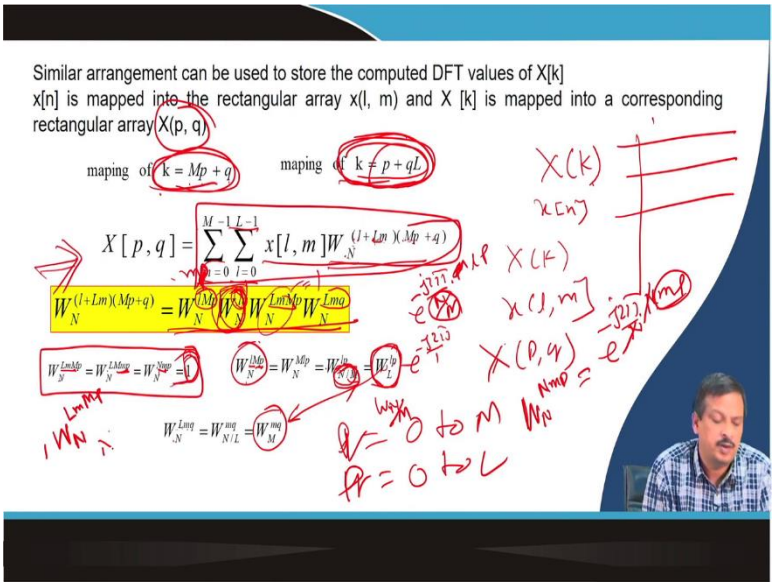
Signal Processing Techniques and Its Applications
Dr. Shyamal Kumar Das Mandal
Advanced Technology Development Centre
Indian Institute of Technology, Kharagpur

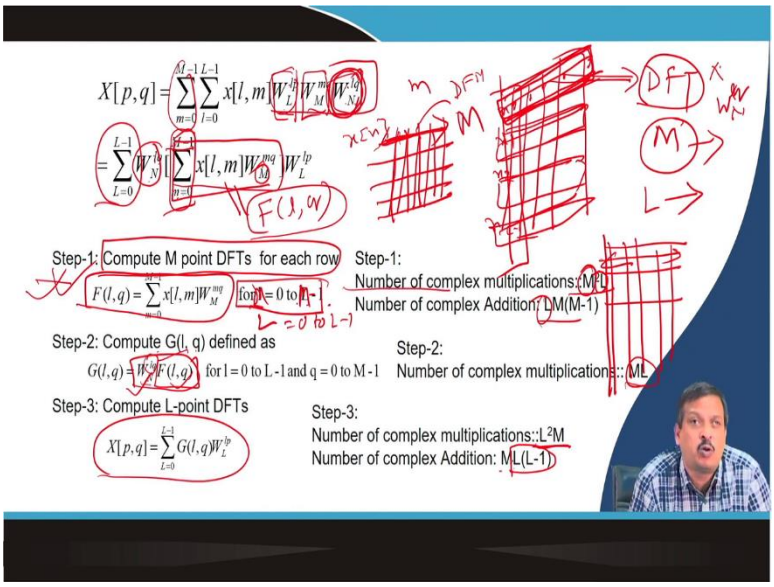
Lecture - 31
Fast Fourier Transform (FFT) Algorithms (Contd.)

(Refer Slide Time: 00:22)



So, in the last class, we discussed this slide. So, say this WN, this product can be simplified to 1 WL lp and WM mq.

(Refer Slide Time: 00:40)



So, instead of this one, I can write down that this is nothing, but a WL lp WM mq and WN nq N*lq will be as it is. So, WN lq and this part is 1 this part is 1. So, this part is nothing but a1. So, I write down the simplified form of the W using the periodicity property of discrete Fourier transform.

Now, if I say if I compute if you see here where the L of L has only WL lp and WN lq. So, I can say these two things, so let us say this one does not have any L, and this one does not have any m; m is not involved here. So, I can say this will go outside of the summation of my part.

So, what I said is although I store the data column-wise, I will first access the data in row wise because m is nothing but a represent the row, although I store the data X(0), X(1) , X(2) up to x L - 1, I access the data in column wise, compute the first DFT. So, I compute the first row's discrete Fourier transform; which order is M. Is it clear? So, although I stored the data column-wise in a two-dimensional array, I now compute DFT row-wise. So, I access the 1-row data and compute the discrete Fourier transform of order M.

