

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

Lecture - 26
FFT - 2

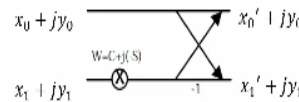
Welcome back to real time digital signal processing course, today we will discuss continue with our fast fourier transform. So, in the last class, we discussed about how to derive the equations from our DFT and then we drew the butterfly structure and then said that how our computation can be improved using FFT.

(Refer Slide Time: 00:49)

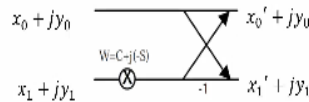
Finite wordlength effects in FFT



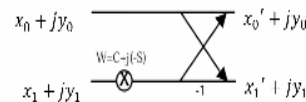
- Roundoff errors, which are produced when the product $W^k B$ is truncated or rounded to the system wordlength



- Overflow errors, which result when the output of a butterfly exceeds the permissible wordlength



- Coefficients quantization errors, which result from representing the twiddle factors using a limited number of bits.



Rathna G N

So, today we will see that what is the finite wordlength effects in FFT. So, what are the first one is we can have we are going to have roundoff errors. So, which are produced when the product $W^k B$ as we can see is truncated or rounded to the system wordlength, we have been seeing that when it is getting multiplied by this butterfly structure will be going to be truncated or rounded. Next one is what we have is the addition so, which is going to cause overflow.

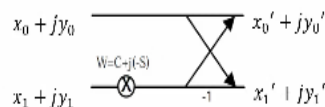
So, error so, which results when the; output of a butterfly exceeds the permissible wordlength. As you can see there is an addition. So, when it exceeds the limit, then we will have the overflow. The other one is our W^k that is coefficients quantization errors; this is the third one what we have it which result from representing the twiddle factors using a limited number of bits. So, because we

cannot represent W^k itself for an infinite number of bits are not available depending on number of bits.

So, we have to truncate our twiddle factors. So, that will be causing us coefficients quantization or sine and cos function what we are going to store them.

(Refer Slide Time: 02:19)

Round off Errors



- Butterfly computation required 4 real multiplication and 6 real additions
- Roundoff noise power (variance) at the output of each butterfly is given by

$$\sigma_B^2 = 4 \times \frac{q^2}{12} = \frac{1}{3} 2^{-2(B-1)} \text{ where } q = 2^{-(B-1)}$$

$$\sigma_0^2 = N \sigma_B^2 = \frac{N}{3} 2^{-2(B-1)}$$

Assuming that each butterfly generates identical but uncorrelated errors, the maximum noise power at each FFT output is approximately

$$W_N = e^{-j \frac{2\pi}{N}} \cos\left(\frac{2\pi}{N}\right) - j \sin\left(\frac{2\pi}{N}\right)$$

$$W_N = C + j(-S)$$

$$A = (C)x_1 - (-S)y_1$$

$$B = (C)y_1 + (-S)x_1$$

$$x_0' = x_0 + A \quad y_0' = y_0 + B$$

$$x_1' = x_0 - A \quad y_1' = y_0 - B$$



Rachna G N

So, we will see first roundoff errors. So, what is it we have taken here $x_0 + jy_0$ instead of representing it as A we have taken split as x as input and this is $x_1 + jy_1$. So, this we have our twiddle factor is $\cos\left(\frac{2\pi}{N}\right) - j \sin\left(\frac{2\pi}{N}\right)$ function. So, you are seeing the multiply sign and this is going to be multiplied by -1 and our output is going to be $x_0' + jy_0'$. So, what we are going to assuming that each butterfly generates identical but uncorrelated errors.

The maximum noise power at each FFT output is approximately given us equivalent to W_N which is nothing but $e^{-j \frac{2\pi}{N}}$ nothing but $\cos\left(\frac{2\pi}{N}\right) - j \sin\left(\frac{2\pi}{N}\right)$. So, we represent cos function as C and then a sin function as S and we are taking the negative sign inside so, it is going to be $C + j(-S)$ and then we know that our $A = (C)x_1 - (-S)y_1$ same way our B is going to be represented in this fashion.

