VLSI Signal Processing Lecture

Professor Mrityunjoy Chakraborty
Department of Electronics and Electrical Communication Engineering
Indian Institute of Technology, Kharagpur

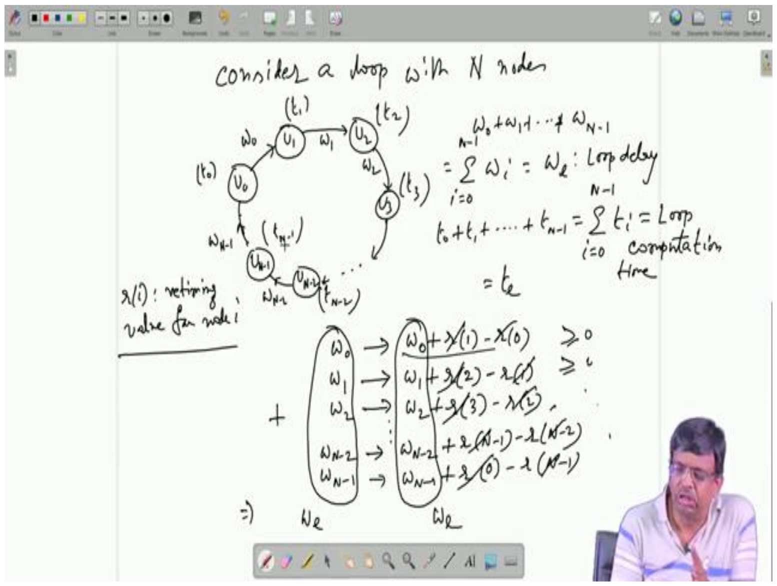
Lecture 07

Loop and Iteration Bound

Okay, in the last class we were discussing retiming a loop, we found some problem. Where it is just a forward path, we can (you know) we can just introduce delays between every two nodes and bring down the critical path then we can get into a node dividing into maybe 2 equal halves again insert a delay between them and so on and so forth. These delays were, they were part of the original DFG but we can introduce them from outside and we can bring down the critical path again still I mean till the point is possible to bring down and node into you know sub-nodes.

But then when we tried to do the same thing to a loop then we had a problem. I mean if we started you know bringing delays from outside in the end we end up with a negative number of billing. So feasibility is gone okay, there was a problem I discussed at length by showing an example. Now why does it happen? That is what we will find out today, why does that happen? What the problems in retiming a loop.

(Refer Slide Time: 01:29)


consider a loop with N nodes
$w_0+w_1+\cdots+w_{N-1} = \sum_{i=0}^{N-1} w_i = w_\ell$: Loop delay
$t_0+t_1+\cdots+t_{N-1} = \sum_{i=0}^{N-1} t_i$ = Loop computation time $= t_\ell$
$r(i)$: retiming value for node i
$w_0 \to w_0 + r(1) - r(0) \geq 0$
$w_1 \to w_1 + r(2) - r(1) \geq 0$
$w_2 \to w_2 + r(3) - r(2)$
$w_{N-2} \to w_{N-2} + r(N-1) - r(N-2)$
$w_{N-1} \to w_{N-1} + r(0) - r(N-1)$
$\Rightarrow w_\ell \quad w_\ell$

So consider a loop, n number of nodes okay like this, U01 node, U11 node, there is a connection there we have W0 amount of delay between them. U0 can take time, T0 micro second, for T0 whatever be the unit, U1 can take T1 unit then there will be another node U2 with delay W1, U3 delay W2 dot-dot-dot. So U, starting with U0 and total n nodes you should go up to n minus 1 this is is U m minus 2. So this has W n minus 1 delay, this has W n minus 2 delay dot-dot-dot. So U n minus 1 after that it comes back to U0.

So together they form I mean and (())(02:53) they form loop. Though it does not mean that there are no other nodes in the DFG, there will be many other nodes around this maybe U0 can be connected to some of them, some other will be connected to U1, U2 will be connected to some other one.

