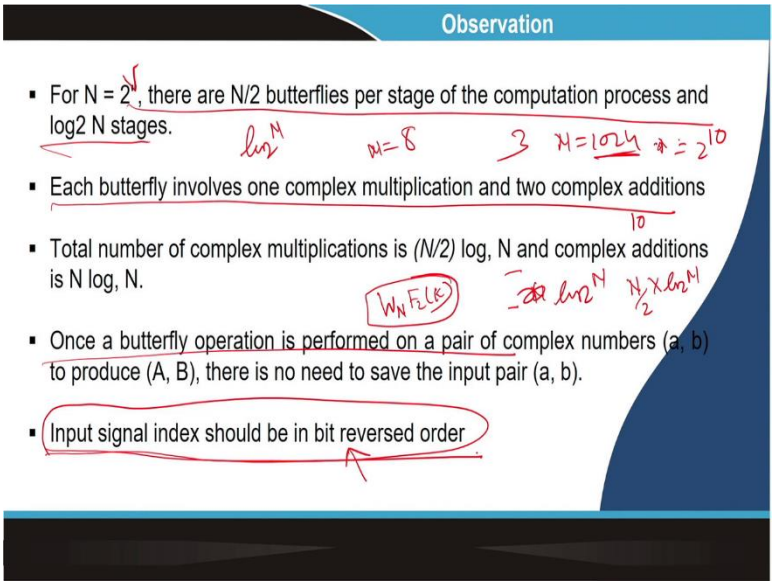
Signal Processing Techniques and Its Applications
Dr. Shyamal Kumar Das Mandal
Advanced Technology Development Centre
Indian Institute of Technology, Kharagpur

Lecture - 33
Radix- 2 FFT Algorithms (Contd.)

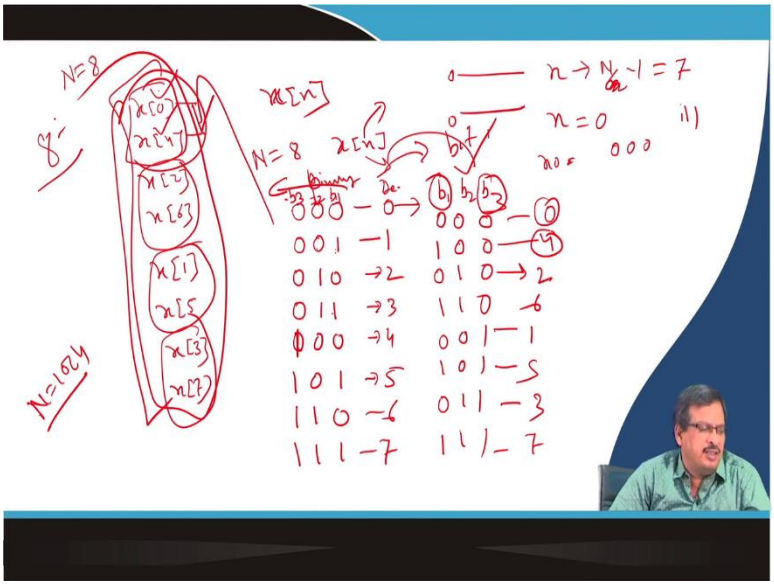
Ok, so we are discussing about this observation.

(Refer Slide Time: 00:24)



So, what is the input signal index that should be a bit reversed? Why it is required? What is the problem?

(Refer Slide Time: 00:29)



If you see, when I say N equal to 8, how do I how do I club the signal x[0] with x[4], x[2] with x[6] and index 1 signal? We clapped with the index 5 signal, and this is the jumble of things I required x[3] will x[7]. How do I do that? In the case of 8 points, I manually did it.

But let us say N is equal to 1024 points. So, if N equals 1024 points, it is impossible to do it manually. So, how can I access and disseminate the input signal x[n] for computing FFT? So, when I want to implement the FFT algorithm, I know I have to implement the butterfly and for each butterfly, I require input signals that are jumbled as odd and even odd and even. If, in the case of 8, I know this kind of jumble is required, this kind of decimation is required in the time.

