Architectural Design of Digital Integrated Circuits

Prof. Indranil Hatai
School of VLSI Technology
Indian Institute of Engineering Science and Technology, Shibpur, Howrah

Lecture – 49
FFT Architecture

Hello everyone. Welcome to the course on Architectural Design of ICs. So, today we will see the FFT Architecture. So, FFT means the Fast Fourier transform. This is one of the most commonly used blocks in, this in mainly for the digital signal processing. So, it has the wide application like in image processing or in wireless communication or in video processing.

So, to design the hardware for let us say for video processing or for audio processing or for image processing you need to design the FFT block ok. So, for that reason you have to know about the FFT architecture. So, today we will see, in these particular lecture we will see the FFT architecture what are the architectures are already there and still people are trying for several decades to design one efficient architecture for a fast Fourier transform.

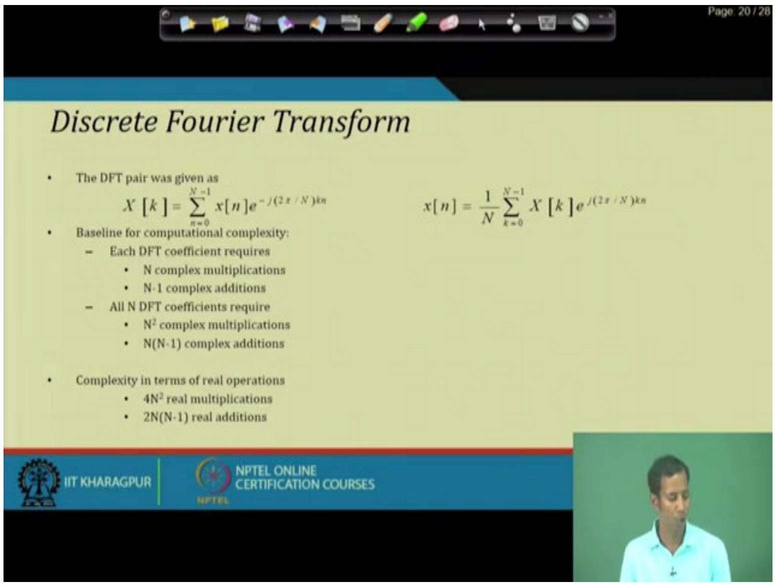
That means why I need this efficient means actually some of the architecture is related to increase the; that means, the computation time or sorry, not; that means, to decrease the computation time so that the throughput of the FFT becomes much more on the higher side; that means, we can process it very faster.

As well as some of the architecture has basically designed with respect to; that means, for the low power application so as a FFT means we have to actually there the memory access is much more so, how to reduce these memory access. So, that it can become or this will be much more applicable for low power applications.

So, those types of architectures are already there. So, various kinds of architectures is already available, but we will see just the basics or the; that means, very few basic architecture of a FFT in these particular; that means, lecture series.

If you want to know more; that means, if you are more interested on FFT architecture the present days you can find it in IEEE website or you can find it in; that means, Google scholar. Or if you; that means, if you and if you are not getting it then please let me know then I will definitely I provide the corresponding information about the recent days FFT architecture, how the people have already developed or still people are basically trying to develop this high performance, highly efficient FFT architecture ok.

(Refer Slide Time: 03:14)



Discrete Fourier Transform

- The DFT pair was given as

$$X[k] = \sum_{n=0}^{N-1} x[n] e^{-j(2\pi/N)kn} \qquad x[n] = \frac{1}{N}\sum_{k=0}^{N-1} X[k] e^{j(2\pi/N)kn}$$

- Baseline for computational complexity:
  - Each DFT coefficient requires
    - N complex multiplications
    - N-1 complex additions
  - All N DFT coefficients require
    - N^2 complex multiplications
    - N(N-1) complex additions
- Complexity in terms of real operations
    - $4N^2$ real multiplications
    - 2N(N-1) real additions

So, at the first actually we have to consider these Fourier transform which came for digital signal passing is these discrete Fourier transform. So, this is the; that means, equation for these generic discrete Fourier transform.

So, whenever we in actually it the Fourier transform has been started with these discrete Fourier transform, and whenever the people have tried to design this DFT. So, at the time the baseline for these computational the hardware requirement; that means, the computational complexity of these DFT requires, N complex multiplication and N minus 1 complex addition for 1 N point DFT.

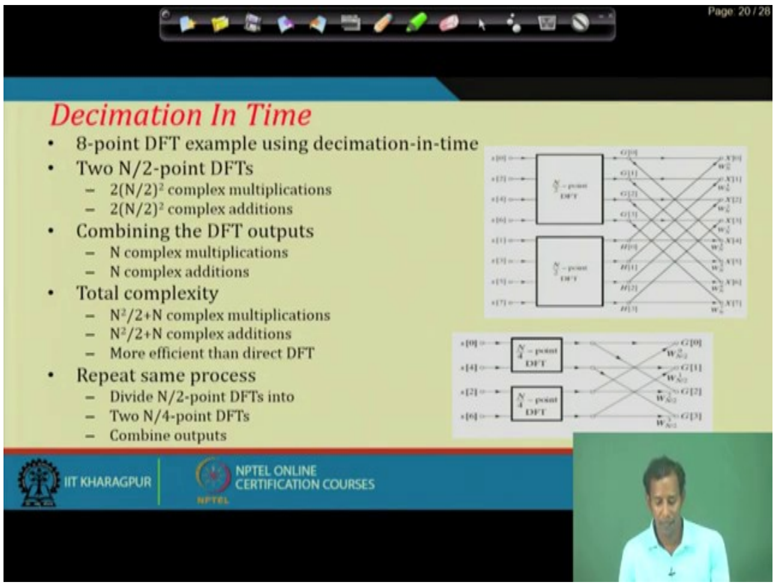
In all N DFT coefficients requires N squared number of complex multiplications and N into N minus 1 number of complex additions. So, therefore, the complexity in terms of real operations it requires 4 N square number of real multiplication and 2 N into N minus 1 real addition.

What does it means is that, to implement this particular function suppose for N point DFT I need the computational overhead; that means, their hardware requirement for these to design 1 N point DFT the requirement is 4 into N square. Let us say if I want to design one 8 point

FFT so at that time so 8 square means 64 into 4 so, 256 number of real multiplications I need, as well as. So, 56 into 2 means 112 number of real additions operation I require.

So, if I consider that 8 point FFT and the FFT lengths of each of this 8 bit so at that time it requires a huge number of hardware requirements ok. So, these as we actually use more number of hardware; that means, more number of multipliers and more number of addition operation so at the time this will consumes more power as well as the speed will be also on the lower side.

(Refer Slide Time: 05:54)



Decimation In Time

- 8-point DFT example using decimation-in-time
- Two N/2-point DFTs
  - $2(N/2)^2$ complex multiplications
  - $2(N/2)^2$ complex additions
- Combining the DFT outputs
  - N complex multiplications
  - N complex additions
- Total complexity
  - $N^2/2+N$ complex multiplications
  - $N^2/2+N$ complex additions
  - More efficient than direct DFT
- Repeat same process
  - Divide N/2-point DFTs into
  - Two N/4-point DFTs
  - Combine outputs

$\frac{N}{2}$ - point DFT
$\frac{N}{4}$ - point DFT
$G[0]$
$G[1]$
$G[2]$
$G[3]$
IIT KHARAGPUR
NPTEL ONLINE CERTIFICATION COURSES
Page: 20 / 28

So, that is why people have actually in the earlier days the DFT actually it has been done in decimation time. So, in decimation time we have just divide the DFT into two part where the half of these is basically done considering these N by 2 point DFT and the lower portion is done by N on N by 2 point DFT ok.

