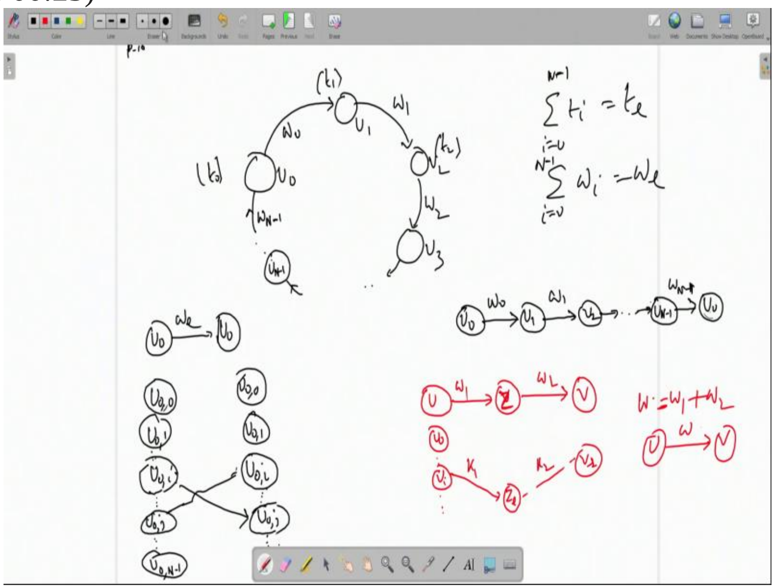
VLSI Signal Processing
Professor Mrityunjoy Chakraborty
Department of Electronics and Electrical Engineering
Indian Institute of Technology, Kharagpur
Lecture 17
Iteration Bound for Loops

(Refer Slide Time: 00:25)



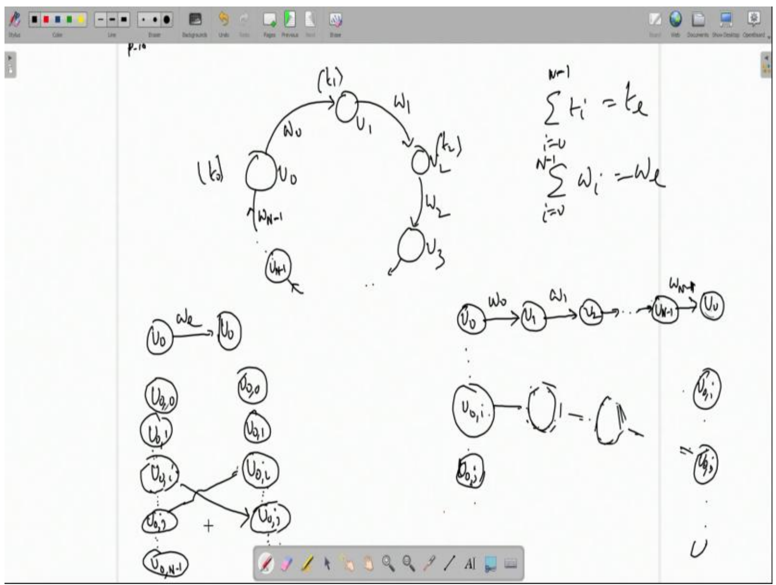
Well, in the last class we are doing this retiming sorry, unfolding of loop. If we have took an example of for loop with three nodes A, B, C and some delays D between A and B 2D between B and C and 3D between C and A. We unfolded by a factor J equal to 3 first and then J equal to 4 and in these two cases, we saw some differences in one case three parallel loops came up each loop having each of the nodes A, B, C is equal to W once in other case I mean two parallel loops came up each loop having A twice B twice C twice.

Then we try to investigate as to why this happened? What is happening? What is there in the background to understand that what I consider is this I considered one result which I can restate again that suppose there is U going to V, going to sorry going to Z, and this has W1 delay this has W2 delay.

If you unfold you will have U0 dot dot dot Ui dot dot dot whatever. So, Ui will go to some node some Z maybe l and Zl will go to some node maybe Vr this may have some delay some delay may be K1 K2 this may have some delay K1 K2. Then if I ask the question that if I start at Ui which Vi finally go to.

Adapter how many systems cycle delay then I said that I can as well drop this Z I can make an single edge of delay W1 plus W2 equal to maybe W just U to V with W delay, then you unfold then from UI you will go to the same here with the same amount of delay K1 plus K2. K and K plus K2 number of system cycle. Therefore if our purpose is to find out which Vi go to adaptor how many system cycle delay, just for that calculation only intermediate node can be dropped that is what I said.