So in the case of 8, I can do odd and even odd, and even, I can do that, but in the case of N equal to 1024, how do I do that; that requires an algorithm. So, instead of reducing the complexity, I am increasing the complexity. So, how should I do it? If you observe minutely that there is a trick to access the signal, which index of the signal should I access?

Let us first explain for 8 bit, and then I will go for the other. So, let us say N is equal to 8. So, if I say x[n], the index of the signal, the sample number, which is represented by the index of the signal, is n, if I want to represent it in a binary number. How many bits are required? So, what is the maximum value of n? N/2 minus N minus 1, which is nothing

but a 7. What is the minimum value? 0. How many bits are required to represent 7 as a decimal number?

So, I know 3 bit. So, if I say 3-bit can represent the index of the signal for 8 point DFT or the length of the signal is 8. So, 0 0 0 0 0 1, so this is 0 decimal, this is decimal, and this is binary, this is 1 0 1 0 2, then 0 1 1 3, then 1 0 1 0 0 4, then 1 0 1 5, then 1 1 0 6, 1 1 1 7. So, the index 0 is represented by a 3-bit binary number which is 0 0 0.

All 3 bits are 0. Index 7 all 3 bits are 1. I want to access the signal in this order. So, first, the 0-th sample, then the 4-th sample, compute the butterfly. Then, 2nd sample and 6th sample compute the butterfly. So, how do I decide that how to do what should be my accessing sequence so that I can club the 0-th sample with the 4-th sample?

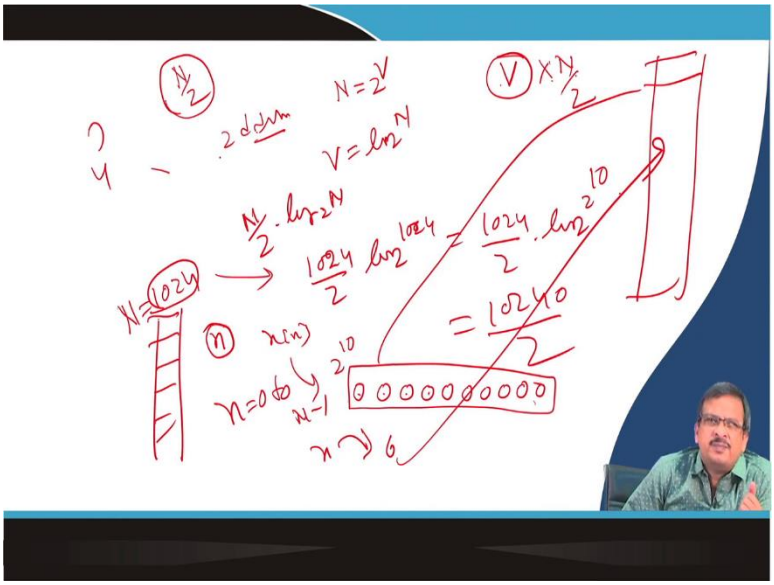
It says that if I access the index in bit reverse mode. So, this is b 1 bit. This is b 2 bit. This is b 3. So, this is LSB this is MSB b 1 b 2 b 3 is the LSB to MSB. If I access it instead of b 1 b 2 b 3 let us say b 1 bit like this way. So, b 1 b 2 b 3 this side, I reverse it. So, b 3 becomes LSB, and b 1 becomes MSB, which is called bit reverse.

So, here I reverse the bit: bit reverse. So, I reverse the index representation bit. So, if I reverse it, what will be there? 0 0 0, so 0 0 1 LSB becomes MSB 1 0 0. So, this is 0 1 0 0 and again LSB becomes MSB 0 1 0 again 1 1 0, so again 0 0 1 again 1 0 1 again 0 1 1 again 1 1 1. Now, see that access 0 4. So, if I access the signal in index bit reverse mode, I get a 0-th sample and the 4-th sample, which I require.

