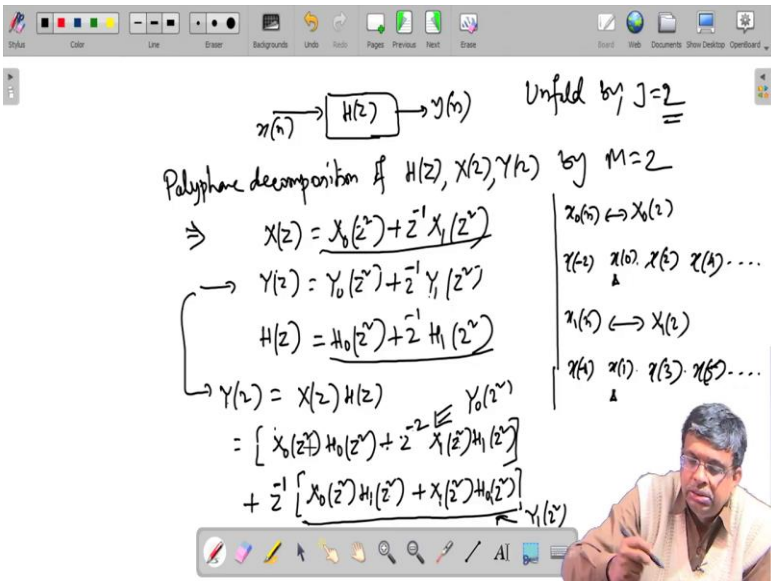
VLSI Signal Processing
Professor Mrityunjoy Chakraborty
Department of Electronics and Electrical Communication Engineering
Indian Institute of Technology, Kharagpur
Lecture – 30
Hardware Efficient, 2-Parallel FIR Filters

(Refer Slide Time: 00:26)



Okay, now, suppose we have got an FIR filter. I want to unfold it. So, you all know this would be called replicated. It will be input buffer, output buffer, hardware complexity will go up by 2 in general. We have seen that IIR filter example also there. Earlier, if earlier I had 1 adder, then because 2 adder, 1 multiplier, 2 multiplier, so hardware goes up by 2. But now using the polyphase decomposition we will show that even if we unfold by j, we can clever, we can play some tricks, DSP tricks by which we can make sure hardware will not go up by 2 but maybe by 1.5, this is the beauty of that polyphase decomposition.

So, we get 2, this j to be 2. If that be, then we pick up this factor 2 from here, we go for H z, X z, Y z by M equal to the same factor 2. I pick up this 2 from here, which means X z, we have already it is of a thing, it will be X0 z to the power M, that is z square, sorry. Since M is 2, M minus 1 is 1, that means between every pair of samples I will have 1 0. So, X of 0 followed by a 0, then followed by X1 then followed by 0, then followed by X2 then followed by 0, likewise. So X of 0 earlier was X0, now also X0; but X of 1 earlier was located at 1, now it will be located at 2Mth position. X of 2 earlier was at 2, now it will be at 2Mth position, 2 into 2, 4; and so on and so forth.

These just from the general description earlier. This will be X z, this and then z inverse 1, we will stop at one because capital M minus 1 is 1 only, the other component. What are the components? What is X0 n which is equivalent to X0 z, what is that? That is this sequence X0, this by X0 n.

