Digital Communication using GNU Radio

Prof. Kumar Appaiah

Department of Electrical Engineering

Indian Institute of Technology Bombay

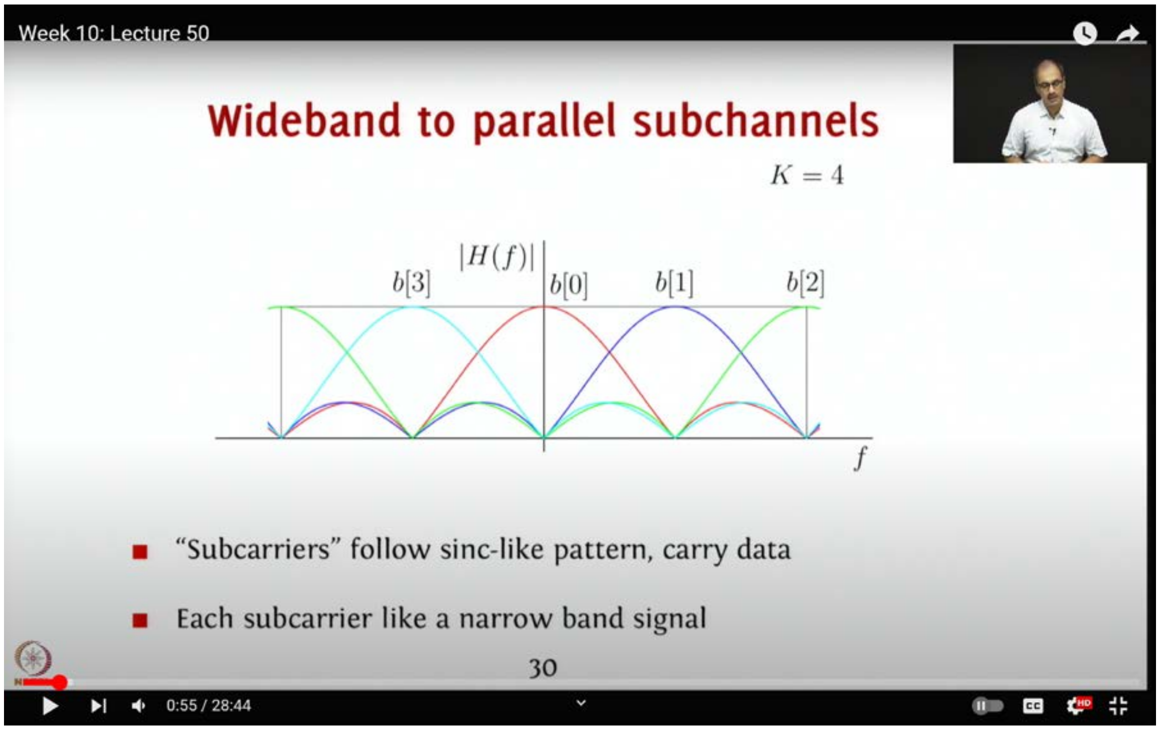
Week-10

Lecture-50

OFDM in the Prescence of Dispersive Channels

Welcome to this lecture on Digital Communication using GNU Radio. I am Kumar Appiah from the Department of Electrical Engineering at IIT Bombay. In this session, we will continue our exploration of Orthogonal Frequency Division Multiplexing (OFDM) and delve into how equalization can be significantly simplified with OFDM.

(Refer Slide Time: 00:55)



Wideband to parallel subchannels

- "Subcarriers" follow sinc-like pattern, carry data
- Each subcarrier like a narrow band signal