Similarly, here are 2 sample 6-th sample, here is 1 sample 5-th sample, here is 3 sample 7 sample. So, if I access the index, I store the signal in memory, and I am accessing the signal in the bit reverse mode of the index. So, the jump in the decimation, which is required in the time domain, is done in bit reverse mode, understand. So I do not have to write a separate algorithm to disseminate the signal in the time domain.

I directly get it by accessing the signal in bit reverse mode. So, that is the beauty of the FFT algorithm. So, if I summarize, then what is the computational complexity of a radix 2 decimation in time algorithm?

(Refer Slide Time: 07:03)



So, I know I require every stage. I required N/2 , the number of butterflies and the number of stages is nothing but a. If N is equal to 2^V, the number of stages is V.

Every stage N/2 number of butterflies and one butterfly required only 1 multiplication and 2 addition: 2 addition 1 multiplication. So, if I say how many multiplication numbers of the stages is V, each stage number of multiplication is N/2 . So, nothing but a V into N/2 . What is V? V is nothing but a log 2 N. So, I can say computational complexity is N/2 into log 2 N.

So, if I say N is equal to 1024, then the total number of multiplication is 1024 divided by 2 log 2 1024, which is nothing but a 1024 divided by 2 into log 2 2^{10}, which is nothing but a 10240 divided by 2 ok. So, the number of multiplications is reduced. Now, if you see how do I disseminate the signal in the time domain?

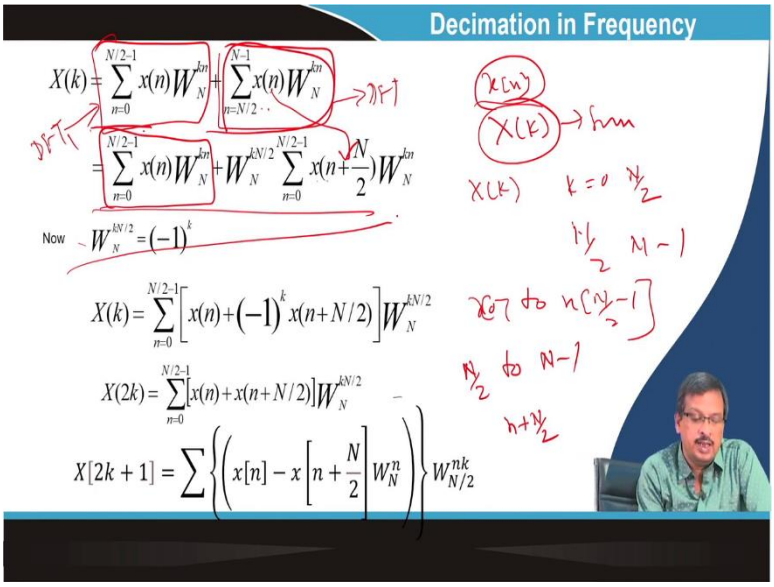
So, in 8 bit, I know it is 0 4, so here N is 1024. So, if I say how many bit is required to store the index n of x[n] 2^{10}, it means 10 bit is required. So, I can say this n is represented by 10 bit, let us say 1 2 3 4 5 6 7 8 9 10 bit is represented by 10 bit index, and it is stored in a memory location. When I access the signal for doing FFT, I will access it in bit reverse mode.

So, whatever the index, index n varies from 0 to N minus 1. So, I get the n, take the bit reverse, and then access the signal. The corresponding signal will be accessed. So, when I

access the 0-th signal, the 0th signal will be accessed; when I access the 1 signal, n is equal to 1, and I just do the bit reverse. So, whatever the index will come, I will access that signal, and I compute the FFT algorithm.

