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

Lecture – 16
Lab – IIR Filter as Resonator

Welcome back to real time digital signal processing lab today. So, we will be seeing some of the implementations, whatever we have done using the MATLAB in the last class. So we saw some sine generation using the DDS in Code Composer Studio. Today, we will see how we can use resonator for generating the sine wave. And then we will see from lookup table, how the real time signal is going to come out today.

(Video Starts: 00:59)

So, you can see that some of the things one has to remember is you can open the project and then some of the files has to be connected for this code basically, in this case, you will be seeing the cmd file which is going to define what are the memory which are going to be used in this. So, the default what it takes and then you will be seeing that what is the buffer length and then what is the audio which is going to be used. So in this case, I am using LCDK 6748 board for real time demo.

So, we have a sine wave that is which is going to have 48 kilohertz, what it is going to run. So, we will be using the interrupt driven. So, you will be knowing different methods of accessing the CPU one is using the polling, the other one can be interrupt driven which we discussed in our theory class also, the other one is DMA based, so their CPU intervention is not going to be there directly from the memory you will be reading into your memory to work on it.

So, some of the cases you may have to have a frame length of whatever the length if you have given you can do directly read from the memory and then put it into the local memory. So, in this case, what we have is the buffer length in this case is taken it as 128 samples and then the loop length is 48. So, how we are going to generate the sine wave usually we use MATLAB to generate the sine wave and then take this 48 samples here and then use them as you can see, it is in the fixed point that is signed Q 15 format what we are using it here.

So, use this in your code, then all of it as you are seeing that it is int16 length of the buffer as well as the computation. So, here we are going to do interrupt driven in this case, interrupt 4 is going to be used and then you will be getting it is the stereo data what we are going to get it so we will be as I have mentioned in the first lab session, so we can have mono input or stereo. So, whenever speech is that either we can take it from the mono that is mic we can connected or we can connect the audio input from our laptop.

Today we are connecting the audio input to the board. So, we will be taking first left sample and then you will be doing the processing here. So, what you have is take from the input buffer left sample pointer and then you are putting into the buffer length basically what you are comparing that is you are incrementing it and then you will be putting back left sample the value from the sine table whatever the pointer is pointing 2 and then sine pointer results are going to be incremented here.

Then we will be seeing that both for the left and right channel, the same left sample is going to be output. So, then we are going to output the sample codec data out. So, in this case, as I have already mentioned, we will be using AC3106. So, now what is the frequency at which it is going to operate that is what is shown it is L138 initialize interrupt driven basically, it has the sampling frequency of 48 kilohertz and then ADC gain what you have given is 0 dB and then DAC attenuation is also 0 dB what you have taken the thing.

And then you will be putting LCDK line input from here input what you are going to take it and then you will be output on the outside of here, or this thing processor basically, both the codec input has in and then out one of the input is for from the I will put it as our computer to input of the board and then output of the board is connected to the speaker here. I think next time we will show the scenario where the board and other things are there. So, we will see that how to run this code.

Some of the important thing what we have to do is when I am experimenting with this or generating the file, one has to keep that in this case, my as you can see that it is LCDKC6748 what I am using it and then here using debug connector, so XDS110 USB debug connector, what we are connecting

it to the board and then the compiler version what it is being used is the 7.4.4 version and we are generating the executable and then here output format has to be legacy COFF file what will be generating it has 2 options. So that is ELF format.

But in our case, it is legacy COFF file what it we need it and device endianness little endian. So, when I talk about little endian and big endian in the little endian lower memory is going to be stored first and then later on the higher part of your 32 bit data, which is going to be stored later, higher part later. Whereas in the big endian some applications need that higher memory has to be first and then lower memory bytes are to be later. So in that case, we will be using the big endian.

For our cases all these a TI board support little endian and then we will be command file what we need it is linker DSP dot CMD these can be downloaded from the TI site or will be providing you whatever necessity to run your codes in the web page basically and we need is rts6740 dot library this the common here 674 series any one of them can be using the library part of it. The next one is what we see here is the compiler. So, in this case, we have to include certain files.

So, these are the support files one has to include where you have downloaded it in my case it is in D colon DSP LIB 6748 What I have taken, where the support file is located, and next one is we have to have the board support library under that there are some include files which are required for real time running. So, then coming to the linker options, we have to provide the search path. So, one of the thing is as you have already taken rts6740.library which reflects here and then the other one is what we call this one although it is LCDKC6748.

We can select evmomapl 138bsl lib is for board support library. So, we need what is the board it is going to separate library what we have to provide and this is has been stored in this location that is how it is coming location from where I have been including which has to be provided. Then next again we have some libraries. So, which are put in the board support library so you can link them. Once you have done the thing you can apply and then close.

