Real - Time Digital Signal Processing
Prof. Rathna G N
Department of Electrical Engineering
Indian Institute of Science - Bengaluru

Lecture - 20
Lab – Real Time Audio Output through FIR Filter

Welcome back to real time digital signal processing lab. So, in the previous class, we saw how the MATLAB is going to run our filters. Today, now we will see that how board is going to respond to, whatever the same corruptive voice, what it is going to be given to board also, and we will be designing the filter and we will see that what will be the response.

(Video Starts: 00:54)

So, as we have been telling that it is the C code what we will be writing, so, which is going to be ANSI C and we will be using the aic3106 is the codec chip on the L138 board here we are using C6748 board, so, which uses this and then we are including the cof file here I have given it as 200 order. So, you will be seeing that this is what I will call it as I have given number of coefficients basically, although it is the same frequency whatever component what it has been taken in FIR filter in the MATLAB same components what you will be seeing it.

So, you are seeing that h of N, this is defined continuously. So, if you want to put it in this format, so you can refer to any of the real time signal processing books or they will be giving how to convert from single line to this. Otherwise it will be if you have given a return it we will take it as a next line. So, we want to arrange it in this way so that we are not going to lose the thing. So, either we can have the here it is 55th order, how it can be represented the coefficients what it is shown.

So, one way of doing it, either we can create a dot h file and then put all the filter coefficients here. And then or we can provide in main dot C itself, the value of it. So, you will be seeing both the ways being used. Now, what we will be telling is the initialization of it that is a codec has to be initialized. So, here we are using the polling technique, there are 2 ways, actually 3 ways of it. Today we will see 2 ways of doing the same thing.

One is using the polling technique, as I have mentioned in my theory class in the architecture, the polling method CPU will be going and then checking whether there is a new data which is present. That is how it will be working on it? So, it will go and then check if there is any data

in LCDK line input that is codec input, we will be using that. And we can specify the sampling frequency what I am going to use it, in this case we have designed all our thing is in 8000, 8 kilohertz, so we will be using FS 8000 and will be looping continuously here.

Because I will be getting the data and then I will be working on my filter. And I will be sending the out onto the codec chip. So, you will have to initialize y of n is my output which is going to be initially 0. So, you will be getting the first sample $x(0)$ is always I will be taking the first sample from my input data that is whatever is connected. So, this is input sample that is from ADC what I am going to take it and then I will be looping it you will be computing filter output basically and this is delaying data in circular manner.

So, for $i = 0, i < N$ actually we will be doing $i++$ and then you will be computing h of i is the coefficient into x of i whatever data is coming in. And then you will be shifting $i = N - 1$, $i > 0$ $i--$ your $x(i)$ will get $x(i-1)$. And then the output $y(n)$ will be whatever you have stored in $x(0)$ what you have computed. So that is how we will be computing for every sample until the order of the filter. So, then what we are going to do is we can directly output left sample.

Because we have taken in from the left sample audio in has both left sample and then right sample, here we have taken only 1 input sample data from left channel and then we are processing and then putting out on the left channel itself. So, I can call it as this thing uninitialized interrupt 16 basically, $y(n)$ what I will be going to output DAC, since the volume is going to be a little feeble. So, what we have done is we have shifted it by 4 basically to increase the amplitude of the signal.

So, we will be putting it to output DAC this value. So, this is how we will run FIR filter. So, we will as we know, first I can do build debug for project this is going to take saregamapa itself as input like MATLAB and then you are seeing that it is up to date because I have already run it if there are any errors. So, you have to take care of them. So, now I can go and then build this project so we have created the project.

And we need the codec initialization and then it is h file also and then linker dsp dot cmd will be specifying what how the memory mapping is going to happen just to give a flavour of it.

So, you will be seeing linker dsp dot cmd, so you are specifying stack, heap and then memory. So, I have a RAM and then there is some shared RAM and then some external RAM also their length and their origin is given in hex basically in this case, and what are the sections I will be using is some of it is dot text wherever it appears, described your comment and other things text has to lie and constant if you have declared that define it will be lying here.