Then I, oh so, I do it. Once I do it, then I take the all so, all column all row wise DFT is calculated first. Then I do column-wise DFT for L, which is the order L. So, if you see that I have a data x[n], I store the data column, then I access the data row-wise and compute the DFT. So, this row is computed, and this row is computed. Now, I again compute column-wise DFT.

So, see how the data is. It is not. I am taking the data from 0 to L - 1 for the first DFT. So, I access the data in row-wise. When storing it, I store it column-wise and then first compute the DFT. So, two times, I am computing the DFT; the first time, the order is M, and the second time, the order is L. So, the whole algorithm has 3 steps. So, step 1 computes M point DFT for each row. For each row, compute M point DFT, which is nothing but a F(l,q).

So, after this I can get F(l,q), l is fixed only q is done. Now, when I do so, I get this one and then define GLA, which is nothing but a product of N lq WN lq multiplied by F l, q. So, I first compute the DFT row wise then multiply each of the coefficients each of the each of the x values with a value of WN l lq, where l is also varies from 0 to L - 1 and q varies from 0 to M - 1 ok.

So, this product is done, then I compute L point DFT. So, then I compute column-wise DFT to get the output row-wise. The output of the discrete Fourier transforms we will store in row-wise. Because I have said X(l,q), q is stored in a wise mapping function, I have taken it that way. Now, you see what is the computational complexity you may say the process is very clumsy. So, it whether it will be computationally easy or it will be computationally more complex.

Now, if you see how much complex multiplication is required in the first step. So, M point DFT M point DFT. So, the first step is required. This will be m, m, which varies from 0 to m - 1. So, there is a mistake: m varies from 0 to m - 1. So, in the first step, we computed M point DFT. So, we have computed M point DFT for every row. How many rows are there?

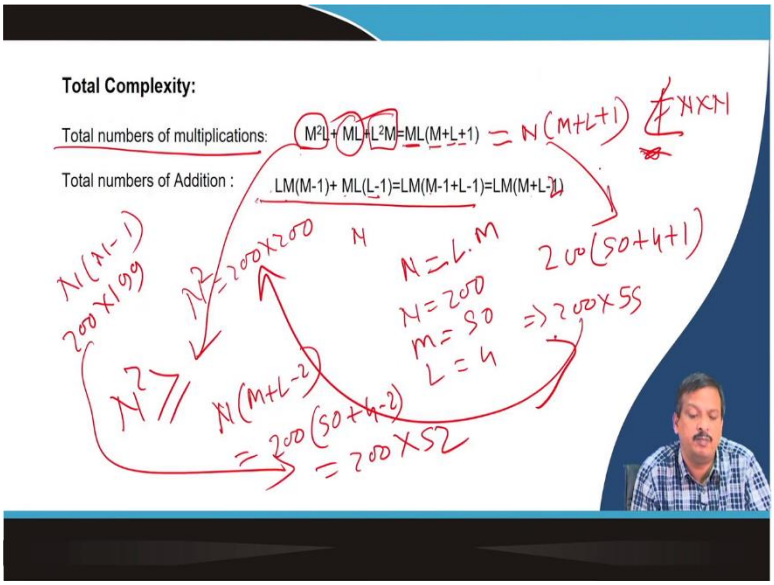
Sorry, this will be so this is DFT is m equal to 0 to M - 1, but how many times do I have to compute this DFT L time? So, L varies from 0 to L - 1. So, I said I store the data in column-wise and compute DFT for each row. How many rows are there? There are nothing, but there is an L number of rows. So, for every row, what is the complexity of the DFT capital M.? So, how many are the complex?

So, computing M point DFT, how much multiplication is required? M^2 multiplication is required. So, the M^2 multiplied by the L is the total number of multiplications required for computing the M point DFT of each row. Again, I will explain it by taking an example. So, I can say M^2 L is the computational complexity, and the number of additions required is nothing but an LM*M - 1 and M*M - 1 for each row.

How many rows are there? L number of rows are there so multiplying by L. Now, L step 2 what is the computational complexity, I have to multiply WN lq with this F l q both are complex. So, I require M*L number of complex multiplication, and then in the third step here, I am computing DFT column-wise. So, for each column, the computational complexity is L^2, and there is an M number of columns. So, computational complexity L^2 M.

Computational addition is the number of radiation L*L - 1 for each DSP and their DFT. There is an M number of DFT, so M*L - 1.