So, these are the necessary things for running it the other files which we need it for running in real time is that is one is L138 LCDK aic3106 dot init dot c, dot h and then linker dsp dot cmd and then

vectors interrupt dot asm these 4 files had to be taken from the support files what we have it and then we have to include. Once everything is done, so, what you can do is whether your code is running correctly or not. So, I can do a compilation. So, if there is any error which it will be listed out here, your syntax error, then you have to go and then attend to that.

If nothing is there, then what I can do is I can go and then build my system. So, once I build the system, you will be seeing that you will be seeing some green light is coming here, which is telling that it is going and then loading onto the board. So, here you have seen that the complete code has been loaded onto the board, you will be seeing that C674X underscore 0 what it has taken the thing. So and then where it has started your entry point what it is showing that it has entered, where the main is there.

So, what we will do is we will run this code, and then you will be hearing a sine wave which is generated at 48 kilohertz, so, I will run it and then you will be hearing the output from the speaker, hope you are hearing it, so, this is how real time sine wave generation using lookup table has happened. So, once I stop also you will be seeing that code has gone and then loaded onto the board. Although I have stopped this has been mentioning from the beginning, this is an integrated development environment code composer studio still our code is running.

So, once I reset by move my board it will be stopping. So, as you will be seeing that I can disconnect my laptop only for debugging session I can use it and then it will go and then load onto the board and then it can continuously run. So, here you will be seeing there are 48 samples in them and then what is the thing the code is going to lie in the board. So, unless I reset the code, it will be running continuously this is what we call it real time. Once I have taken the debugger, I have downloaded all my code into the dumped into the board.

Then it will be continuously running for whatever input you are going to give it. So, this is what we call it as real time. So, here what is the thing happening is we are going to interrupt for every sample my CPU and collect the data into my buffer here. And then I am taking it out and then playing it. Here it is only in this case, what we have done the thing is because we have generated

the sine table, the samples are taken from the table and it is sent it out to the codec channel that is DAC output. So, that is what what you heard the thing.

So, the next one what will take the example is so, this is the second sine generation next one will take it resonation using a resonator how we can generate our sine wave. So, we said that in IIR filter, we can push it to the resonator thing and then generate our sine wave. So, here what is the thing here, so you will be seeing that we are calling it as sine generator. And then we will be redefining my Y1 sample that is value is given here.

So, what is the value how we have calculated that we said that 3 samples what we have to give it after that the IIR filter is going to resonating on its own, so here it is sine f tone what you have taken the thing divided by f sample, so into 360 degrees, what you are converting into, so in this case sine we you want to generate 500 hertz divided by 48 kilohertz into 360. So, what is it? My f tone whatever the frequency I want to generate 500 hertz and sampling frequency I have chosen as 48 kilohertz.

So, which will be equivalent into sine of 3.75 which is my Y1 value what it has been taken, then we define our equation AA as 1.9957178. So, this we call it as 2 star cos 3.75 you can go back in the literature, where we have discussed about the IIR filter application as a resonator. So, you will get the filter structure and from there you will see that these are the coefficients what it has to be provided initially, then this is the value what it has been taken for this, then you are calling y 3 as you will be providing 0 for sample.

Then Y 1 is the other sample what you are providing it and then 0 and then our amplitude basically A is given as AA value, then, you are going to run the code that is you will be making your later on computation, y_0 is my output, what is $y_0 = y_1 * A - y_2$, this is our IIR filter equation, and then y_2 will be y_1 and y_1 becomes y_0 and then y_0 what you will be continuously multiplying is you have given a scale of full 16 bit range, what we are giving it.

So, multiplied with 32000 what we have taken. So, that is what it says using a number slightly less than this such as 32000 maximum, if y_0, it becomes 1 I can go up to 32767. So, if there is any

overflow or underflow, so you are multiplying it by 3200 that is what it says prevent overflow, then you will be returning short y_0. So, we recast the result as short and then value upon returning it since your D to A converter is programmed basically to accept 16 bit signed values in this case.

But here you will be generating it in the on the board, but real time we would not be will not be running it today in this class. So, you will be what is it then sine data using sine generation. So, you are block sign value what I had to calculate continuously. So, you will be putting it in the loop $i = 0$ for 0 to length, whatever you have defined, you will be calculating your $i++$. And then buffer of i is going to call the function sine generation basically.

So, then what happens, you are calling your function block sine from the main and then we are calling the passing the parameters g buffer and buffer size. So that is what it says fill buffer with the sine data and then return. So, your g buffer will be containing your output value, we will see how it is going to show that whether we are getting IIR filter in the oscillatory mode, we are going to generate a sine wave for our application. So, I am doing again now debugging the code so the code has gone and then loaded onto the board.

Then we will run the thing because I am showing you in the waveform format. So, I have given a breakpoint in this case. So, the code will be stopping there was a as you will see sometimes you will be getting the debugger is going to misbehave because my breakpoint it has not taken the thing it has completely gone and then run and then it is unable to reach whatever you have given sometimes return 0. So, it was unable to locate it. So, hence it gave you an error. So to I will be re allocating my breakpoint here, and then we will rerun it.

