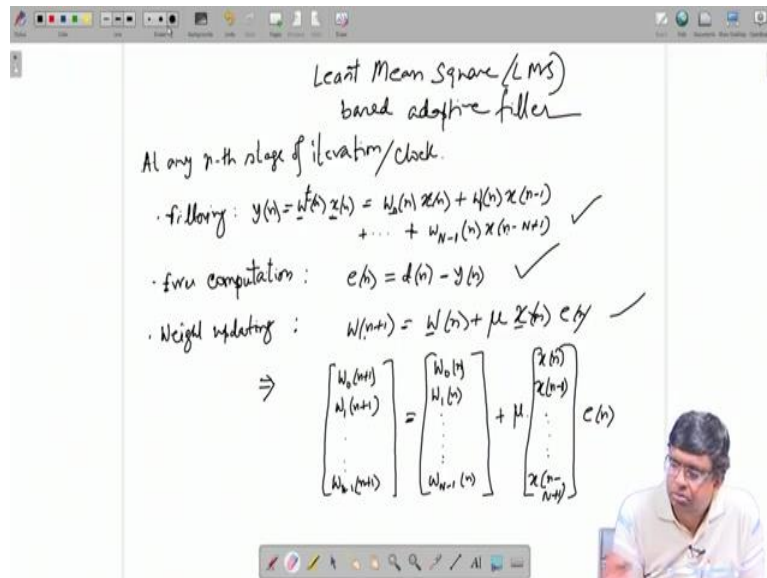


VLSI Signal Processing
Professor Mrityunjay Chakraborty
Department of Electronics and Electrical Communication Engineering
Indian Institute of Technology Kharagpur
Lecture 11
Retiming LMS

(Refer Slide Time: 00:37)



So, we start from how we ended the last time we derived this adaptive filter not derived. I presented this famous adapted filter algorithm call LMS least mean square based adaptive filter. The equation was like this. At any nth stage you do three operations at any nth clock. ((1):16). Then it is clock, nth clock because one interaction per clock. You are doing three operations one is filtering. That is filter output is y_n is equal to $w_n^T x_n$ vector. Which is nothing but you can write w_0 have to put n because even if it is a 0th coefficient it changes from n to n . x_n plus next coefficient $w_1 n$ dot there is a filter output which is nothing but $w_n^T x_n$. Let me remove this.

Last but we say this. All are function of n because they are adaptive they are changing from clock and the data sample here is... this is the meaning of this filter output $w_n^T x_n$ so that is first what to do. We are assuming w_n is already available that is a filter coefficients all these coefficients to be used at the nth clock are already computed are evaluated from the previous clock. So we will just simply filter the input like a FIR filter and calculate the output y_n and that is number one. Then either computation, you calculate either e_n as given desired response d_n minus y_n , this y_n is filter output. Then you upgrade the vector w_n to a new value

w_{n+1} by the LMS formula and the w_{n+1} you hold till the next cycle comes then you give it to the filter.

Then w_{n+1} vector will be used as the new filter at $n+1$ th clock. So weight you can say weight adaptation adjustment update I prefer the weight updating, you can also say weight adaptation or adjustment. So you generate new weight vector from previous weight vector and μ times x_n vector e_n remember x_n is Vector but e_n is a scalar so μ multiplies all the elements of x_n . Typically it is the vector typically it means if $(\dots)(5:23)$ 0th coefficient w_{n+1} will be w_n or maybe I just. If I want to write the vector form. It was like this $w_{n+1} = w_n + \mu x_n e_n$ this is vector nothing but $w_{n+1} = w_n + \mu x_n e_n$ into scalar e_n but if I write in expanded form it is like this.

Now I have to develop an architecture a circuit which will do all this calculations. That is filter output calculation, calculation of this and calculation of this. At reunite cycle, ok. Therefore for that first start with input x_n . You have to do it like an FIR filter x_n into w_0 it is a function n but suppose $w_0 x_{n-1} + w_1 x_{n-2} + \dots + w_{N-1} x_0$ into this. w_{N-1} and they are sum it is like a convolution sum typical FIR filter for this you have to proceed like FIR filter to assume that these coefficients w_0, w_1, \dots, w_{N-1} all these are known they are available next it will be subtract it is output y_n from d_n .


Then is the most important step from using e_n and then using x_n vector. Calculate μ weights each coefficient. So all the coefficients are used here w_0 used here w_1 used here all these are used. So now we have to be updated by an either extra term will be the new values for $n+1$ th clock. So I will be holding them if $n+1$ th clock. I will be using them for filtering to generate y_{n+1} . I will go on like that. This is the meaning of this.