And then we know that $x_0' = x_0 + A$ what we have it A is given by this equation and $y_0' = y_0 + B$ given by this equation and $x_1' = x_0 - A$ and $y_1' = y_0 - B$. You can verify whether we have got


these equations correctly or not, then what happens? Our butterfly computation requires for real multiplication what we have, as you can see the thing here what we needed.

So 1, 2, 3 and then 4 real multiplications and 6 real additions. So, how many of them you can count here this is 1, 2, 3 and then 4. So, we have another 2 coming from your these two A and B here $4 + 2$ what we needed. So, 6 real additions taking into account our subtraction is equivalent to addition that has to be kept in your back of mind whenever we do this additions and next what we say is then what happens to our round of noise power that is we call it as variance at the output of each butterfly is going to be given by.

So, we have derived our noise power as q^2 by 12. So, because we have 4 real multiplication, so, it is going to be $4 \times \frac{q^2}{12}$ which is nothing but $1/3$ and then we are substituting $q = 2^{-(B-1)}$ in terms of how many bits it becomes $\frac{1}{3} 2^{-2(B-1)}$ and then this is at each butterfly. So, then we have how many butterflies we are going to have.

So, we approximate it as although we have $N - 1$ is the thing, number of stages what we are going to have it, we have rounded it off to N . So, the noise power for N stages is going to be σ_0^2 is nothing but $N\sigma_B^2$. So, when we substitute our butterfly σ_B^2 which becomes $\frac{N}{3} 2^{-2(B-1)}$.


(Refer Slide Time: 06:23)




Round off Errors (2)

- Total number of butterflies required to produce an output sample is

$$\begin{aligned}
 \frac{N}{2} + \frac{N}{4} + \dots + 2 + 1 &= 2^{M-1} + 2^{M-2} + \dots + 2 + 1 \\
 &= 2^{M-1} \left[1 + \left(\frac{1}{2}\right) + \dots + \left(\frac{1}{2}\right)^{M-1} \right] \\
 &= 2^M \left[1 - \left(\frac{1}{2}\right)^M \right] = N - 1
 \end{aligned}$$

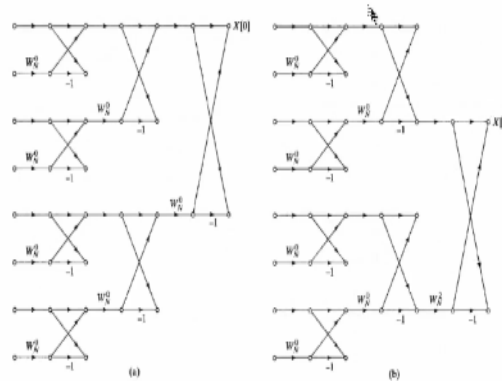



Rathna G N

So, we need total number of butterflies required to produce an output sample is so, we will be seeing $\frac{N}{2} + \frac{N}{4} + \dots + 2 + 1$. So, when you take the series expansion, so, you will be seeing that it is nothing but $2^{M-1} + 2^{M-2} + \dots + 2 + 1$. So, when you substitute it, so, which becomes equivalent to $N - 1$ butterflies are required, so, which was approximated as N .

(Refer Slide Time: 07:00)

Round off Errors (3)



Rachna G N

So, now, you will be seeing that how the error is going to have its impact. So, here you want to compute $x(0)$. So, how many butterflies we need it we have to go back from here. So, I need one butterfly here and the previous stage to compute this and this output here one butterfly is required in the stage 2 and this is the other butterfly required to compute the output of this thing or we need W_N^0 in this case.

So, 3 butterflies and in the previous case, you will be seeing that we need all the 4 butterflies here. So, this is $4 + 2, 6 + 1, 7$. So, it is nothing but $N - 1$ butterfly contribute for one output. So, this is x_0 if you take x_4 also what you will be needing the same thing. So, as an further example, if you take $x(2)$ this is x_0 and this is going to be x_1 , this is going to be x_2 and this is x_6 will be the output. So, you will be seeing when you count how many butterflies are required, you will be needing $N - 1$ butterflies to compute one output.