Now, the thing is that after actually dividing these coefficients into even and odd we can do these butterfly operations, which is basically related to the corresponding twiddle factors multiplications. This is very much related to this particular term which is e to the power j 2 pi by N into k n.

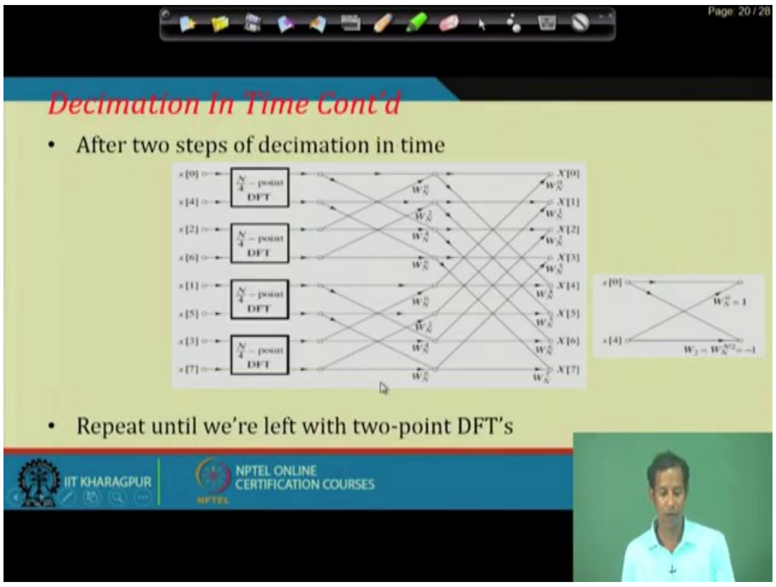
So, now for these 2 N by 2 point DFTs we need 2 into N by 2 square number of complex multiplication as well as 2 into N by 2 number of 2 square number of complex addition. So, combining these two; that means, DFT outputs we can get; that means, which requires total number of N complex multiplication as well as N complex addition. So, the total complexity at that time it becomes N square by 2 plus N number of complex multiplication and N square by 2 plus N number of complex addition.

So, now in these particular case if we consider that ok, we need; that means, 8 point FFT if again we consider so at that time 32 plus 8. So, 40 number of complex multiplication we require if we actually divide the coefficient; that means, if we divide these the corresponding; that means, sample in even term and then odd term and both if we do the; that means, if we run these computation of the DFT in parallel so at that time the requirement for these complex multiplication and complex addition they become little bit lower than the original DFT implementation ok.

So, actually this is, if we consider only about; that means, radix 2 radix 2 means as we have only read if only divided 2 part; that means, the even part and then the odd part. If we increase the radix; that means, if we consider 4 parallel DFT which is running at N by 4 point DFT so at that time the corresponding computation time that will be much more faster ok.

So, we will see how these actually the computation time at that time becomes lower whenever we consider more of the radix and if we consider more of the radix then what also it; that means, what it happens then that we will also; we will also see.

(Refer Slide Time: 09:25)



Decimation In Time Cont'd

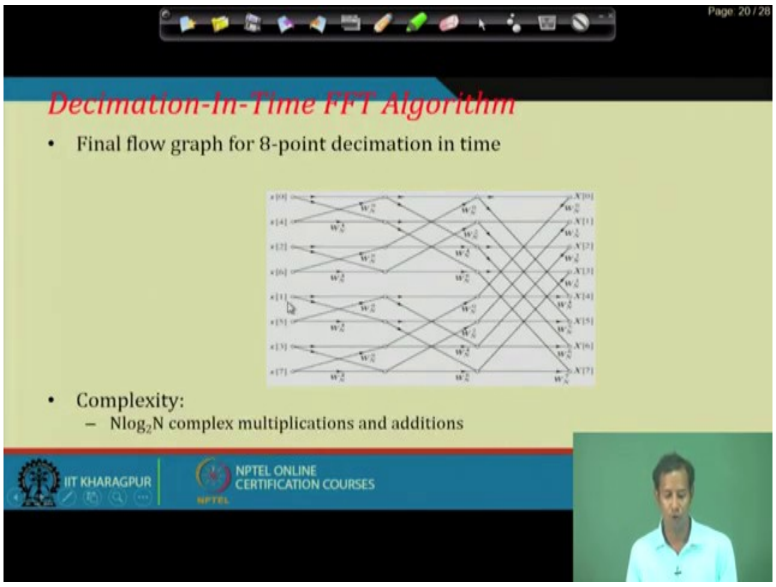
- After two steps of decimation in time

- Repeat until we're left with two-point DFT's

Now, depending on these 4 point where the radix 4 DFT in decimation in time what we can do is that, we have in these particular N by 4 point DFT what we have considered we have considered x 0 x 4 x 2 x 6 x 1 x 5 and x 3 x 7. Then they are being basically here we are doing these butterfly operation which is nothing, but these twiddle factor multiplication which is coming because of these e to the power j 2 pi by N because of this that particular factor.

So, whenever we will do at that time that corresponding functions it will give you two parts one is the real part another one is the imaginary part ok. So, in these butterfly operation you can see that this these particular actually node is multiplied with the coefficient values of 1 and these particular actually branch or these particular edge is basically multiplied with the term of minus 1 ok. So, these values of 1 and minus 1 they are being calculated or computed based on the corresponding twiddle factors ok.

(Refer Slide Time: 10:58)



Decimation-In-Time FFT Algorithm

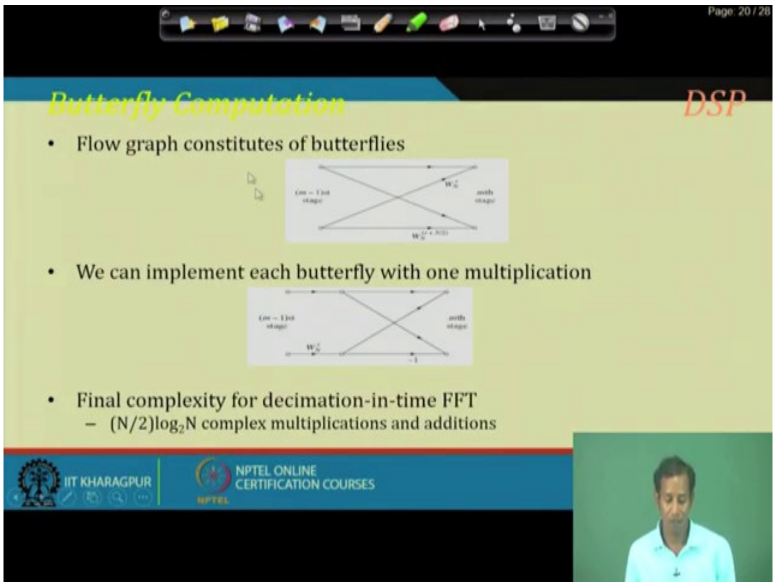
- Final flow graph for 8-point decimation in time
- Complexity:
  - $N\log_2 N$ complex multiplications and additions

So, now and these will be actually repeated until and unless we are getting these 2 point DFTs. So, this is another example of this decimation in time these Fast Fourier transform and basically fast Fourier transform is nothing, but these division of these N by 2 point; that means, DFT. In that means, if we divide these two; that means, odd part and the even part and then if we run them on parallel computation that is the, and by that technique we increase the computation; that means, we increase the throughput of the Fourier transform, which is nothing, but this fast Fourier transform.