(Refer Slide Time: 8:26)

Least Mean Square (LMS)
based adaptive filter

At any n -th stage of iteration/clock.


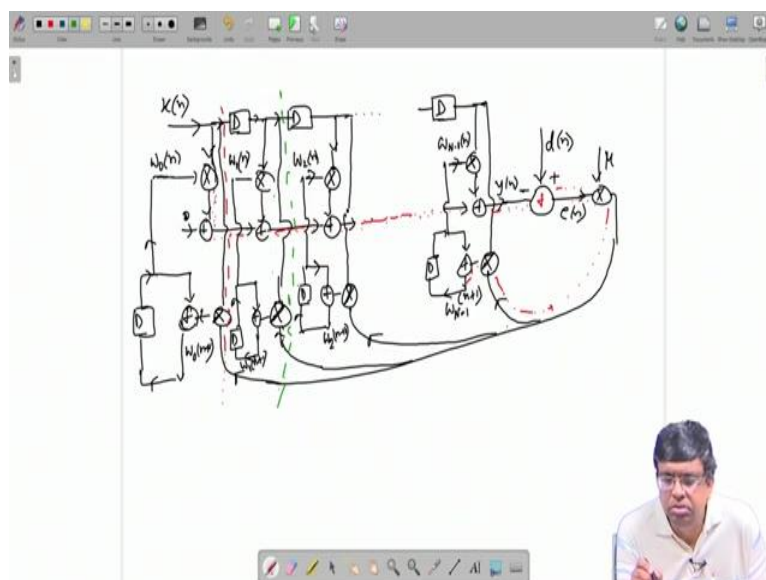
- filtering: $y(n) = \sum_{k=0}^{N-1} x(n-k) = w_0(n)x(n) + w_1(n)x(n-1) + \dots + w_{N-1}(n)x(n-N+1)$ ✓
- error computation: $e(n) = d(n) - y(n)$ ✓
- Weight updating: $w(n+1) = w(n) + \mu \sum_{k=0}^{N-1} x(n-k) e(n)$

$$\Rightarrow \begin{bmatrix} w_0(n+1) \\ w_1(n+1) \\ \vdots \\ w_{N-1}(n+1) \end{bmatrix} = \begin{bmatrix} w_0(n) \\ w_1(n) \\ \vdots \\ w_{N-1}(n) \end{bmatrix} + \mu \begin{bmatrix} x(n) \\ x(n-1) \\ \vdots \\ x(n-N+1) \end{bmatrix} e(n)$$


Least Mean Square (LMS)
based adaptive filter

At any n -th stage of iteration/clock.

- filtering: $y(n) = \sum_{k=0}^{N-1} x(n-k) = w_0(n)x(n) + w_1(n)x(n-1) + \dots + w_{N-1}(n)x(n-N+1)$ ✓
- error computation: $e(n) = d(n) - y(n)$ ✓
- Weight updating: $w(n+1) = w(n) + \mu \sum_{k=0}^{N-1} x(n-k) e(n)$

$$\Rightarrow \begin{bmatrix} w_0(n+1) \\ w_1(n+1) \\ \vdots \\ w_{N-1}(n+1) \end{bmatrix} = \begin{bmatrix} w_0(n) \\ w_1(n) \\ \vdots \\ w_{N-1}(n) \end{bmatrix} + \mu \begin{bmatrix} x(n) \\ x(n-1) \\ \vdots \\ x(n-N+1) \end{bmatrix} e(n)$$



So therefore I go to the ((8:27)) directly I will come back to this page again if required. Take the Input x_n , x_n will be multiplied by like an FIR filter w_0 but here w_0 is a function of n , it changes from clock to clock. Then there is a delay multiplied by w_1 create a delay multiplied by w_2 dot last one more delay and multiplied by Last coefficient it is a function on n . And then you have to add them. You can put a plus and 0. Basically whatever you have. That only comes. That this gets added with this.

We have only done these kind of things so while doing signal for graph for FIR filter you just have to refer that only coefficients are here function of n multipliers, then again this addition multiply this addition it will get added like this dot. So, y_n will come up. This up to is subtracted from d_n there is your e_n multiplied by μ . So μ into e_n comes. Now have to generate w_0 n plus 1 w_1 n plus 1 w_2 n plus 1 and holding in flip flop at then release in the next cycle, Right? How to generate w_0 n plus 1 so you come back to the previous equation, the previous page w_0 n plus μe_n into x_n . w_1 n plus 1 is w_1 n plus μe_n into x_{n-1} so and so forth.