(Refer Slide Time: 08:27)

Round off Errors – 2



- Doubling N , which is equivalent to adding a stage to the FFT, doubles the noise power. To retain the same noise power, we increase the wordlength by 1 bit.

$$\sigma_0^2 = \frac{2 \times N}{3 \times 2^{2B}}$$

- Assume that the real and imaginary parts of the input sequence are uncorrelated and that each has an amplitude density that is uniform between -1 and $+1$

$$\sigma_x^2 = \frac{2^2}{12} = \frac{1}{3} \Rightarrow \sigma_x^2 = N \sigma_0^2, \quad SNR = \frac{\sigma_x^2}{\sigma_0^2} = 2^{2(B-1)}$$

- Note that for FFT algorithm, a double-length accumulator does not help us reduce round-off noise, since the outputs of the butterfly computation must be stored in B bit registers at the output of each stage

≡

So, that is the reason why what we call it as that is when we increase N that is doubling N , which is equivalent to adding a stage to FFT. So, that is what is going to happen doubles the noise power to retain the same noise power, we increase the word length by 1 bit. So, we will be seeing that sigma naught square is nothing but we are doubling the $\frac{2 \times N}{3 \times 2^{2B}}$ and then increasing 1 bit.

So, it becomes to be $-1 + 1$ which is going to be 2^{2B} , then what happens the real and imaginary parts of our input sequence are uncorrelated and that each has an amplitude density that is uniform between -1 and 1 , then our σ_x^2 in our input noise power is nothing but $\frac{2^2}{12}$ which is nothing but $\frac{1}{3}$. So, we can equate to signal to noise ratio $\sigma_x^2 = N \sigma_0^2$. This is σ_0^2 .

So, then signal to noise ratio $\frac{\sigma_x^2}{\sigma_0^2} = 2^{2(B-1)}$. So, what the note we are going to carry from here for FFT algorithm. A double length accumulator does not help us reduce roundoff noise, since the outputs of the butterfly computation must be stored in B bit registers at the output of each stage. That is what the note what we will be carrying from this, just like in FIR filter by increase then the length of our accumulator, we could hold on to the result till the end of it. But here, we would not be able to do it because we have to use it for the next stage.

(Refer Slide Time: 10:36)

Overflow Errors



- There are 3 methods to avoid the overflow:
- Static scaling: dividing the input at
 - First stage by N
 - Any stage by 2
- Dynamic scaling dividing the input at any stage by 2 if the largest absolute input value is greater or equal than 0.5

$$|X[k]| < 1 \quad 0 \leq k \leq N-1$$

$$|X[k]| = \left| \sum_{n=0}^{N-1} x[n] \cdot W_N^{kn} \right| \leq \sum_{n=0}^{N-1} |x[n]| \quad 0 \leq k \leq N-1$$

So now, we will see how the overflow is going to cause error. So, how we can avoid it overflow cannot be avoided. So, how you can by scaling and other things, how we can avoid the overflow errors, there are 3 methods one is we can do a static scaling that is dividing the input at first stage by N that is a maximum this thing or division what we are going to do it or any stage by 2. So, that way we can incorporate the static scaling.

The other method what we have is that is basically based on the dynamic scaling that is dividing the input at any stage by 2 if the largest absolute input values greater than or equal to 0.5. So, then what happens that is we are taking the norm of magnitude of $X[k]$ which is less than 1. So, $0 \leq k \leq N-1$. So, how you will be taking the thing to find the norm, so, we will be equation is nothing but $\sum_{n=0}^{N-1} x[n] \cdot W_N^k$.

So, we have to take the norm of it, which is less than or equal to, so, we have assumed that twiddle factors have been scaled and then that within the limit, then it becomes the magnitude of x of n by scaling my input then that is $0 \leq k \leq N-1$ I can overflow error can be avoided just like our FIR filters. So, if we have more than $N-1$ stages, the addition what we can take care of in our accumulated by having guard bits and then later on, we have to do the scaling.