(Refer Slide Time: 02:48)



Then, I consider it loop like this U0 goes to U1, W0 delay U1 goes to U2 W1 delay their timing T0, T1, T2 node times finally U1 minus 1 and from here you go back to U0, this you can draw in this way also as a straight line in a linear manner U0 to U1, W0 delay U1 to U2 here also U1 to W1 delay dot dot dot and finally U N minus 1 it should come back to this U0 but this U0 I repeat here just for convenient they are actually same. So, you have this delay.

Then the question is now unfold by a factor J either U0 0 U01 dot dot dot U0I dot dot dot U0 Z. Now, suppose U0i goes to U0i only...U0i only. Then I will have a loop form because from U0i, I go back to U0i through the intermediate nodes. Like, U0i will go to some node of U1 time that some node of the U2 type dot dot dot.

But finally if it comes back Ui then the loop is formed. So, I go through each node only ones like the first case in our previous example go through A1 go through B1 go through C1 just three parallel loops. But it be so happy that I started U0i after unfold I do not hit U0i I go to U0j. What

I am doing is maybe I should have explained I should say we will more. That, if I my purposes to find out this that if I unfold these and find out from any of the nodes. So, U0i which U0 this U0 that here will also UV no U0 to U0, U0 to U0 total loop is... total delay is W0 plus W1 plus dot dot dot W8 minus 1 which is the loop delay Wl.

So, I told you that if my purpose is to find out this then if I started U0i that is I am unfolding I am getting U01 U02 dot dot dot U0i dot dot dot U08 minus 1 take any one U0i then which of the U0 here I go to after how many system delay. To find that out I can (())(05:10) ignore the intermediate nodes take the total delay, which is Wl and forward node like, for the edge like this this U0 the H starts from U0 goes to U0 again. Total delay Wl, if you unfold that then U0i. Where go to U0i or we not in general go to U0i, if it goes to U0i them basically a loop is formed.

That is in this actual diagram U0i will go to some U1k, U1k will go to may be U2M, U2M will go to something but finally it will hit U0i. So, it will come back but it may not happen there is as I go through I will end up with you some other node U0j where here also I am unfolding, so this is your U0i here, this is your U0j here.

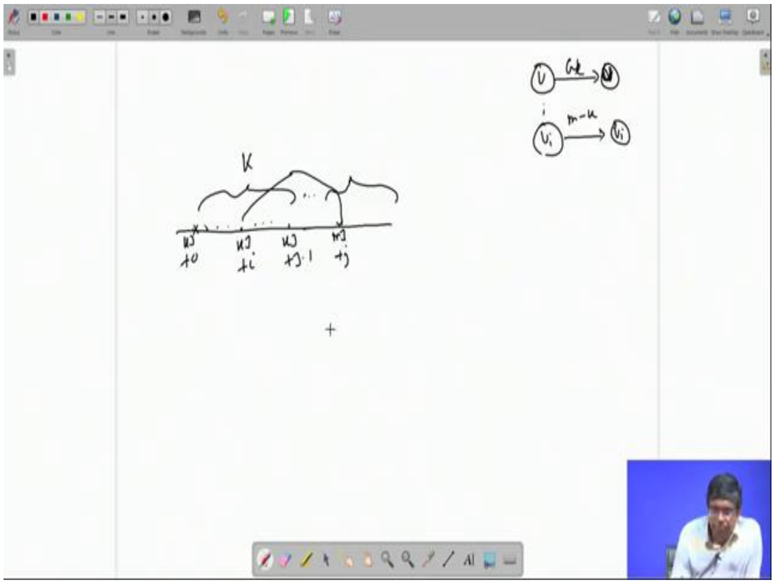
So, I may instead of hitting here I may hit here. So, I go through various intermediate nodes all these intermediate nodes something from here, something for here, this is coming from these, this is coming from this, one of the copies one of the copies, but finally if I hit U0j this is not what I started from.