But when you consider them, U0 to U n minus 1 along them they form a loop because you start from U0 and go back, come here then you I mean. You finally arrive at the same U0 again okay so loop. If loop, in the loop I have shown the computation time for every node and the delay between every two pair. If you add up all the delays in the loop that is if you say W0, W1 dot-dot-dot up to W n minus 1 that is sigma Wi, i equal to 0 to n minus 1 that is the total delay available in the loop it is called loop delay, loop delay, total delay in the loop.

Similarly if you add up all the computation times T0 plus T1 plus dot-dot-dot T n minus 1 that is it is called total loop computation time, loop computation time figure calling TL. Okay, now suppose I retain the loop that means I will be assigning retiming values to them, so I have R of 0 for U0, R of 1 for U2 like that. So basically I have 2 things, R of i retiming value for node I, okay so after retiming W0 this delay will turn out to be what after retiming it will be original W0 plus retiming value of the destination node for this edge.

This edge hits at U1, so destination value is R1, retiming value of the destination node minus retiming value of the source node. Source for this edge I mean, This (edge) node that is R 0. Okay, similarly W1 it will be original W1 plus R of 2 minus R of , okay dot-dot-dot may be one more I take, W2 will become W2 plus R of 3 minus R of 2 dot-dot-dot, W n minus 2 it will become W n minus 2 plus R of n minus 1 minus R of n minus 2. And W n minus 1, it will become W n minus 1 original plus R of 0 because U0 is the destination here minus R of n minus 1.

Now if I add them these fellows if I add I get the total delay in the loop that is (sorry) that is WL. Around this side that is before retiming total loop delay is WL. After retiming these are the delay values this is the new delay. They are all greater than 0 by the way. The retiming values are chosen such that they all are feasible that is nobody is negative, fine. So then if I add them after compute that is after retiming what is the new loop delay?

New loop delay will be summation of the new delays, so this much in 1 delay, this much is 1 delay like that okay. So if you add them you see these six cancel. So you will be left with only this part, so that is again WL. This is the issue that when you retain a loop before retiming loop delay is WL, after retiming also loop delay be WL. Total delay in the loop cannot increase. You have to just manipulate so that some delay from some edge is move forward or backward and like that.

But total delay must be same. That is unlike the forward path pipelining, forward path retiming you cannot bring additional delays from outside world and insert them in various edges. This total delay in the loop cannot be changed, they remain fixed I cannot add extra delays from outside. I can move some delay forward or backward by cleverly choosing retiming values or the nodes. Only that much can be done. This is a fundamental problem of retiming a loop. Fine the loop is very essential.

For instance if you have IIR filter, the IR filter you know you have an equation Yn which depends on Y n minus 1, Y n minus 2 that means there is a feedback from Y nth through a delay back to the input side, so that forms a loop. If it is Y n minus 2, through 2 delays Y n Cos back gets added with the input side and comes to the output. So that forms a loop. So loops are very essential, loops come in various you know DSP circuits, DSP and communication circuits.

Okay, in communication there is something called decision feedback equalizer, where the output decision about the what symbol was transmitted that is estimated and then fed back to the processing equalizer for refining some estimate there is some structure okay, so there is a loop again. Loops occurred frequently and always I tell you loops are the main cause of problems okay, now under this then what best can be done theoretically at least.

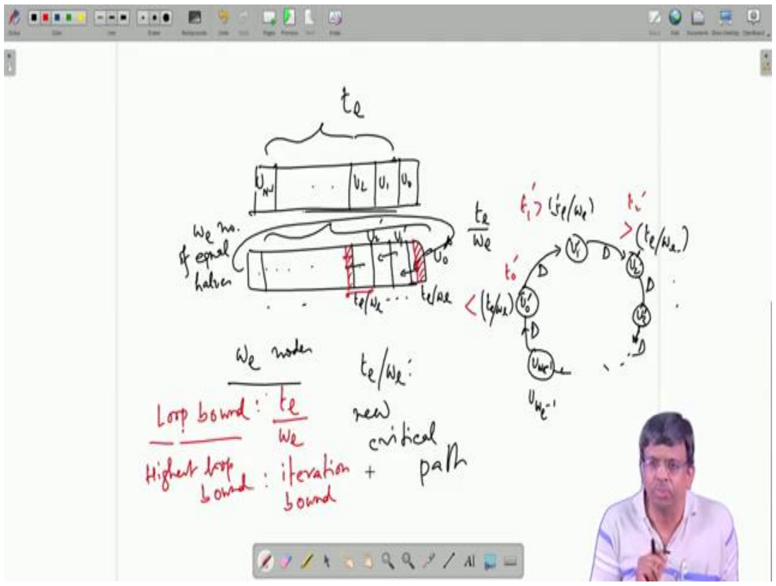
What best we can do? How to what is the minimum value we can bring down the critical path 2? Okay, here what is the critical path? If every delay is positive then it will be either time taken by this node or this node if this node or this node whoever has the maximum. Okay because if they are separated by delays the 2 times will not be like T0 and T1 will not get added. Because in one cycle U0 does a job next cycle there is used to U1 and likewise. Otherwise if you have delay free path in the loop you have to add those node computation times.