So, what I request is that all of you, once you understand the radix 2 decimation in time algorithm, you can implement it in C programming. Do not write the FFT of the signal in MATLAB and do that. Can you do that? Later on, you can think about how efficiently I can implement it. So, that is the decimation in time algorithm.

(Refer Slide Time: 10:12)

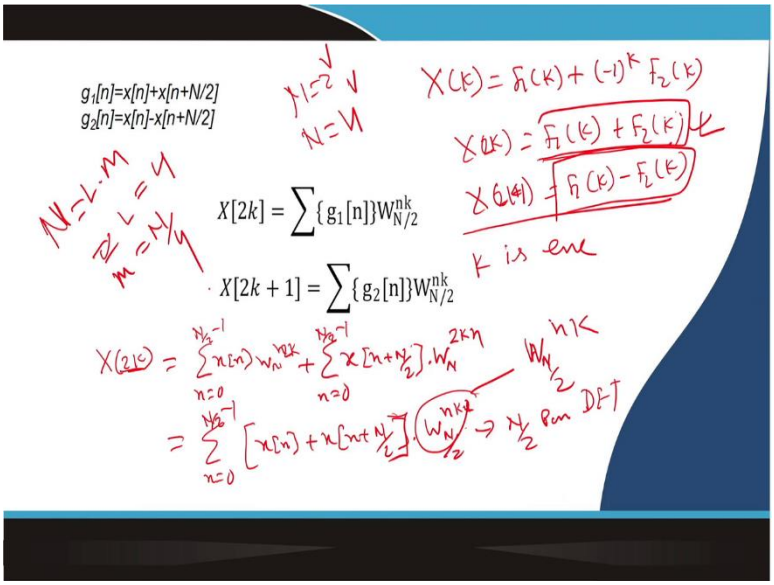


Now, we go for the decimation in the Frequency algorithm. Here, so instead of dividing the signal odd and even in the time domain, now I divide the signal in the Frequency Domain. So, what is the frequency domain? If x[n] is in the time domain, then capital X(k) is in the frequency domain. So, instead of dividing x[n], I am dividing X(k); let us compute that X(k) first, k, equal to 0 to N/2 and then N/2 to N minus 1. So, I can say instead of directly computing N point DFT, I divided this DFT equation into 2 N/2 point DFT.

First, I access the signal x[0] to x[n] by 2 minus 1 and compute the DFT, then I access the signal x[n] by 2 to N minus 1 and compute the DFT. So, this is the DFT number 1, and this is the DFT number 2. Now, here I can directly know that this is nothing but the same. But if you see here, here summation varies from N/2 to N minus 1, but the length is N/2 , I know that.

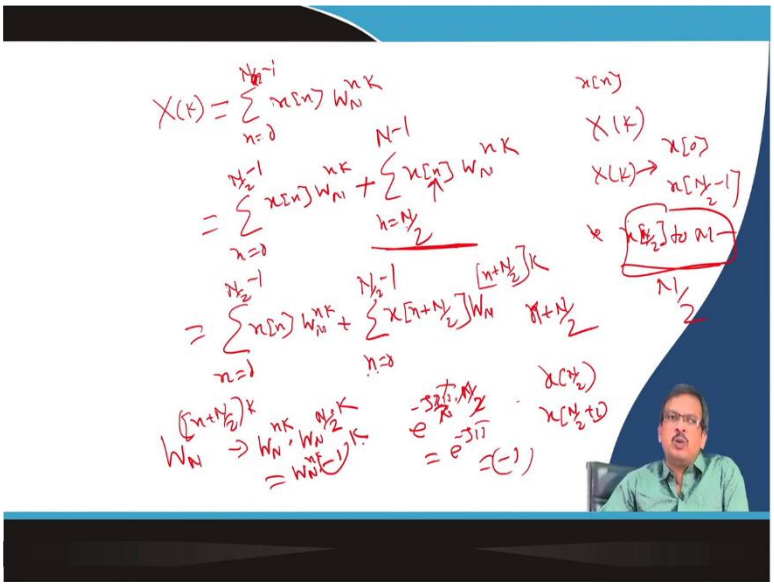
But, if I say this index is changed to n plus N/2 . Then I can say small n equal to 0 mean x of ok you may not be understanding, so I will take a slide here.