(Refer Slide Time: 09:14)



Now, if I compute the total computational complexity and total number of multiplication for row-wise M^2 L for second stage ML for third stage column-wise L^2 M. So, if I take the ML is common, it is nothing, but an M + L + N which is nothing, but a N*M + L + 1 which is not N*N.

It is less than N*L. Why? Because if I say N is not a prime number and it can be divided*L*M so, let us say N is equal to whatever the example I have given 200. So, m is equal to, let us say, 50, then L is equal to 4. So, if it is N^2, then it is 200*200.

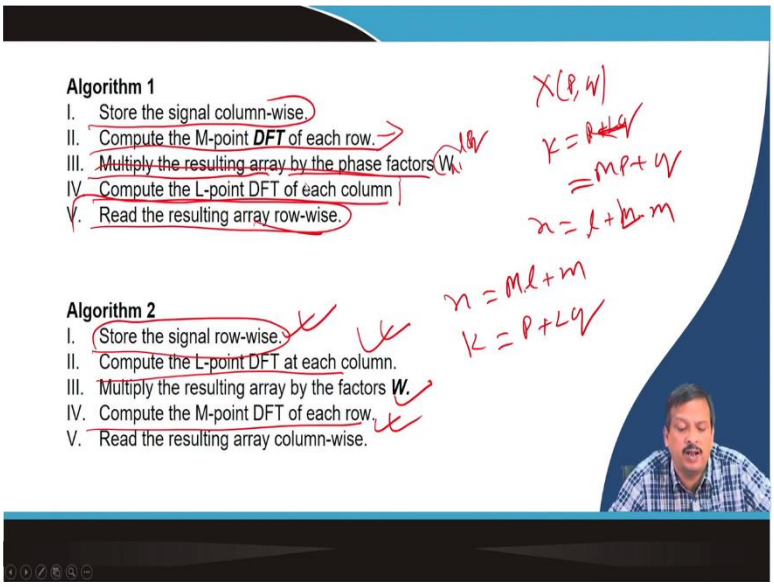
Now, what is the computational number of computational complexity here it is 200*50 + 4 + 1. So, it is nothing but a 200*55, which is much, much less than this one, so this is much less than this one. So, this N^2 is much greater than this one.

Similarly, how many total additions are required? Only a two-stage addition is required. The second stage I does not have any required addition, so LM*M - 1 ML*L - 1. So, I can say N*M + L - 2.

So, what is nothing, but if, in this case, for example, it will be N means 200*50 + 4 - 2 is equal to 200*52, which is much much less than N*N - 1, that is nothing but a 200*199, so I am reducing. So, somehow, if I am able to implement the divide and conquer methods I am in, I will efficiently compute the discrete Fourier transform, which is much, much less multiplication complex compared to the first one.

But what is required is that I have to store the data column-wise, then I have to compute a discrete Fourier transform row-wise, accessing row-wise first and then column-wise. So, it is a stage process, or I can say I can store the data row-wise and first compute the discrete Fourier transform column-wise and then row-wise. So, if I store the data column-wise, I have to first compute row-wise; if I store the data row-wise, I have to first compute column-wise.

(Refer Slide Time: 12:43)

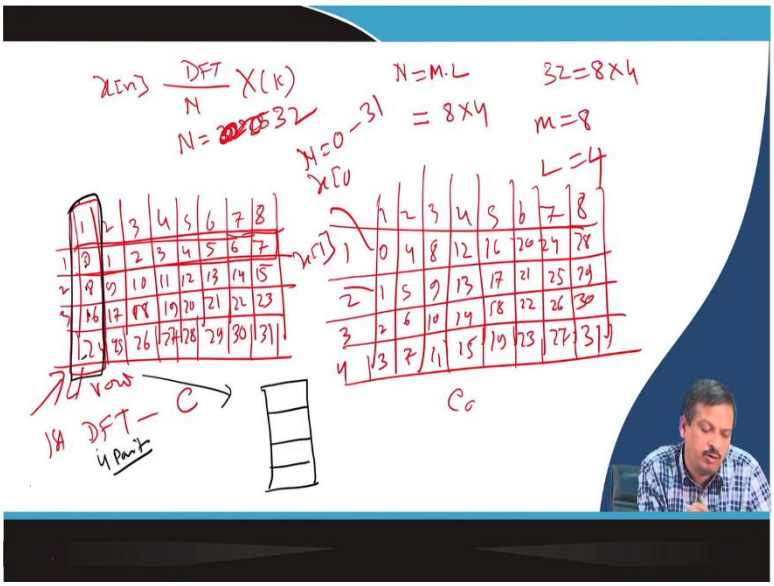


So, there are 2 algorithms that are possible. So, algorithm 1 stores the signal column-wise, computes M point DFT for each row, multiplies the resulting array by a phase vector WN lp and then computes the L point DFT for each column, reads the resulting array row-wise, reads the result row-wise. As I said, when I write X p and q, so K is in the form of P + Mp + q, the form we have taken for that mathematical derivation this form we have taken here is Mp + q.

