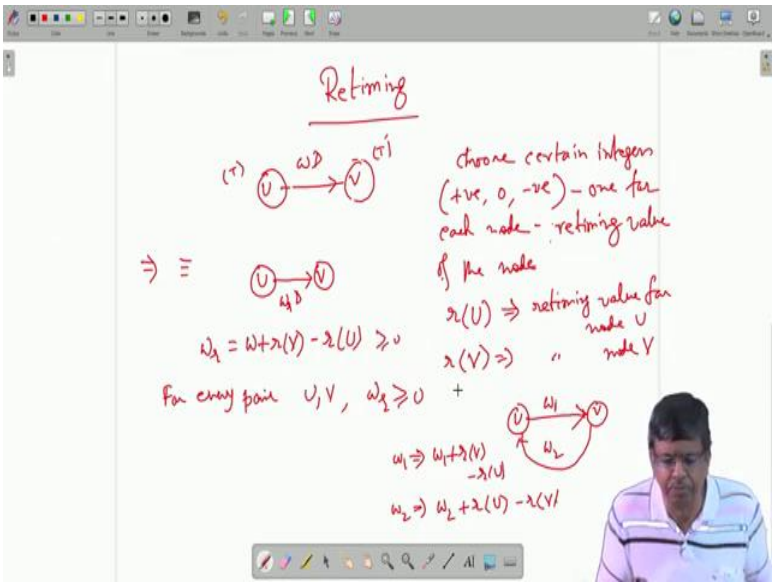
Course on VLSI Signal Processing
Professor Mrityunjoy Chakraborty
Department of Electronics and Electrical Communication Engineering
Indian Institute of Technology Kharagpur
Lecture 05
Retiming Theorem

(Refer Slide Time: 0:30)



So the previous class we started with retiming and quickly what we did is these we said retiming is a method by which you can change the delays or the various edges in a giving data flow graph, maintaining equivalents. That is the new DFG you will be doing the same job at every node as the original DFG was doing. But this gives a formula to change and that is retiming and there I said that every sector of DFG if you pick up it will be of this type.

There will be node U from which a edge may come up and go to another node and DFG is nothing but repetition of these, from this node and edge goes to another node from that node and edge goes to another node so on and so forth (())(01:09) delay. So, I just take one such sector and whatever you say that will be applicable to all other sectors like node edge node, that kind of sectors, same thing.

We said that for retiming you choose some integer values could be positive, could be 0, could be negative for each node, one for U, one for V, one for another node and so on and so. Each integer value is called the retiming value of that particular node. I denoted by this, like for node U, r of U, r stands for retiming, r of U, it is an integer plus minus or 0, which you chose and assigned to U, it is called the retiming value of node U.

Similarly r of V, retiming value of V could be positive, negative or 0, U assigned to V it is the retiming value of V so on and so forth. There is no restriction as such on your choice, but there is some restriction that will come now, the remaining theorem says that if the edge at w number of delays, w can be 0, 1, 2, 3 dot dot dot.

Then after retiming you can change it to new value of delay wr, r for retiming where wr is original w plus the retiming value chose for the destination node destination in terms of the arrow that is r of capital V minus retiming value of the source node. Source means in terms of this edge, this is coming from this, so r of U. And this is a new value and this you do for every edge, so new DFG will be equivalent to this. That is it will generate the same sequence at every node, which it was doing earlier, but it will be delayed by the retiming value number of cycles for that node.

And there is one point of question that seems there is a minus sign you must make sure the whole thing does not turn out to be negative because negative number of delay has no meaning. This is very important, because suppose of the have a loop, there I said that in a forward path suppose you had a w 1 amount of delay for difference for w 2 amount of delay. So w 1 will become w 1 plus or r of V minus r of U after retiming this edge.

And retiming this edge it will become w 2 plus r of U because this is the destination now minus r of V. So, if you want to make r of V positive very highly high valued positive number r of U 0 or negative number, so that this is positive and does not turn of it negative. Here it will have a reverse effect because here r of U has positive sign, r of V has negative sign. So it will turn out to be negative. So, you have to balance between the two.