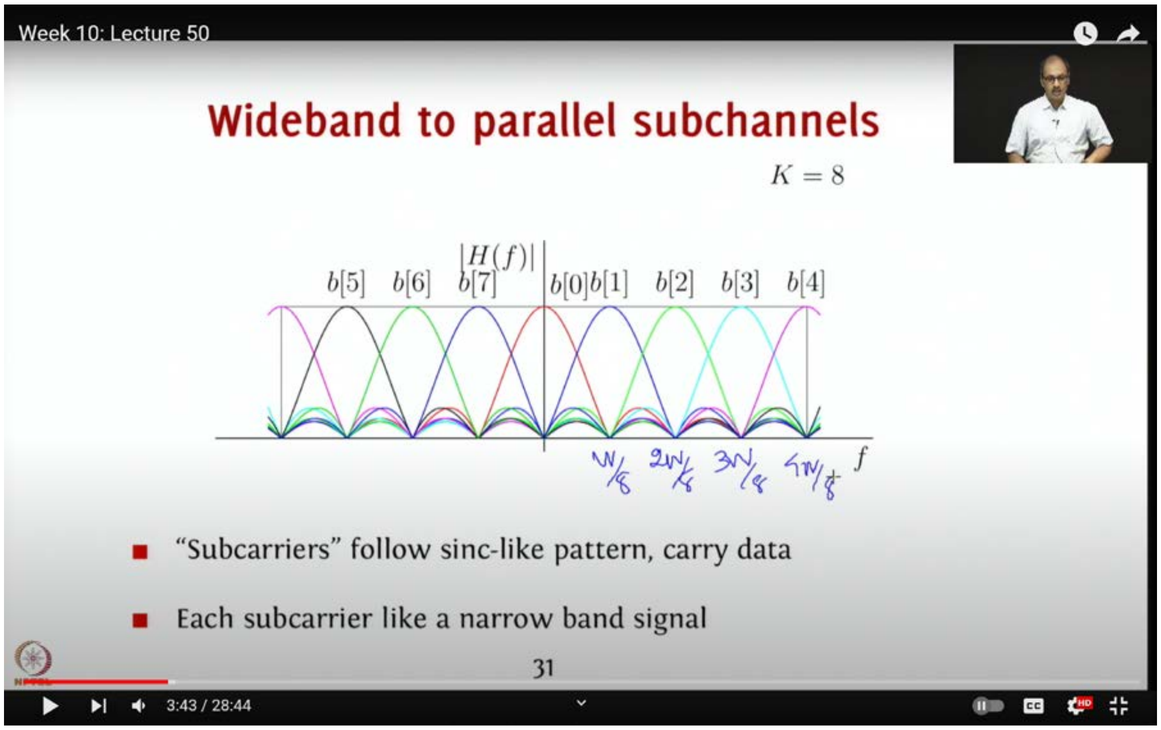
30

To recap our previous discussion on OFDM, we established that viewing OFDM as a combination of symbol repetition and frequency shifting provides valuable insights. Specifically, if we repeat the symbol b_0 multiple times, and similarly for b_1, and apply appropriate frequency shifts, the resulting waveform in the frequency domain displays a distinct pattern. In this pattern, b_0 occupies a spectrum close to 0 Hz, b_1 is near $\frac{W}{4}$ Hz, b_2 is around $\frac{W}{2}$ or $-\frac{W}{2}$ Hz, and b_3 is close to $-\frac{W}{4}$ Hz. These are known as sub-carriers, and they follow a sinc-like pattern in the frequency domain while carrying data.

One important point to remember is the need for precise sampling in the frequency domain. Accurate frequency domain sampling translates to precise time-domain frequency adjustments. Therefore, for OFDM to perform optimally, it is crucial to have accurate receiver carrier calibration and phase estimation. Failing to sample at the correct frequency points can lead to inter-symbol interference due to contributions from adjacent symbols.

Let's proceed with the next topic.

(Refer Slide Time: 03:43)



Wideband to parallel subchannels

$K = 8$

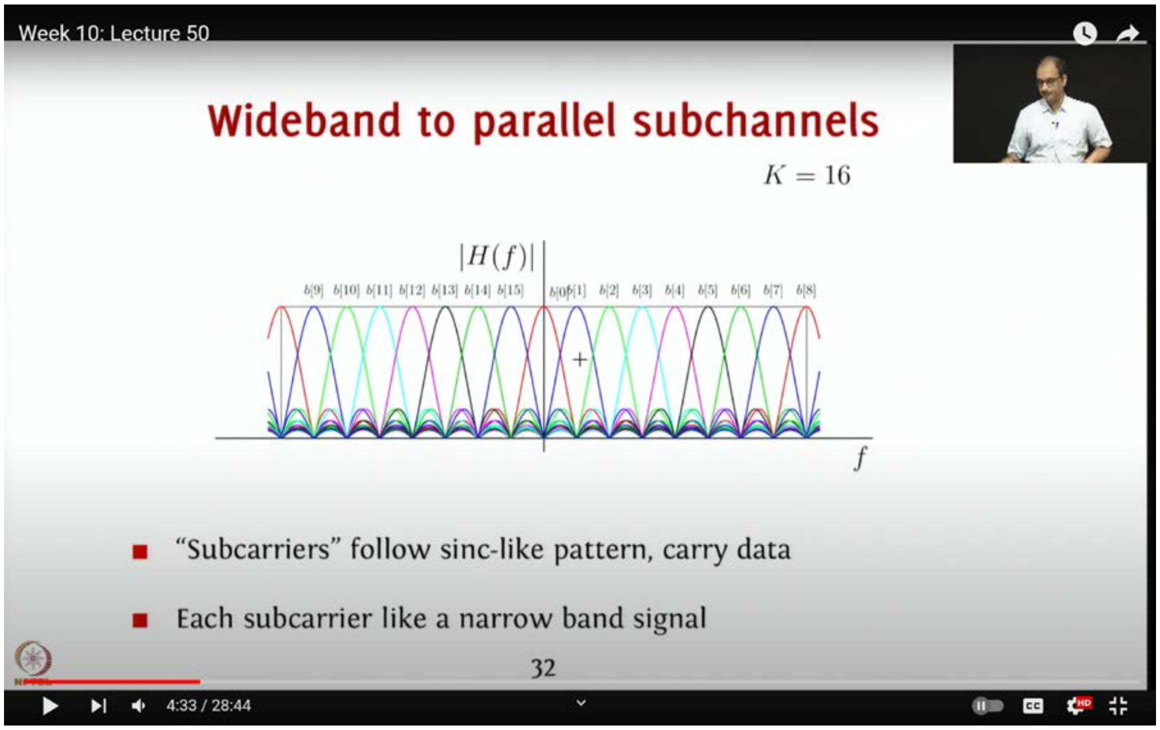
- "Subcarriers" follow sinc-like pattern, carry data
- Each subcarrier like a narrow band signal

31

We were discussing how the Discrete Fourier Transform (DFT), or more precisely, the inverse DFT matrix, can be used to map your symbols b_0, b_1, b_2, and so on, into a format where they are distributed across narrow-band sub-carriers in the frequency domain. As you increase the number of sub-carriers, you'll notice that the spectral footprint of these sinc-like pulses becomes narrower, allowing each to occupy a more confined bandwidth. Despite this narrowing, you still achieve parallel transmission.