And whenever we do like these so at that time the complexity becomes N into log 2 base N number of complex multiplication and addition well, what is this N? N is nothing, but the corresponding number; that means, the point; that means, N point DFT or N point FFT what we require. So, in these particular case as we have considered 8 point; that means, FFT so at that time the corresponding complex multiplication and the addition requirement will be 8 into log 2 base N equals to; that means, 24 number of complex multiplication and addition operation.

And here you consider each of these is basically requires these multiply and accumulation; that means, operation.

(Refer Slide Time: 12:38)



Butterfly Computation

DSP

- Flow graph constitutes of butterflies
- We can implement each butterfly with one multiplication
- Final complexity for decimation-in-time FFT
  - $(N/2)\log_2 N$ complex multiplications and additions

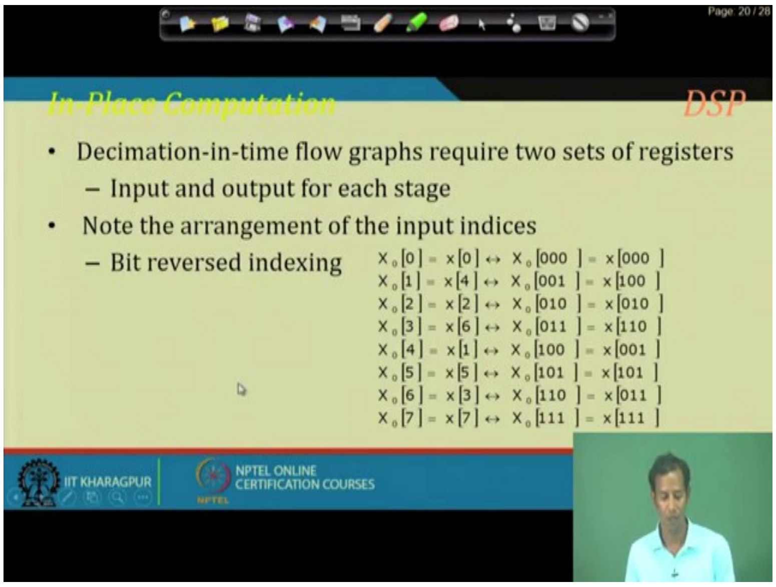
Now, what is these butterfly computation? So, the butterfly computation is that it actually generates from the m minus 1 stage, it has to multiply it with the corresponding twiddle factors to generate the next stage, which is the m th stage ok. So, in these upper ward actually this upward; that means, branching it has to multiplied with W dash W r N whereas, this lower branch has to multiply with the twiddle factor values of W N r plus N by 2.

So, what we can do this upper branch will be actually depending on the this value if we know the N so at the time we know what will be the values of these W N and W N r plus N by 2. So, from there we can we know that these lower branch will be this value of these will be minus 1 and these value will be the 1. So, the final actually complexity for decimation in time

FFT will be if we consider this kind of butterfly computation so at that time it will be N by 2 into log 2 base N complex multiplication and addition operation.

So; that means, in the earlier case in these case we are having 8 into 3 for 8 point FFT so 24 number of complex multiplication and addition operation so if we just constitute this butterfly unit and if we use these corresponding these the twiddle factors values as 1 and minus 1. So, at that time the requirement for this multiplication complex multiplication and these addition will become 4 into 3 means only 22 sorry, only 12 at that time.

(Refer Slide Time: 14:41)



Page: 20 / 28

In-Place Computation

DSP

- Decimation-in-time flow graphs require two sets of registers
  - Input and output for each stage
- Note the arrangement of the input indices
  - Bit reversed indexing

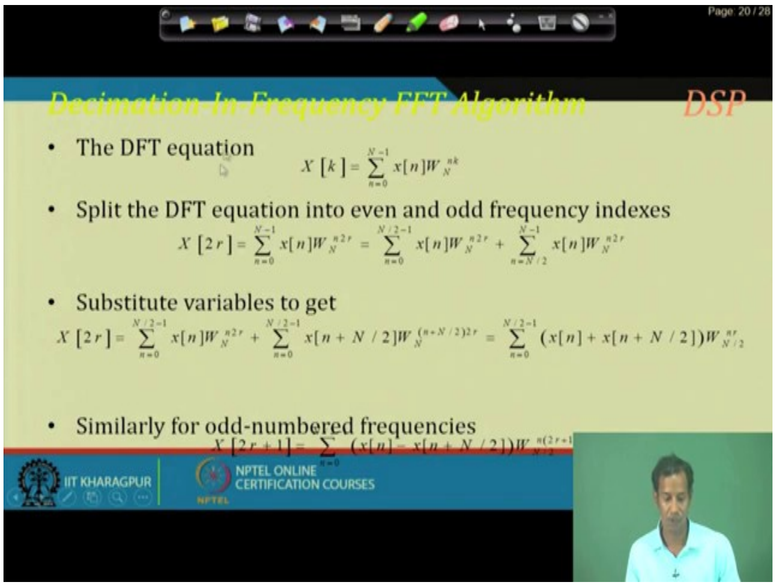
$$\begin{aligned}
X_0[0] &= x[0] \leftrightarrow X_0[000] = x[000]\\
X_0[1] &= x[4] \leftrightarrow X_0[001] = x[100]\\
X_0[2] &= x[2] \leftrightarrow X_0[010] = x[010]\\
X_0[3] &= x[6] \leftrightarrow X_0[011] = x[110]\\
X_0[4] &= x[1] \leftrightarrow X_0[100] = x[001]\\
X_0[5] &= x[5] \leftrightarrow X_0[101] = x[101]\\
X_0[6] &= x[3] \leftrightarrow X_0[110] = x[011]\\
X_0[7] &= x[7] \leftrightarrow X_0[111] = x[111]
\end{aligned}$$

IIT KHARAGPUR | NPTEL ONLINE CERTIFICATION COURSES

So, in decimation in time flow graphs require two sets of registers one for the input and other for the output for each stage. And note the arrangement of the input indices are in bit reversed in indexing.

Bit reverse indexing, means; the output the if I apply these input at x 0 x 1 x 2 x 3 x 4 x 5 x 6; that means, this is the input sequence the output from this decimation in time FFT the output will become in a bit reverse manner which is nothing, but this x 0 then x 4 then x 2 then x 6 then x 1 then x 5 then x 3 then x 7 something like these; that means, 0 this will.