So, that is why the output will be stored in a row. Wise now you can say the store the signal row-wise. So, here, K is this form, and n is l + capital M capital L*m form. Here, I can reverse that n is equal to capital M*l + small m and K is equal to P + Lq. Store the signal row-wise, compute the DFT column multiplied by W, and compute the M point DFT row-wise.

And read the result column-wise. So, both are possible, so let us give an example for your better understanding. So, let us use an example for this purpose.

(Refer Slide Time: 14:26)



So, let us say I have a x[n] and I want to compute discrete Fourier transform of n, then I get X K. Let us say N is equal to, let us say, 32 or let us say 25, let us say 20 purposefully I think I am taking 25 N is equal to 25. So, if N is equal to 25, how can I express N in terms of the M*L product of two numbers so N is a non-prime number?

So, I have to express myself in M and then in L form. So, I can say no if I say this is okay for both sides. Let us say 5*5, both sides will be equal, and L and M will be equal. So, instead of equal, I want to take, let us say 32. So, 32 means I can express in 6 8*4.

So, N is equal to 32, so 32 can be expressed as 32 is equal to 8*4, so I can say M is equal to 8 and L is equal to 4. So, if I want to compute the apply divide and conquer methods I can say if I want to store the data. So, there is a M number of columns 1, 2, 3, 4, 5, 6, 7, 8. So, there is an 8 column starting from 1 to 8, and there is a 4 number of rows: 1, 2, 3, 4.

Now, I have to store my x[n] if I store my x[n] row-wise, so there will be an index 0, 1, 2. So, I am writing only the index 3, 4, 5, 6, 7. So, first, 7 samples will be written in the first row second 7 samples; that means 8, 9, 10, 11, 12, 13, 14 and 15; then I have 16, 17, 18, 19, 20, 21, 22, 23 then 24, 25, 26, 27, 28, 29, 30, 31. So, N varies from 0 to N - 1, which means 31, so 31 samples I have written.

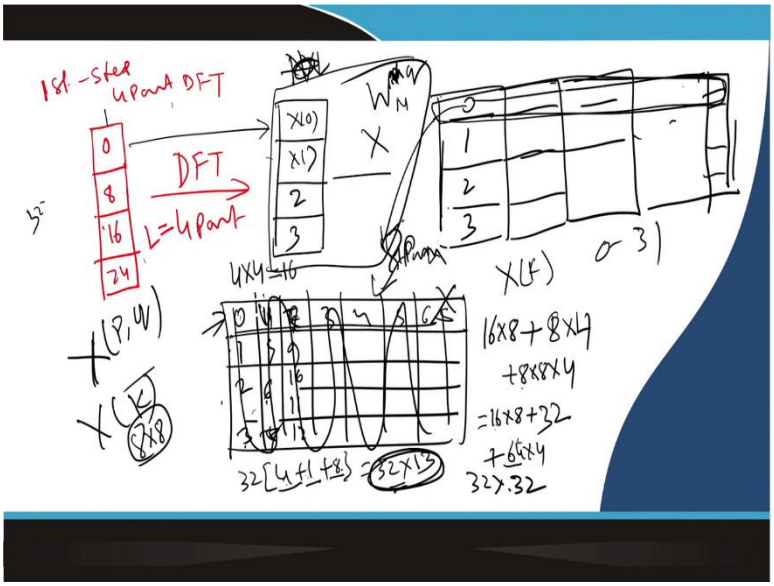
So, I have written the sample row. I can also write the sample column-wise, 1, 2; 1, 2, 3, 4, 5, 6, 7, 8; 8 column and 4 rows 1, 2, 3, 4; 1, 2, 3, 4. So, again I can write this way also

0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31. So, those numbers represent the index.

