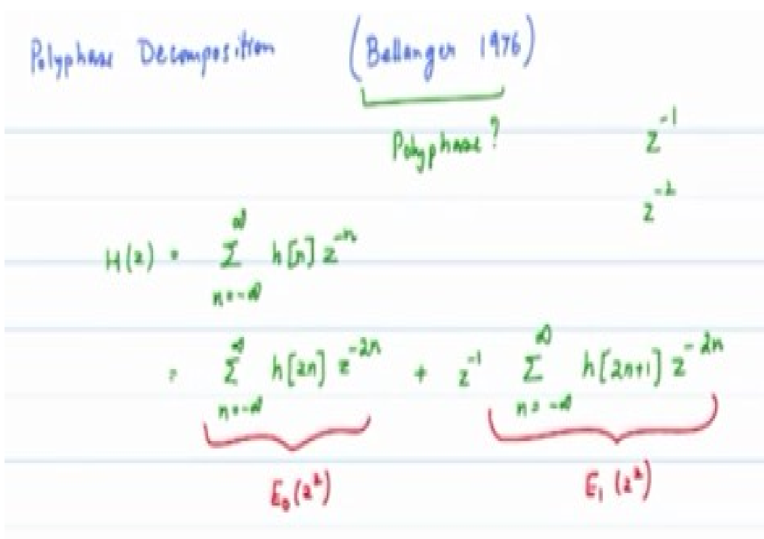
Multirate Digital Signal Processing
Prof. David Koilpillai
Department of Electrical Engineering
Indian Institute of Technology – Madras

Lecture - 14 (Part-2)
Polyphase Decomposition Continued – Part2

Now I would like to move on to our discussion on the polyphase decomposition, which we had already described. I would like to pick it up from there and then move forward and develop that further okay. So the polyphase decomposition, we introduced in the last lecture. Let me give a little bit of background to it, since this is the first time we are formally introducing it, because it is a, so polyphase decomposition.

(Refer Slide Time: 00:48)



Polyphase decomposition : probably the result that really opened the doors for multi-rate signal processing to come into widespread use was first introduced by a French researcher. His name is Maurice Bellanger which in the year 1976 okay. So that was probably the opening point and his observation of multirate and polyphase was in a completely different context, but we have translated it into our notation.

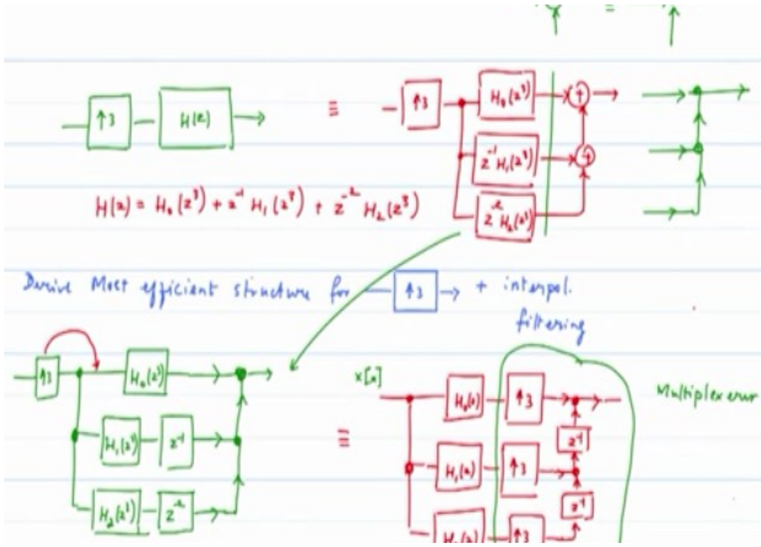
So maybe at some point some of you who are interested should take a look at more Bellanger's original paper and try to see why he used the term polyphase, maybe a hint to that is, what is the magnitude response of this element? Unity, all pass filter right. It is Z inverse is an all pass filter, but it has got a phase okay. Now Z-2 also an all pass, it has got only phase okay. So basically he was looking or thinking about some mechanism by which he will translate a filter into sub filters which have got these components okay.