(Refer Slide Time: 12:02)



And then, I explain it very clearly, then I come to this slide. So, what I said is forget about the slide. What is there forget about that.

(Refer Slide Time: 12:11)



So, I said I wanted to compute N point DFT. So,

$$X[k] = \sum_{n=0}^{N-1} x[n] W_N^{nk}$$

which is the DFT equation.

Now, I said instead of dividing the signal in odd and even, I said no, I want to divide X(k) in two parts

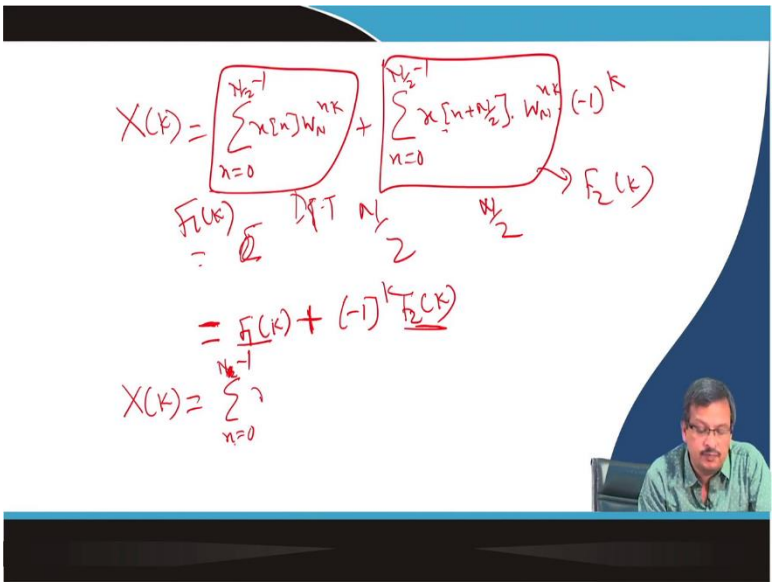
$$X(k) = \sum_{n=0}^{\frac{N}{2}-1} x(n) W_N^{nk} + \sum_{n=\frac{N}{2}}^{N-1} x(n) W_N^{nk}$$

So, I can say n varies from N/2 to N minus 1 x[n] W N n k. Now, if you look at this one here, index n starts from N/2 ; let's suppose I want to start from 0. The meaning is that the index is changed from instead of index n, I can write n plus N/2 . When I write n plus N/2 , then I can rewrite it this way: n equal to 0 to N/2 -1 x[n] W N n k plus n equal to 0 to N/2 -1, because the length of the signal is N/2 . Now, I want to write 0 instead of N/2 to N minus 1.

So, what I am doing is that I am changing the index, which is nothing but n plus N/2; when n equal to 0, I am accessing x[n] by 2; when n equal to 1, I am accessing x[n] by 2 plus 1. So, this part I am doing. So, the n index is replaced by n plus N/2 into k. So, if you see that W N n plus N/2 into k, which is nothing but a W N n k into W N N/2 into k.

So, $W_N^{\frac{N}{2}} = e^{-\frac{j2\pi}{N} \cdot \frac{N}{2}}$. So, 2 2 cancel N N cancel so $e^{-j\pi}$, which is nothing but a minus 1. So, I can say it is nothing but a W N n k into minus 1 to the power k minus 1 to the power k.