So, basically, this is x 0, and this is X(1) . So, I store the data row-wise in this case column. Okay, so I just store the data row-wise and column-wise as much as possible. Let us say I want to compute when I store the data row-wise. So, if I store the data row, the first step is to compute DFT column-wise. So, in the first column, DFT, I have to compute. So, I have to compute DFT for this column.

So, if I want to compute this column DFT. So, after computing the DFT, I will store the DFT in columns 1, 2, 3, and 4. So, I accessed those samples; that means the 0th sample, 8th sample, 16 samples, and 24 samples. I access those samples and compute 4-point DFT. So, I am computing the first step, which is to compute 4-point DFT. Let us try to go to the next slide. So, here I am computing the first step, 4 point DFT column.

(Refer Slide Time: 20:09)



So, the data is in red, so let's say the first column is data. So, this is the 0th index data, this is the 8 index data, this is the 16 index data, this is the 24 index data, I think so, yes. So, I have taken this data, and I am computing the DFT discrete Fourier transform of 4 points, which means L is equal to 4 now. So, what I will get is transform data, so transform data is also stored in a column.

So, this is X(0) X(1), so the index is one capital X(2) X(3). This will be multiplied with WN lq ok. So, instead of here lq, yeah, here will be mq because I am computing column-wise, so it will be mq. Then, the again data will be stored in columns 0, 1, 2, and 3. Now, again, for the next column data, then for the next column data so I have a how many columns there will be 8 columns so there will be 8 such column.

Now, I am computing the DFT row-wise. So, here, data is stored row-wise first, and then I compute the DFT column-wise, then again, so when I compute it, so what? I will get the first row, so 4 point DFT. So, there will be 8. How many columns are there?

So, in the column, there will be 8 cells in each row: 1, 2, 3, 4, 5, 6, 7, and 8. So, when I compute DFT row-wise, I have to compute 8 points DFT, so I get again, let's say, 8. So, 1st row, 2nd row, 3rd row, and 4th row I will get, and then I have to access the data X of when I represent X(p,q) or when I said X(K), then this will be 0, 1, 2, 3, 4, 5 not 6, 8 it will be not like that.

It will be since I stored the data in columns, row-wise it will be 0, 1, 2, 3, 4, 5, 6, 8, 7, 8 like that 0, 1, 2, 3, 4, 5, 6, 7, then 8, 9, 10, 11, 12 like that way, that way I will get how many X(K) also varies from 0 to 31. Now, do I have the computationally efficient algorithm, or is it not? So, how much multiplication is required at this stage?

It is N cross L, so N is the length of the DFT is 4 points. So, for each DFT, I require 4 and 4; how many columns are there? It is an 8-column. So, I can say 4*4 is 16, so for each column, I require 16 numbers. So, in 8 columns, I require 16*8 numbers. Plus, what do I require? I require an L*M number of multiplication here. also, once I get that matrix, I require an L*M number of multiplication.

So, I also require here L, so M is 16. What is my M? M is 8, and L is 4. In the last stage, for each row, I have to compute 8-point DFT. So, each DFT computation is 8*8, and how many rows are how many rows are there 4 rows + 8*8*4? So, what does it become? 16*8 + 32 + 8*8, 64*4, and this is 16. If I say that it is 32, if you have taken out, then it becomes 16*8. So, it will become 4 + 1 + 8.

So, it is nothing but a 32*13 ok. So, if I take 32, it is common, so it remains 4; 4 16*2 is 32. So, remain 4 32 go out 1 this will be 32 go out remain 2 so 2*4 is 8 so 4 + 8 + 1 so it is nothing, but 32*13. If I directly compute DFT, what is the computational complexity

32*32, 32*32, which is much, much less here it is much much less. So, I have efficiently computed the DFT using this divide-and-conquer rule.

Now, in the next class, I will talk about the radix algorithm. Here, I am talking about the divide-and-conquer rule. I use the same principle, but I use different radix to compute this efficiently. I will use the same principle, periodicity and symmetry, and the divide and conquer rule, and then I will implement a fast Fourier transform, which will be much more efficient. You can say this will come in the order of $n\log_2 n$, or if it radix is 4, then n log 4.

So, 4 n that, I will show you what I will discuss in the class in this syllabus. I will discuss radix 2 full that time dissemination and frequency decimal radix 2 full, and I will just touch the radix 4 because radix 4 may not be that much of required. In most of the cases, we implement the radix 2 algorithm.

Thank you.