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

Lecture – 32

Lab: Different ways of implementing FFT in CCS

Welcome back to real time digital signal processing lab course basically. So, last class we had seen some of the FFT how it is going to run in MATLAB. So, today we will see how we can implement the same thing in our code composer studio that is on the hardware DSP processor. So, the first thing what we have here is a C code what it is written for FFT. So, you will be seeing that we will maximize this, so that you can see the thing the code. So, what is it? So, we define the math function because extensively we will be using the sine and then cos functions in this case, otherwise we have to derive it in hardware using series expansion.

So, here number of points DFT points what it is defined is 64, we call it as FFT or DFT interchangeably you have to know about that. And then in this case I said in MATLAB we are defined π as it is, but here we have to specify the value of it. So, it depends on what is the length you want to define it either 3.1415 usually we stop at and then if you want more accurate value you can take it up to this.

And then the structure what defined is float real and imaginary is defined which is a complex variable. And, then it is going to call FFT function in this case. So, the prototype whatever return basically and then I O buffer is going to take the input and then both output buffer. So, it takes the input values and then output is also going to the buffer that is the reason why it is called I O buffer. And, then your intermediate buffer is also float you are seeing that some of them are defined as float some of them are short values.

So, the index variable i is defined as short and then buffer count is also initially it is made 0 the samples in the I-O buffer. And, then the flag what you are going to set 1 by interrupt service routine when I-O buffer is going to be full. Then, you have to use the complex that is w points basically you have defined points as 64 in this case, we have it twiddle constant stored in w. The next one is samples that is primary working buffer in this case that is also of length this. and the frequency what will be generating is 10 hertz in this case.

So, we will be computing twiddle factors using cos and then sine function that is what one is the real part of it what we have twiddle constants and imaginary this thing twiddle constants are computed with this. Now, using this you will be what is it generating the I-O buffer of I with sine function with 10 hertz. So, that is frequency what it is given and

then 64 is the sampling frequency what you have chosen in this case. So, you will be initializing the samples initially to 0, then you will be calling the buffers basically to swap them with new data and then later data. So, you will be going up to 64 that is 0 to 64 you will be computing your FFT.

So, first is imaginary component you are making it 0 and then you are calling the FFT function by sending samples, points in this case it is 64. So, samples are also going to be 64 in this case call function FFT dot c in this. case. And then once you have called the thing, so you will be getting both real and imaginary values as written. So, you will be doing i is equal to 0 to i less than points compute the magnitude of it.

So, how you are going to calculate the magnitude? So, x1 of i is nothing but square root of samples i dot real with real squared basically plus samples of imaginary dot imaginary. So, both the squared under square root what you will be calculating. and then you will be incrementing your p with p plus 1 and then this will be ending our main function. So, we will go to this thing. So, it will be in line with it.

So, we as usual we will be calling the project as I said one of the student who has done the thing. So, we will do the first So, you will be seeing that it is intact here it is using the inbuilt FFT function which is being called with whatever samples you have it. So, you have the sorry FFT function we did not see the thing. So, we will see it with the thing. Here also it is points is 64 and structure is your float and then complex basically and then you will be defining this complex you are calling it as extern because you will be passing from between function and then your main function and then FFT function.

So, you are calling this as a FFT function which is. complex y is the output and int is our length of our FFT. So, input sample array and number of points as you are seeing it. So, these are the temporary storage variables and these are the loop counters and then we have to calculate Upper leg and then lower leg are calculated separately that is even and then odd part of it are computed of the butterfly separately. So, then difference between the upper and lower leg what you will be calculating and you will be defining number of stages.

So, that is to have the interaction and then you will be defining some of the step through twiddle constant and then i is equal to log base 2 of N points that is number of stages what you will be doing it. So, you will be doing up to number of stages plus equal to 1. i will be i to the power of i is 2 times i what you will be doing it and defining your while function this way. So, you will be calculating the leg difference that is difference between upper and lower legs, step between values in twiddle dot hedge what you will be doing it and then for end point FFT you are calculating i is equal to 0 to number of stages what you have it. and then initially index will be 0 and then the other loop is going to be j is equal to 0 to less than leg difference.