In the current scenario, we have divided the bandwidth from -W/2 to W/2 into 8 parallel OFDM streams. These 8 parallel streams enable you to transmit data at the same rate, but now in separate frequency bands. To illustrate, b_0 occupies frequencies close to 0 Hz, b_1 is near W/8, b_2 at W/4, b_3 at 3W/8, and b_4 approaches W/2. This distribution continues accordingly, with 8 sub-carriers effectively representing narrow-band transmissions in the frequency domain.

(Refer Slide Time: 04:33)


Week 10: Lecture 50
$|H(f)|$
$b[9]$ $b[10]$ $b[11]$ $b[12]$ $b[13]$ $b[14]$ $b[15]$ $b[0]$ $b[1]$ $b[2]$ $b[3]$ $b[4]$ $b[5]$ $b[6]$ $b[7]$ $b[8]$
f
4:33 / 28:44

Wideband to parallel subchannels

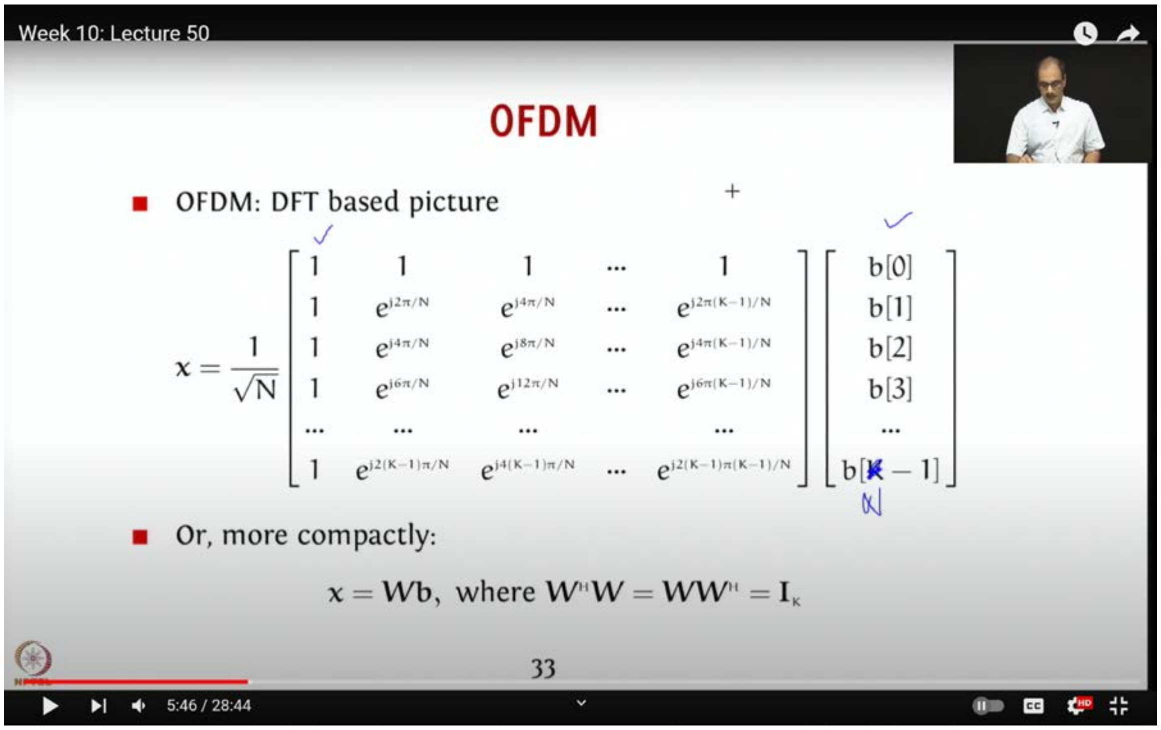
$K = 16$

- "Subcarriers" follow sinc-like pattern, carry data
- Each subcarrier like a narrow band signal

32

If you further increase the number of sub-carriers to 16, they become even narrower and more closely spaced in the frequency domain. It's worth noting, though not covered in this course, that improper frequency sampling, such as failing to correctly calibrate the receiver frequency, can lead to significant inter-symbol interference. This interference, appearing in the frequency domain, can severely impact performance. Keep this in mind as we move forward with our DFT-based discussion.

(Refer Slide Time: 05:46)



Week 10: Lecture 50

OFDM

- OFDM: DFT based picture

$$x = \frac{1}{\sqrt{N}} \begin{bmatrix} \overset{\checkmark}{1} & 1 & 1 & \cdots & 1 \\ 1 & e^{j2\pi/N} & e^{j4\pi/N} & \cdots & e^{j2\pi(K-1)/N} \\ 1 & e^{j4\pi/N} & e^{j8\pi/N} & \cdots & e^{j4\pi(K-1)/N} \\ 1 & e^{j6\pi/N} & e^{j12\pi/N} & \cdots & e^{j6\pi(K-1)/N} \\ \cdots & \cdots & \cdots & & \cdots \\ 1 & e^{j2(K-1)\pi/N} & e^{j4(K-1)\pi/N} & \cdots & e^{j2(K-1)\pi(K-1)/N} \end{bmatrix} \begin{bmatrix} \overset{\checkmark}{b[0]} \\ b[1] \\ b[2] \\ b[3] \\ \cdots \\ b[\underset{N}{\cancel{K}} - 1] \end{bmatrix}$$