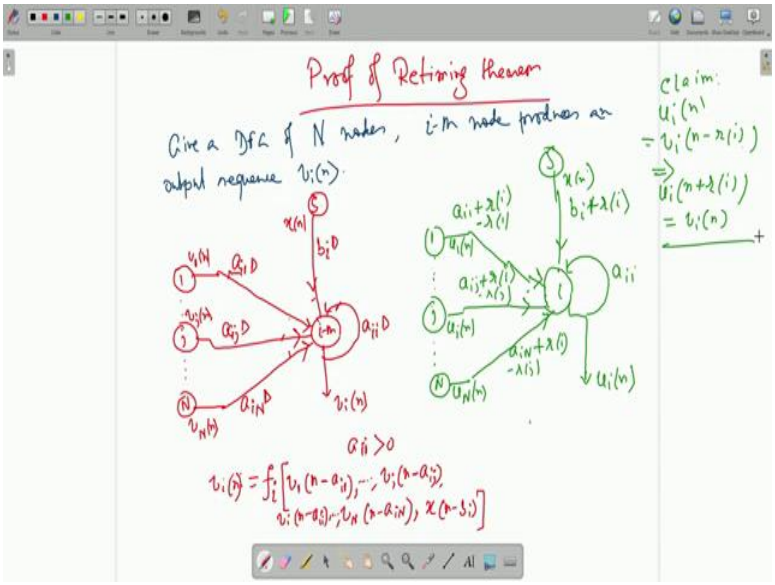
So, basically for every edge you get one such inequality, the state of inequalities is called the feasibility inequality. Whatever waits choose satisfy those inequalities, that will be acceptable. You can choose those waits that is those retiming values anytime.

If you are give me the set of inequality suppose hundred such an inequalities because there are hundred edge, you cannot solve them and solution space may exist, may not exist. If does not exist, then no retiming possible. If it exists, you can give it to a computer package it will generate to your solution space you can pick up any choice of retiming values for the nodes from that space and that will be fine.

Now where from does it come? That is the question that comes from retiming theory. So I will give a general statement of the retiming theory which I was doing last time. And then the

proof can be omitted but I will give a proof because proof does not require any hard mathematics, just mathematical logic. You can skip it or you can follow it.

(Refer Slide Time: 4:43)



Next thing that I did last time, I said that suppose you consider a DFG of capital N number of nodes. So node 1, node 2, up to node capital N but the i th node, it is the vi at that produces a sequence outputs sequence be vi n. So, if you focus on that particular node and i could be 1, 2, 3 anyone. So i general i th node, so its structure could be like.

This is a i th node, it is produces an output Vi N. At other nodes are shown separately here it may depend on other notes like node 1 producing V1 N it may go there if there is a connection. It may not go if there is no connection, I make a provision of a connection with a delay ai 1, i from here, 1 from here ai1 number of delay.

If ai1 is 0, 0 delay if ai N is 1 1 delay, if ai 1 is infinity, that means infinite delay, which means no connection because if it is infinite delay, the signal can never reach here. So that takes care of no connection be this also, then j th node, Vj N is the output. It has to be delayed by some number which is dependent on i and j, so aij D N th node aiN D.

So I am making a very general case where all other nodes connect with this with the respect delays ai 1, aij, ai N. Some of them could be infinity also which means no connection. This can depend on its own its own its own output, so output we have a feedback that must have a delay aii D and aii must be positive because if it is 0 delay, not positive but 0 that means current output to calculate unit the same current output then the input which is not possible,

but it brings at least 1 delay or 2 or 3 delays that we from past output. I am working on the current output which is fine.

It may depend on an external signal source S which produces our signal xn. I am taking one signal source there can be multiple signal sources also that can be easily generalized. It is generally xn. There may be a path here with some delay or may not be a path here so I give a permission of a path or edge with bi number of delay. I do not need 2 subscripts here because this is a universal source, so it does not need a subscript. That is why bi there i comes from where it is going bi, bi number of delay.

If bi is infinity, no connection that is single source does not come to, does not connect to this i th node directly or bi can be 0, 1, 2 as before. Then what does this node do? It takes all the incoming information and does something and produce an output. What it does? There is a computation load whatever you have done inside that you know very well. So, I denoted by your function f dedicated to i th node fi.