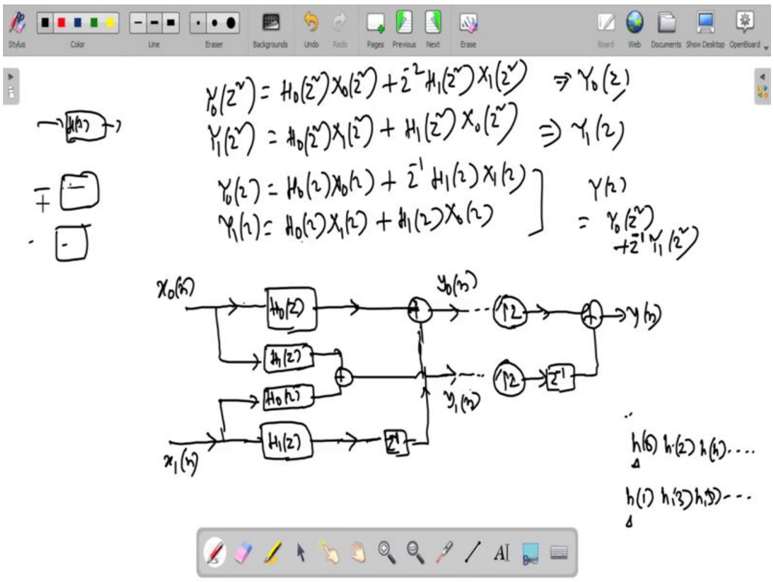
Then we expand by 1, expand by 2, and we conclude to that this between every pair of samples, we insert M minus 1, there is one 0, so one 0 will be here, one 0 here, one 0 here. And similarly, X1 n, other polyphase component before expansion Z-transform is X1 z, it will have X1, X3. Again, if you expand it, it will be one 0 here, one 0 here, one 0 0 here, then you shift it to the right by 1, add with this with a 0 here, 0 here you will get back X n.

So this side before expansion is X0 n, corresponding Z-transform is capital X0 z. If you insert zeros, it becomes a capital X0 z square. Similarly, if you insert zeros, it becomes capital X1 z square then you shift it to the right by 1, so z inverse this and then add, you get back original sequence as Z-transform, this just repetition of what we did earlier. This I am doing for capital X, I can do for Y also. Similar thing, Y0 z square. You all know the meaning of all these polyphase components. Now, what is Y z? Y z is this, but Y z also equal to X z into X z. If will replace X z by these, X z by this.

Multiply X z, then this two direct terms X0 z square H0 z square plus z inverse 1 z inverse 1, z inverse 2, X1 z square H1 z square, this is one expression plus the cross terms, z inverse, you take common, capital X0 z square H1 z square. Now you equate the two, this Y z with this Y z. You see here, in the first quantity under bracket, everything is a function of z square. z square, z square, this also z square to the power minus 1, z square, z square. All are function of z square.

Y0 also here, function of z square. So, these actually expression is Y0 z square, this entire expression. And then z inverse common, z inverse common after that Y1 function of z square. So here again, this one, you see function of purely z square; z square, z square, z square, z square. So this is your Y1 z square, all right. So, in the next page we can write.

(Refer Slide Time: 07:52)



Cross terms H0 X1, H1 X0 outside, I mean, you put it in a bracket multiply by z inverse, z inverse times this plus this; that is your Y z. So, we will generate instead of z square, we will take Y0 z and Y1 z. We will first generate Y0 z, Y1 z. From z to z square means corresponding sequence, you have to expand by a factor of 2. So, z will become z to the power M that is z2. Expanding by 2 bits between every pair of samples M minus 1 zeros that is one 0.

So initially, we will have Y0 z, we will have Y0 z, we will generate which is nothing but H0 z X0 z plus z inverse 1, H1 z, X1 z and Y1 z is your, these two. Ordinarily, how will I generate? X0 z corresponding sequence that polyphase component X0 n, suppose it is available, how available we will easily see. So that is, in Z-Domain you say X0 z that should go to a H0 z.

Similarly here, X1 H0 or X0, if I take the same X0, just a minute. I have suppose the other component which is polyphase factor 1 of order 1, polyphase component 0th order, polyphase component first order. X1 n means in Z-Domain capital X1 z. Not expanded, so z remains z not z square. So, we will have X1 z multiplied by H1 z, so there will be a H1 z filter.

And then it has to be output has to be multiplied by a delay and then added with these output. This will be your Y0 n. And then X1 H1, this is done. Y1 how to operate 1, X0 will be through H1 z. So, we will get H1 z. Same H1 z repeated again. X0 z H1 z, X0 H1, X0 H1 and H1 and X1 H0, X1. So, there will be H0 once again. These two will have to be added. This is Y1 n.