(Refer Slide Time: 15:38)



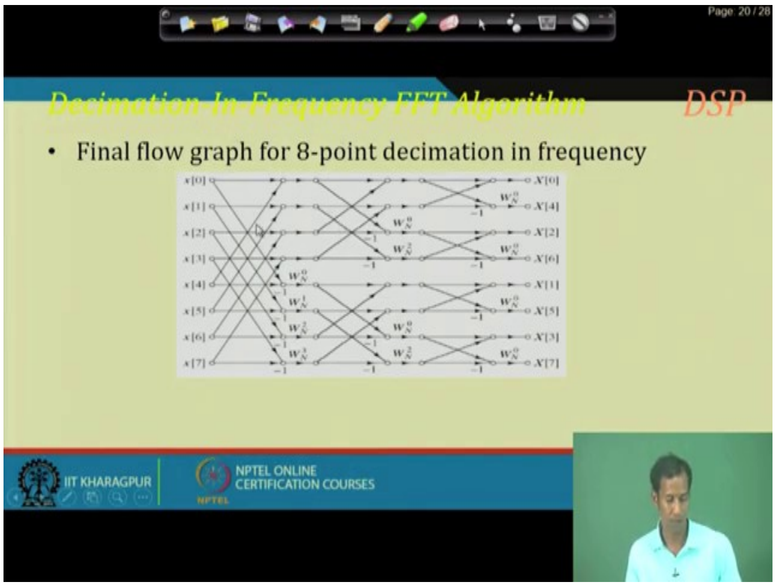
Decimation-In-Frequency FFT Algorithm

DSP

- The DFT equation $X[k] = \sum_{n=0}^{N-1} x[n] W_N^{nk}$
- Split the DFT equation into even and odd frequency indexes
$$X[2r] = \sum_{n=0}^{N-1} x[n] W_N^{n2r} = \sum_{n=0}^{N/2-1} x[n] W_N^{n2r} + \sum_{n=N/2}^{N-1} x[n] W_N^{n2r}$$
- Substitute variables to get
$$X[2r] = \sum_{n=0}^{N/2-1} x[n] W_N^{n2r} + \sum_{n=0}^{N/2-1} x[n+N/2] W_N^{(n+N/2)2r} = \sum_{n=0}^{N/2-1} (x[n] + x[n+N/2]) W_{N/2}^{nr}$$
- Similarly for odd-numbered frequencies
$$X[2r+1] = \sum_{n=0} (x[n] - x[n+N/2]) W_{N/2}^{n(2r+1}$$

So, these decimation in; that means, FFT algorithm. So, this is the initial DFT equation. Now if we split this DFT equation into even and odd indices; that means, frequency indices then these particular equation can be written as n 0 to N by 2 minus 1 summation of these n 0 to N by 2 minus 1. Where x n into W N into n 2 to 2 r whereas, the; that means, this is the even part and this is the odd part; that means, the DFT now it has been split into two. So, that I can run parallelly ok.

(Refer Slide Time: 16:29)



Decimation-In-Frequency FFT Algorithm

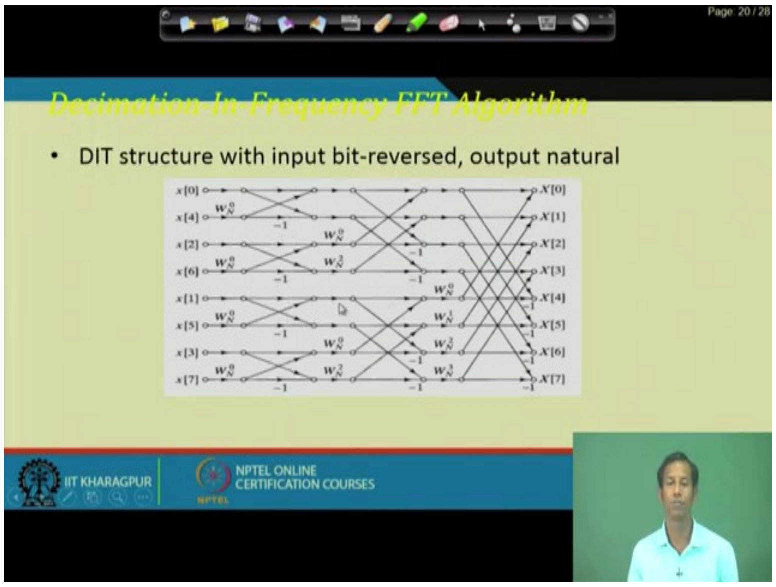
DSP

- Final flow graph for 8-point decimation in frequency

So, then this the corresponding graph for this decimation in time sorry, decimation in this is. The earlier one was the decimation in time so this is for decimation in frequency. So, in decimation in frequency what we have to give is that the input should be in the in order whereas, I will get the corresponding output in the bit reversed manner. So; that means, this is not the act, so x 0 is producing x 0 the x 1 is then its producing x 4, x 2 is producing x 2, x 3 is producing x 6, x 4 is producing x 1 so something like that the output will come comes in the bit reverse manner.

So, at the time to get the output should be also like x 0 x 1 x 2 x 3 x 4 x 5 something like that. So, you have to do some bit reversing; that means, you have to do the bit reversal so that these particular sequence became in order ok.

(Refer Slide Time: 17:45)



Decimation In Frequency FFT Algorithm

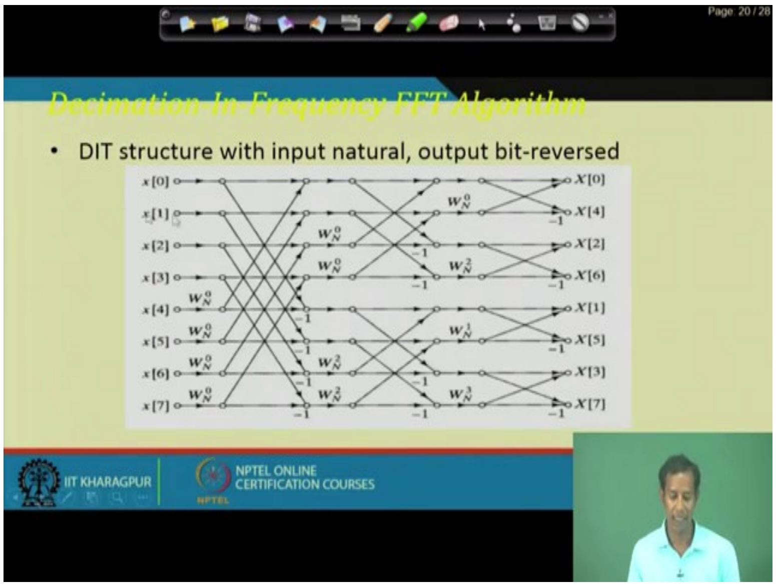
- DIT structure with input bit-reversed, output natural

So, in decimation in time actually then this DITs; that means, decimation in time structure with the input bit reverse the output will be natural; that means, if we apply the input in a bit reverse manner which is these x 0 x 4 x 2 x 6 x 1 x 5 x 3 x 7 so at that time the output will be comes in the bit reverse manner.

So, it; that means, you can do in; that means, in either you can put either you can give the input in the original sequence you get the output at the bit reversed manner or you put the input in the bit reverse manner. So, that the output would comes out from these that FFT block which will be in original or in original sequencing or in particular order ok.

So, and here you see all these; that means, these particular the lower branch they are basically multiplied these, butterfly these cross multiplication is known as the butterfly operation so the butterfly lower branch of these butterfly they are being multiplied with minus 1 whereas, the upper branch has been multiplied with positive 1.