It is a function of all the incoming components. So, here the incoming components is V1 N minus so much of delay, n minus ai 1 dot dot dot vj N minus aij dot dot dot V capital N small n minus ai n dot dot dot you can take like of these, Vi n minus aii and input x, n minus bi these what it is producing and that is done not only at i th node, it could be node 1, so it is V 1 N it could be node 2 there V2 N dot dot dot. This describes the DFG, general structure with the DFG.

Now, I change I retain the same structure, but I changed the delay values by a retiming formula as I already discussed that is I denote, I dedicate some retiming values to all the nodes. What I do for the source node? That I do separately, I will come later forget about source nodes for the time being all the internal loads, you know I give 1 retiming value r of 1 here assign. I assign 1 retiming value r of j here, r of i here, r of capital N, so on and so forth. And using that formula I changed the delay values by that formula.

So, I get another structured. So, that will not generate Vi N as such because the structure has changed where delay values have change. So, outputs maybe Ui n, U1 n, Uj n, UN n all that. And then I will so I equivalents between Ui n and Vi n. So after the retiming, the new structure will be like this.

Look at the arrow, arrow is heading here. So, that is why this is a destination. So, r of i minus the source r of 1, this is producing U1 n. No longer V1 n, U1 n, because new structure has come up so I cannot assume the same input. This is node i dot dot dot, j th. I am dropping that later capital D from now onward. D just indicates delay. So this is a integer that may times delay and I make sure that this is not negative. None of the fellows even though there is a negative sign, it is not negative that I have taken care suppose by my choice of the retiming values.

So it will be generating Uj n, delay is original ai j plus r of i th minus r of j. Original ai N plus ri minus rj, destination retiming value when a source retiming value. This output which is not Vi n it is Ui n. The feedback path, feedback path I had original aii then it should be aii plus the retiming value of the destination node. In this case, destination is i th, so aii plus r of i minus retiming value of the source which is again is i th node. So for minus ri, so aii plus r of i minus r of i. So it remains aii only remains unchanged. This is source it is generating the same xn.

Here I say something new ordinarily I had bi or it comes to source node. We assume this source node has retiming value 0 that is the bi should b, this bi plus r of i, as though it is bi plus r of i minus 0. As though the source node signal source node has retiming value 0, bi should be changed to be bi plus ri, this is for the i th case. So, i can be 1 here to here or here, i can be 2 here or here like that.

So, this is after retiming, this is before retiming and my claim is their equivalent in the sense that Ui n and Vi n will be identical just separated in timescale by a known delay and known delays is nothing but the retiming value of this node which you chose and therefore you know and therefore known that is we will have claim. Claim is Ui and this is the delayed version of this. This will be Vi n minus the retiming value here, Ui n is nothing but original Vi n minus some delay and delay is nothing but the retiming value of this node ai 1.

That is Vi n minus ai 1, ai 1 is a sorry I am sorry this is not the thing, minus r of i. That is Ui n at anything this is nothing but the same Vi n but delayed by ri or equivalently Ui n if you take ri on this side n plus ri this will be your Vi n. That is Ui n is an advanced version of this. As the Ui n is a delayed version of this or Vi n is an advanced version of Ui n.

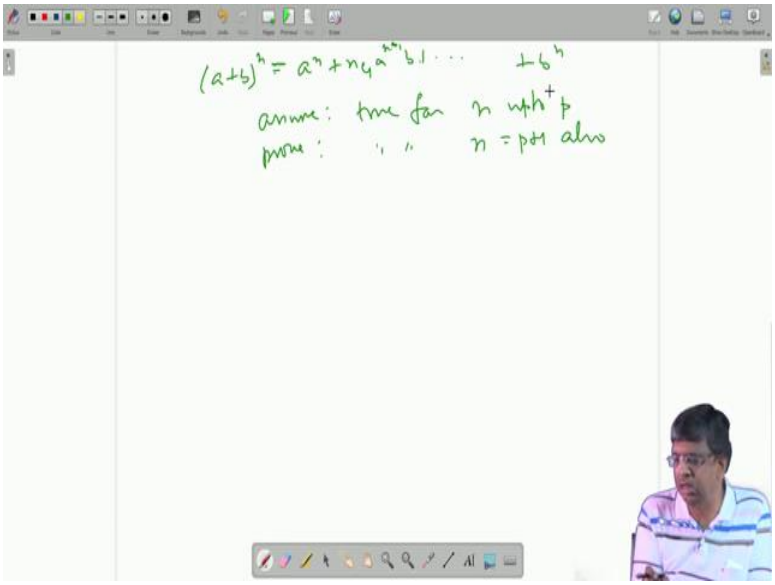
That is if Ui n if I go further up Vi n is the same thing. It is Vi n minus something is Ui n, so Vi n is n plus the same thing. I take minus are on that side, this is a claim. So that equivalent