(Refer Slide Time: 12:54)

Static scaling – Dividing the input at any stage by N



- The noise-to-signal ratio increases as N^2 , or 1 bit per stage. That is, if N is doubled, corresponding to adding one additional stage to the FFT, then to maintain the same noise-to-signal ratio, 1 bit must be added to the register length.

- The assumption of a white-noise input signal is, in fact, not critical here. For a variety of other inputs, the noise-to-signal ratio is still proportional to N^2 , with only the constant of proportionality changing.

$$|X[k]| < 1 \quad 0 \leq k \leq N-1$$

$$|X[k]| = \left| \sum_{n=0}^{N-1} x[n] \cdot W_N^{kn} \right| \leq \sum_{n=0}^{N-1} |x[n]| \quad 0 \leq k \leq N-1$$

$$|x[n]| < \frac{1}{N} \quad 0 \leq n \leq N-1$$

$$\sigma_x^2 = \frac{\left(\frac{2}{N}\right)^2}{12} = \frac{1}{3N^2} \Rightarrow \sigma_x^2 = N$$

$$SNR = \frac{\sigma_x^2}{\sigma_0^2} = \frac{2^{2(B-1)}}{N^2}$$

So, the first one that is what we said is static scaling that is dividing the input at any stage by N . So, what happens to the noise to signal ratio increases as N^2 or we call it as one bit per stage. That is, if N is doubled, corresponding to adding one additional stage to the FFT, that is what we just now saw it, then to maintain the same signal to noise ratio, one bit must be added to our register length.

So, this is our equation what we have it, so, we will be adding one more bit to that. So, we say our $x[n]$ we say that it is scaled by $\frac{1}{N}$ then what happens to our sigma noise σ_x^2 input basically $\frac{\left(\frac{2}{N}\right)^2}{12}$ which is nothing but $\frac{1}{3N^2}$. So, we call it as this is our signal power σ_x^2 is approximately equal into N and then signal to noise ratio will be our signal is $\frac{\sigma_x^2}{\sigma_0^2}$, which is going to be $\frac{2^{2(B-1)}}{N^2}$.

So, what in this case the assumption of a white noise input signal is considered in that case it becomes $\frac{\left(\frac{2}{N}\right)^2}{12}$ is our signal power, for a variety of other inputs, the noise to signal ratio is still proportional to what we call it as N square with only the constant of proportionality is going to change as you can see the thing here.

(Refer Slide Time: 14:52)

Static scaling – Dividing the input at any stage by 2



- Very little noise (only a bit or two) is present in the final array. Most of the noise has been shifted out of the binary word by the scalings

$$\sigma_0^2 \cong 4\sigma_B^2 = \frac{4}{3} 2^{-2(B-1)}$$

$$|x[n]| < \frac{1}{N} \quad 0 \leq n \leq N-1$$

- It is important to note again that the assumption of a white-noise signal is not essential in the analysis.

$$\sigma_x^2 = \frac{2^2}{12} = \frac{1}{3} \Rightarrow \sigma_x^2 = \frac{1}{3N}$$

- The basic result of an increase of half a bit per stage holds for a broad class of signals, with only the constant multiplier in eq. being dependent on the signal.

$$SNR = \frac{\sigma_x^2}{\sigma_0^2} = \frac{2^{2(B-1)}}{4N}$$



Rachna G N



So, if we do dividing the input at any stage by 2 what is the thing is going to happen? So, I think there should be some trigger in your mind, because why we call it by 2 is we have a barrel shifter. So, we can shift the dividing is nothing but right shift by one bit is going to give us divide by 2. So that we need not have to spend any time in shifting our input, what happens? Very little noise only a bit or 2 what we are going to get affected is present in the final array and most of the noise has been shifted out of the binary word by the scalings.

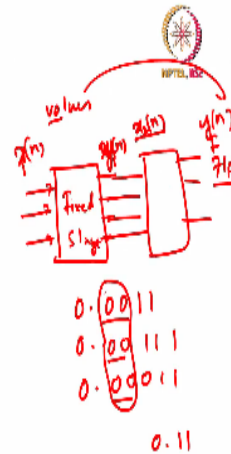
So, you will be seeing that $\sigma_0^2 \cong 4\sigma_B^2$ just we saw that because of the 4 butterfly, which is nothing but $\frac{4}{3} 2^{-2(B-1)}$ and then we are assuming our $|x[n]| < \frac{1}{N}$ in this range, then what happens to our input noise power, so, which becomes 2^2 assumption here also it is white noise signal what it is assumed, then it will be $\frac{2^2}{12}$.