So, you will be computing it and then here it is the upper leg what you will be calculating it, some of the temporary variables you will be seeing it and then y is calculated in this fashion and you will be indexing is going to be incremented depending on the step size. and then the leg difference is going to be leg difference by 2 and then step is going to be multiplied with 2. So, again j is equal to 0 you will be doing the as we know that bit reversal has to be done for our c calculation which is done using this computation. So, you will be seeing that hardware is not being used. So, if you are writing in assembly as I have been telling in the thing, you can use the hardware to do the bit reversal.

Here it has to be done manually as you can see the thing using the code. So, this is how you will be calculating your this thing imaginary and real part are arranged for your FFT. And then you will be coming back from this FFT function. So, we will run this and then see what is the magnitude squared function what we have got it. So, this is we are debugging our DFT.

So, we are seeing that build has finished. you are seeing that it is connecting it to the board basically. So, you will be seeing that where the target has been connected and memory map whatever there is going to be cleared and it will be set up for this function. and it will be calling the memory is DDR2 in it at 150 megahertz is going to be done. So, this processor is running at 150 megahertz at present.

So, it goes and loads the code basically. So, we have to see the debug. So, you can see that the debug window has got opened. So, it has gone and loaded in the this is the entry point basically main function where it starts the thing. This is the entry point of the code in our board.

and then you will be seeing that here the pointer is pointing to the main basically. So, it is ready to execute. So, as we have seen in the last class when I took up the demo of the code composer studio. Either you can continuously run put a breakpoint and run. So, here there is a breakpoint as you can see which is been set.

So, we will run the code continuously. If you want to do this there is any error if you find then you can go with single stepping. So, we will run the code. there was a little this thing memory map issue with the thing. So, we will set the debugger again and then I can either do a debugging or I can because it is already debug has been done we can go and then from the run command.

So, here I can you will be seeing a lot of variations will be available. So, you can go and load your program. So, if it is already finished the debug and you have not modified anything and then if you want to run a resume from the run stage you can do that. or if you want to terminate or if you want to go to main or if you want to reset the processor whether you want to do the CPU reset or system reset or reset the emulator basically

whatever is running on the thing you can do it. And then if you want you can restart or you can step into or step over that function if it is a loop then I do not want to see every step of it I can do the step over in that and then if I want to do assembly step into I can do the thing.

So, these are the functions what you have it. Now, what we will do is we will have a run here either from here or here I can give the thing, I will give the run from here ok. So, because this is not finding the exit dot c that is why what it is little bit complicated. So, we will see that what is our this thing output basically. So, what I have to see is my magnitude square function.

So, we will we can go to tools, we will go to graph, we will find it a single time and then we have said that it is 64 point FFT what we have run. So, input data will be 64. And, then this is a floating point what we are running the thing. So, I can define it as 32 bit floating point number and then the start address in this case is x 1 is our output where we are calculating our magnitude and even the display data size I will change it to 64 and then we will say ok. So, you will be seeing that what was our input frequency it was 10 hertz ok.

So, you will be seeing computing FFT from your thing which approximately gives you peak exactly if you want to look at the thing it is somewhere around 10 ok. So, you can see that your code is running fine. So, how to check this whether you have got your frequency correctly or not. What we can do is our samples what we have generated here you will be seeing that I O buffer is going to give a sine function. So, we will see that whether that also has 10 hertz as a sampling frequency.

So, we will go to graph again. So, sorry I will cancel it because single time for we will see single time and then come back. So, I can give 64 and then this is again a floating point number 32 bit floating point number and then the start address in this is going to be our I O buffer. So, I o buffer what I can give the thing and then even display points will be 64 because we have not gone beyond it. So, we can give the thing. So, you can see the sine function how it has with 64 point it is not a complete pure sine wave you will be seeing some of the points here has a problem, but it is approximately going with the thing.