And this is the bss section basically where the entry point and other things is going to have and dot far will be some memory which is away from the thing which is going to be stored in this dsp 12 RAM and then you will be seeing even the stack and then switching and even dot data it is getting stored here. And initialization part of it we call it as where the entry point to program is going to be that we call it as initials thing which is here.

So, if there are system memory is also in this and then you will be having i o whatever also the location is in that and then vectors also whether it is polling or interrupt driven. So, the vectors have to be specified where it is going like this the memory and then you will be calling that external RAM as your external RAM. So, this is how you are memory storage is going to be in your DSP processor. So, we will see that how the thing is going to run in processor.

So, you will be seeing that it is doing that debug, so, you are seeing that it is going and then loading on to the board. So, it has finished and you will be seeing that this is ready to run, either I can do single stepping or I can run continuous since if it is not codec based you can do single stepping and then see each step what is the value which is getting computed. So, here you have seen that it has come down to entry point here in main is entry point.

And you will be seeing h and x values where they have been defined and h value has already gone it is a float which has been taken in this case if you are taking it as quantized one you can go and then take them so these are the values of coefficients basically and these are the input initially what it has. So, what we will do is here it is sa re ga ma with noise what I will be putting it so we can run this from here, which is the input that is a through the output of laptop jack you can connect it to the input of the board.

And then the output we are connecting it to the speaker for you to hear the thing. So, this is running continuously. Now we will see what is the thing is going to happen when I run my board. So, what it says is if you are not going to hear anything, you have to go back and then

reset the board and then you have to restart it, so you are unable to hear any of it. So, we will see again do the debug whether we can hear something you will be seeing that even after stopping get the board is running.

Because we have given it is a while loop until I reset my board the code will be running on the board. So, you will be seeing the complete FIR filter you are hearing the noise because even in the passband we have little noise, so, as I mentioned in while running my MATLAB code, so, this is how there will be a little noise also which is going to be amplified in boards. So, I had to go for very high order filter here although it shows it is 200 order FIR filter.

Still I may have to increase but most of the time as when we took up the stability of the filter for FIR although we said it is stable, but at least up to 256th order it will remain same, but beyond that, it may become unstable. So, better to design all your filter orders for FIR in for the board less than or equal to 200 what I say my students not to go beyond that, somewhere around 120 to 150 what we do it? So that is what you saw that little noise is left out.

Now we will see the same thing whether I can use the interrupt driven so what is the difference between polling and then interrupt driven we will see in a while, it is the same code what I will be having it. So, you will be seeing that same sa re ga ma FIR dot c. So, I can use the lp55 coefficient I have not renamed it or I can use this as renaming for both the coefficients are there I can try on both the thing so I will be taking this itself 200 dot coefficient whichever previous one what we have done.

So, you will know that there is no difference between polling and then interrupt driven. So, here what it says same thing will be initializing the codec. But what is it I have to write the interrupt service routine here, because I am going to interrupt the CPU whenever there is a new sample which is coming in then what the CPU is going to do is there is a location where it is specified here it is interrupt 4 is the one being used. So, you will be seeing that there is an assembly file for vectors interrupt dot asm.

So, you will be seeing that it is calling all vectors I have the entry point is underscore int00 were your main basically we will be entering it here and then there are predefined vectors here 1 2 3 which is not accessible to the user and then interrupt 4 is accessible to the user and rest of it is not accessible to the user. So, you will be seeing that this is the entry point and where

they will be branching you will be seeing that this is the entry point and then how it is going to branch.

This is the assembly code sample in ti processor what you will be seeing it, this is store word basically B0 in B15. This becomes a stack basically and then you will be moving constant here from this B0 lower portion of it and then a higher all the registers are 32 bit so we will be loading the address 16 bit lower version and then higher here and then you will be branching wherever this is pointing 2. This is the pointer what is given and then this is usually we as I said be B15 represents stack pointer where the entry point has been given.