So just that μe_n is already available. That into x_n . μ into x_n plus w_0 n . w_0 n plus μe_n into x_n they are added the output is your w_0 , w_0 n plus 1 by the formula what is w_0 n plus 1, w_0 n plus μe_n times x of n . μe_n times multiplication x of n drawn from here that and w_0 n as of it the 2 available is giving here so w_0 n plus 1 come. This I will be holding in flip flop in delay.

In delay and release in the next cycle so next cycle this will come here. It will generate μw_0 , w_0 n plus 1 and loop will continue. There is a weight update loop what w_0 n coefficient. Then w_1 n you see. How to generate w_1 n plus 1. μ I mean 2 w_1 n even μe_n times x_n minus 1, w_1 n plus μe_n times x_n minus 1 so what you do is μe_n is already taken x_n minus 1, x_n here x_n minus 1 here so from here now you take a same line. Sorry there will be a multiplier, this multiplier so μe_n times x_n minus 1 that is to be added with w_1 n . They get added and the output i hold in a flip flop holding register delay.

This is your w_1 n plus 1. In the next cycle it moves here. You can do a few more also. If we had w_2 n plus 1 w_2 n plus μe_n times x_n minus 2. So, w_2 n and μe_n times x_n minus 2, x_n minus 2 is here x_n minus 1 x_n minus 2, so from here you take it at same line here it gets added with... so this will be your w_2 and again this will be put in delay like that. And lastly if I verify and you can even change now, last coefficient so this much of x_n minus capital N plus 1 that is to be multiplied by μe_n added with this coefficient. So you can take again another

tap from here. Do you think you can have multipliers? Same structure. This is your w . So this is the basic structure of an LMS adaptive filter. What is the critical part? critical part will be, you start from here these multiplier.

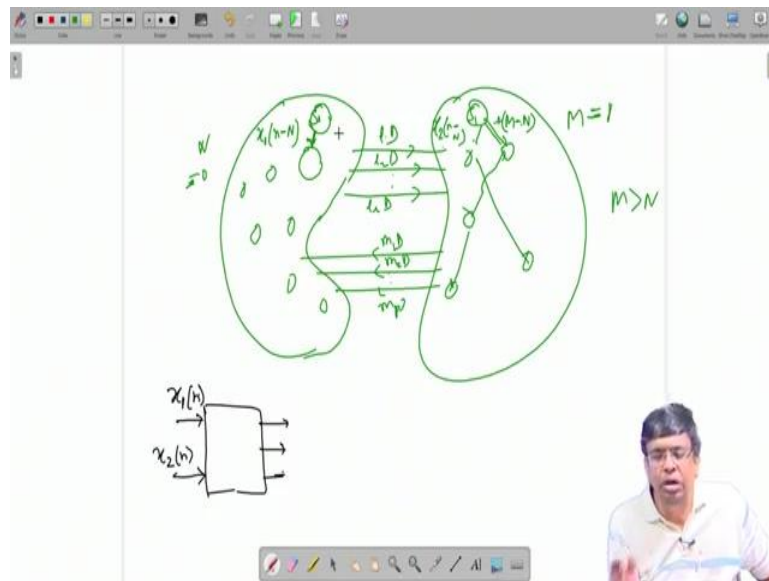
The multiplier in parallel to take one like FIR filter multiplier is parallel through so take one of them and go on following the adder line so one multiplier and so many adders. Then again this also an adder subtractor because subtraction and addition they are equivalent. So this is the line. One multiplier so many adders. Then again a multiplier. Then these are parallel structure so you can go up to either these. And then up to this that is all. And then there is delay or you can try here they are parallel. So this a critical part. I want to bring down the critical part. I want to like FIR filter entirely here also in the adder chain all the additions are coming one after another in the same cycles I want to insert delay here.

If you want insert delay here I will try to do (())((18:23)) so what I will do here I want a delay. So suppose I cut like this. So this is one direction edge this is one direction. And this is reverse direction. I want to have a delay here so if you want delay here. This will become D this will become $2D$ because they are in the same direction and from the opposite direction I should take out one delay but is there any delay in this edge? No. So, it will become negative so I cannot do retiming. Similarly if you want to do here, if you want to insert a delay here you have a cut set hold on. I do have a delay here between these two adders. So I will be cutting like this.