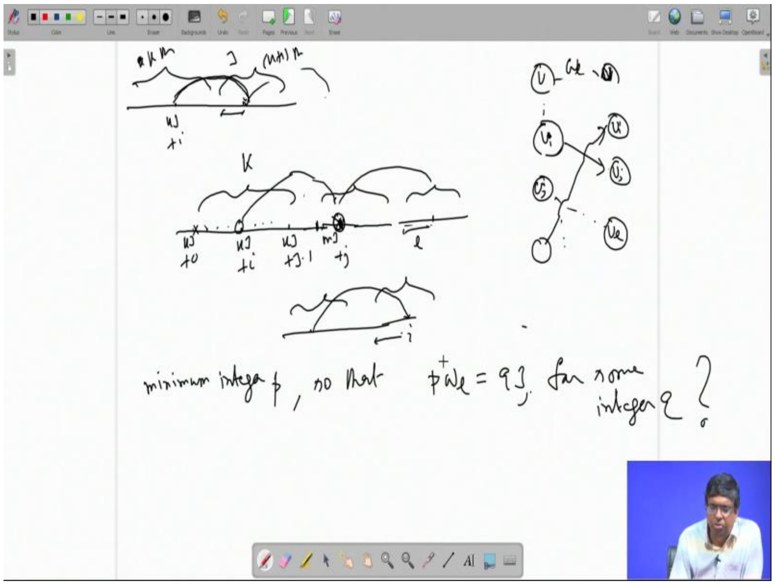
So, loop is not formed then from U0j I will again start. So, this U0j and this U0j actually same for our convenience only I am drawing separately from this U0j again I start I go through intermediate nodes if we takes me to you U0i, then a will loop is formed but in this loop every node of the U0 type executed twice, every node of U1 type executed twice, every node of E2 type exited twice like that.

So, total competition time will be twice the competition time by this loop, competition time by loop where you go through U0 once, U1 once, U2 ones and UN minus 1 once only then the competition time is tl loop competition but here you are going through all those nodes but not stopping again going through all those node then only stopping because you are getting U0i back. So, tl at tl it will become 2 tl this is situation.

So, in general in U0i will not take you to U0i it will hit U0j... may not even go to U0i it may go to some U0l, U0l may go to U0i. So, that times three times I have to hop from this U category to back to U category maybe U0i to U0j then U0j to U0k or U0l from U0l to U0i like that.

(Refer Slide Time: 07:59)





So, this can be further understood through the timing diagram, suppose I am in the k th system clock. So, the starting point is KJ plus 0 as you know then Kj plus 1 dot dot dot in general say Kj plus i dot dot dot Kj plus j minus 1, that is data at this point at this point all this point they are parallel everywhere together in the system.

Here is the original time access where they are coming any faster rate? This is followed by this, followed by this, followed by this. So, short period j time shorter by the systems they all available parallely. So, the clock period has gone up j times, so they are all available parallely out of the way I am crossing kj plus i, this data will be given to node Ui if it is just U to V it will be given to Ui that is if it is some U to V.

Now, original edge has edge has Wl I want to delay is a loop U to U here I have got U to U down Wl amount of delay U to U. So, Wl amount of delay means if I start counting 1, 2, 3 I will finally hit something here. If it is again, i maybe KJ plus I and this is some MJ plus 0 dot dot dot MJ plus i. So, i th data goes to i th. So, from Ui I go to Ui and amount of system delay will be M minus K.

This M th system clock this is K th, this MJ plus 0, MJ plus or MJ plus i this KJ plus 0, KJ plus 1 KJ plus i but i if it goes to i then Ui will go to Ui and total... So, it will for a loop because I started UI and I go through intermediate nodes which are not visible here which you ignore but finally it will end up at Ui. So, I go through the nodes only once all intermediate nodes only once and I happily come back to Ui.

So, in this case a loop will be formed from UI similarly from UJ similarly for UK same thing Ui has been taken just denied? So, this case part loop I have computation time same as, computation tl because every node is executed only once never repeat twice or thrice, but this may not happen, this cannot happen because it may not hit MJ plus I it may not hit i th it may hit some point here some MJ plus may be small j.

So, i th Ui will go to Uj then here I started Uj these toward same but I write is on this side they actually same Ui going to Uj. Actually it should have been like this Ui going to Uj so it has been like this. But, I am drawing them separately input side output side but there actually same nodes. So, Uj if I start it might again go to so maybe MJ plus J this point for it hits.

So, I started here hit here and i and j am same that is why Ui going to Ui going to Vj that Uj then Uj again start U0 will go to some other place? Maybe this much is l. So, it will go to Ul like that, finally after sometime. What will happen is this if we extract these diagrams here, finally from some point it will come back to the, i th point of some system clock.