(Refer Slide Time: 19:14)



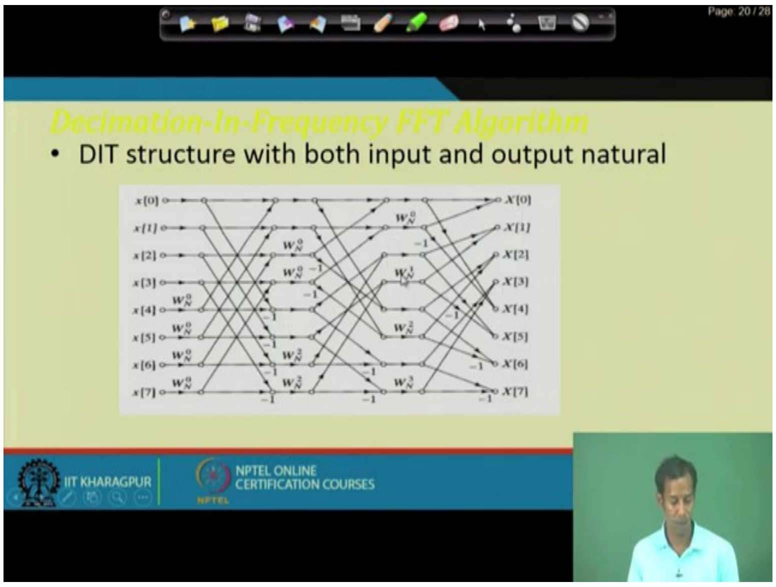
Decimation-In-Frequency FFT Algorithm

- DIT structure with input natural, output bit-reversed

Now, whenever these in so this is in decimation in time and in case of decimation in frequency the input is in original sequence, but the outputs are coming in bit reverse manner. So, here you see the difference is that initially actually we do these; that means, the radix 2 butterfly ok. And then actually it has to multiplied with initially the x 0 with x 4 then x 2 with x 6 x 1 with x 5 x 3 with x 7.

Then in the next these x 3 x 1 x 7 x 5 so it has to multiply with that, then again this x 1 has to multiplied with x 0 x 5 we has to multiply with x 4 x 3 has to multiplied with x 2 so x 7 has to multiplied with x 3 ok. So, which will be just reversed in case of decimation in frequency ok.

(Refer Slide Time: 20:29)

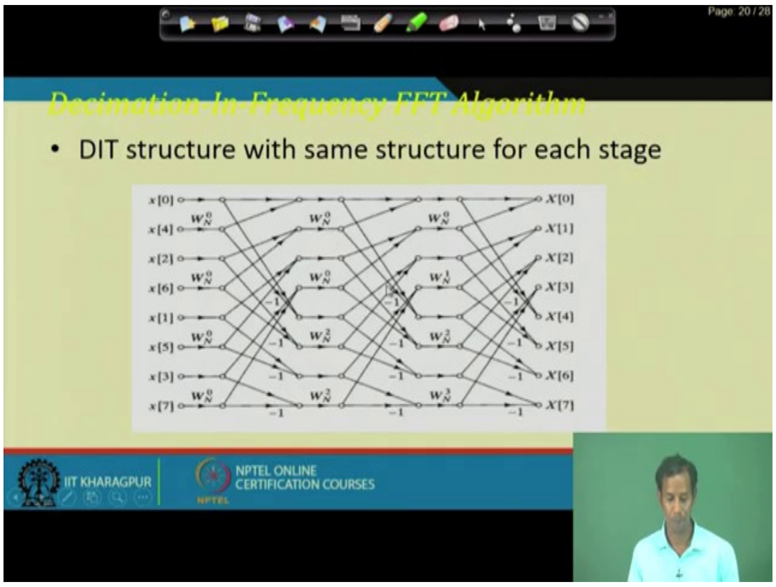


Decimation-In-Frequency FFT Algorithm

- DIT structure with both input and output natural

So, in decimation in frequency case so that the initial computation will be x 0 with x 4 then x 1 with x 5 x 2 with x 6 x 3 with x 7 and then we will get the corresponding output something like this.

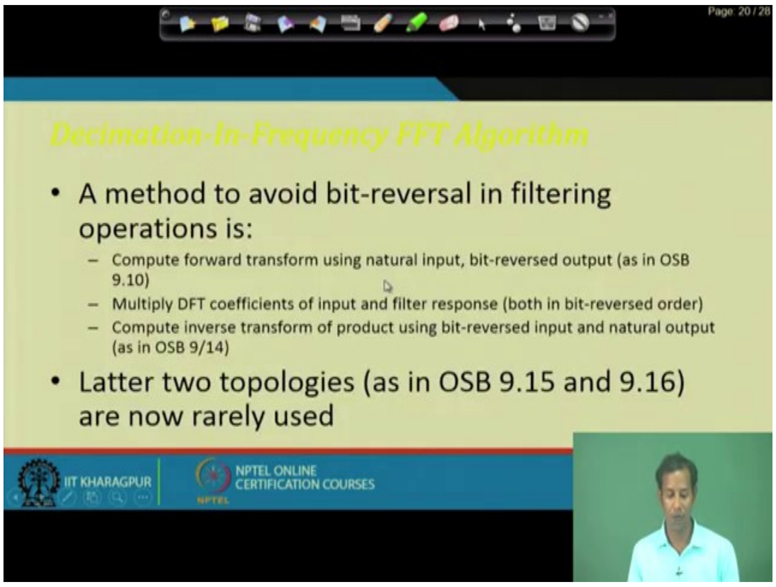
(Refer Slide Time: 20:48)



Decimation-In-Frequency FFT Algorithm

- DIT structure with same structure for each stage

(Refer Slide Time: 20:51)



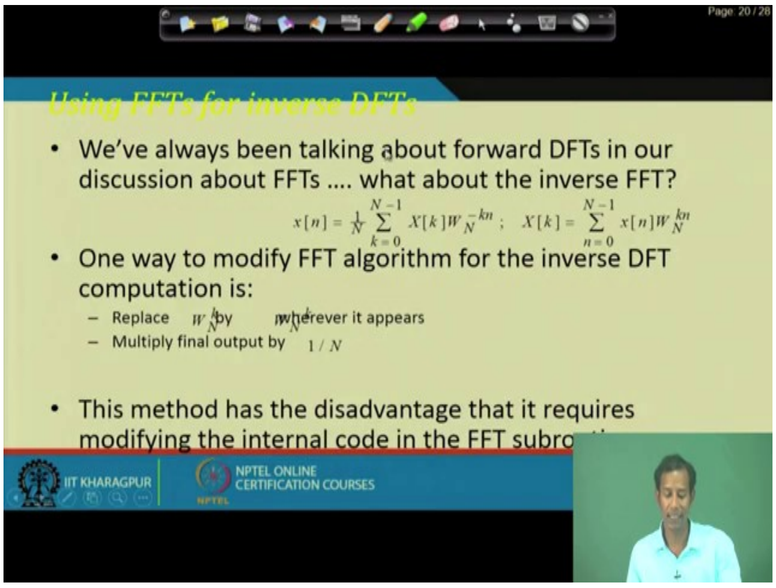
Decimation-In-Frequency FFT Algorithm

- A method to avoid bit-reversal in filtering operations is:
  - Compute forward transform using natural input, bit-reversed output (as in OSB 9.10)
  - Multiply DFT coefficients of input and filter response (both in bit-reversed order)
  - Compute inverse transform of product using bit-reversed input and natural output (as in OSB 9/14)
- Latter two topologies (as in OSB 9.15 and 9.16) are now rarely used

IIT KHARAGPUR NPTEL ONLINE CERTIFICATION COURSES

So, the decimation in frequency FFT algorithm it has to; actually this is a method to avoid; that means, what method we will adopt to avoid these bit reversal in filtering operation is to compute the forward transform using natural input and bit reverse output. And these multiply these DFT coefficients of input and filter response. Compute these inverse transform of the product using bit reversed input and the natural output.