So, I want to have a delay here. This is one direction same direction in this side. So if you want (())((19:45)) to delay here it will become D it will become $2D$ from the reverse direction I have to take out delay, now reverse direction there is one edge, another edge but none of them has delays so I go and take out their delay so on and so forth. So this structure is good but you cannot do any retiming on it. Because there is a loop, feedback loop at the feedback direction there is no delay in the edges, there is a reason I cannot take out delay from that direction and it is leaving the forward direction to bring down the critical part that is I want delays to be between every two adders here like I did in the case of FIR filters earlier to bring down the critical part but the moment I tried to put a delay in this direction and therefore I have to have another delay in this edges, that is the same direction I have to take out delay from the opposite directions from this edges where it is cutting but there is no delay in this direction. That is the reason we cannot do.

So, as such LMS algorithm is not is good but when it comes to retiming and all that to bring down critical part this algorithm is no good. You cannot retain the architecture to speed up that is to pipeline it. For that Elizabeth algorithm is to be changed to another version called delayed LMS. Another version called delay LMS. I will go to that delay LMS but that something more we have to see.

(Refer Slide Time: 21:21)



I come back to that retiming theorem part. For say this special case. I said that if I divide the giving ((21:28)) into two parts. And that time I took one source, single source and I put it happily on this drive. Now I will have two sources one on this side one on this side. And suppose I have got some nodes.

They are internally connected as before. And there are 1 setup in this direction. They are not changing. And another set of using these direction. Some delay 11D, 12D, 1rD I do not know what notation I use that time it could be small m1D, small m2D dot Small m may be pD. So from one direction we have add constant number of delay and from the opposite direction take out the same amount of delay from all the edges make it sure that none of them ((22:40)) to negative but we said we got 10 considered only one source. Now I am persuing two sources, now suppose it is coming to this node and this is coming to this node. If I choose n here and the retiming value m here and suppose m is greater than n.

So, as I know. If I follow retiming theory if this as original delay what about that delay that class n will come. That many delay and here class m what about that delay in this edge class

m will come, if m is larger than n what you can do, you can absorb this n delay in this edge here also n delay. That is if there are two sources ok if you have to take I mean absorb constant number of delay then same must be true for all sources. You cannot give a delay single here and not delay single here.

That is if there are two signals say maybe x_1n here you are getting suppose x_1n . Here I am explaining this, here you have given x_2n . So, x_1n so there is the system where just a minute, over the system that is taking 2 singles it is deriving various outputs. Every node output is an output right. Now if you want to delay one input we have delay the same number as we both that inputs are delayed by the same amount. From that is both the inputs arrive after same amount time together your system outputs will not change every output we be delayed by the same amount it is just delaying the entire thing. But you cannot give one delay here another delay are defeated then you will be completely different.

That is instead of giving x_1n , x_1n coming down and x_2n coming down. If they come after two cycle here after two cycle here inter operation of the circuit also will be delayed by the same two cycles provided both are delayed by the same amount. But if you want to delay x_1n by say two cycles and x_2n by three cycles. Then there will be problem then the outputs will no longer be same as previous one with some constant delay, it is not. So as long as both the inputs are delayed by the same amount entire thing of the circuit also gets delayed by the same amount just delaying the inter operation, so supposed m is greater than n.

So, I delay in this input here. By n here also I delay by n which means here I will absorb the delay. So, my input will be $x_1 n$ minus N. Here again capital N same delay has to be given. So delay that will be left with is m minus n, n will go inside and the input here will be x_2 this also delayed by the same amount. So both the inputs to the intersystem delayed by same amount capital N. Therefore all node outputs all will be delayed by the same amount N.

But I am left with what in this side I mean whatever with the delay that i had here. That remains as it is nothing gets added. Here whatever was to be added plus n that gets reduced by a factor n. As a special case suppose n is 0, as a special case and n is 1. So, before retiming I had whatever delay this is 0 here. Whatever delay I had this will remain unchanged as before and this will get reduced by $m1 - n0$. So it will just added by 1. It will be added by 1. That means this edge will have no addition of delay, this edge will have addition of one delay. Because n is 0 m is 1 as a special case. So whatever delay I had here that class 1 minus 0 that we want. So, whatever delay i had in this edge that will get added by one delay here,

here nothing. This you remember this I will be using while retiming that delay LMS algorithm which will discuss first. Thank you very much.