(Refer Slide Time: 15:24)



So, what I can say that

$$X(k) = \sum_{n=0}^{\frac{N}{2}-1} x[n] W_N^{nk} + \sum_{n=0}^{\frac{N}{2}-1} x\left[n + \frac{N}{2}\right] W_N^{nk} \cdot (-1)^k$$

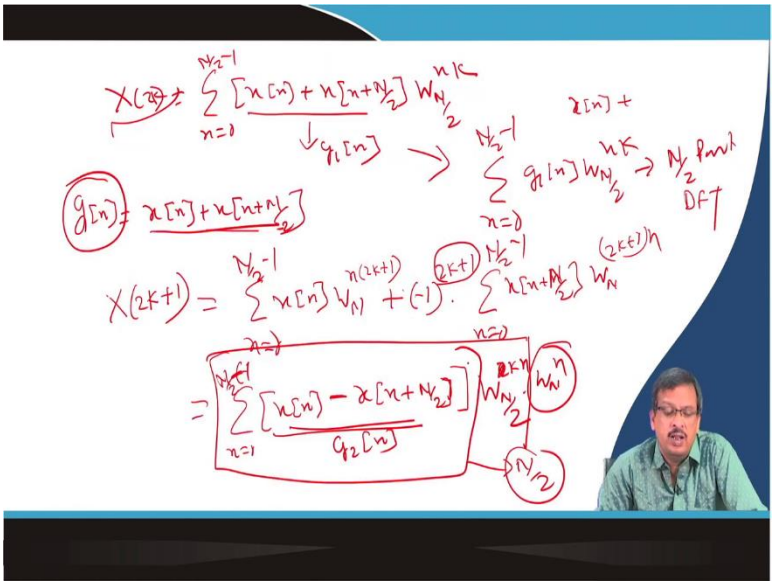
So, if I say that this one is in DFT length is N/2, this one is in DFT length is N/2. So, if this one is F1 k and this one is F 2 k, then I can say X(k) is nothing but an F1 k plus minus 1 to the power k. So, there will be another term. I have forgotten, let us see this one: F 2 k, ok? So, I write down F1(k) is nothing, but this part I write down ok.

So, now if you see that, I can say that X(k) here, X(k) is equal to these two sums is same. So, why do I write two sums? Let us say n is equal to 0 to N/2 -1. So, F1[k] and F2(k) are almost equal. So, I can say if n is equal to 0 to N/2 -1 or N minus 1, up to this part is ok. So, let us say so X(k). Forget about this slide. Think about here: X(k) is nothing but an F1(k) plus minus 1 to the power k F2(k), ok or not.

So, if it is that so when sometimes X(k) will be F1(k) plus F2(k) or X(k) will be F1(k) minus F2(k). So, when will this be a plus? When k is even, so if I say x[2] k in that case it will be positive. If I say x[2] k plus 1 for odd, if k is even, then it is F1(k) plus F 2 k; if k is odd, then it is F1(k) minus F2(k). So, if I want to write down the equation of F 1 X of 2 k is nothing but a n equal to 0 to N/2 -1 x[n] W N n k plus 2 k W N 2 k; k equal to 2 k plus n equal to 0 to N/2 -1 x[n] plus N/2 into W N k is replaced by 2 k; 2 k n.

So, if you see, I can say that n is equal to 0 to N/2 -1 x[n] plus x[n] plus N/2 multiplied by W N n k 2, which is nothing but a this can be replaced by W N/2 n k. So, this is a signal, and this is N/2 n k. So, it is nothing but a N/2 point DFT. Whose N/2 point DFT? I will take another slide here.