And then you will be a loading back wherever you have stored the where was the entry, point that back so that it can return to its safer place. And you will be seeing that there will be some NOPs 2 account for the delay. So those who are interested in assembly programming, if time permits, I will show one of it, how I can exactly count, how much clock cycle it is going to take? So, this is how your entry points are going to be where it, it says it is reset.

As you can see that entry vector, vector 1 is non maskable interrupt I think there should be some trigger in your background, what is happening and then the store result. And this is an interrupt service routine what we have it so other server kept it as dummy. So, this is where our interrupt routine is going to start. So, the CPU will jump and see whether there is any data and then it will take it and then process it.

So, what we are going to do is we will be writing FIR filter code in the interrupt service routine. So, x0 what we will be taking the sample from this thing codec chip and then we are going to process it just like polling. And then we will be doing the delay lines assignment circular buffer what we have to call it, we are doing that and then we will be outputting this sample same way as the previous one right shifted by 4 bits basically.

And then here also we are using the sampling frequency as 8 kilohertz and these are the ADC and DAC gain basically we keep it as 0 dB and then LCDK line input is the codec input and output and we will be looping forever. So, we will see that this how it is going to behave whether we are going to have a different output or the same way what polling gave so we will be looking at it. So, only the difference in polling as I have been mentioning CPU goes and then checks for the data.

So, we will see whether thing is running so it is still running. So, noise whatever we have given it for the polling, it is still running so we will see whether. As you have seen it after breaking it also it has not stopped so interrupt driven is also going to behave in the same way. So, this is how one is with the polling and with the interrupt what we have done the thing so now we will move on to IIR filter.

So, here are it is a notch filter what it has been designed like the previous one so we will see that how this? This we call it as interrupt driven only what it has been taken. So, if you want to run polling, you can very well use this as a sample this thing the first one there you can go and then put the code. So, you will be seeing that this is my thing LCDK initialization. Next, I am going to define my constant it is here int coefficient what it has been taken.

So, for 900 hertz to remove that these are the coefficients what it has been given. So, you will be seeing that and then for the 2700 hertz, what I have to remove it, these are the coefficients values, which have been designed in MATLAB and then we have taken the coefficients and then we had to this thing sections what we will be having, so $x(1)$ because it is the 6th order filter. So, you will be seeing that $x_1$to x_6, what you are calling it as, signed integer. So, you are seeing not a float here it is running in integer format.

So, this is how you will be generating the thing? So, and then $y_1$to y_6 also has to be defined for both numerator and that denominator you are initializing them separately. Now since it is that interrupt driven. So, you will be, service routine has to be return. So, some of the variables what we need it for IIR filter what is shown here. So, output there are temporary files what it has been created.

So, you will be seeing that for 900 coefficient 900 common input so and then input in this case is the sample which is coming from my codec that is ADC basically for sample what I take it in input, and I will be passing it. So, next one will be passing it as you can see that output what we have to pass it to temp 1, you are doing the filter, IIR you are going to pass it through that, that is the second stage of it, this is the first section, this is the second section which is running and this is the third section.

So, after that, because it is a 6th order filter what you have it as I have mentioned earlier, so the other for the 2700 hertz, what you will be doing? You will be taking this from the input of the section you will be feeding into this, this is what we call it as series what we have fed in the thing, then from this output will be feeding to this one, fine, output of temp 3 goes to temp 4 which is eliminating my this thing 2700 hertz sine wave what is peak I we call it as a noise.

So, this is finally the output temp is going to have the last stage output, then we will be doing output left sample 2 that is codec DAC output we will get from the temp that is left sample what it is going to be. So, now will be after that we will return from the service routine this is my main function. So, which is also running at 8000 and then line in now we have to see that what is my IIR filter is going to do because I am calling how many times 3 + 3, 6 times I am calling the filter.