And then this is $\frac{1}{3}$ which is equivalent to $\sigma_x^2 = \frac{1}{3N}$. And so, the result of an increase of half a bit per stage holds for a broad class of signals with only the constant multiplier in this equation being dependent on the signal. So, what happens to a signal to noise ratio in the final sigma x square by sigma naught square, you will be seeing that $\frac{2^{2(B-1)}}{4N}$ in this case after substitution.

(Refer Slide Time: 16:57)

Dynamic scaling – Block floating point

- The original array is normalized to the far left of the computer word, with the restriction that $|x[n]| < 1$
- The computation proceeds in a fixed-point manner; except that after every addition there is an overflow test.
- If overflow is detected, the entire array is divided by 2 and the computation continues
- The number of necessary divisions by 2 are counted to determine a scale factor for the entire final array
- The SNR depends strongly on how many overflows occur and at what stages of the computation they occur.
- The positions and timing of overflows are determined by the signal being transformed; thus, to analyze the SNR in a block floating point implementation of the FFT, we would need to know the input signal.



So, the other one is we call it as dynamic scaling. That is, we use the block floating point. So, what is its definition you may be wondering what is that thing, what we say is the original arrays normalized to the far left of the computer word with the restriction that our $|x[n]| < 1$ and the computation proceeds in a fixed point manner, except that after every addition there is an overflow test, fine. If there is an overflow detected, the entire array can be divided by 2 and the computation is going to continue.

So, the number of necessary divisions by 2 are counted to determine a scale factor for the entire final array, then we will be calculating the signal to noise ratio depends strongly on how many overflows occur and at what stage of the computation they occur. So, the positions and timing of overflows are determined by the signal being transformed. Thus, to analyze the signal to noise ratio, in a block floating point implementation of the FFT, we will need to know the input signal.

So, when we call the block floating point is because each stage, we have different stages. So, we know each stage, what is the input required. So, we will scale this input $x[n]$ what we will call it the first stage, or I can call it as $x_1[n]$ and then keep those values here. Do this in the fixed point mode operation, then when we are coming out of it, either take these values and scale back so we will call it as $y_1[n]$ which goes as input to the next stage, which we call it as $x_2[n]$.

We can multiply and then do the rescaling of this or in the end, what we can do is if there is a overflow, then we will be dividing it by 2 in these cases, if it is not, so then we will keep these values till the end. And when I come out of here, whatever final I will put it stage, I will multiply with these values and bring it back into floating point value. So that is one value where I know that just as an example, so we have 0.0011, and the next one is 0.00111 or the next value may be 0.00011.

So, I know that these 2 bits, I can call it as a left shift and then keep it as 0.11 and then I will be working that is block floating point I will calculate whichever the common take it out and then use the rest of them to for my computation of FFT here, stage one or whatever may be the thing, this is how our block floating point is going to work in our case.

(Refer Slide Time: 20:37)

Real Time FFT Considerations

- Signal Bandwidth
- Sampling Frequency, f_s
- Number of Points in FFT, N
- Frequency Resolution = f_s/N
- Maximum Time to Calculate N-Point FFT = N/f_s
- Fixed-Point vs. Floating Point DSP
- Radix-2 vs. Radix-4 Execution Time
- Windowing Requirements



So, coming to the what are the real time FFT considerations, we are going to have it so, one of the thing is we have to consider the signal bandwidth, sampling frequency, number of points FFT N what we wanted and what is the resolution? So, we have seen the DFT resolution, so, what we need it f_s/N and maximum time to calculate our endpoint FFT. So, that is N/f_s what we are going to have it whether we want to do fixed point versus floating point DSP.