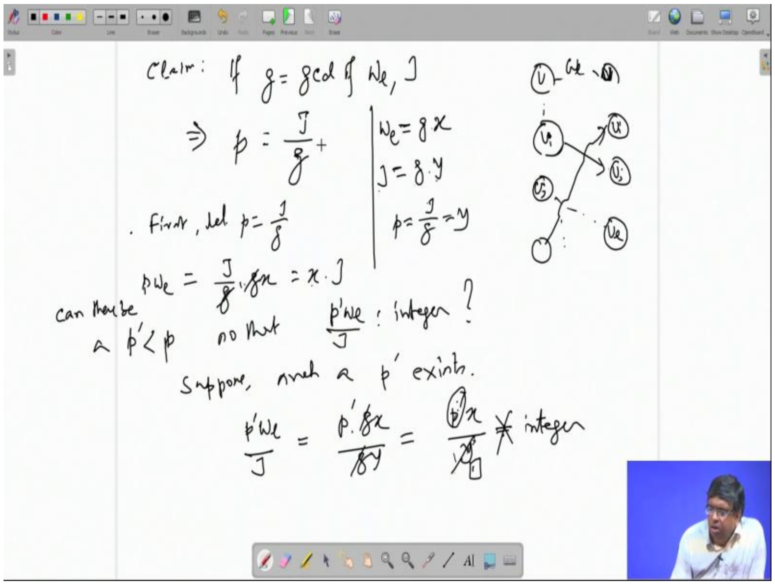
So, then it will come back to Ui from somewhere it will go back Ui so loop will be formed every time you cross one from here to here tl amount of the competition time. So, if you have three such parallel things you know that 3 into tl or 4 into tl, 5 into tl like that. That means from here I go to the right by Wl I do not find i th then again I go by Wl.

I will find i th quite I will find i th. If from KJ plus I, I go to the right by J if I will KJ plus i suppose I go to the right by J then from this clock I will go to the next clock it is KJ it is K th, it will be K plus 1 th but this will take me to the i th. If I go by 2J it will again take me to the i th. So, KJ plus i it will go to K plus 1into J plus i, then it will go to K plus 2 into J plus i like that.

So, i th will always be you were going to i th if the amount by which I am jumping is a multiple of integer multiple of J there is one J times jump from i to i go to i another jump by J from i again I go to I. So, here the point is I am not jumping by J, capital J. I am jumping by Wl another Wl another Wl.

So, what is the minimum integer p? So, that p time this Wl this supposed to be at integer multiple of J then only i will come back to i the loop will be formed. It is qJ for some integer q this a question. What is the minimum p integer p? So, that p time Wl there is a if I hop p times that length will be an integral multiple q times J. So, that from i th if you start you end up with i th a loop is formed and since it is minimum that many hops is required you cannot avoid it. So, (())(15:05) what is that p that is the question?

(Refer Slide Time: 15:17)



Claim if g be the greatest common divisor gcd of Wl and J then claim is that p which we are looking for it is J by g. This is claim, how to verify the claim first let p be as before J by g. Now, since g is the gcd of Wl and J, Wl will be this g times some x for x is an integer and J will be same g times another integer y and they are cannot be any common factor between the X and Y that is the definition of greatest common divisor.

That is all such common factors are part of g x and y there are called co-prime there is except for one there is no factor common between them and this was the claim suppose this, we start with this p then is p Wl and integers times J will verify that yes, then the question is (())(16:42) the minimum p for which it will be... for which pWl will be some integer times J or can there still be a lesser value of p for which also pWl will be some integer times J first verify whether this p?

Whereas this p into Wl is it and integers times J you can replace Wl by gx and p by J by if you take this p J by g is p and W is gx and g and g cancels, so you see x times J. So, actually x is an integer. So, q takes the value x the value of x one thing is sure for this p, p times Wl will be an integer times J. So, if I hop by this p from i th node, I will end up with i th node, but is it the minimum size p or can I still have a p prime?