Effectively z inverse, z-2, z-3 which is nothing but the polyphase decomposition and the polyphase. So that is where it is coming from, but he came about it in a very different way and of course as in all inventions or discoveries, the original person you know thought of it in a very different way, but then we are able to give interpretations that are interesting and so the example that we looked at in the last lecture.

If you have a filter H (Z), summation n equal to minus infinity to infinity H (n) Z power –n, we said that we could write it as two summations : n equal to minus infinity to infinity H (2n), grouping all the even coefficients or the even powers of Z, Z power -2n + then the odd terms. I will factor out a z inverse, within says I get n equal to minus infinity to infinity H (2n+1) z power -2n, that additional.

So you can see how the polyphase in this case just another term with just another phase is emerging and this is what we introduced in the last class. We called this as E0 of Z squared. We call this as E1 of the Z square right. That is what we said and of course if you had to apply the polyphase decomposition with the down sampling, we showed how this structure could be utilized.

(Refer Slide Time: 03:51)



Now if you remember, we left the discussion with a task to apply the polyphase decomposition for the up sampling case. So maybe we just write that down just for completeness sake okay. So H (Z) is being written as either E0 or H0, does not matter polyphase components of this form. So the this structure can be re-written as upsampling by a factor of 3. Implementing this structure, this would be called a parallel form implementation of a digital filter.

So whenever you have a filter, which has been written as to some terms that are added together then it is a parallel form implementation. So H0 of Z cubed in parallel with Z inverse H1 (Z cubed) in parallel with Z-2 H2 (Z) cubed and all of these terms will get added together okay. Now this type of structure, we are going to see again and again and again. So this writing these plus signs so many times is actually quite cumbersome.

So what we would like to do is actually introduce notation, which makes it makes it simpler for us. So what we are going to do is, this notation I think you are familiar with some digital filters. What that means is, that signal is getting split into 2 branches. The same signal is traversing on the 2 branches. Now on the output side, so if I have 2 signals which come like this, it is an adder. In multirate structures, we often do the following. This is also an adder okay.

So you do not have to keep drawing this plus signs. So this structure could be written in the following form with the proper arrows, that means the 2 lower branches get added and then they add to the top right. So this these are this portion and this are identical. So again rather than keeping adding those multiple times, we will do that okay. So now of course, we would like to use the noble identities, but before that this structure can be re-written as instead of Z inverse H1 (Z cubed), interchange the order.

Those are 2 linear blocks. So up sampling by a factor of 3 using my convention, the convention this is H not of Z cubed the lower branch is H1 (Z cubed) followed by Z inverse, signals to be added and then the other branch is H2 of Z cubed followed by Z power -2 and this signal also to be added. So please draw the arrows. Once you draw the arrows, you can interpret them correctly, which way the signal is flowing, which signal is getting added, which 2 signals are getting added at each point.