(Refer Slide Time: 20:19)



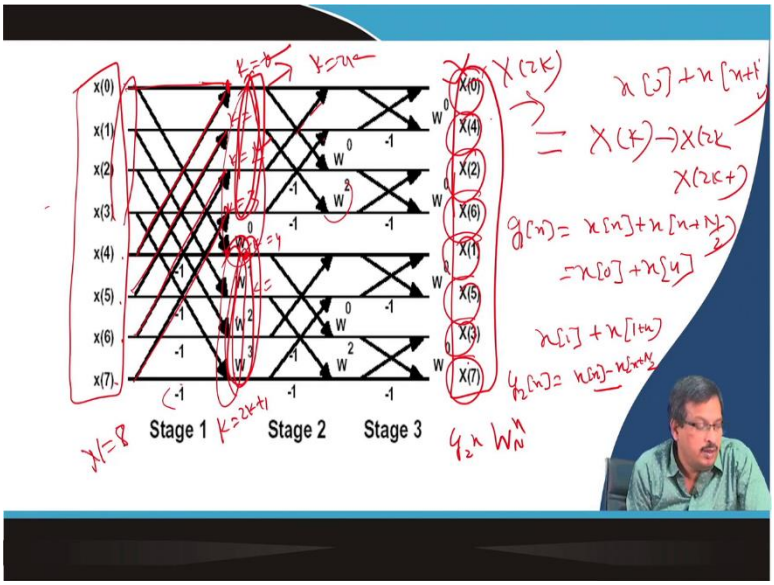
I can say X of 2 k is equal to 2 k is equal to n equal to 0 to N/2 -1 x[n] plus x[n] plus N/2 into W N/2 n k. So, I can say this is a signal. Let us say this is g of g 1 n. So, g 1 n is nothing but a x[n] plus x[n] plus N/2. So, I am creating a signal sequence accessing the signals g x[n] plus n by a n plus N/2 signal ok.

So, these two signals combined together create my g 1 n, and then I can say n is equal to 0 to N/2 -1 g 1 n W N/2 n k. So, it is nothing but a N/2 point discrete Fourier transform. Now, when X is odd 2 k plus 1, so for an odd index of frequency, the even index of the frequency is this one. So, what is the odd index of the frequency? I know this is nothing but a n equal to 0 to N/2 -1 x[n] W N to the power n k n into 2 k plus 1; plus 1 k is replaced by 2 k plus 1.

Minus plus minus 1 to the power 2 k plus 1 into n equal to 0 to N/2 -1 x[n] plus n by capital N/2 into W N to the power 2 k plus 1 into n. So, I can combine together n equal to 0 to N/2 -1 x[n], see this is 2 k plus 1 is odd is nothing but a minus; minus x[n] plus N/2 ok W n to the power 2 k n into W N to the power n 2 k plus 1. So, this is nothing but a N/2 .

So, this is if it is g 2 n, then I can say this is again N/2 point DFT N/2 point DFT, which is multiplied by W N to the power n. So, though, if the output is odd, then it will be subtracted and 2 point DFT the N/2 point DFT; if the output is even index, then it will be added and N/2 point DFT.

(Refer Slide Time: 23:35)



So, if I want to draw the signal flow diagram. So, I have x 0, x 1, x[2], so here I am not decimating the signal; the signal is accessed as per index x[0] plus x[n] plus N/2 ok. So, see how the signal flows; when I compute decimation in time, my signal flows from the east side to this side; here, I am flowing the signal from input to output. How? So, 0 to 3, 4 5 6 7.

So, if I say this is my k index 0, k index 1, k index 2, k index 3, k index 4, k index 5. So, if I say this is an even index, is this an even index, okay or not? So, if I say that in the first part, let us say this part is even and odd. So, I have divided X(k) X of 2 k and X of 2 k plus 1. So, 4 points will be the even point, and 4 points will be the odd point. Let us say the first 4 points are my even point, which is k equal to 2 k, and the last 2 points are my odd point, k equal to 2 k plus 1 last 4 point.

So, in the case of the first 4 points, it will be signal x 0. So, what is g 1 n? g 1 n is equal to x[n] plus x[n] plus N/2 . So, if the 8th point N is equal to 8, then I know that N/2 is 4. So, when I say accessing x[0], then I have to access x of 4. So, x[0] and x[4] will be multiply add together.