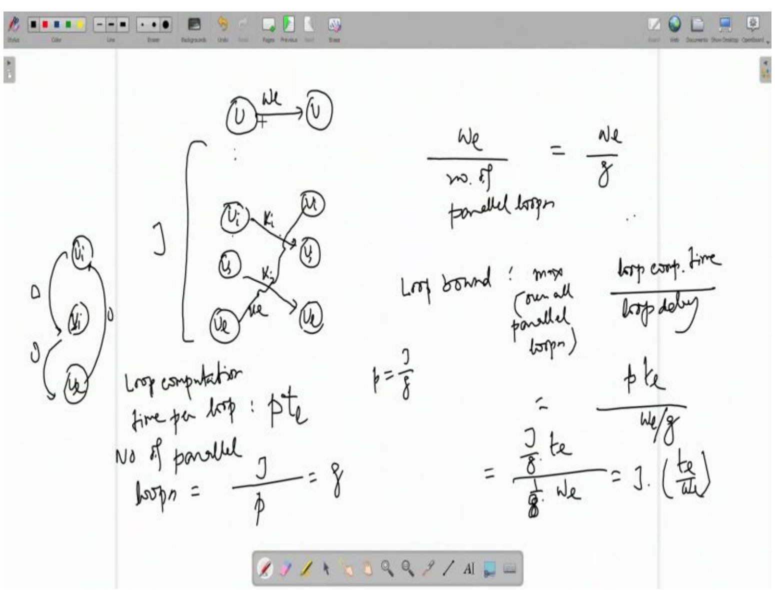
Less than p integers p prime. So, that for this p prime also if you take p prime into Wl divide by J. You get an integer? When is can there be a p prime less then p. So, that for p prime Wl also it is an integer times J that is if it divide by J you get integer like q this a question, answer is no this is the minimum p, why it is a minimum p, because suppose such a, p prime exists.

Then p prime Wl by J, p prime Wl is gx J is gy. So, gx by gy. So, g and g cancels p primes x by y. Now, what was this, this p, this p was J by g. Now, what is J by g, J by g is y. This p is J by g but J by g is y. So, p actually is? That p by definition is J by g which is y. Now, p prime is less than p, so that is means p prime is less than y because y is p.

So, p prime is less than y. So, even if p prime suppose is a factor of y and it cancels fully you will have something here or greater than 1 because p prime is less than p. That is less than y whereas p and y are same. So, p prime by y even if p prime is fully contented y as factor after division there will be something in the denominator which is greater than 1 but that cannot be common with x because x and y are co-prime there cannot be any common factor between them.

So, this can divide be and integer, this can divide be an integer. Because even if it p primes cancels with y there will be something here that cannot be part of x. Because y and x have nothing common when they are co-prime. So, x by something it will be a fraction it cannot divide by an integers that is why such a p prime is not possible, alright. So, this how many p? J by g where g is the gcd between Wl and... with this suddenly else we can derive.

(Refer Slide Time: 20:37)



Suppose I have got original loop or something like instead of U0 U1, you can even call it EU U prime instead of U0. I am putting U prime just changing the name sorry something like this. That means up W0 delay W1 W2 and loop competition time is Wl loop delay. Loop computation time that is I am not changing anything, just changing the name instead of U0 and all those things I am saying U instead of U0 U1 U2 I am must calling it U, U prime, U double prim like that.

And then the loop we can. So, we can drop the intermediate nodes you can take it from U to U itself Wl that is I am showing U2 U Wl. So, no U0 unlike the previous example just U to U, I have put U, U prime, U double prime just change them notation nothing else nothing conceptual here. If I am unfolding by J so suppose I consider particular one Ui it will go to some UJ then you might take you to some Ul with some delay.

Delay I am not writing maybe you can write Ki, KJ whatever then Ul might take you here Ui. So, loop will be form Ki KJ Kl like that. So, this example just three, three times some hopping Ui to UJ one hop again UJ to Ul one hop as I go through, I will go through the intermediate nodes not shown here not visible here they are not shown here exclusively. I will go through those intermediate nodes finally hit back J th node of the U type from J th again I will go through the intermediate node. So, I will basically come to Ul actually Ui to UJ to Ul and then in this example from Ul to Ui.

Actually it goes like this, goes like this. But I am showing input and output side separately that is why same node is repeated. Now, how many times how many hops will like, that will be p every time I every time I go through Wl basically I am hopping I am going through all the nodes once. So, Ui to you UJ that is one hop that is what one hop.

So, I will go through all that node intermediate nodes. Here one hop, but UJ will not take me here how many times I should go p? Then only p into Wl will turn out to be some integer times J and from Ui I will go back to Ui. So, every time I go in the timing diagram Wl to the right I go through the intermediate nodes, one hop. So, p such hopes will take me back to Ui.

So, one hop, another hop, another hop, p hops. So, that means and every time how much competition time tl. Because as I go from U to U as I go from Ui to UJ intermediate nodes and the Ui that will take tl amount of time tl is the loop competition time.