I had to pass what are the values and then it has to run the thing. So, you will be a calling with star x, star y it is their locations what you are giving and then constant signed int even the star h and then integer x_1 what you will be giving it? So, now we are initializing here also this is a local variable as you can see, temp is 0 and then it will be taking x1 as the input that is copy input to this memory. And $x(0)$ is going to be your signing this value and then temp will be calculated that is h_0 into your $x(0)$.

So, h_0 is int your predefining the thing input and then $x(0)$ is already int so you are doing the integer multiplication and then you will be what is it, adding it to temp so h_1 into x_1. So, you will be continuing up to h_6 into $x(6)$. So, you can see that sequentially what the computation is happening, this is for the first stage of the this thing 0 basically and this is going to happen for your poles that is divide temp by coefficients is a_0 what you are going to do that?

So, if you are want to normalize the thing then you have to divide by that. Otherwise, you are passing it through your y_0 to y_6 here. So, then what you are going to do? You are going to because as we know that multiplication and addition is going to result in 32 bit. So, we have to take only we will say higher order 16 bit. So, because Q13 format what we are using for the coefficients so we can shift it by 13 and then see that if temp is greater than 32767 so you will be what is it, saturating it manually here temp = 32767.

So, else if temp is less than or this thing -32767 this is overflow what you are avoiding it, here we are avoiding the underflow if it is less than this, you will be making it maximum as -1. So, the thing is varying between -1 to +1 that is what, what we are restricting and then we are putting as y_0 as temp value. So, you can shuffle the values for the next stage, because it has to take y_6 should be taking the value from y_5, $y(n-1)$. So, you will be and even $x(n)$ should take it from $x(n-1)$, do this, and then you will be returning that temp value.

So, this is how the code is going to run so we will take a demo of it so here my input is going to be corrective voice with this is the sa re ga ma what we are running. So, we will take the corrupt voice what we have run it with the MATLAB. So, sample files we have it so this will be IIR corrupt voice what I will be feeding into my system. So, this is running, let it be running. By the time we will run code on the board, we will see that thing.

So, I will be doing the debugging. First, what you have to do is the compile it see for any errors, since as I have been telling you that I will be usually running it first and then because time shortage and other things, it will take a little more time to figure out the errors and then do the debugging. Usually everything is corrected and once it is running what I will be putting it for the demo. So, we will run the thing now.

As you listen output is still remaining. So, you have seen that both whatever frequency that is 900 hertz, and then 2700 have been removed and you are hearing the clear voice from it. So, if you want to hear the corruptive voice, what I will do is the connection what we have given it to the board I will remove it. So, you will hear that 2 tones as you heard in MATLAB so now I am connecting back to my board.

And then if I read on the thing I can here I need not have to run debug every time I can go to the project, I can what I will say I can go to run I can load the project now whichever because all have been compiled. So, I need not have to debug it again. If I go and then click on them, it will be loading onto the board and then we can run it directly so, it will take. So, you have heard the clean speech now coming out of it.

So, once I reset the thing it stops. So, you will not be hearing the thing it is running. So, usually it goes and loads after only for debugging purpose what we have to use laptop once all the code

has been running on the board. So, till you remove the power supply, so it will be continuously running on the board. So, if there is any power failure, then you may have to read that is go and load into the board and then run it. So, this completes demo of FIR and IIR filter both in MATLAB and then in hardware.

Any way we have seen that resonating frequency how we are generated using the C code is shown in the previous class. So, if you want to make this a real time that is you can either write polling driven or interrupt driven. So, one of this can go as an assignment for you to run this in your hardware so that how I can generate my tone and how multiple frequencies I can generate using the sine wave. So, that within just as you can see that with y_0, y_1 and y_2 and then x_0, I will be able to generate my required frequency component using IIR filter in resonator mode. So, thank you and then all the best in your labs.

(Video Ends: 30:32)