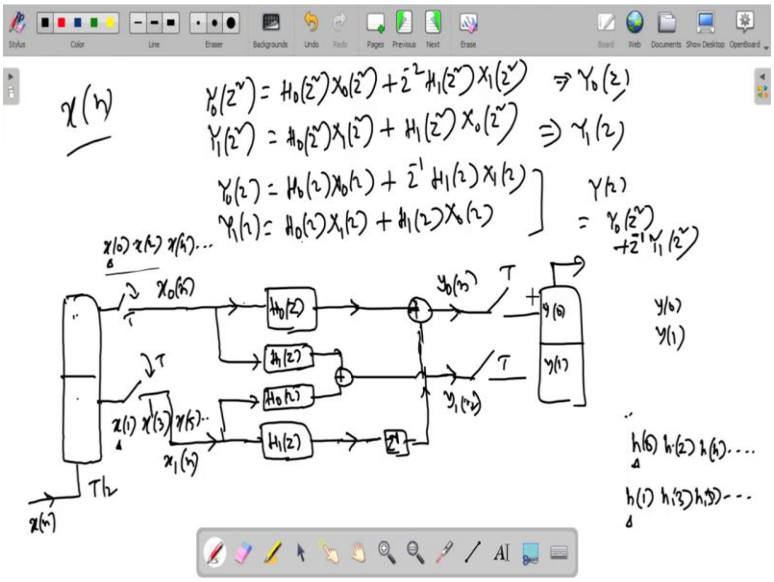
How to get back Y n? Now I know in Z-Domain, Y z means from Y0 z that is here, I have to make it Y0 z square plus z inverse Y1 z square, that means I have to expand this by 2, so that output is Y0 z square, I have to expand this by 2 so that output is Y1 z square then shift it by or delay it by z, one delay that is z inverse then add that is this where a delay, so Y0 z square here Y1 z square delay and then you add, this your Y n. Now, let us look at this structure.

Suppose H z is an FIR filter, say it has got 100 coefficients, then a H0 z will be what? It will take h of 0, h of 2. h0 then, polyphase factor, right? h0 it will take h4 dot, dot, dot; with origin here. And the other filter will be H1 will be h1 h3 h5 dot, dot, dot, dot; with origin here. So, half of the coefficients will be here, if it is an FIR filter of how many terms then? 50, if total is 100, 100 terms. Here also 50. So I will have in each filtering, how many multiplications? 50 multiplications.

h0 into X n, H0 into X n plus H2 into, I mean h, g0 into X0 n plus h2 into X0 n minus 1 plus h4 into X0 n minus 2 and dot, dot, dot, dot. So, there will be hundred terms apart conclusion output. Per output there will be 100 multiplications, sorry, 50 multiplications here. So 50, similarly 50, similarly 50, similarly 50. So here, same for additions. So, 50 50 – 100, 50 50 – 100, so it becomes 200.

So, if I have not done any of these, I simply, I was given a filter H z with 100 coefficients, if I simply unfold it by making it parallel 100 here 100 here, it will be 200. So, by this I am not getting anything here 50 50, 50 50. So if total is 200, here also 200, 100 here 100 here. I am just making parallel versions of this filter, buffer and all that. Here, there is no gain, but I will show how to get gain? How to obtain gain? Probably very interesting, but before that certain basic things you see.

(Refer Slide Time: 18:03)



What is X0 n? This sequence as we know. X0, X, dot, dot, dot. What is this? Maybe put an origin here. What is this? X1 origin here X3 X5 dot, dot, dot.

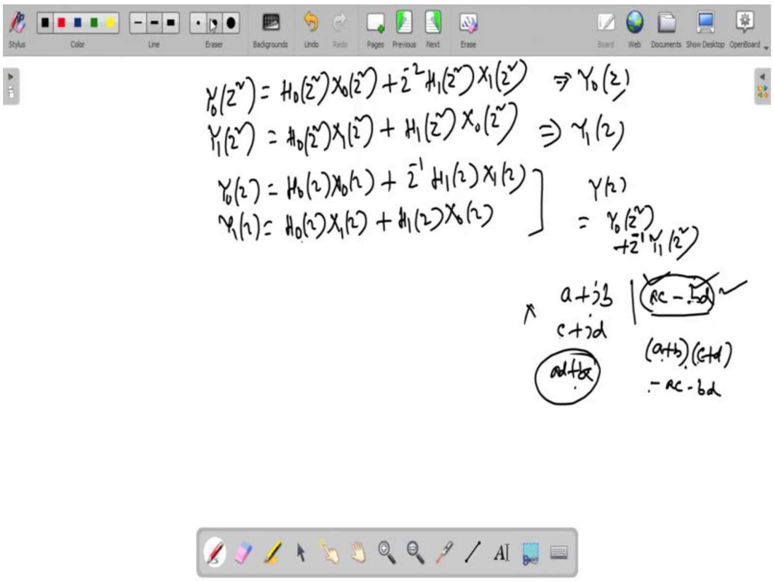
Now, my input is X n. How will I obtain all this? One thing is, in DSP you download it, decimate it by a factor 2 and all that. Similarly, you can generate this, but actually you do not do that. If I have that same buffer which I used in unfolding, X n and it is T by 2 down, T is the sampling period, period, clock period of the system. T by 2 sorted period, faster rate of the input.