So fine, so there is a loop given with some critical path some delay distribution fine. Now how to adjust the delays? Okay, the computations with the nodes so that I can achieve the best possible that is I can achieve the minimum possible critical path. We can directly see that we cannot bring down the critical path below that is easy. What is that? Okay, that is what is my next topic for that just look at this thing. Just if you see the net processing that we are doing.

First we are doing a processing by U0 then followed by U1 there is a delay ofcourse but otherwise processing wise U0 it takes some amount of time T0, next processing is U1 it takes amount of time T1, then U2 taking some amount of time T2, T3 this is T n minus 2 sorry I forgot to mention this here okay, so on and so forth. So what I do I first consider a total processor which consist of first U0 then U1 then U2 that is I am finding out how much is the net job I have to do.

So first I have to start with U0 processing, then U1 processing, then U2 kind of processing then U3. So I built up a net super processor okay which carries on all the jobs. Okay then we examine.

(Refer Slide Time: 11:23)


t_l
U_{n-1}
U_2
U_1
U_0
W_l no. of equal halves
t_l/W_l
$\frac{t_l}{W_l}$
W_l nodes
Loop bound : $\frac{t_l}{W_l}$
Highest loop bound : iteration bound
t_l/W_l : new critical path
D

So start with U0, so suppose this is the job. I am just considering the total job, delays I will definitely require I will require to insert but net job that is maybe U0, did you go to U1 from right to left I am showing U2 dot-dot-dot up to U n minus 1.

These are net job and how much time it takes? T0 plus T1 plus T2 dot-dot-dot up to Tn that is net time takes is nothing but the loop computation time TL okay then I take a re-look at this job, net job, total job and instead of having nodes as before here like here U0, U1, U2 what I do, I divide into equal halves if possible. Hardware may not part then I cannot do this optimization. But suppose hardware part means then I divide them into equal halves how many equal halves? WL number of, WL is the loop delay, number of equal, equal in terms of time taken, equal halves.

Okay, total time is T suppose the hardware permits to divide the job into exactly WL number of sub-jobs each taking the same time, so is taking time how much? If they take the same time total time is TL, WL number of new jobs, new nodes okay, so it will be TL by WL. So everybody takes TL by WL okay dot-dot-dot. Everybody, everybody takes TL by WL,TL by WL, Tl by WL dot-dot-dot like that everybody takes, if it is possible. If it is not possible I cannot come up to this. But suppose it is possible to look at the total job and divide it equally, equally means in terms of time divided into WL number of sub-jobs.

Okay, so if they are equal in time everybody will get a share of TL by WL then what I do, I dedicate WL number of nodes. So this is one node you can call it U0 prime now because U0 location has been used earlier I cannot hear so I cannot use the same notation. Then the next job you call U1 prime. Next one you call U2 prime this is the way. First U0 prime, this is U0 prime then it goes here U1 prime, then it goes here U2 prime I am showing from right to left and so and so forth, okay.

So U2 prime dot-dot-dot total WL okay maybe one more I show U3 prime. WL is total so we started from 0 so WL minus 1 then only total is WL. So I formed a loop the same thing okay, U0 prime, U1 prime last one and last one should come back here because that is a loop where it is coming finally is a loop and then so everybody gets how much time, takes how much time? TL by WL, everybody.