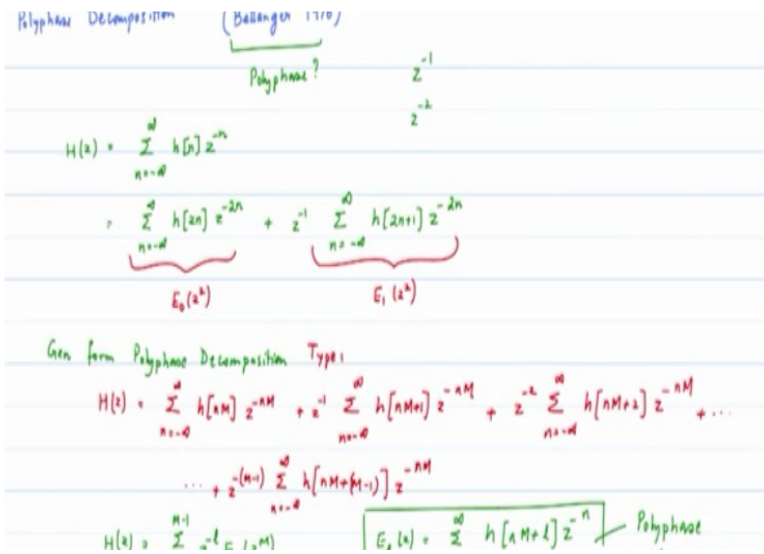
And this is also a node on which the signal just splits into multiple branches. Now up sampler inside of the where the signal splitting occurs, you can see that there is a opportunity for us to apply the noble identities. The up sampling followed by filtering where the filtering has got a power, Z to the power L where L is the up sampling factor. So using that we now have the following structure you have H0 of Z followed by the up sampler 3. The second branch has got H1 (Z) and there is a Z inverse.

I can move it past the H1 (Z cubed), but I cannot move the up sampler past this Z inverse. So this is up sampled by a factor of 3 followed by a delay. So the delay, I am going to capture it on the output branch Z inverse okay, and of course the third branch is H2 (Z) followed by up sampler, followed by Z that power -2. So already there is a Z power -1 on the other branch. So I will just add a Z power -1 onto this and please draw the arrows in the right direction.

So that we know which way the signal is flowing and this is the efficient, probably the most efficient implementation that we can get of this up sampling interpolation filter and of course this portion of it is a multiplexer. Multiplexer with the arm going in the anti-clockwise direction. So this is a multiplexer operation. So in other words what it says is take the data, take input X (n), filter it by sub filters, just like you would do an LTI system, interpolate them right.

I mean filter them convolution, and then multiplex the data and that is the most efficient because you are not doing any operations on 0 valued samples. All of these 0 valued samples and their effect on the structure is actually taken up into the multiplexer. So there is an efficient implementation that is available to us okay.

(Refer Slide Time: 09:33)



So now we go back to our discussion, what is the most general form of the polyphase decomposition? General form of polyphase decomposition, we actually refer to 2 types of polyphase decomposition and they may seem very similar, but there is a reason why they are. So we call this as type 1. Most of the time, 95% of the time when we talk about polyphase decomposition, we are talking about type 1 polyphase decomposition.

Type 1 polyphase decomposition says take your filter, Z transform and split it into sub summations. So n =-infinity to infinity H (nM) Z power –nM. So I am taking all the m-th coefficients in the first summation. The second summation will be Z inverse summation n equal to minus infinity to infinity H (nM+1). All of them offset from the M-th position by 1 unit of time. Z power –Mn. Please note or Z power nm okay.

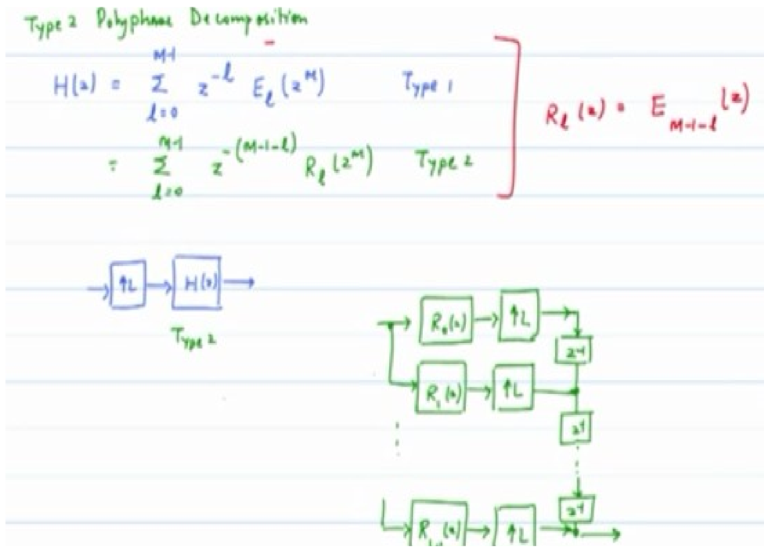
Next is a summation with Z power -2 summation n = - infinity to infinity, n equal to minus infinity to infinity. This would be H (nM+2) Z power nM, okay and so on. The number of sub computations will be since you have divided, split it up in the multiples of M, there will be total of M. The last one will be a summation n equal to minus infinity to infinity H (nM +M-1). That is the last of the when you do the subsequences.