So, in the lab, we have done floating point implementation to do the fixed point implementation, we may have to consider the block floating point and then do it. So, whether we want the Radix 2

FFT or Radix 4 FFT. So, if it is power of 4 whatever input it is advantageous to use Radix 4 because it is going to be -1 or $-j$ coefficients what we have to compute in Radix 4 you can refer to the literature's to how to do the Radix 4 implementation.

So, we have seen the Radix to here, but divide that is even and odd parts what we have taken and then how we have computed. So, the next one is because most of the application is going to be, we will be using in the windowing filtering. So, what are the windowing requirements one has to consider, so, we saw an example in the lab that is a case a window and then the rectangular window. So, how our output is going to get affected with the thing.

(Refer Slide Time: 22:26)

DFT Computations



N	Real Multiplications				Real Additions			
	Radix-2	Radix-4	Radix-8	Split Radix	Radix-2	Radix-4	Radix-8	Split Radix
16	24	20		20	152	148		148
32	88			68	408			388
64	264	208	204	196	1032	976	972	964
128	72			516	2054			2308
256	1800	1392		1284	5896	5488		5380
512	4360		3204	3076	13566		12420	12292
1024	10248	7856		7172	30728	28336		27652

So, now, to put the thing different Radix implementation can be done. So, if it is power of 2 all of them what is taken is power of 2 some of them are power of 4, some of them are power of 8 what you will be seeing it so, if it does not fit into any of this category, and you are taking some Radix format, how we can do the split Radix format also one can be 8 and then if it is order is 2^{12} if we take the thing, I can have Radix 8 one and Radix 4 what I can combine and then how I can do the Split Radix.

So, that is what, what is shown in this table. So, as an example, this is the least $N = 16$ what is considered and Radix 2 needs real multiplications of 24 whereas, as you can see it has come down to 20 because this is power of 4 number of multiplications real multiplications has come down to

20, in the Split Radix it needs also 20 in that case, whereas real additions in case of Radix 2 we need 152 additions in Radix 4 it is 148 and Split Radix is 148. Just we will see the last one 1024 rest of it you can look at yourself.

So, what we Radix to as you can see that it needs 10248 real multiplications and 7856 in the case of Radix 4, so, you will be seeing that how much difference we are going to have with Radix 4 with Radix 2 and then we cannot implement this in Radix 8 so that is why no nothing has been given. And in this Split Radix still you can come down to 7172. And then same thing with respect to your real additions it takes 30728 whereas Radix 4 takes 28336 and then Split Radix is still lesser.

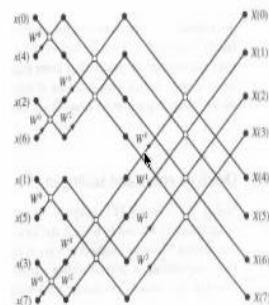
So, depending on your application, which Radix format, what you want to have it that is the application may need 1024 or 2048 or 4096 or if you want to implement much more than that, you can see which Radix format or different Radix format at each stage what you can have it and then combine and then take the output if it is not power of 2 also.

(Refer Slide Time: 25:20)

Flowgraph - I



- For 8-point, radix-2, decimation-in-time FFT Algorithm



14

Dr. K. S. K.



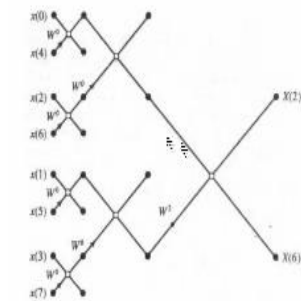
So, we will see that how the butterfly structure is represented in a flow graph that is Radix-2 8-point decimation in time what algorithm what it has been taken. So, you will be seeing each point what we have it, so, you will be seeing that, here you are seeing the 4 butterflies here, and in the

next stage also 4 butterflies. And then in the last stage also what you have the butterfly and multiplication and other things are represented with your weight factors as it is shown.

(Refer Slide Time: 25:59)

Flowgraph – 2

- Contribution of butterflies to roundoff noise at the outputs $X(2)$ and $X(6)$



15

Rathna G N

And then we said that to contribute for any of the two 1 input, we need $N - 1$ butterflies. So, there we have drawn everything with the flow graphs, it is easy to see that we need $N - 1$ butterflies.