(Refer Slide Time: 21:33)



Using FFTs for inverse DFTs

- We've always been talking about forward DFTs in our discussion about FFTs .... what about the inverse FFT?

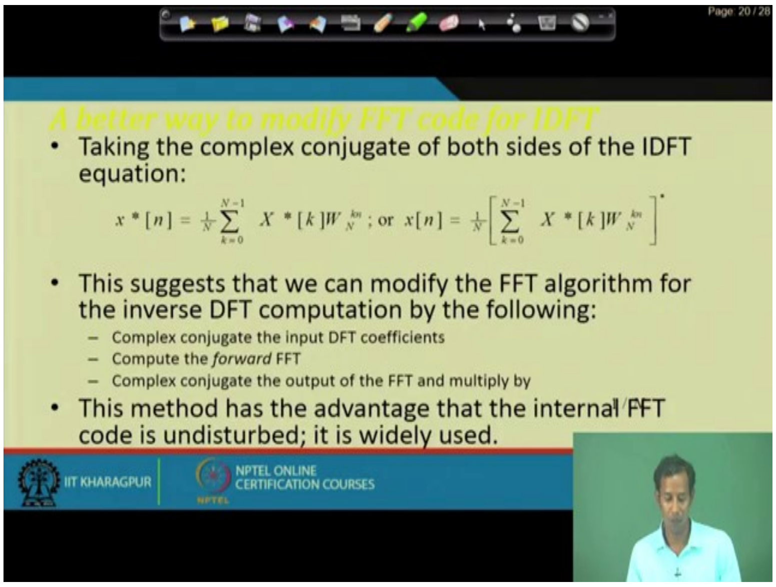
$$x[n] = \frac{1}{N}\sum_{k=0}^{N-1} X[k]W_N^{-kn}; \quad X[k] = \sum_{n=0}^{N-1} x[n]W_N^{kn}$$

- One way to modify FFT algorithm for the inverse DFT computation is:
  - Replace W_N^k by W_N^{-k} wherever it appears
  - Multiply final output by $1/N$
- This method has the disadvantage that it requires modifying the internal code in the FFT subro

IIT KHARAGPUR NPTEL ONLINE CERTIFICATION COURSES

And, now so whatever we are considering that is only for the forward DFTs or forward FFT we have only considered, but what about the inverse FFT ok?.

(Refer Slide Time: 21:55)



A better way to modify FFT code for IDFT

- Taking the complex conjugate of both sides of the IDFT equation:

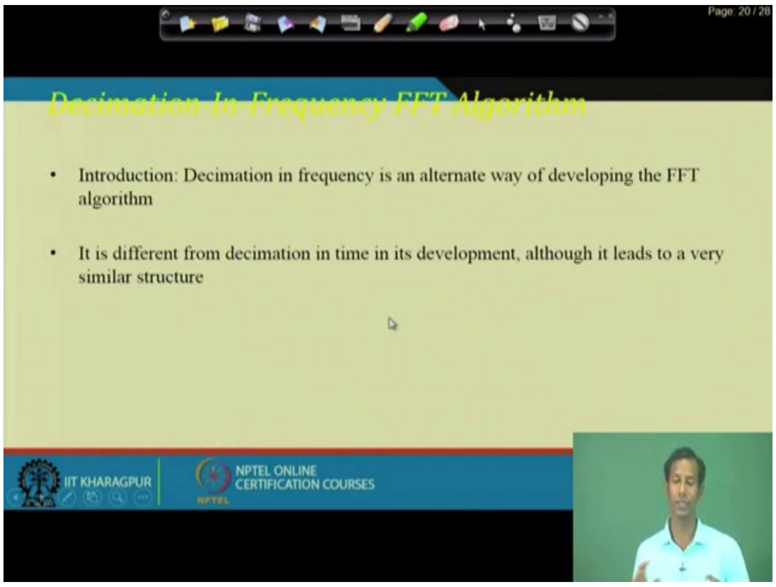
$$x^*[n] = \frac{1}{N}\sum_{k=0}^{N-1} X^*[k]W_N^{kn}\text{; or } x[n] = \frac{1}{N}\left[\sum_{k=0}^{N-1} X^*[k]W_N^{kn}\right]^*$$

- This suggests that we can modify the FFT algorithm for the inverse DFT computation by the following:
  - Complex conjugate the input DFT coefficients
  - Compute the *forward* FFT
  - Complex conjugate the output of the FFT and multiply by
- This method has the advantage that the internal FFT code is undisturbed; it is widely used.

IIT KHARAGPUR | NPTEL ONLINE CERTIFICATION COURSES

So, what will happen if we design the I FFT block? So, at that time how we can do; that means, how we can modify the FFT code so that it becomes the inverse; that means, inverse fast Fourier transform ok. So, what we can do is that at that time it will be just 1 by N into k; that means, k 0 to N minus 1 with x convolution with these W N into N base k n ok.

(Refer Slide Time: 22:34)



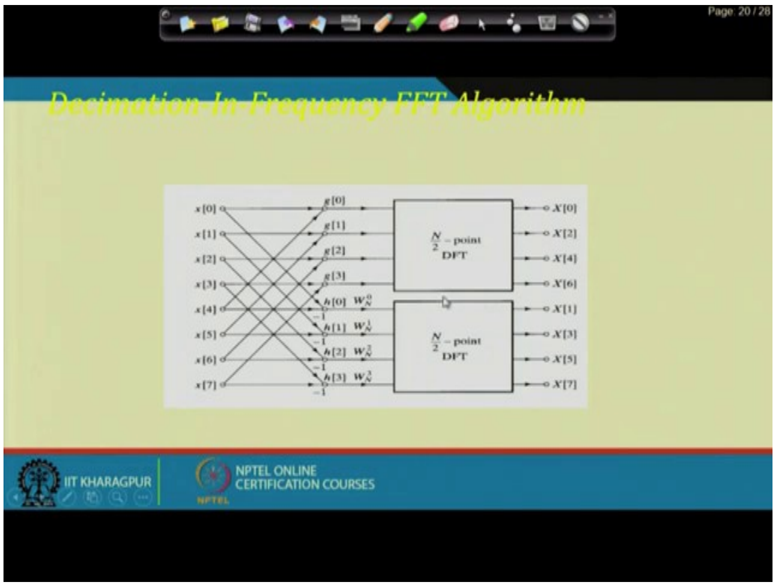
Decimation-In-Frequency FFT Algorithm

- Introduction: Decimation in frequency is an alternate way of developing the FFT algorithm
- It is different from decimation in time in its development, although it leads to a very similar structure

IIT KHARAGPUR | NPTEL ONLINE CERTIFICATION COURSES

So, how we can basically design the hardware for this is that.