Again from UJ as I go through all the intermediate nodes and reach up to hear another tl and again from Ul as you go through all the internal nodes internal intermediate nodes with here, again another tl. So, general competition time will be for loop will be p times. How many loops? No, actually is how the loop is formed you see here Ui to UJ, UJ to Ul dot dot dot and finally UI.

So, I have unfolded U. So, I have got total J is copies of U out of which certain copies they form a loop we loop within themselves again another set of nodes will form a loop with themselves they cannot have anything intersection they cannot have. Because if I... timing diagram If I started Ui there is a KJ plus i if I go somewhere here if I started at some other point, I cannot hit the same point I will hit another point.

So, if KJ plus i take this to UJ then KJ plus M will can do something else it cannot be the same UJ which means a set of nodes will form a loop within themselves among themselves, another set of nodes will form a loop amount themselves, but there is no intersection there is no you know common thing between the common node between the nodes, between the loops.

So, out of capital J how many has loop? In one such set I have got p times because one hop, another hop, another hop. So, total p hops will make one loop after unfolding. So, p nodes out of J are taken away one loop is formed another p nodes, for being another loop taken away.

So, how many such parallel loops. So, number of these are all after unfolding will be total number of nodes by p, p was (())(27:03) from J by g, so this is actually g and then what would be the loop delay this loop like K1 plus K2, Ki plus Kj plus Kl like that. So, total there will be some delay here, some delay here, some delay here. So, is a loop delay per loop.

Now, you remember we evaluate an expression that if add all the delay free guts Ki, Kj, Kl all the delay free guts will accounts of all the edges the sum total is equal to Wl. So, Wl will be distributed equally because it is very symmetric equally between the loops. That is if I take one set of nodes they form a loop.

Fraction of Wl which comes in the total delay count, in this loop that will be same as the fraction of Wl that is given to another set of nodes for being another loop. That is Wl will be divided by number of parallel loops. Because all the edge delays Ki, Kj, Kl and then all if you add total is equal to Wl that we have proved earlier.

So, out of which I am taking away sum like here. So I have got certain number of delays. So, that is due loop delay for that loop. Again I am time taking away another set of nodes. They are forming a loop within themselves and the system loop delay. Since, this is very symmetric total delay total number of group delay. That is Ki plus Kj plus Kl here.

Another, there will be same for all of them does not it? So, if same and the total is Wl per parallel loop how many delays Wl the total divide by total number of parallel loops. Where is parallel loop they are identical reception symmetric. So, if one set of one parallel loop get certain number of delays say Ki plus Kj plus Kl other also will get the equal amount of delay and total is J, total is Wl. So, WL by total number of such parallel loops and total number of parallel loops is G. So, Wl by g.

So, what is the loop bound now, loop bound you remember the best we could do that is if you do retiming the best we can again you can achieve the best in terms of reduction of critical path. So, what is the loop bound? Loop bound will be per loop take any loop, loop competition time to take the max overall loops.

All parallel loops, next up what for any loops say you find out? Loop competition time divide by loop delay. Now, per loop how much competition time up in p times tl and per loop how much delay? Wl by g and what is p was J by g. So, if we replace this by J by g times tl by 1 by g times Wl. If it is J times tl by Wl which is common for all the loops. No need to maximize... So, you see this beautiful thing before unfolding loop delay was tl total sorry loop computation time was tl, loop delay was Wl, some are iteration bound or loop bound was tl by Wl.

So, you could bring down the critical path by retiming to that level, but now it has just increased J times. There is we cannot not reduce critical path below J time original loop bound, original loop bound was tl by Wl down this is gone up. So, I cannot reduce critical path below this now and this is happening because of only one thing.

As I told you after unfolding delay counts go down, that sometimes delay free path emerge. That is why critical for cannot be you know, reduce arbitrarily that is a reason. Because delay count goes down on the other hand number competition per loop goes up because p time you are hoping means tl plus tl plus tl, p times.

So, computation time is increasing by p but in that loop I am not getting the full set of W, full amount of Wl just a fraction of Wl, Wl by g I am getting. So, loop delay decreasing loop competition time imply overall loop bound is increasing which means I cannot being down the critical path after unfolding, of course after unfolding. I cannot bring down the critical path below this J times these. So, that is all for Loop Unfolding. Next, I will go to what is called bit level and digit level structures, thank you.