1 is the delayed version of the other but delay is known to us because that is retiming value of your choice and that is the reason they are equivalent.

And this is true of all the nodes so they are equivalent. We had to just prove it, proving is required because I do not want to just, you know, give some theorem and then you know without leave there a without proof.

Because I mean the person who is attending it, he must know where it comes from. If you do not like you can skip the proof also, but it will take some time, maybe forty minutes or so. So I will just carry out the proof. But the proof does not require any advanced mathematics it just requires basic mathematical logic and mathematical induction okay, induction.

(Refer Slide Time: 16:26)



Proof by induction, what are we doing induction? We have always done induction in our school. Suppose you are approving something like say binomial theorem, say a plus b, whole to the power n. We have all that proved done this probe nc1 a to the power a minus 1 b and dot dot dot. Finally b to the power n, it was as a formula, binomial theorem.

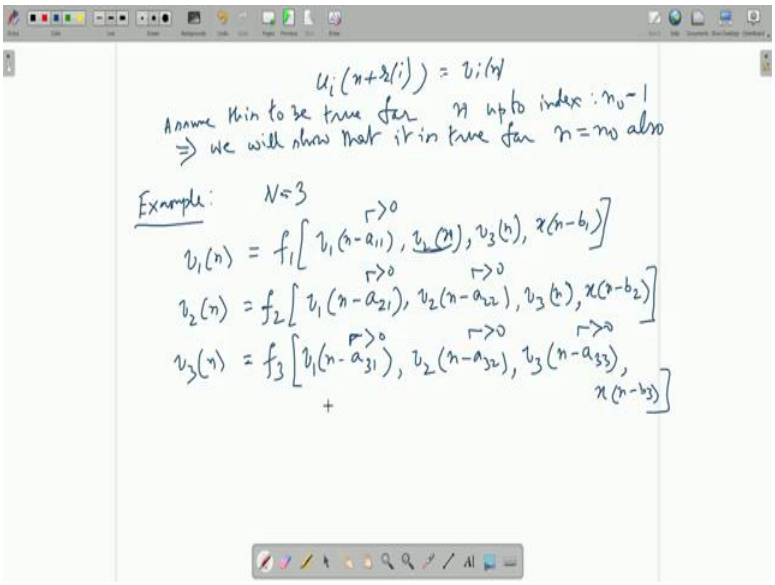
We have proved it, if we say proofing by induction, what do we do is this? We assume that this is true. We assume that this is true, true for n up to some integer p then we proof, prove for n equal to p plus one also. So, if it is true for any n equal to p, then it will show it is true for p plus 1 if it is therefore, by the same logic if it is true for p plus 1 here, it is true for p plus 2. If it is true for p plus two it is true for two plus 3 there is the same logic.

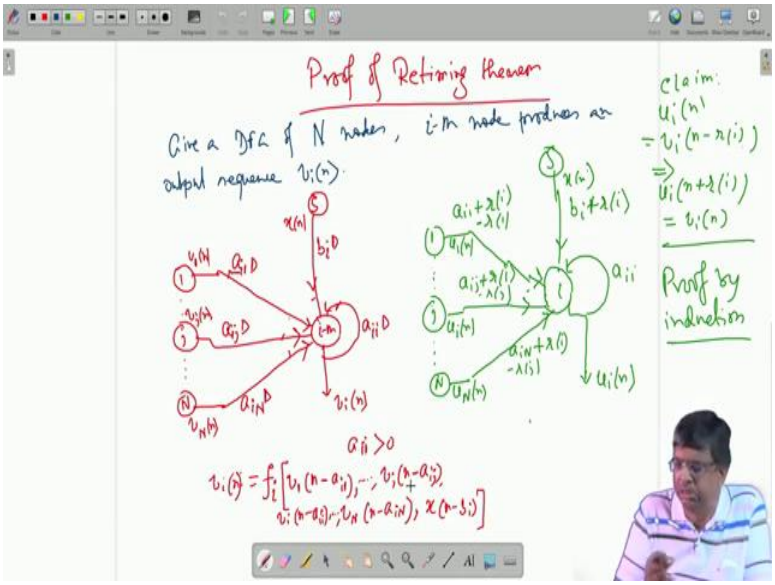
But there is a assumption which is called induction hypothesis. Then I am assuming it to be true for n up to some p. So I should I at least demonstrate one case where it is at least one case where it is true that I have to do directly.

