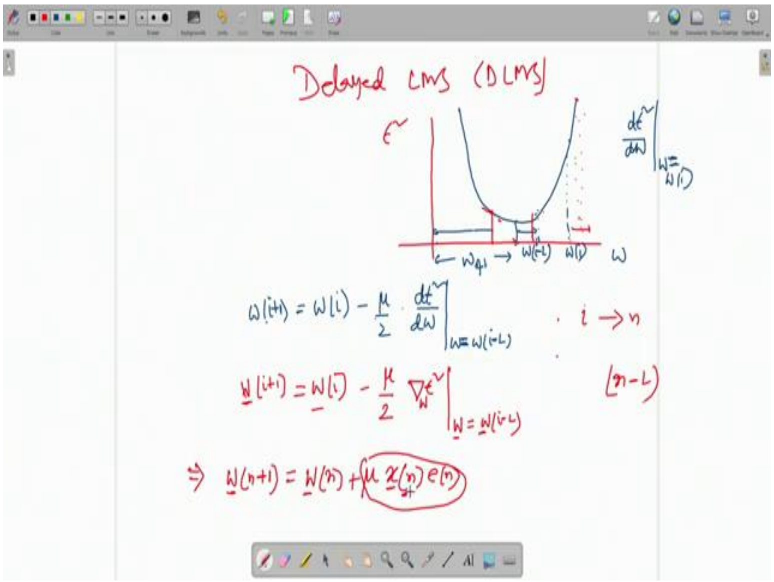
VLSI Signal Processing
Professor Mrityunjoy Chakraborty
Department of Electronics and Electrical Communication Engineering
Indian Institute of Technology Kharagpur
Lecture 12
Retiming delayed LMS

(Refer Slide Time: 00:32)



So we have seen the problem of retiming and regarding LMS based adaptive filter so, now we will go to what is called, delayed LMS, DLMS it is a modified version of LMS algorithm. Now, remember how we derive the LMS algorithm, I took a gradient descent search that time I took only one coefficient as an example, epsilon square was the inner variants if you plot it with the coefficient w single coefficient I say there is a coordinating function it will be only one unique minima, so it was here, so this is your w watt, now that time suppose at the ith step of interaction and here, wi I take a gradient of this external square vis-a-vis w that is, evaluate it at w equal to wi.

Then, from current wi I subtract a quantity proportional to this, proportionally constant in Mu by 2, so, mu eta rate will be that is wi plus 1 was wi minus some mu by 2 into this gradiant. Why I did it? Because, if it is positive like here, so I will be subtracting from wi I will go towards left which is the proper direction because w s is located to the left, if I am on this side, gradient is negative, negative negative positive so, I will be adding not subtracting, then I will go in to the positively, the correct direction.

So, I will always go back and forth in the correct direction and finally hit upon this, that was my goal. But, then that landed being to an adaptive filter which is not retimeable. So, I will take a re-look, we will propose and modify the approximate version which was watts, theoretically also it has been proved we will work, it will converge but, this is called delayed LMS what I do, is this at the Ith step I generate new eta rate w I plus 1, by subtracting as before from wi a quantity proportional, constant mu by 2, d epsilon square dw but this gradient I do not take at this point, I take at a point where I was several iterations ago, that is if this is may be w i minus l.

So, l iteration ago I was here may be, or I was somewhere else and just took this point as an example that time what do I with the gradient, that I use here. That is w equal to, okay. Then you can say, I mean am I not losing anything, the point is suppose, l cycles ago I was here to the right here also I am to the right that time also I was to the right, then gradient down is positive, that time also positive, so that time from gradient is positive so here if I do the subtraction, this is the positive quantity, still a positive quantity so I will be subtracting.

So, I will go in to the correct direction, nothing wrong. Only the magnitude of the gradient here was steep. So, I have been jumping by a longer margin here the gradient magnitude is not so steep, so I will jumping towards the optimal 1, which is smaller amount but, direction of my movement will be correct in this case. Because, gradient here also positive, here also positive, so here also I would have been moving to the left, so I would have been subtracting from wi something positive. Here also something positive I am subtracting because gradient is positive.