This will be Z power –nM, still with a leading polyphase term okay. This will be plus Z power minus M-1. So notice that you have power of 0, power of -1, -2, -M-1. So these are so, if you were to label these subfilters in the following way, we can write down H (Z) as summation L=0 to M-1. Basically summation of M subpolynomials Z-l El Z power M, where El (Z) is defined as summation n equal to minus infinity to infinity H (nM + l) Z power –Mn.

I think I wrote the wrong one. This should be an nM, Z power –n. Please be careful about the definition and no confusions because when it is Z power M, it is different but now this is the definition of the polyphase component. We refer to this as the subfilters or as the polyphase components themselves. Basically what this says is whenever there is a structure that introduces these polyphase decomposition blocks, then we have a efficient way of implementing them.

Again we have just now looked at a case where L=2, we saw a case when L=3, of course you can see that there is a generalization that is easily possible. Now I would like to introduce the second version, which is not used as frequently, which is the type 2 polyphase decomposition. So it is useful for us to just keep that in mind and occasionally when we use it, I will highlight it to you that it is a type 2 representation.

(Refer Slide Time: 13:32)



Type 2 polyphase decomposition, again it is for certain introducing certain symmetries in the structures. So notationally, it is exactly the same. So I take a polynomial H (Z), split it into subpolynomials exactly like before l=0 to m-1, Z power -l and then the sub polynomials. If I write it as El Z power M, this is type 1. Now type 2 is summation l=0 to m-1. I am writing them side-by-side, so that you will see the difference Z power –m-1-1* Rl Z power m.

This is type 2, okay. So what are the range of the polyphase components. It can go all the way from Z power 0, Z power -1 to Z power minus –M-1, in both type 1 and type 2. So which means that El and Rl are, they are basically the same subpolynomials, but they have been index differently. So that is the only difference and the result and that observation can be actually captured by comparing these two.

You can write down the fact that Rl (Z) is nothing but EM-1-l (Z), okay and now probably the question that is on your mind is you know what is the reason for even introducing the type 2 polyphase decomposition and I will answer that when we start looking at the structures and it will become easy or obvious, why we have to do, why we saw that. Now before we go any further, I would like to maybe just draw a type 2 filter implementation just for visualization.

And again the reasons for using type 2 will become apparent once we start doing them. So I would like to do the following, up sample by a factor of L followed by H (Z) okay. You may say well just now we did. Did not we do this? Yes, we did it with type 1 polyphase decomposition, please do type 2 polyphase decomposition. Type 2 polyphase decomposition again I would not go through all the steps.