(Refer Slide Time: 26:23)

Signal to Noise Ratio in FFT

- We can associate four round off noise sources to each butterfly, one for each product round off noise power (i.e., variance) at the output of each butterfly is given by

$$\sigma_0^2 = 4 \times \frac{q^2}{12} \text{ where } q = 2^{-2(B-1)} \text{ and the system wordlength is } B \text{ bits}$$

- The Noise generated by a butterfly at one stage is fed into subsequent stages ($N - 1$ stages)

$$\sigma_0^2 = (N - 1)\sigma_n^2 \approx N\sigma_n^2 = \frac{N}{3} 2^{-2(B-1)} \text{ (when } N \text{ is large)}$$

- The signal-to-noise ratio in this case is approximately equal to

$$SNR = \frac{1/3}{\frac{N}{3} 2^{-2(B-1)}} = \frac{2^{2(B-1)}}{N}$$

16

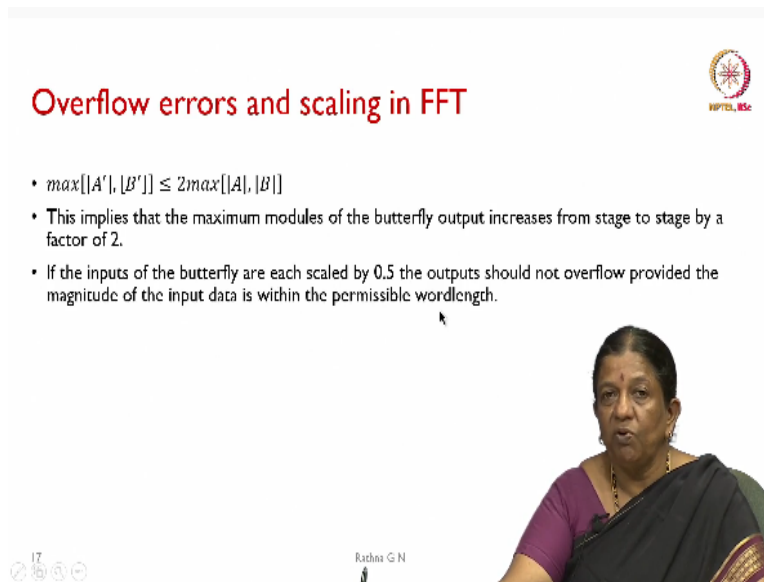
Rathna G N

So, we saw just now the signal to noise ratio. So, you can go through the thing, we have number of bits is equal to B bits. And then we have seen that noise generated by a butterfly at one stage is fed into the subsequent stages that is in this case $N - 1$ stages, what we call it. So, then are we said

that our noise is represented in this way that is $(N - 1)\sigma_B^2$. And now we will be realizing for large N we can represent it as $(N)\sigma_B^2$ with assumed in the previous case.

And then we have signal to noise ratio in this case is approximately given by this equation that is $\frac{2^{2(B-1)}}{N}$. So, if I want to this thing signal to noise ratio, we call it as proportional or inversely proportional to N and directly proportional to number of bits, as you can see that N is in the denominator and B is in the numerator.

(Refer Slide Time: 27:37)



Overflow errors and scaling in FFT

- $\max[|A'|, |B'|] \leq 2\max[|A|, |B|]$
- This implies that the maximum modules of the butterfly output increases from stage to stage by a factor of 2.
- If the inputs of the butterfly are each scaled by 0.5 the outputs should not overflow provided the magnitude of the input data is within the permissible wordlength.

17

Rathna G N

So, that is how by increasing one bit you can nullify on whatever signal to noise ratio what you want to get it. So, we said that overflow errors and scaling in FFT what it has to be done maximum what we call it as that is now you will be making it $A' = 1$ and $B' = 1$ in that case it will be $\leq 2\max[|A|, |B|]$. So, this implies that the maximum modules of the butterfly output increase from stage to stage by a factor of 2.

So, if the inputs of butterfly of each scale by 0.5, the output should not overflow provided the magnitude of the input data is within the permissible wordlength what we call it.