Amount the volume of that amount the magnitude of that is less now because, here it was steep here the gradient is no so steep. But, still it is okay, now you can say that if I am not here, If I am here, suppose if I am here, then my gradient would have been negative, so instead of subtracting I will be actually minus minus plus so that I will be adding so, it means from wi I will move to this side, wrong side, so I will go somewhat here, and like that. After some point of time, suppose I am here after some iterations then when I go back by l cycle I will be still on the right hand side of this optimal.

So, then my gradient here or gradient here will have the same side so, I will start walking back to the correct direction, so basically it will be just, a little delayed, okay it will be, it might be back and forth I will be going in a wrong direction for a while and then I will correct myself. Then how do you… this was for one case then I made it for the case where we need only one coefficient, we had so many coefficients.

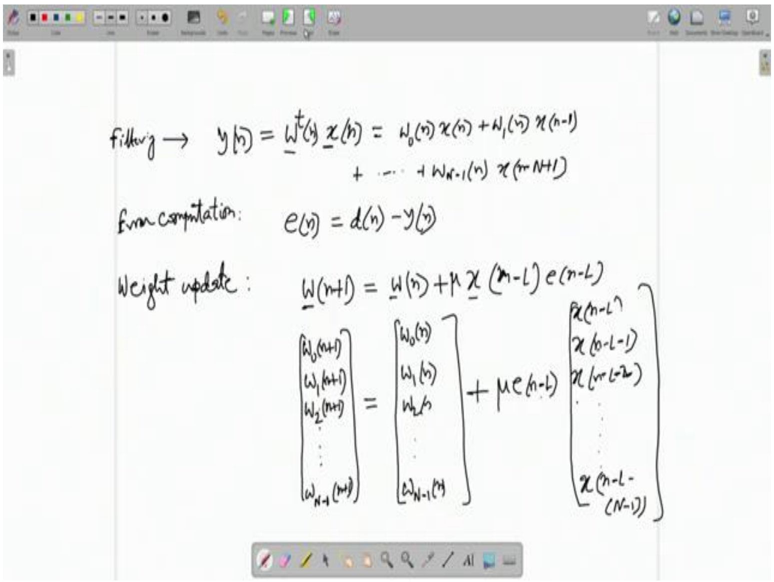
Capital N number of coefficients so that resulted in last time w vector i plus 1 was wi vector minus mu by 2, all the gradients put together in a vector from that was del w working on this guy epsilon square and to be evaluated at w equal to, all the coefficients delayed by the same amount. That is w0 coefficient not for i index for i minus l, w1 coefficient not for i index but i minus l dot.

This is the delayed steep is decent from this how did I go to LMS I said I will make it online as though iteration is carried out in real time, ith iteration is done in ith clock so I said index i I will replaced by N because we are more conversant with n being the index of clock real-time clock so, nth index, nth time index. So, at nth clock nth iteration so, instead of i here we replaced all i index by e, n index basically same just changing the name, n index means nth iteration and that will be doing at the nth clock, so nth iteration to be done in nth clock, n plus 1 th in n plus 1 th clock, n plus 2th in n plus 2th clock and so and so forth.

That is what they did, and then the gradient expression had auto correction matrix r (())(07:18) vector p so those we replaced by some approximate values and we work out this, we worked out and eventually we have got the LMS formula. This is the approximate, approximation of this gradient part, gradient update part, weight update part, gradient means weight update part, approximate because r matrix auto correction matrix was replaced by some ordinary estimate cross coefficient vector p also was replaced by some ordinary estimate, where estimate in a crude estimate.

That is why this is not correct update but approximate. But it does serve the purpose that also LMS. Now, we are doing the same thing but gradient is computed not at n, I replace by n nth clock so, I am taking that gradient of n minus lth clock so, I will have xn minus l, en minus l so now will be if the delayed LMS it will be n minus l, this minus l. so w n plus there is a gradient not at nth clock but in n minus nth clock, that is all.