So, now what we will do is. We will see FFT of this. So, again I will go to tools and then I will go to my graph in this we will calculate FFT magnitude. So, this is 64 points what I have it 64 and this is again our what is it 32 bit floating point number and then what I can give is I O buffer. and the number of points what I want is 64. So, we will give stages 6 in this case and then we will see what will be R.

I did not give the sampling frequency I should have given it. So, you have to multiply 0.15 into 64 which is going to give you 60 hertz. So, if you are unable to see this what we will do is we will again do go to tools and then graph we will do FFT magnitude and then this is 64. And then this is my this thing 32 bit floating point and here sampling frequency what I have to give is 64 and then start address is Ivo buffer.

and then this is our thing is 64 what we wanted. So, we will give this and we will plot. So, you will be seeing that it is showing 10 hertz is the input frequency computed from the built in function. So, what calculated you will be seeing that in the single time. So, you will be seeing that by writing your own FFT code.

So, you are generating the 10 hertz signal what you have computed without knowing what is the frequency component of the input signal. Here we know the thing in normal cases we may not know what kind of signal we are going to get it. So, once I find the frequency spectrum. So, what are the frequencies present? But I should know how many what is the sampling frequency at least used most of the cases as I have mentioned in my theory class that speech if it is a narrow band speech we use 8 kilohertz as a sampling frequency and if it is a wide band speech it is going to be 16 kilohertz. and then we know the rest of the thing audio and then video thing.

So, those are the sampling frequencies. So, from that what you should be able to find out that whether what signal you want to keep it and then eliminate by doing filtering one of the application. So, using from the frequency domain you will know what are the frequencies present which you want to keep it and which you want to eliminate also you can see. Most of the ECG signals as you will be knowing that line frequency in India is 50 hertz whereas, in US it is 60 hertz that will be present in the ECG signal. So, first thing what you will be doing is eliminate that frequency using filters. So, this is one of the example of FFT in using our board basically.

So, now we will see we have seen the overlap add and save method using what I will call it as MATLAB we had seen in the last class. So, we will see how we are going to do using our C code in hardware. So, here we will be defining number of points there it was 64 points we had defined here it is 512 points and the length of it is 512 and you will be seeing that our filter length is gone to 351 and then input signal length is 3000 length and then our signal block length is 162. So, this is again pi is defined and then you will be defining variables just like the previous FFT case and then what you will be having is you are going to contain the filter coefficients in column dot that. So, there you are generating your filter coefficient.

So, here you can compute and keep it in your data as the file. So, you will be seeing that filter coefficient is called this is the 351 using MATLAB what it has been taken and then put it in a file as that file. So, the other one is you will be calling fft.c function here this

thing it is 512 points. So, we have seen this fft with what is the thing bit reversal also which contains the thing there it was 64 bit here it is going to be 512 bits what we will be doing it.

So, coming to the main dot C what is it? So, we have to compute our this thing input signal, what are the frequencies that are generated here? It is 200 hertz just like in MATLAB what we seen the example for these frequencies, same thing what will be implemented in C code also on the board. So, 200 hertz, 267 and then 400 hertz has been generated using sine function in this case. So, you must be thinking why sine is used. So, it will take time by either you can use the IIR filter in oscillatory mode to generate our sine wave or we can use this.

So, taking the FFT of this one and then saving it. So, we can compute our this thing you will be seeing that it is using the convolution using a overlap save method. So, that is how you will be calculating your output. So, you will be calling the IFFT and then you will be computing IFFT using the new twiddle factor array. So, and then ifft is just do fft in this case by providing your twiddle factors with the complex conjugate. And then you will be doing a stage y in this case for k is equal to 2 and then you are doing all this computation.

and then later on k is equal to 3 and then k is equal to 4 to that is floor of length s divided by s length that is in this case what it is taken as 64 because you have done already 3 of them. So, 64 is the length of your FFT what you are do using it in overlap save mode for computing the length of your data. So, you will be doing it and the last one the 62nd one you will be repeating it by providing what is the k value of it. So, you can see the length of the code what it is running here. So, again we will put a break point and then see whether there is any error in the thing.