So, 1T by 2, X of 0 comes next T by 2, X of 0 here. X of 0, X of 1 here and then the switches close. X0 moves out, X1 moves out. Then switches open next time in the first T by 2. X2 comes here than X2 moves here, X3 comes here, X2 X3 again they move in. So, X0 X1, next time X2 X3, next time X4 X5. So, by the simple switching arrangement, switching and buffering arrangement, we get these sequences here, so no need to have decimator and all other things.

Similarly, here as such I know if we expand between every pair of samples we will get 1 zeros similarly one 0, then shift by 1, then we add, the sequence will build up. But I do not need all these things, because what I will do is simply, what we will do is unfold it. So in 1T by 2 is Y0 0. What is Y0 0? Y0 0 is Y0. What is Y0 1? Y1 0. Y n0 is Y1, is not it? Y0 0 is nothing but Y0. What is Y1 0? It is Y1. So, in one clock Y0 Y1 comes and they go out. So, first Y0 followed by Y1. Next clock, Y0 1 which is Y2, then Y1 1 which is Y3. So, Y2 Y3, they move out.

So, you get the sequence back Y0, Y1, Y2, Y3, like that. You do not need to have delay, I mean, all those decimator or expander and all those things. It is very interesting. Now, here I did not have any gain, but I can have some gain, so let me erase this first.

(Refer Slide Time: 21:50)



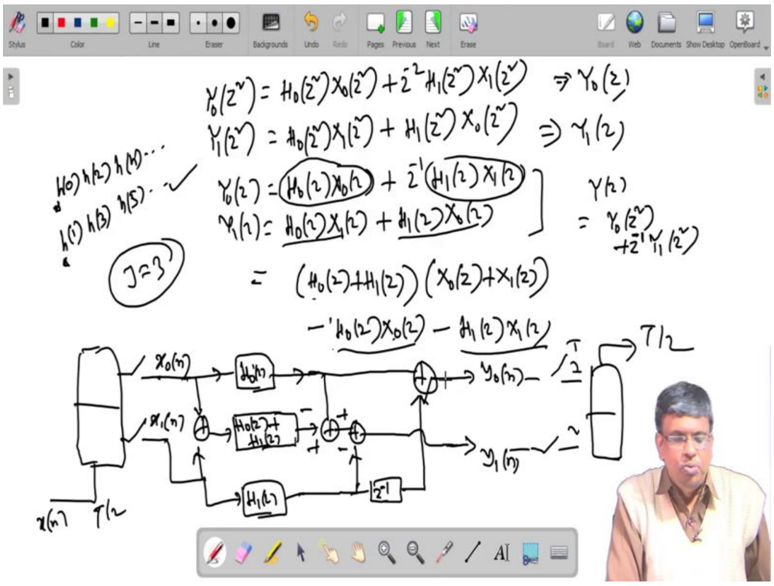
Now, before I get into this thing, I will tell you something very interesting we just practiced today. Suppose you got 2 complex numbers a plus jb, c plus jd and you want to multiply them in hardware, now there is nothing called j, j is just for reference you have a and b one pair, c and d one pair. You treat this as real part, imaginary part; this as real part, this as imaginary part. So, your product will be if you were to multiply the two, a c minus b d, this your real part and a d plus b c, this your imaginary part. So a c minus b d plus j times this. This what you would do. How would you multiplier?

a into c, 1; b into d, 1; a into d, 1; b into c, 1. 4 multipliers and 2 addition or subtraction, subtraction is another form of addition, so 2 addition. So, out of 4 multiplier, I can bring it down to 3 at the cost of some extra addition, but additions do not really create any problem, they hardly require any hardware, multipliers do. The way it is done is this, you evaluate this a c minus b d and then calculate this a plus b, no j here, c plus d. So, a plus b one extra addition, c plus d, one addition, one multiplication.