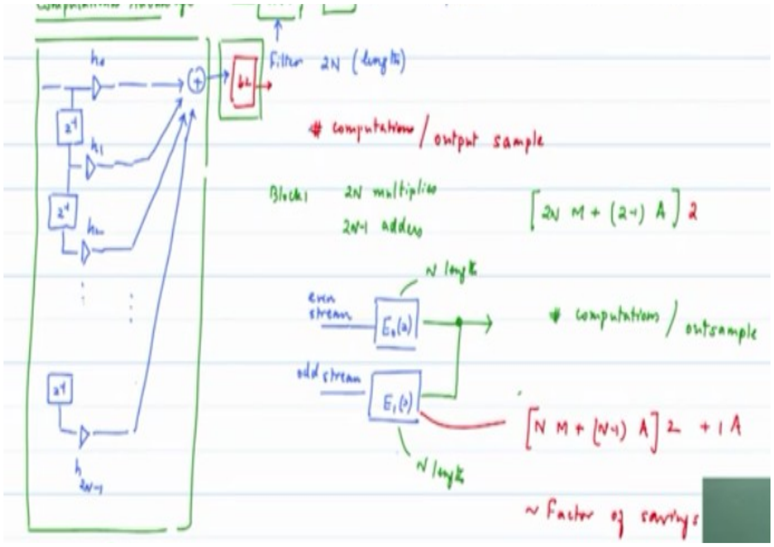
What you would get is R0(Z) second branch R1 (Z), the L-th branch is RL-1 (Z), then the up sampling factors, up sampling by L in each of those branches. There are total of l branches and here is where the difference comes. Previously the arrows were pointing upward towards the summation; now because R0 has got Z power L-1, so which means that the delay units are still there, but the arrows are now pointing downward.

So this gets added to this followed by another delay chain dot, dot, dot and then the last one comes with the Z inverse dot. So notice that the arrows are downward on the upper, on the branch on this side and if I changed the order of the multiplexer, of course this happens. So basically it just says that the commutator arm is rotating in the different direction compared to previously, but this is a form that results when I use type 2 instead of type 1 okay.

So now you can see why type 2 has been introduced is so that you can avoid having advanced elements if you want the structure because if you had advanced operations, then what would happen; these would have to go in the upward branch and the upward direction now. You can now see that there is a way to use that, but again it will become apparent when we start looking at more of these structures.

So for now I hope the definition of type 2 is clear and then the implementation of a filtering operation with type 2 and this is something that, once you are comfortable with this, we can build on. Like as I mentioned type 1 is the predominant structure that we are going to be working with.

(Refer Slide Time: 18:23)



So here is the computational advantage, computational advantage of using the polyphase structure. Computational advantage; so I want to look at the simplest case where I have H (Z) followed by a down sampling by a factor of 2. Now what is the what is the benefits of using polyphase decomposition versus the direct form. So polyphase the comparison is between polyphase structure versus the direct form.

The direct form is a form that you would have looked at. Basically it is a combination of delay elements multipliers and then the adder. So let us take the case that H (Z) is a filter FIR filter of length 2N. This is the length of the FIR filter, so which means that there will be 2N-1 delay elements and 2N multipliers. So the implementation of the direct form would be first multiplier is H0, input goes through a delay element.

Second multiplier is H1 delay element multiplier H2 dot, dot, dot, the last one comes through a delay element Z inverse and gets multiplied by H (2N-1). Now these are the and all of these signals have to be added together. This is the direct form implementation. I am sure you would have studied this in basic DSP. So this is the direct form implementation of the filter alone, but then I also have to have the down sampling by a factor of 2.

So in the direct form, I can of course move the down sampling to the other side, but you know what really does not help much, because I end up having to compute the multiplication. So let us look at how many computations I am doing per output. So number of computations per output sample, computations per output sample. So think of it as a hardware implementation. You are counting in a hardware implementation, where the first block is doing the direct form filtering.

First block is doing the direct form filtering and then the second block is doing your sampling, down sampling. So computations per output, so this is the step 1 or block 1. Block 1 at every clock cycle what is the computation that has happened? Block 1 every clock cycle, there will be 2N multiplies, you agree with that. All the filter coefficients will multiply whatever is there in their input register, 2N multiplies. How many adds? 2 element adders.

There will be 2N-1 adders. It effectively that is like you know the last 2 get added and then the previous 1 and the previous 1. So there will be 2N-1, 2 input adders. So this is at every clock cycle right and then, but you are looking at computations per output sample. Block 2 says compute, compute, throw away 1 sample, keep that. So basically this 2N multiplies +2N-1 adds *2 is the number of computations per output sample.