If it is at all possible to break this total job equally like this then only I can talk like this. Again TL by WL dot-dot-dot. And how many delays I have? WL okay so I give 1 delay here D because how many nodes I have? WL number of nodes. So start with 0, U0 prime in front of it you get one delay. U1 prime in front of it will be one delay, U2 prime in front of you give, U3 prime give one delay okay. And finally this U, WL minus 1, this is actually U Wl minus 1, as total, so in front of it will be one delay. How you give the delay? By just retiming.

Because by retiming we know we can move some delay forward or backward. So by doing so you keep maintain everybody gets one delay. Okay then according to me this is the best. It is because every edge has a delay, so the time of (no) two nodes will get added up because critical part will be either time of this node or this node or this node or this node whichever is maximum but here all are same, so critical path will be TL by WL. That is new critical path okay this is the new critical path. The time should not get added because they are separated by delays.

So these are not delay free path. They are not delay free I guess, so I take individual times of each node all are same so critical this okay. And then another nice thing you see, I mean I am using only one delay per edge, I am not wasting any extra delay just minimum delay required to separate the 2 nodes is D so that is why I am doing. So it is the optimum use of the total available delays and further and under that per node I have TL by WL of time, so critical path is TL by WL. My claim is this is the best that can be done.

Now you can say why do you say so? Suppose, I mean one of the nodes maybe this one, the first one instead of making it TL by WL I make it shorter that is less than means time T0 prime less than this. Then you can say that look I have made it less so why to stick to TL by WL? I can by designing these processors I can make one of the nodes or few of the nodes shorter than TL by WL.

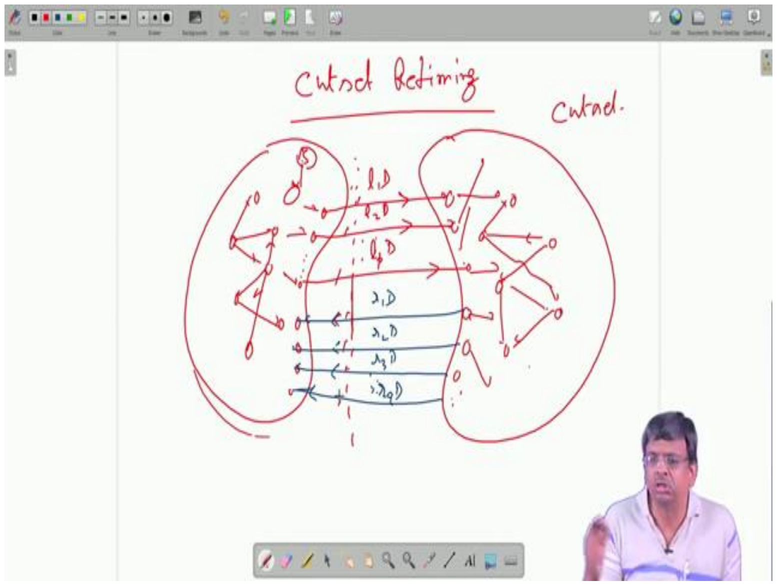
But then if I make anyone shorter at least one of the other nodes or several of the other nodes will get thicker okay, so the time there will be more maybe here the time T2 prime maybe more or maybe T1 prime may be more. Now what is critical path? I have to take the maximum of the individual computation times here because everybody has to finish the job, so in this case critical path increases. Okay, so the best that can be done is this, so this is a figure of merit. It is called this figure is called loop bound.

Okay, that is total computation time of the loop TL divide by WL. What does it stand for? That is this will give the best possible critical path minimum possible critical path you can arrive at in the loop by retiming, you cannot go down below this. This is the best and then suppose you have very loops in the DFG then whatever the highest, highest loop bound is called iteration bound, highest loop bound, highest loop bound is called iteration bound. So for the entire DFG that will be the best that can be done okay for the entire DFG that is the best that can be done okay, so this works like a figure of merit.

So we have understood why we take this thing I mean what is the problem in this retiming of the loop? Now after this so we have obtained this figure of merit for loop we can take some examples. Okay maybe I will take 1 or 2 examples after a while and then we will see how to calculate the loop bound, iteration bound and all that. Iteration bound for the entire DFG is the figure of merit as far as loops are concerned.