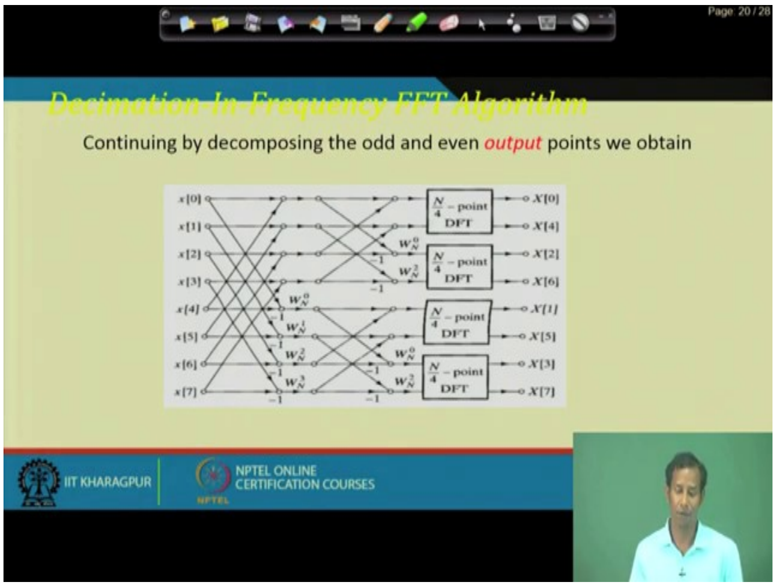
(Refer Slide Time: 22:47)


Decimation-In-Frequency FFT Algorithm

So, actually this is yes. So, here you see initially what we are considering is that we are considering N point 2 DFT and then we are multiplying these with each of these twiddle factors.

Now here in decimation in frequency FFT algorithm so what we have done is that we have initially we have done these x 0 multiplied with x 4 x 1 multiplied with x 5 x 2 multiplied with the x 6 x 3 multiplied with x 7. And then each of these in the next step we are having these N by 2 point DFT and here we are having N by 2 point DFT which basically generates these x 0 x 2 x 4 x 6 and here x 1 x 3 x 5; that means, all these odd even part here from here and even sorry, odd part from the lower DFT ok.

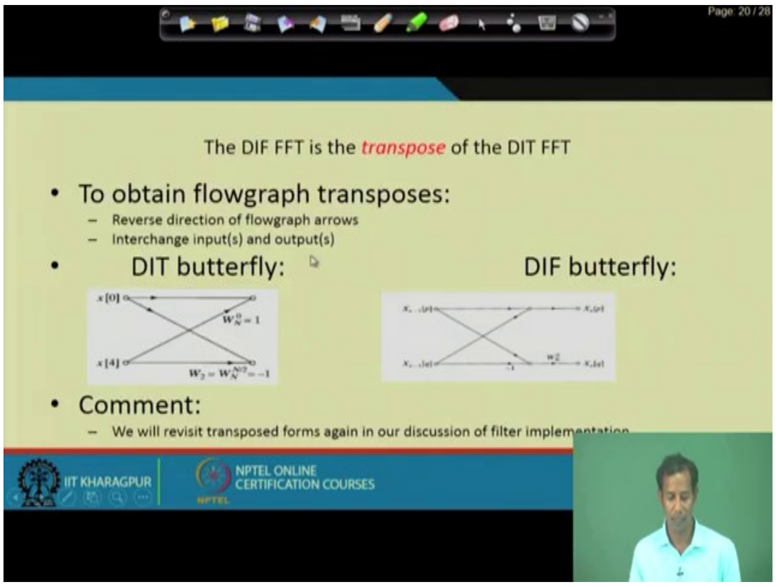
(Refer Slide Time: 23:47)


Decimation-In-Frequency FFT Algorithm

Continuing by decomposing the odd and even *output* points we obtain

So, then the same things using; that means, N by 4 point DFT also we can easily compute, just it is just the reverse case of the; that means, these fast Fourier transform what we have already seen.

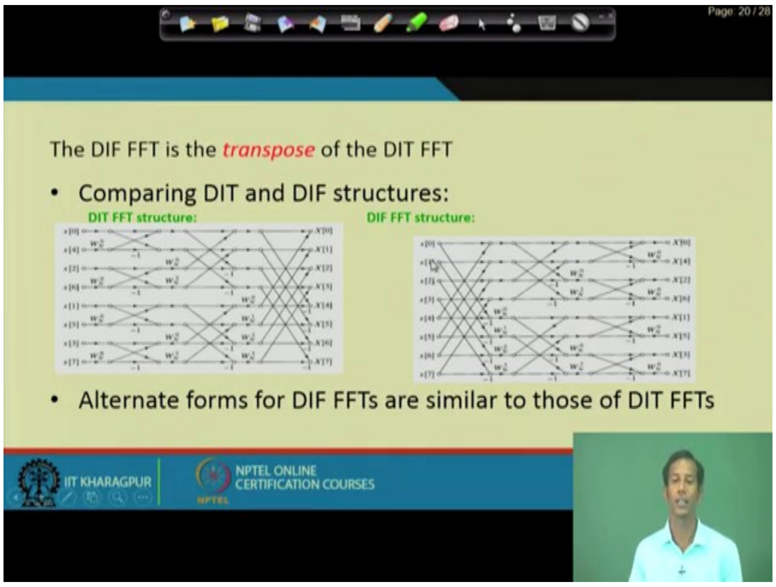
(Refer Slide Time: 24:11)


The DIF FFT is the *transpose* of the DIT FFT

- To obtain flowgraph transposes:
  - Reverse direction of flowgraph arrows
  - Interchange input(s) and output(s)
- DIT butterfly: DIF butterfly:
- Comment:
  - We will revisit transposed forms again in our discussion of filter implementation

So, the decimation; that means, this actually the DIT butterfly if you see actually it corresponds with this x 0 and x 4 whereas, this decimation in frequency they are being multiplied with; that means, in the. Here we are considering the input in the bit reverse manner, but here sorry, here we are considering in the original sequence, but here we have to consider the inputs in the bit reverse manner ok.

(Refer Slide Time: 24:47)


The DIF FFT is the *transpose* of the DIT FFT

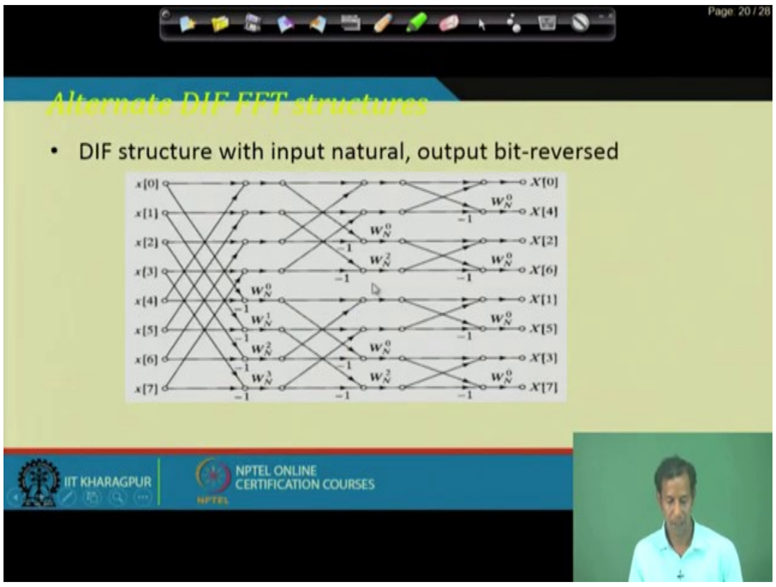
- Comparing DIT and DIF structures:

- Alternate forms for DIF FFTs are similar to those of DIT FFTs

So, this is the corresponding decimation in time FFT structure and this is decimation in frequency structure so here you see here we are considering x 0 x 4 x 2 x 6 x 1 x 5 x 3 x 7, but here the inputs are already in sequence. So, the major difference is that, whatever we are getting that with inputs are bit reverse manner in case of decimation in time the outputs are in the original sequence just the opposite in decimation in frequency the inputs are in the order, but the outputs are in bit reverse manner.