- Or, more compactly:

$$x = Wb, \text{ where } W^H W = WW^H = I_K$$

33

5:46 / 28:44

Technically speaking, what we discussed was that you can use the inverse DFT matrix, which, for example, is represented by the matrix

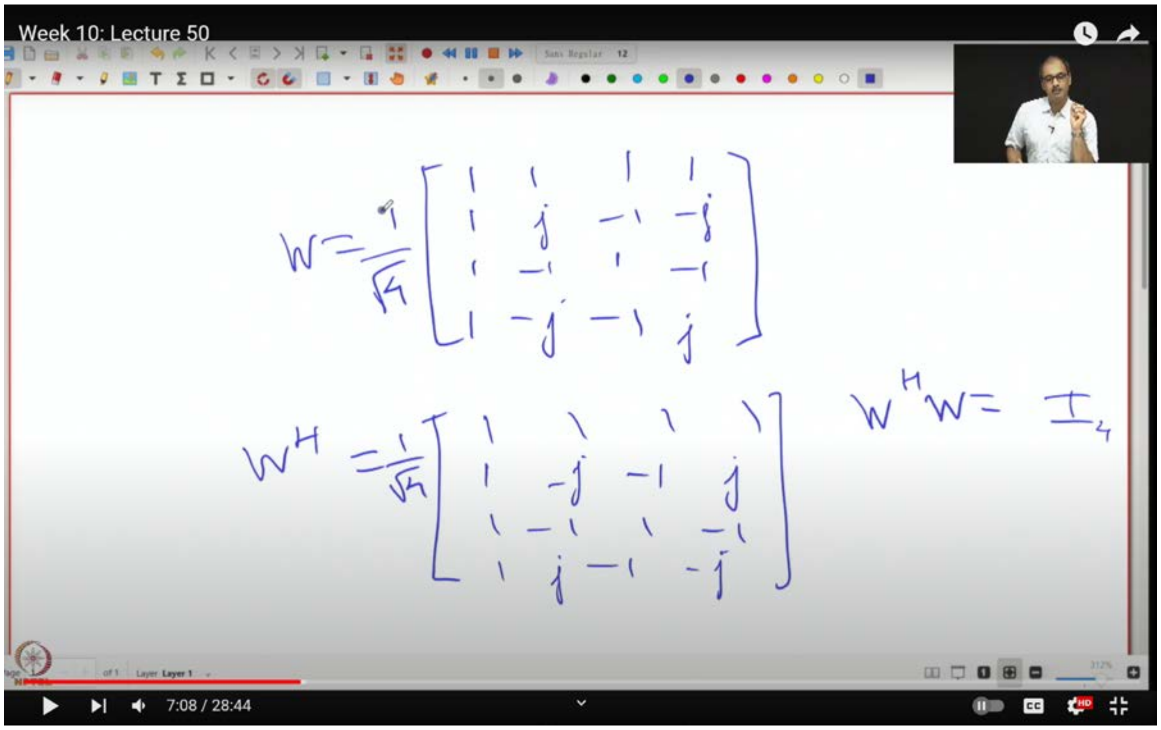
$$\begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & j & -1 & -j \\ 1 & -1 & 1 & -1 \\ 1 & -j & -1 & j \end{bmatrix},$$

to facilitate this type of modulation where your symbols are distributed across parallel frequency bins. How exactly does this work? Let's reframe our understanding using slightly different notation, though the underlying concept remains the same.

Consider collating your data into $b_0, b_1, b_2, \ldots, b_{n-1}$. For a single block of data, the idea is to repeat each b_i multiple times. For instance, b_0 is repeated n times and multiplied by a column of all ones. Similarly, b_1 is repeated n times but multiplied by $e^{j\omega_0 n}$, where ω_0

represents different frequency components like $e^{j2\pi/n}$ and $e^{j4\pi/n}$. This approach is applied to each b_i, so b_2 is multiplied by $e^{j4\pi/n}, e^{j8\pi/n}$, and so on, up to b_{n-1}.

(Refer Slide Time: 07:08)



$$W = \frac{1}{\sqrt{4}} \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & j & -1 & -j \\ 1 & -1 & 1 & -1 \\ 1 & -j & -1 & j \end{bmatrix}$$

$$W^H = \frac{1}{\sqrt{4}} \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & -j & -1 & j \\ 1 & -1 & 1 & -1 \\ 1 & j & -1 & -j \end{bmatrix}$$

$$W^H W = I_4$$

The matrix is scaled by $\frac{1}{\sqrt{n}}$ to ensure it is unitary. From our previous discussion, we defined W, the inverse DFT matrix, in a 4-point example as follows:

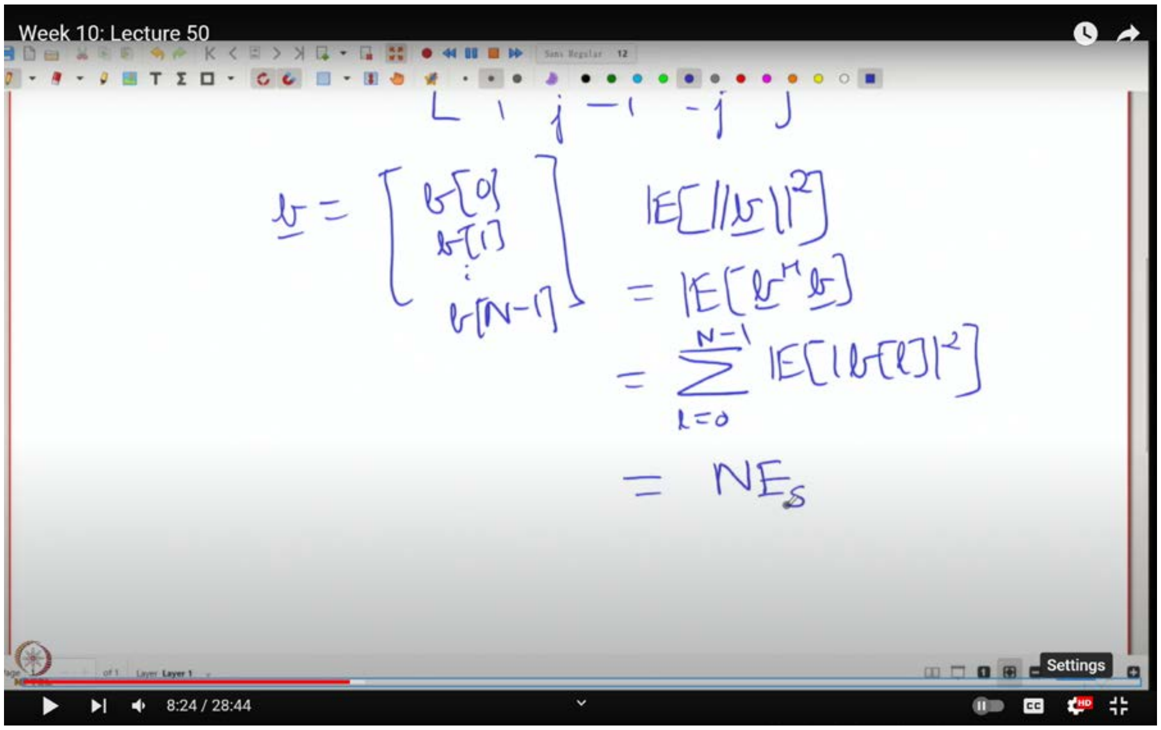
$$W = \frac{1}{\sqrt{4}} \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & j & -1 & -j \\ 1 & -1 & 1 & -1 \\ 1 & -j & -1 & j \end{bmatrix}.$$

We also considered W^{H}, the Hermitian (conjugate transpose) of W:

$$W^{\mathrm{H}} = \frac{1}{\sqrt{4}} \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & -j & -1 & j \\ 1 & -1 & 1 & -1 \\ 1 & j & -1 & -j \end{bmatrix}.$$

When you compute $W^{\mathrm{H}}W$, you get a matrix that is $4 \times$ identity. To normalize this, you divide by $\frac{1}{\sqrt{4}}$, resulting in the 4x4 identity matrix.

(Refer Slide Time: 08:24)



$$\mathbf{b} = \begin{bmatrix} b[0] \\ b[1] \\ \vdots \\ b[N-1] \end{bmatrix} \qquad \begin{aligned} &\mathbb{E}[\|\mathbf{b}\|^2] \\ &= \mathbb{E}[\mathbf{b}^H \mathbf{b}] \\ &= \sum_{l=0}^{N-1} \mathbb{E}[|b[l]|^2] \\ &= N E_s \end{aligned}$$

This scaling by $\frac{1}{\sqrt{4}}$ is crucial because it ensures that $W^{\mathrm{H}}W$ equals the identity matrix, meaning the energy of the vectors is preserved. In other words, multiplying by W does not alter the energy of the vector b. This normalization is fundamental for maintaining the integrity of the signal's energy in the transformed domain.

Let's examine the relationship between our signal vector x and b, where $x = Wb$ and $\boldsymbol{b}$ is our vector of symbols $b_0, b_1, \ldots, b_{n-1}$. We want to evaluate the expectation of the squared norm of $\boldsymbol{b}$, which is given by:

$$E[|b|^2] = E[b^H b].$$

This simplifies to:

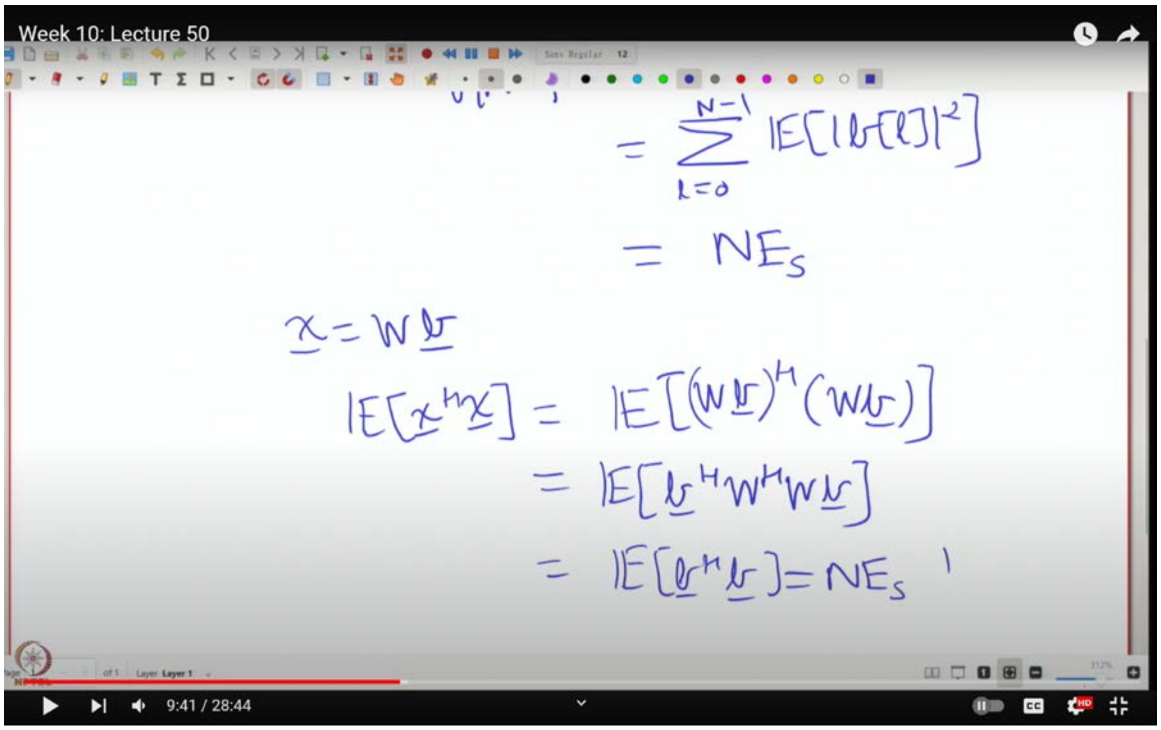
$$E[b^H b] = \sum_{l=0}^{n-1} E[|b_l|^2].$$

Here, $\boldsymbol{b}^{\mathrm{H}}\,\boldsymbol{b}$ represents the sum of the magnitudes squared of the elements of $\boldsymbol{b}$. Assuming that our symbols are generated as independent and identically distributed (i.i.d.) with the same average energy, this sum becomes:

$$\mathrm{N} \cdot E_s,$$

where E_s is the average signal energy per symbol. Therefore, $n \cdot E_s$ represents the total signal energy in the vector $\boldsymbol{b}$.

(Refer Slide Time: 09:41)



Week 10: Lecture 50

$$= \sum_{l=0}^{N-1} \mathbb{E}[|\underline{b}[l]|^2]$$

$$= NE_s$$

$$\underline{x} = W\underline{b}$$

$$\mathbb{E}[\underline{x}^H\underline{x}] = \mathbb{E}[(W\underline{b})^H(W\underline{b})]$$

$$= \mathbb{E}[\underline{b}^H W^H W \underline{b}]$$

$$= \mathbb{E}[\underline{b}^H \underline{b}] = NE_s$$

9:41 / 28:44

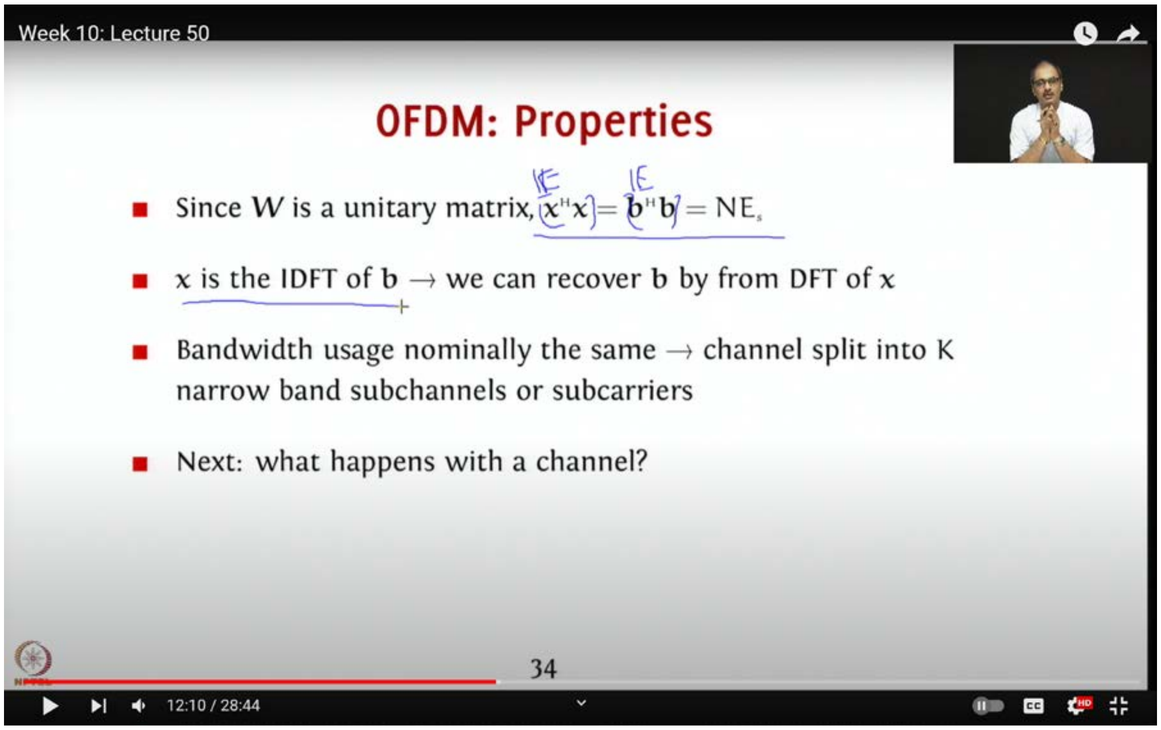
Next, consider the vector $x = Wb$. To determine whether we use more or less power after applying the inverse DFT matrix W, we evaluate the expectation of $x^H x$:

$$E[x^H x] = E[(Wb)^H (Wb)].$$

By substituting $x = Wb$, this becomes:

$$E[(Wb)^H(Wb)] = E[b^H W^H W b].$$

(Refer Slide Time: 12:10)