(Refer Slide Time: 28:30)

Overflow errors and scaling in FFT (I)



- In fact, in some cases scaling by 0.5 is not sufficient to avoid overflow even if the input is less than unity. To illustrate, consider the explicit expressions given

$$A' = A_r + B_r \cos(X) + B_i \sin(X) + j[A_i + B_i \cos(X) - B_r \sin(X)]$$

$$A' = A_r - [B_r \cos(X) + B_i \sin(X)] + j[A_i - [B_i \cos(X) - B_r \sin(X)]]$$

$$A' = 2.4142 + j; \quad B' = -0.4142 + j$$

- If $X = 2\pi k/N = 45^\circ$, then $\cos(45^\circ) = \sin(45^\circ) = \sqrt{2}/2$. Without scaling and with real and imaginary parts of inputs each set to 1 (the limiting case), we have, from the equations above,

$$A' = 1.2071 + 0.5j; \quad B' = -0.2071 + 0.5j$$



Rathna G N

So, as in this thing scaling, how we have considered 0.5 as the scaling factor and then deriving what will be the magnitude of it is illustrated by this example. So, we consider $A' = A_r + B_r \cos(X) + B_i \sin(X)$ and this is our imaginary part and then we assume $A' = j[A_i + B_i \cos(X) - B_r \sin(X)]$. So, how we have got this if $X = 2\pi k/N$ which is 45° , then our $\cos(45^\circ) = \sin(45^\circ) = \sqrt{2}/2$.

So, without scaling and with real and imaginary parts of inputs each set to 1, the limiting case we have from the equations from these above what is it? $A' = 1.2071 + 0.5j$ and then $B' = -0.2071 + 0.5j$ if all of them are set to 1. This is what we will be getting it. So, we know that what will be our maximum output from this butterfly structures.

(Refer Slide Time: 29:55)

Problem



- A hardware FFT processor uses fixed-point arithmetic in its butterfly computations. Estimate the maximum wordlength required to perform a 1024 point FFT with an output SNR of 40 dB. Assume that the input to each butterfly is scaled by 0.5 throughout the FFT.

$$SNR = \frac{1}{2N} 2^{2(B-1)}$$

$$40 = 10 \log \left(\frac{1}{2N} 2^{2(B-1)} \right); \quad 10^{\frac{40}{10}} = \frac{1}{2N} 2^{2(B-1)}$$

$$B - 1 = \frac{1 \log_2(2N \times 10^4)}{\log(2)} = 12.14 = 13 \text{ bits (approximately)}$$

- System wordlength, $B=14$ bits

19

Rachna G N

So, to see that by just in how many bits are required to keep our signal to quantization noise ratio required. We will see that processor one problem in this case so what it says a hardware FFT processor uses a fixed point arithmetic in its butterfly computations, so estimate the maximum wordlength required to perform a 1024 point FFT with an output signal to noise ratio of 40 dB, in this case you are lucky to have just 40 dB so assume that the input to each butterfly is going to be scaled by 0.5 which is nothing but $\frac{1}{2}$ throughout the FFT.

So, then signal to noise ratio become because we are scaling by 2 it is going to be $\frac{1}{2N} 2^{2(B-1)}$. so we have been given signal to noise ratio in dB as 40 dB substitute this put it as 10 log on the right hand side so you will be solving this then I need $B - 1 = 12.14$ bits which is nothing but 13 bits approximate it to the next higher number, then system wordlength for this is going to be $B = 14$ bits, so you can see if I want to have the CD quality which is 91 dB.

So, if you are this thing scaling is 0.5 so you can see that how many bits are required you can work it out and then come back with the answer. So, this we have seen that how the quantization noise is going to affect our number of bits and then how many end points what we have to choose it.

(Refer Slide Time: 31:58)



- Overlap-add and overlap-save techniques to compute long Input Signal



And in the next class we will be covering overlap add and overlap save techniques to compute long input signal because as you are seeing in the present case the length of the input is also restricted to $N - 1$ whereas in our real time application input is coming continuously so how we can apply this FFT using overlap add and then save in the next class thank you.