So, by seeing this I can say I can say or that the corresponding FFT architecture whether that is following decimation in time or decimation in frequency.

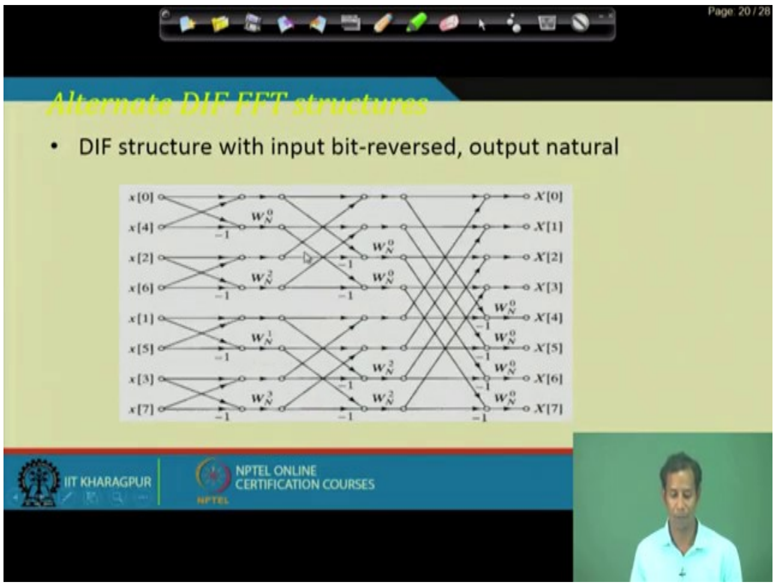
(Refer Slide Time: 25:42)


Alternate DIF FFT structures

- DIF structure with input natural, output bit-reversed

So, this is another actually alternate FFT structure, this is the basic FFT structure what people have designed in the earlier days of FFT computation ok.

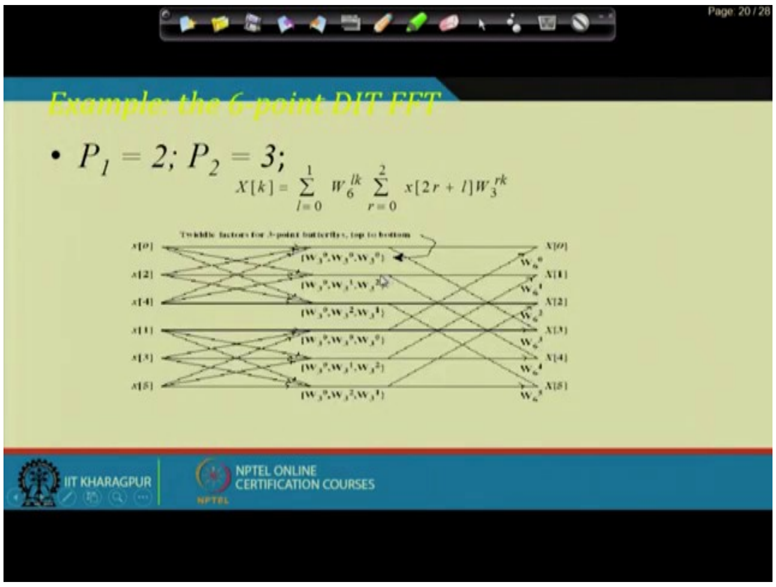
(Refer Slide Time: 25:57)


Alternate DIF FFT structures

- DIF structure with input bit-reversed, output natural

So, in all these case we have already know that in decimation in frequency the input in decimation in frequency the input should be in the corresponding original sequence.

(Refer Slide Time: 26:25)



Example: the 6-point DIT FFT

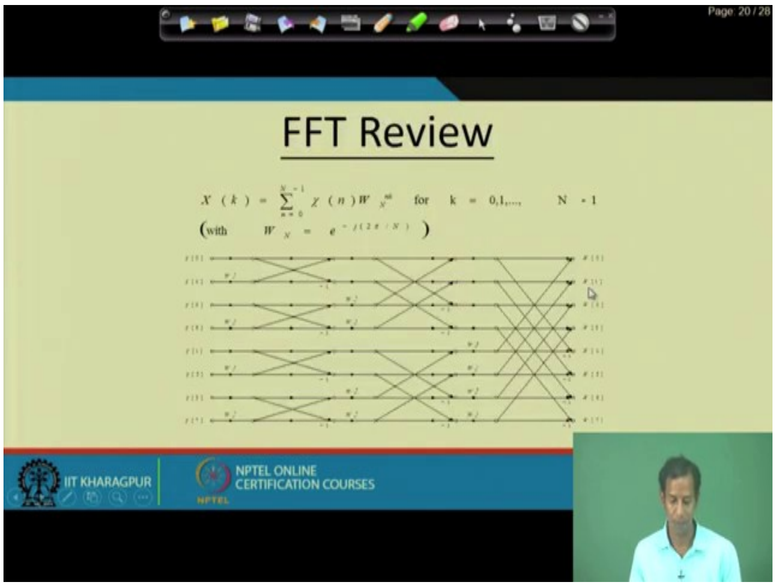
- $P_1 = 2; P_2 = 3;$

$$X[k] = \sum_{l=0}^{1} W_6^{lk} \sum_{r=0}^{2} x[2r+l] W_3^{rk}$$

Now, so whenever we suppose consider one 6 point decimation in time FFT. So, at that time what will happen? So, initially in the earlier example we have considered that 8 point FFT, but you now if we consider these 6 point FFT so at that time what will happen we have to consider these this x 0 x 2 x 4 whereas, the odd part are x 1 x 3 x 5.

So, this x 0 is multiplied with x 2 and then this again it is multiplied with x 4 here x 1 is multiplied with these; that means, it is the, but come it is the butterfly operation with the x 3 and it is the butterfly operation with the x 5. And we give the input in the bit reverse manner we get the output in the original sequence ok.

(Refer Slide Time: 27:25)



FFT Review

$$X(k) = \sum_{n=0}^{N-1} x(n) W_N^{nk} \quad \text{for } k = 0,1,\ldots, N-1$$

$$\left(\text{with } W_N = e^{-j(2\pi/N)}\right)$$

So, this is the; that means, this decimation in time fast Fourier transform architecture. Now, actually whenever we have to design these particular circuit so at the time the hardware implementation this is the just that the flow of; that means, how this data flow diagram of this fast Fourier transform. Now these I have to convert or this I have to this data flow diagram will be implemented via some hardware.

So, what will be the hardware or structure to implement this particular data flow diagram of this fast Fourier transform that we will see in the next lecture.

So, thank you for today's lecture.