Week 10: Lecture 50

OFDM: Properties

- Since W is a unitary matrix, $\underbrace{\overset{\mathbb{E}}{x^H x}} = \overset{\mathbb{E}}{b^H b} = NE_s$
- x is the IDFT of b → we can recover b by from DFT of x
- Bandwidth usage nominally the same → channel split into K narrow band subchannels or subcarriers
- Next: what happens with a channel?

34

12:10 / 28:44

Since W is a fixed, non-random matrix, the expectation operator can be factored out:

$$E[b^H W^H W b].$$

We previously established that $W^H W$ is the identity matrix I_n, so:

$$E[b^H W^H W b] = E[b^H I_n b] = E[b^H b].$$

Therefore:

$$E[x^H x] = E[b^H b] = n \cdot E_s.$$

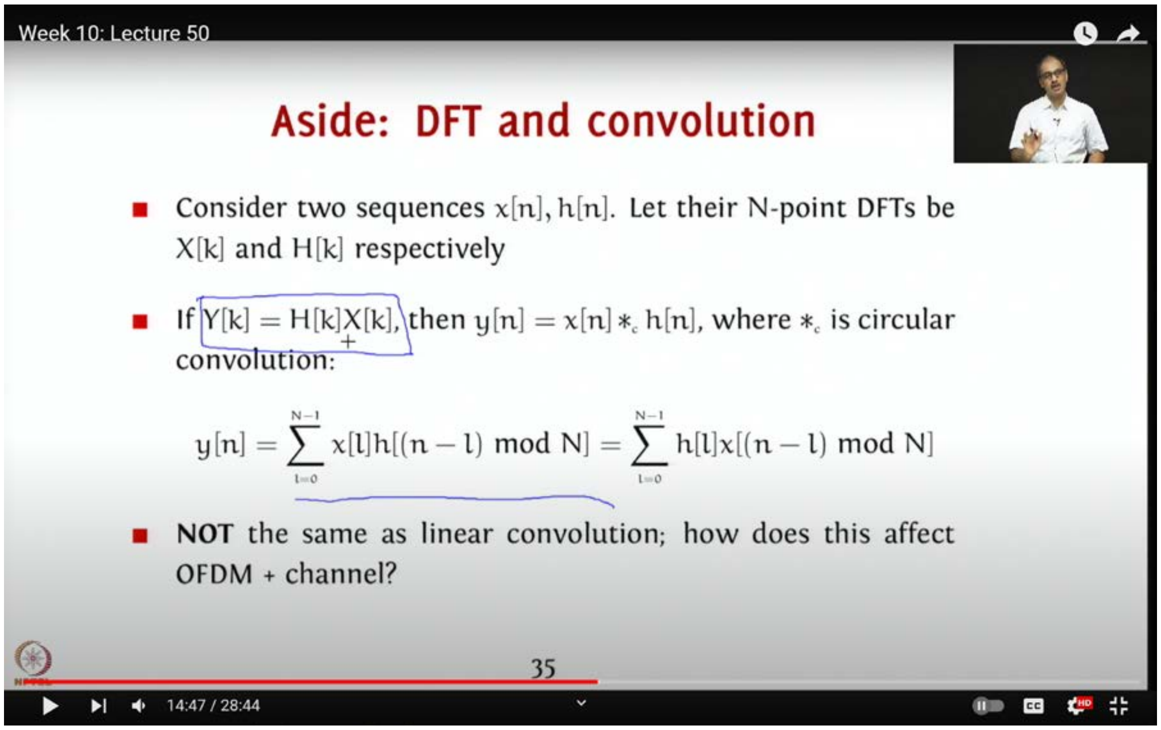
This result is significant because it shows that using this scaled version of W (with the $\frac{1}{\sqrt{n}}$ scaling) ensures that the energy consumption remains unchanged. In other words, the energy usage is preserved when transforming $\boldsymbol{b}$ into $\boldsymbol{x}$ using the IDFT matrix.

However, a word of caution: in practical implementations, such as those in GNU Radio, the DFT matrix may have scaling factors associated with it. This scaling needs to be taken into account to correctly assess the signal's energy. It’s a straightforward adjustment, but an important one to remember.

What you need to do here is simply multiply by a factor, but just keep in mind that we will rely on this particular DFT matrix framework because it has very useful properties. For instance, you can verify that $W^H W = WW^H = I_n$, where I_n is the identity matrix. This makes our analysis much more convenient.