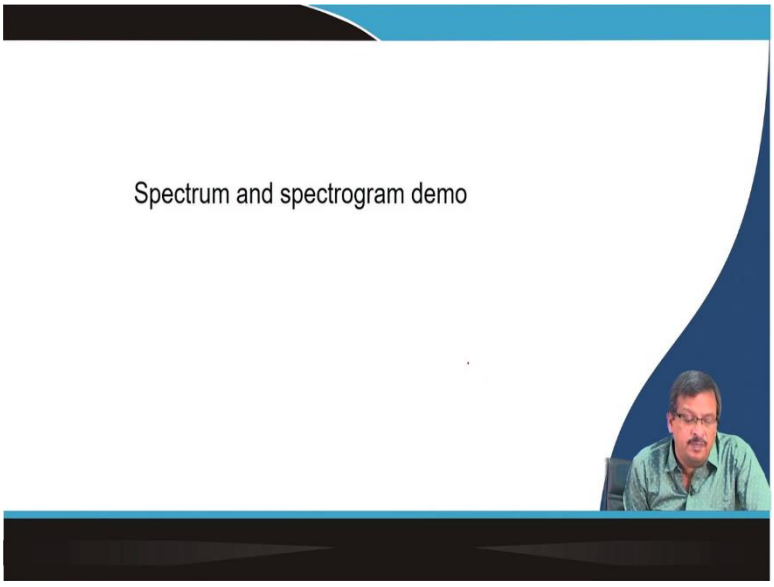
There is nothing there. Just add; now, again, when I say x of 1, then, I know it is nothing but a x of 1 plus 4, which is 5. Add together, add together, add together. So, the first 4 points are my when x of 2 k, and the second 4 points are x of 2 k plus 1, which is negative.

So, I know that is g 2 n is equal to x[n] minus x[n] plus N/2 . So, when I say this index n is so, in this case, this will be if you see the first one and last one negation minus ok.

So, instead of plus I make it minus. So, I get this one now. Once I get g 1 and g 2 n, then I know g 2 n has to be multiplied with W N n ok. So, n is varies from 0 to 4, so w to the power 0 1 2 3. So, here, see that the decimation in the time weight vector multiplies and then adds and subtracts. Here, first add and subtract, and then the weight vector multiplies and is propagated for the next section.

Again it will be 2 propagated for the next section because N/2 point. So, it will be W N 2 to the power square, so I get that jumbled version of the output. So, if you see the output sequence is in bit reverse mode, the output sequence is in bit reverse mode. So, if I want to access the output sequence, I have to do the bit reversal, and I get the sequence. So, when I want to do the x[0], I then x[1], I access the index in bit reverse mode, and I access that x value. So, this is called decimation in frequency. Is it clear?

(Refer Slide Time: 28:14)



So, decimation in time and decimation in frequency this is the two algorithms you have to know it and the maximum use cases is time and in some cases, frequency is also used. So, which one is more convenient for you to use? Now, if you see there is a radix 4 algorithm. What is the radix 4 algorithm? Radix 2 algorithm I know that N is N can be expressed as a 2^V, and radix 4 algorithm N is expressed in 4^V.

So, I can say if N is equal to L and M. So, if L is equal to 4, then I know M is equal to N/4 . So, I have divided the signal into 4 sequences instead of 2 sequences. I divided the N/2 sequence, and I divided the N sequence into a 4 sequence. Similarly, if I say decimation in time, so I divide the input signal in 4 sequences. When I say decimation in frequency, I will get the output as a division. So, in the input sequence, I am not jumbled.

It is as it is given an output sequence I get in jumbled order, which is in bit reverse mode. I access it ok. The most commonly used case is the Radix 2 algorithm. The most used algorithm is the Radix 2 algorithm.

So, I will stop here. So, I have explained the time domain of the radix 2 algorithm decimation in time and decimation in frequency. I will request all of you please implement it, and if you have any confusion, write down it and then learn it and try to develop an algorithm on it, and you can implement that algorithm using C programming.

Thank you, thank you very much.