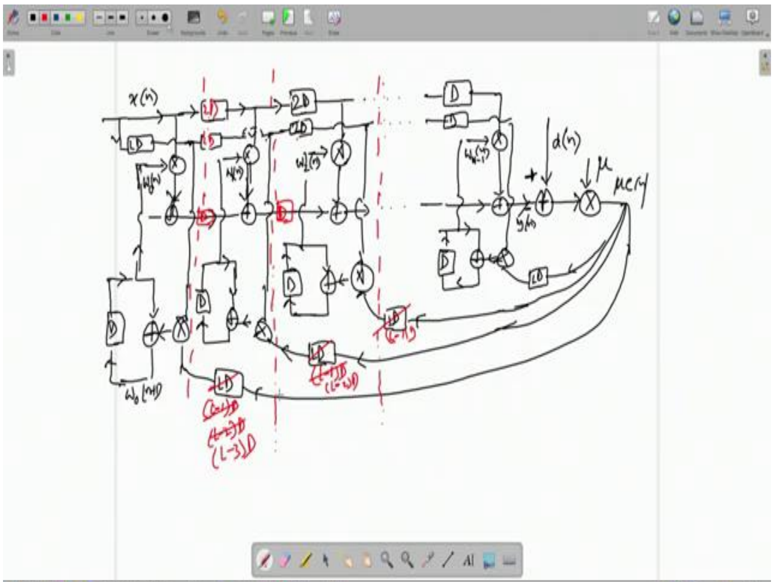
(Refer Slide Time: 09:04)

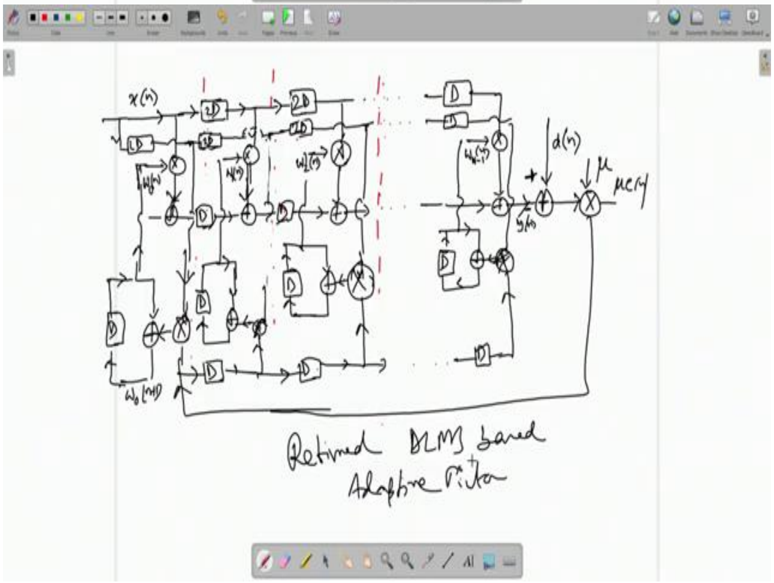


So I can rewrite the equations delayed LMS its equations will be basically, it does three things one is filtering as before, so this is no different, assuming that coefficients are available at n th clock, wn vector or all its components w0n, w1n dot they are all available at the n th clock, I simply have to find out the filter output as before.

This is fine there they get (())(10:06) calculation, this is also as before. Now, weigh update, this is what the difference with, LMS will come as I have told, explained wn plus 1 will be wn plus mu xn en, now sorry just a minute, okay l cycle delayed, delayed gradient, this is what we have derived here, so now I will just draw the architecture of this in the next page and we will see that now, it retimeable, it is because there is delay that has come up, the delay that has come up in the ni expression e n minus l, earlier it was en, en was feedback because no delaying the direction that is why I could not take out delay from that side and put them in the forward direction, but now (())(12:06) we have l cycle delay which we could now manipulate. So, I could go the next page and draw the architecture in the same way as we did last time.

(Refer Slide Time: 12:17)





Sorry this is multiplier, all these are same as before they can FIR filter w 0 times xn w1 time xn minus 1, w2 times xn minus 2 only the coefficients are not fixed they are flying with time that is why they have the function of n, and this getting added. This is the yn to be subtracted from dn remember, this DFG has two inputs tau one single source xn another source is dn, that is where, by previous discussion on retiming DFG with two single sources one on left side other on right side that whatever I discuss that time, that will very relevant applicable here, because I have got now a circuit where there are two single sources xn and dn. It multiplied by mu bring it back , now if you see, as before if I writing in vector form, expanded form, it is w0n plus 1, w1n plus 1, so same as before, here will be (())(17:15) because there is a mu missing here sorry, mu into en minus l you can write together first, times xn minus l so starting coefficient will be no longer xn itself will be xn minus l then we will be one cycle delayed n minus l minus 1, then n minus l minus 2 dot, minus, so n comes to first coefficient w0n you are updating as w0n plus mu into e n minus l into xn minus l. mu into e n minus l you can generate by first take the product mu into e n then delay it by l cycles.