So maybe you can take p equal to 1 or 0 or p equal to 2 whatever and so that left hand side equal to right hand side directly. If that is true, then for p equal to 1 here it will be by this logic it will be valid for p equal to 2 again for p equal to 2 if it is true it will valid for p equal to 3 by this logic and it will go on. This is called induct proof by induction.

There is assume this to be true for n up to some index and then proof, it is true for p plus 1 also. And then validate that assumption by working out as direct case that for p equal to certain value this is the left hand side equal to right hand side.

(Refer Slide Time: 18:17)





Same thing we will assume here, what you have to do? As I told you, we have to prove that for every i th node, n plus ri is equal to bi n for every i th node. So, we as you by prove it by induction, so assume this to be true or n less that equal to some index n naught or less than maybe less than 1 minute up to index some n naught minus 1. This is true for n up to n naught minus 1 that is true for n up to n naught minus 1. That is if you put n equal to n naught minus 1 this is true here also it is true. If you put n equal n naught minus 2 here, here also it is true a like that.

Then we will show that it is true for n equal to n naught also, n equal to n naught also. This is by induction. And then one case I will show directly to validate this assumption that at least for 1 n naught, this is true. That are n naught for n naught minus 1 left hand side is right hand side.

If that is valid, this will continue by this recursion. Now, before I do that there is one property I want to show that is if you go to the previous page, look at any of this any of this two. So here suppose these delay is 0 or some of the delays are 0, then it might appear that vi n here will depend on simply (())(20:37) n only. Suppose ai 1 is 0, so it will depend on v 1 n only. That is it will depend on the current cycle n th cycle output of 1 or if it is 0, it will depend on the current cycle output of vj, I mean j that is vj n.

It will appear because this fellow will be 0, this fellow will be 0 and so on and so forth. Here ai 1 will be 0, ai u will be 0, so on and so forth. But we will show that it will actually depend not on the current cycle of v 1 n or vj I mean node 1 that is v 1 a or node j that is vj n and so and so. So, it will actually depend on their past values.

This vi n will depend on that node outputs of previous cycles only not on current cycle even if some delay is apparently 0. It will not depend on the current cycle into enter it will eventually depend on the past values of node 1, node j, node n, not in the current value that is what I will show.

That it will effectively depend on v 1 n minus some value vj n minus some value like that. If any ai 1 is 0, aij is 0 that is apparently they are 0 delay. But it will effectively depend on their past values only. That is very important for these. So, that is what I will show? Take example, to as an example of that case and to show we say n equal to suppose 3, n equal to 3.

So I am got 3 nodes same v 1 n, it may be f 1 v 1 n minus ai 1, a 1 1 sorry where i.is 1 here. This is not i, a 1 1. Let me make it cleaner here then v 2 n minus a 1 2. But suppose a 1 2 is 0, if it is a 1 is positive then v 1 n depends on v 2 n minus some value that is the past output of node 2 which is fine which is fine which is what I was trying to say it will. But in case that delay is 0, a 1 2 suppose is 0 that is a delay from node 2 to 1 suppose 0. Apparently it will be like this, as though v 1 n is a function of v 2 n same cycle.

But a 1 1 is a self-loop that I told you aii that is the the feedback of the node i from output to its own input that cannot may not be 0 that has to be positive. That means this fellow has to be strictly positive. They begin say v 3 n. In general it is n minus a delay delay is a 1 3 because destination is 1, sources node 3, so a 1 3, a 1 3 if it is positive, I am done because v 1 n is a function of past value of v 3 n, v 3 n minus 1 or 2 or 3 whatever.