So, it says it is up to date because. I have checked it earlier and then brought it and then. So, we will be doing the debugging as usual. So, one of the thing one has to remember is what we have to look for here in this case. So, what is it? Here it is out s what we are going to look at it. So, it has gone and then loaded on to the board and then it is pointing at the main.

So, the break point what we have provided is here. This is out s is what we have to look at it. I do not think there is a calculation for our magnitude squared. So, one has to remember or to see that which one I want to monitor ok. So, here what is my input output if I know the thing then I will be able to see that what is my output. So, here what has been declared is product of A50s and IA50 in the out temp.

So, that is what I have to look at the thing. So, anyway we will run the code if there is any difference. So, we will find out with the thing. So, now, again as I said because it is

unable to see exit dot c sometimes in c function. So, that is why the code has gone beyond and then it is looking for the thing.

So, anyway we will look at what is our single time graph. So, we said that it is O temp is our thing. So, we can give it as. Our acquisition buffer of 3000 what we have it and then it is a floating point. So, I can give it as 32 bit floating point and then the sampling rate I will not mention, start address what I will mention it as out s ok, we will try to map the thing. So, you will be seeing that okay, so something I made a mistake here anybody can look at what I made the mistake was only 200 samples are getting mapped.

So, but my length is 3000 what I have given the thing this is the sine wave because you have done the I a 50 also in this case. So, we can see what is the frequency we have it 32 bit floating point and then start address is out s sorry, where my output are there and then here acquisition also I have to make it that is display data size for 3000. Now, something you will notice I want you to look at the thing. So, for 3000 samples what we have done the thing.

So, what I will do is I will make it full scale. So, you are seeing your this thing what is the frequency here somewhere around 260 hertz retaining it and filtering out the other two frequencies. So, there were three frequencies what were present others have been eliminated. So, one of the thing what we can do is. we can see in the other points basically. So, I can see samples ok. So, what it is going to give you my it should give me my inputs. So, you can see my some of the samples what it looks like this ok. This is not the complete sine wave. So, this is not the input my input is going to be somewhere generating it as SI only the problem is. Here I would not be able to see this si, why is it going to strike something to you because this is si is a local variable.

So, for me to plot it I had to declare it as a global variable and then I can do the plotting. So, this exercise you can take it up and then look at it. So, like this you can go and then run your codes and then see whether your FFT code is working.

One more is DFT code what it has is. this is from the book. So, if you are able to download the code book and then run your code. So, you will be seeing that he is it is from the real time digital signal processing fundamentals implementation and application. This is the third edition most of the time I will be using it also which has the C code to complete compute your float FFT test dot C. So, even MATLAB codes are there. So, you people can use the thing and then some of the functions that is floating point complex header files is created and then fft dot h and then data file input also is going to be you can declare and then you will be using it.

So, if you say that your input f dot data you will be seeing it as. So, this is your input data what you have it input 7 f floating point data for experiments ok. So, you can look at running these book codes in the thing. So, just we will run the thing. So, we will see that

how it is going to run in our board. one has to keep it in mind that it because I have already checked the things there are no errors.

So, that it can go and then load on to our this thing board and then it is going to run. As you can see the thing it is pointing here. So, I can put a breakpoint here in the printf statement ok. So, and then I can run the code. So, he will be running FFT frames basically there are 13 frames run in this case and which are the ones running and what will be the value of it you will be seeing it.

So, it will take a little time and then you can observe it. So, you can see that experiment is completed at the 13 frames basically frame wise computation what it is happening. So, that means, to say that either you can sample wise what you can do it or the complete frame. run and then have a look at it. So, here output is going to be present in the spectrum of i.

So, we will display that basically. So, tools go to your graph single time. So, this has a 64 sample thing and we will call this as 32 bit floating point and start address is spectrum. and even the display we have to have it as 64 because that is what we will be getting it. So, you will be seeing that I will magnify the thing. So, you will be seeing that this is what somewhere around 12, 7 hertz or whatever 6 or 6.9 to 7 hertz is the frequency what it is getting generated through this. With that we will close the FFT computation on the DSP processor board and then we will look at the other applications in the next lab session. Thank you.