So, you have seen that it has come the breakpoint has come and then stopped here. Now I will be able to see the output once I complete it comes out of it my memory gets erased so I would not be able to see my what buffer has so if you keep the thing also you will be seeing that what are the hex values your g buffer where it is located default it is going to show hex value decimal value and octal binary all the format whatever you want to view it, you can see that.

Now what we will do is we will see using a graph, so for that I will be going to tools so if you are holding on in the with the break point, then this tools option will be coming for you while running your code otherwise it goes off we will see it in a while, so I can do a single time so I will got some 50 samples what I will be collecting it so I can say that it is 16 bit signed integer what I am going to get the output so then my start address is what we have is g buffer where the data is stored.

So, if I give g buffer so you will be seeing that the output is going to come here. So, you can see that only half the sine wave what it has come because I have chosen number of samples small so what I can do is hopefully I have control on the thing data what I can specify display properties we will see it access and other things what it is given so otherwise what I can do is I can close it and once again go to the tools and then increase my data size here.

So, I will give it as something 250 let us see whether I have the thing and then this is 16 bit signed integer what I can select and again I have to do this g buffer and then I can give it. So, what is the thing happening since my it is generating only 1 single cycle basically after that you will be seeing whatever is in the memory is junk. So, if you want to increase this so you have to have more data what you will be generating it number of samples what you will increase so that for loop will be much more than the thing.

That length what you can increase in this case length is chosen as we had we have defined our length, buffer size is our length which is getting passed on there. So, buffer size is what we have taken is 128 samples. So, you will be seeing that at 128 samples I am going to get 1 cycle so you can increase the buffer size and then collect more data so you can get more sample seen your waveform what you are looking at it. So, this is how we make IIR filter oscillate generate sine wave.

Now we will see one of the example what we have seen is sine wave generated only we have output. Now the other one is what we have is audio in and then out in the board that is it takes the input from any of your external devices. And then whatever comes out of it is going to be sent out directly to check that whether your path from ADC to DAC it is through the board it is working or

not how we are going to do that. So, this is audio in and out here you will be seeing that it uses the polling mechanism.

So to differentiate between your polling and then interrupt driven so the CPU has to go and then check if there is any data in the buffer. So, once in a while it will go and then check and then collect it. Whereas in the interrupt driven so as you know that whenever the person comes to your house, they will be pressing the calling bell. So, then you will go and then attend to that. But if you are expecting the guest at that time, and if he is a chief guest or whatever maybe the thing once in a while you have to go out and then check the person has come or not.

That we call it as a polling that is every 5 minutes or something you will be going and then checking. So, if he has come then he will be bringing him that is how your polling is going to work. Whereas in interrupt driven as an example, the person comes and then rings the bell, then we know that somebody has come, you will go and then attend. So, that is the difference what I have given with an example. Now we know that using the polling, and then we are using the sampling frequency as 48 kilohertz.

So, we have taken both ADC gain and DAC gain as same way, and then we will be using line input. So, this is what the board will be using it, then sample I, what I am going to get is input sample, I will get it and I will be sending it out output sample. So, what happens in this case, so you will be seeing that it is a 32 bit uninitialized int what you are calling it, whatever data comes here, you are sending it out. So, we will see that how this is going to run in real time. so you will be see hearing some, whatever speech signal if I gave it, it will be coming out.

If it is tone, it has to come out, that is how it will be working. So, we will do the debugging because I have checked for all the errors and other things so directly, I can run the debug if you are sure that there are no errors in your code directly, I can go debug and then test them. So, now we will run the thing. So, there is nothing coming out of it. So, just we will go and then see that some speech signal what I will be sending it remember the force will be with you always, remember the force will be with you always, remember the first little bit.

So, you have heard the speech coming out of it. So, I can select if you want any audio you can select the audio also. So, we will select here what I have is some tones different tones are there which has been generated. So, you will be seeing that how they behave with, as you are seeing it is a 500 hertz sine wave what it is getting played. So, we will see for the other tones anyway the music generating the music you have heard of it. So, how the pitch is going to change.

So, we will see one kilohertz how it is going to have a behavior. So, different, what you can generate the tone, so you will be seeing the shrill, most of the things will say it as a noise basically. So, this is how your board takes audio in and then sends out it is not doing any processing. So, most of the time, what we do is if we want both audio in and out we will be taking here, sample here input sample, then we do the processing and send it out.

So, as an example, we will see here there is a audio is getting what I will put it as attenuated. So, but still you may be able to hear a little bit of the thing. Otherwise you can try on yourself. We have seen a sine wave generation and in the next class we will see how FIR filter can be put in between our audio in and out with noisy signal, how we can remove the noise. We will look at it on the board we have seen already on using our lab MATLAB.

So, we will be designing the coefficient using MATLAB FTA toolbox. So, I had shown you how to store those FTA coefficients in our file, which we will be using it in our FIR filter design in hardware. Thank you have a nice day.

(Video Ends: 29:19)