But if it is, if it is a function of the current index n current index n even then I will show the v 1 n actually will not going function of v 3 n. Eventually it will be a function of v 3 n minus some positive number. There is some past output of node 3.

And then these 3 n input x n minus b1 can be 0 or positive it does not matter I am not bothered. If that b now let us consider v 2 n, f 2 v 1 n minus a, now destination is 2 source is 1, so it will be 2 1, a 2 1, it has to positive because here v 2 n here v 1 n dependent on v 2 n same cycle and if v 2 n depends on v 1 n that is if a 2 1 is 0 that will be v 1 n is a function of v 2 n, v 2 n is the function of v 1 n which means v 1 n is a function of v 1 n, which as I told you is not possible absent.

That is current cycle node output depending on the same node output, you cannot do that there is a absent, you see v 1 n if I took a function of v 2 n I did not put any delay here, v 1 n as this case I took as a function of v 2 n. Then if v 2 n when I write as a function of v 1 that

delay must be positive cannot be 0. Because if it is 0, if it a 2 1 is 0, that eventually it means v 1 n is a function of v 2 n, v 2 n is a function of v 1 n which means v 1 n is a function of v 1 n. which is impossible, v 1 n is the function of v 1 and so you can never calculate v 1 n.

Therefore, this is strictly positive. Then v 2 is a self-loop n minus a 2 2 say 2 2 must be bigger than as I told you every aii greater 0 then v 3. Again here v 2 n is a function of v 3 n minus ideally it should be a 2 3, 2 is the destination, 3 is a source a 2 3. In this case a 2 3 can be positive can be 0. If it is positive, I am done because that v 2 n will depend on past output of node 3. But like me not take up that case, let me take that case where it is actually n only, no delay.

And then xn minus some delay b 2 which I do not care whether it is positive or 0. And last one v 3 n, if 3 this is v 1. Again v 3 n, n minus now we will have a 3 1, a 3 1 must be greater than 0. Because here v 1 n was a function of v 3 n, no delay, There if v 3 n also the function of v 1 n, no delay. That but the same logic v 1 n is a function of v 1 n which is not possible.

So, if it was a function of v 3 n only, no delays, so v 3 n must depend on past value of v 1 where is v 1 in minus some delay there is a 3 1 positive. Then v 2 n minus a 3 2 again this must be positive here, simply because v 2 n here as a function of v 3 n, no delay. So v 3 n when is a function of v 2 n there must be a delay, otherwise v 2 n will be a function of v 3 n, v 3 n function of v 2 n which means which v 2 n will be function v2 n itself which is not possible. So it's positive.

Then I have got the self-loop n minus a 3 3 positive and then aii you know self-loop delay is always positive I have tell n minus b 3. Now, I will go back look at v 1 n, I have got first time v 1 n minus a 1 1 which is a positive term. So this strictly delayed term n in minus some positive. So, the whole thing is strictly delayed, v 2 n minus nothing n, the moment I find 0 delay v 2 n then I replace by this f of 2 whatever it is I bring it here.

So, it is a function within a function, within that function I have got a strictly delayed strictly term but no delayed term. The moment I find it no delay term v 3 n, they are again bringing the corresponding function.

So this f 3 comes up there. So, it is a function within function within function, but they are all the terms are strictly delayed then I go to v2 n, v 2 n again I replace the moment it is not delayed, I replaced by this function again. These two are strictly delayed term again v 3 n I replaced by these the third one expression. There is all the terms are strictly delayed, so on

and so forth. So, eventually it will be a nested function, function within a function within a function kind of thing, but all the terms as you can see will be strictly delayed that is, with n whatever will come that will not be 0 that will be positive.

So, this is very important that every v 1 n, v 2 n, v 3 n, the eventually function of the past values of v 1 n, v 2 n, v 3 n. Even if the corresponding delays in the circuit in the DFG has found to be 0, but that means nothing. Effectively each of the node output is dependent only their past values, that is why for these up using these I will proof the retiming theorem with the next up. Thank you.