So, it will become mu en minus l but now you are multiplying by xn minus l earlier it was xn now it is n minus l, next time you will be multiplying not this with not xn minus 1 but, n minus l minus 1, now remember that. So, here I have got mu en so this i have to delay by l cycles and multiplied by xn minus l, so xn I will pass through l number of delay. So, xn minus l is here, this i take that to multiplied by mu en minus l, so l delay of this and then with update part as before, so this with w0 n plus this product this is your w0n plus 1, here there is no difference, I put it in delay.

Then next coefficients w1n this one, for that w1n is updated as w1n plus same mu en minus l, and xn minus l minus 1, so already had n minus l delayed further by 1, to be multiplied by the same mu en minus l, added with w1n. So, already xn minus here, so one more delay, we go with update loop, no different from earlier, this I put back here, next what if you want to see, you can take one more, same mu n minus l, but n minus l minus 2.

So already we have n minus l minus 1, one more delay, so n minus l minus 1, so one more delay, multiplied by the same, sorry, just a minute, multiply then we erase this, dot and last one again, now we can just extend, it will be like this, multiplied by the same mu en minus l, and then. Now, if I want to, look what is retiming so, my target is this adder chain, this adder chain everywhere between this adder and this adder, this adder and next adder, I want to insert a delay.

How to do that, so start with this, suppose I cut, so this is forward direction, this is forward direction and this is forward direction and this is backward direction, in the backward direction, now I l number of delays so I make it l minus 1d, I inserted delay here, I make it twice D, and I have a d here, then I want to have between these two adders again another delay, we need these two adders so again I cut, like this but further if I cutting this, I have to cut this also, so this will be cut again, now here, the forward direction, so I doing a delay here and change these delays from one delay to two delays.

And from here I changed to, l minus 1d and now one more delay gone from here, so it will be changed to l minus 2d, next time again I will be cutting something like this, this will become l minus 1d, this will become l minus 2d, whether to cut all of them, when I cut this, I cut this also, this also, so this last line gets cut always, this last line gets cut almost always , I mean leaving the first case and so on and so forth, and this becomes l minus 3d, so eventually I will have delays in these paths and last path will have 0 delays, if l is just equal to the number of, forward paths you have retiming, then every time you do the retiming one delay gets reduced from here, so it may turn up to be 0.

Next one will be one delay, next one will be two delay like that, and always 2d, 2d and 1d. Again 2d, 2d 1d again 2d 2d 1d like that so, then the circuit can be modified like this on the final circuit I draw, so this a will be having no delay, next will having one delay. Because if this has come to 0, next will come to 1, next will come to 2 etc. keeping that in mind mu en will directly come to this, then this will be delayed by one cycle and then it will come here, now only one delay, then it goes here, one more delay it goes here, dot lastly again one more delay, goes here and let me change the color here, it will 2d, will be just d, gain 2d, 2d will be just d, so on and so forth, okay this is the retimed DLMS based adaptive creator, I believe this gives a good idea about how to retiming complicated circuits, you take some examples may be from (())(30:01) book or some other book and try your hand.

Retiming is a very tricky issue I mean it requires, specially (())(30:10) retiming requires, I mean puzzles solving capability I mean you should have some mental clarity it I mean it requires some sharpness of mind, but it very facinating and also, as I told you it is a quantum study, it is a foundation what about this entire topic of VLSI Signal Processing that is when we move to our next topicwhich is parallel processing even there we will see, retiming comes in other ways, it is not the retiming is always used to minimize the critical path, it can be used to minimize other things also.

So, with this I conclude this chapter on, section on retiming, retiming one application we have seen how to minimize the critical path, so that the speed of the circuit throughput, goes up. Another way of increasing enhance the throughput is going parallel, parallel processing and parallel processing you pay in hardware, pay in power, but in gaining speed, parallel processing when applied to single processing DSP has a particular name unfolding. So, we start with unfolding in the next class. Thank you very much.