Then if you only focus on the loops that is critical path cannot be brought down below this thing iteration bound. Okay this is for the loops. Our next topic is cut set retiming which is a very important topic and very fascinating topic it requires some intelligence also.

(Refer Slide Time: 20:38)



Here cut set retiming, this is a special way of doing retiming of course we do not violate retiming theory. Because if you violate retiming theory and do some arbitrary (you know) adjustment of delays the new circuit will no longer be identical or equivalent to the original circuit you are lost. So we will not do that. But then you must have seen that we had lots of liberties in choosing the retiming values only some inequalities were to be satisfied. But after that we had lots of liberties, lots of ways one can choose a particular way, particular systematic way of choosing this is called cut-set retiming which is basically done by visual inspection of giving DFG and then carrying I mean assigning certain retiming values.

Okay that is called Cut-set retiming this requires lots of intelligence. Okay I give you a circuit you know there is a unique way of doing cut-set retiming it is up to your choice. You can choose the cut-set to be this, you can choose your cut-set to be that so on and so forth okay. So in cut-set retiming what you do, you look at the entire DFG by visual inspection you cut it into 2 halves, separate into 2 halves okay, one half has some nodes inside dot-dot-dot maybe there is a source node S.

I am considering single source, so single source could be either on this side or this side I am talking to be on this side. These nodes they are internally connected. No node will be hanging separately okay, maybe this way, maybe this way this connection, this connection all this but there are some nodes, they have their own connections okay and there are some nodes here like that okay. Same here, there may be some kind of internal connections arbitrarily I am drawing.

There are some nodes here alright, so there will be AVS because after all the DFG is a connected graph. So it is not that it is separated by 2. So from some nodes here there will be edges to some nodes on this side dot-dot-dot okay and from some nodes here there will be edges on the reverse side. They may have delays, may not have delays so I will just remain you know general so I say it may have a delay L1 times T, it may have L2 D dot-dot-dot maybe total I have got LP, total P number of edges in this direction LP number of D.

So all this delays L1, L2 they are positive unless non-negative cannot be negated either 0 or positive. Similarly in the opposite direction okay in the opposite direction we will have maybe odd 1 D, odd 2 D, odd 3 D dot-dot-dot, odd Q D. There is Q number of paths in the opposite direction it is the most denial case. You may not have paths in both the direction. Okay, you may have paths in one direction but I am taking more general. And these nodes are all internally connected.

This may have connection here, this may have connection, this may have connection and like that, nobody is hanging separately, it is a connected graph, connected graph and the source node maybe it is connected to one node called one linked, maybe here one node it will be connected to other nodes and all those. Okay, now this is the structure I just separated out one segment under this region another segment. But actually this various nodes are there they are connected between them like this and one single source would be connected to one node here.

May be have source may have connections to multiple nodes also that is easy that I will come to but single source I have put it deliberately on these. Now these edges you know from left to right and right to left they qualify to be a cut-set. If I remove them and after removal of them the original DFG is separated into one full sub-DFG where all the internal nodes are internally connected nobody hanging separately and another separate sub-DFG.

Again all the nodes are internally connected and nobody is separated out, no one is hanging separately. Okay so then I indicate them by passing a dotted line through them, dotted line. So whichever edges the dotted line cuts those edges are part of the cut-set. That means dotted line goes to these, these, these, these, these, these. If you remove them obviously I am left with this sector and the other sector. No connection between them but this itself is a sub DFG by itself, internal nodes are connected, here also internal nodes are connected.

So I cut it into 2, not 3, not 4 only 2 then this set of edges form a valid cut-set, whereas as cutting them I cut the (dividing) given DFG into 2. So first task, we need to identify such cut-sets. But remember there is nothing viewing it, one DFG you can view it one angle way, you can take some of the edges to form a cut-set, your friend can take some other edges to form a cut-set and you can do a go on doing cut-set retiming again-again-again.

Okay, well some people are smart they will arrive at the ends quickly, otherwise they will choose wrong cut-sets or not very useful cut-sets and go on doing retiming. Okay they are like the intelligent. So in the next half we will see how this can be used for doing retiming. Thank you very much.