Basically I computed twice, the block 1 would have done 2 computations block 2 says throw away 1 of those, keep 1. So effectively at the output side this into 2. Now had you done it using the polyphase structure, what would have been the implementation? It would have been a demultiplexer at the input site right. So basically you would have done a polyphase E0 (Z). The odd stream, the even stream would go to this side. Even stream will go here.

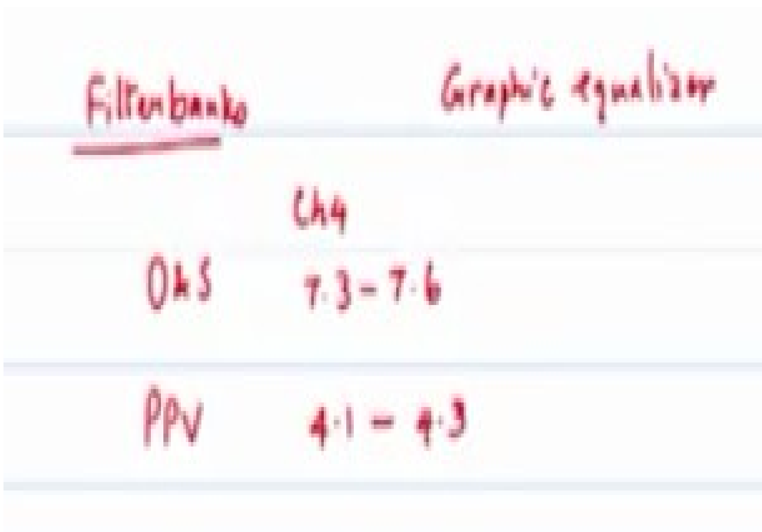
The odd stream goes on the lower branch that will be E1(Z). Now E0(Z) has how many taps, N taps, because H (Z) has got 2N. I have split into 2 polyphase components. This has got length is N. So this 1 also has got N length. So these 2 have to be added, output of these 2 get added at every instant of time and that is the output. So per output sample, number of computations per output sample, per output sample, I would like to do that.

Each of these blocks basically requires N multiplies +N-1 adds. There are 2 of them and then there is 1 more add at the output right. So this but the one that kind of scales with the length of the filter is the first term. So this is the output. Now if you actually and this is the output per output sample and if you compare the 2, you will see that there is roughly a factor of savings is 2, savings 2.

Now if you are doing this for down sampling by a factor of M, this would become a savings of m and of course you can show that the same is true when you if you are doing the up sampling as well. So the advantages of the polyphase implementation actually come through very strongly once you are in the domain of the hardware implementations.

Now what we would like to do next is having given the broad framework, we now would like to talk about the filter banks and for this I would request you to start reading a little bit ahead, because this is a part that is actually interesting, easily, easily understood and so it is helpful for us.

(Refer Slide Time: 24:51)



So filter banks is the topic that we are going to be looking at. Filter banks is nothing but splitting a signal through a set of filters. So application such as your graphic equaliser, in the stereo that you listen to. It splits it into high pass, low pass and medium frequencies and then boosts or. So that is a filter bank. Basically that is what it is doing. So example would be a graphic equalizer.

So filter banks come in various forms and very, very useful, a very powerful tool that we get from the multirate side. In this context, please read Oppenheim & Schaffer, chapter 7, no sorry chapter 4 sections 7.3 to 7.6. That is the part where it actually covers the multirate, the noble identities, polyphase decomposition and gets into filter banks. Also look at P. P. Vaidyanathan’s book, multirate signal processing. There it is chapter 4.1, section 1 to section 3.

Again these are the parts that we are going to be looking at. It builds up a framework, as I told you which is a very powerful framework extensively used in communications as you can see every time you do an FFT, you are actually doing a filter bank operation. So that is the power of this. We will build and develop this in the next class. Thank you.