So, here I have got a d one multiplier, b c one multiplier. So, 2 gone, here third, then minus this fellow, this fellow I bring back here, which is already calculated, that is a d minus b c, no need to calculate them, this already calculated here. So, if you subtract out a d and b c from this result a c, sorry, there was a mistake. You calculate this real part, then a plus b, c plus d from the two subtract a c and b d, see you are calculating this, you are calculating this to obtain a c minus b d you have to do this multiplication a c, you have to do this multiplication b d, so, those results are available. So, now, to obtain the other result that is a d plus b c, the imaginary part, this is real part.

Real part you calculated by finding out the product a c, finding out the product b d and subtracting. Now what we are doing we are doing? We are doing a plus b, one addition, c plus d, one addition, multiplying. From this product, subtract out a c, which is calculated, subtract out b d, so only the cross terms remain b c and a d. So, you have got 1, 1, 1, 1; 4 additions and only 3 multiplications. One multiplication here, one here, one here. So, by having 2 extra addition you can bring down the number of multiplications by 1 and this is how it is done in practice today. Same theory you will bring it here.

(Refer Slide Time: 25:39)



I will calculate these as it is, Y0 z as before and Y1 z, I will calculate as, forget about the z inverse. It is like a b, okay c d. So, I will have, it is like a d plus b c like that. So, I will have a H0 z plus H1 z times X0 z this. So, already I have, minus I will take this product, minus I will take this product. So, the two cross terms H1 X0 here and H0 X1, they will remain. This is already evaluated, already evaluated no need to revaluate. So here, so these fields bring down my hardware cost easily, which we will see now.

So as before, I have buffer, we have already seen this give rise to X0 n polyphase factor about our 0, this we have already seen. It is coming at T by 2. T is the clock period of the system. I need to have H0 z X0 z, so let it go to H0. I need to have H1 X1, so let it go to H1. As before, Y0 z, no change. This my Y0 n, simple? Z inverse X1 H1, X1 H1 and H0 X0, H0 X0 that is Y0 n.

Other one is what? First another filter whose transfer function is neither H0 nor H1, but H0 plus H1. And this input is neither X0 nor X1, but their addition. So, these two filters will get added there the extra additions, they are coming, coming in multiplied. From these I have to subtract out H0 X0. H0 X0 is here, so I subtract out from that further I have to subtract out H1 X1. H1 X1 is coming here. This is by Y1 n and then as before I will give it to a buffer. I will generate Y n, absolutely fine.

Now here you see how much is the hardware increase. This is H0 n H0 z. See, for example, I had 100 coefficients, this will have roughly half, 50. H1, half 50. How about this? Interestingly, this also will have 50 because a H0 z and H1 z means what? H0 z is h0 followed by h2 followed by h4 dot, dot, dot. And H1, with origin here. And this is h1 with origin here, h3, h5, and dot, dot, dot. And you are adding. So 50 here, 50 here, but added also if we add the two sequences, resulting sequence also will be having the same length 50. So, this will be a filter with 50 coefficients only. So, now 50, 50, 50, 150. Earlier it was 200. This is the beauty of it, just by using these tricks.

This is a very simply one. In the next class I will consider j equal to 3. And obviously, I have to consider the polyphase factor of, polyphase decomposition of order 3, which is more complicated and cumbersome, but we will just give the results. So, you get a feel that I mean, in the case of FIR filter hardware increase can be brought down. Why not IIR filter? IIR filter, even if you take the H0 z, original H z, original H n in time-domain had infinite number of terms because IIR. If you decimate by 2, if you take 0th order polyphase component, even that also will be infinite in length, H1 will be infinite in length, and their summation also infinite in length. So IIR, III, IIR, infinity plus infinity plus infinity.

So that really does not change. In the case of FIR, originally 100 means 50 here, 50 here and another 50 here, so 150. That is all for today. See you next time. Thank you.