In other words, if the energy budget for your signal at the transmitter is E_s per symbol, then after multiplying $\boldsymbol{b}$ by W to get $\boldsymbol{x}$, the energy per symbol remains the same. Since W is a unitary matrix, we demonstrated that $x^H x = b^H b$, or more precisely, the expected value of $x^H x$ equals the expected value of $b^H b$, which is $n \cdot E_s$. For modulation schemes like QPSK, this holds true without any adjustments. However, for other modulations like 16-QAM, you might need to consider the expectation when calculating power.

(Refer Slide Time: 14:47)


Week 10: Lecture 50

Aside: DFT and convolution

- Consider two sequences x[n], h[n]. Let their N-point DFTs be X[k] and H[k] respectively
- If $\boxed{Y[k] = H[k]X[k],}$ then $y[n] = x[n] *_c h[n]$, where $*_c$ is circular convolution:

$$y[n] = \underbrace{\sum_{l=0}^{N-1} x[l]h[(n-l) \bmod N]} = \sum_{l=0}^{N-1} h[l]x[(n-l) \bmod N]$$

- **NOT** the same as linear convolution; how does this affect OFDM + channel?

35

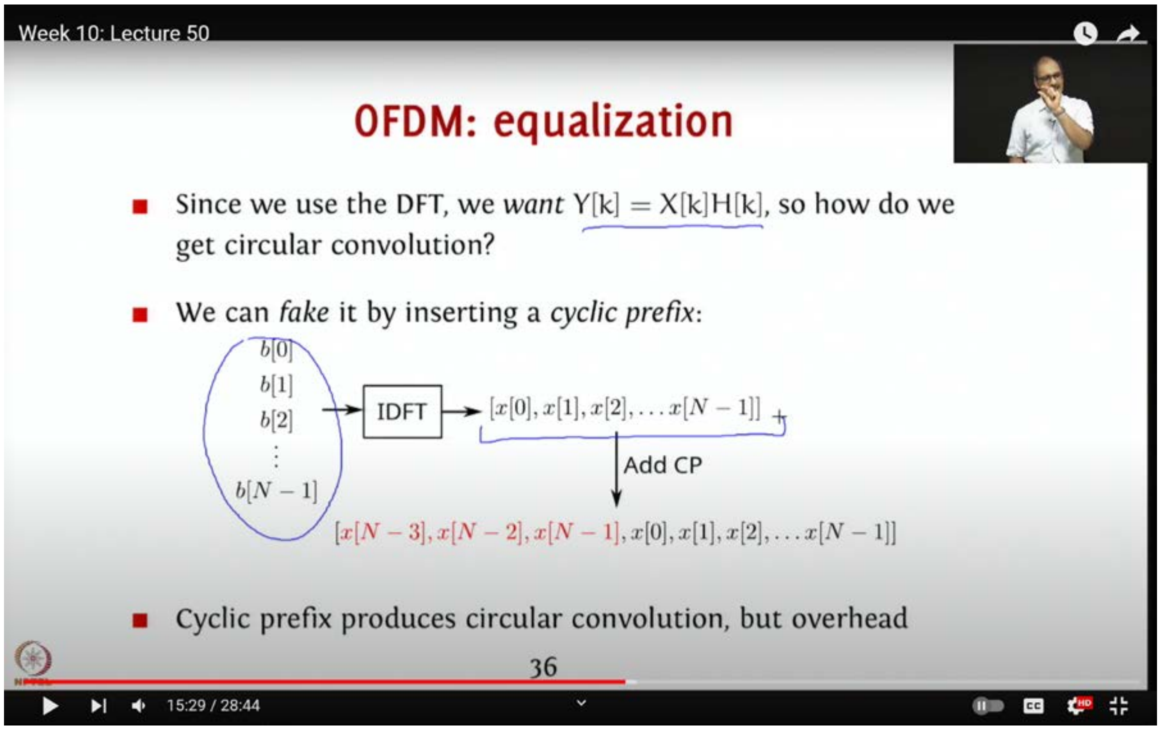
14:47 / 28:44

Now, ***x*** is essentially the inverse DFT of ***b***. The important question here is: how do we retrieve ***b*** from ***x*** at the receiver? Let's assume, for simplicity, that there is no noise and no channel interference. In this ideal scenario, since ***x*** is the inverse DFT of ***b***, you can simply apply the DFT to ***x*** to recover ***b***.

The reason this works is because the DFT and IDFT are perfectly lossless and reversible operations. So, at the receiver, you can group the n symbols of ***x***, perform the DFT, and retrieve your original symbol vector ***b***.

However, the real challenge arises when there is noise or when the signal passes through a channel. This is where things become more complicated, and we need to account for issues like noise interference and channel effects. In the next part of the discussion, we will take a closer look at how the presence of a channel introduces convolution and noise addition, and we will explore strategies to handle these complications effectively.

(Refer Slide Time: 15:29)



OFDM: equalization

- Since we use the DFT, we *want* $\underline{Y[k] = X[k]H[k]}$, so how do we get circular convolution?
- We can *fake* it by inserting a *cyclic prefix*:

$$\begin{tikzcd} \begin{matrix} b[0] \\ b[1] \\ b[2] \\ \vdots \\ b[N-1] \end{matrix} \arrow[r] & \boxed{\text{IDFT}} \arrow[r] & \underbrace{[x[0], x[1], x[2], \ldots x[N-1]]}_{} \arrow[d, "\text{Add CP}"] \\ & & {[x[N-3], x[N-2], x[N-1], x[0], x[1], x[2], \ldots x[N-1]]} \end{tikzcd}$$

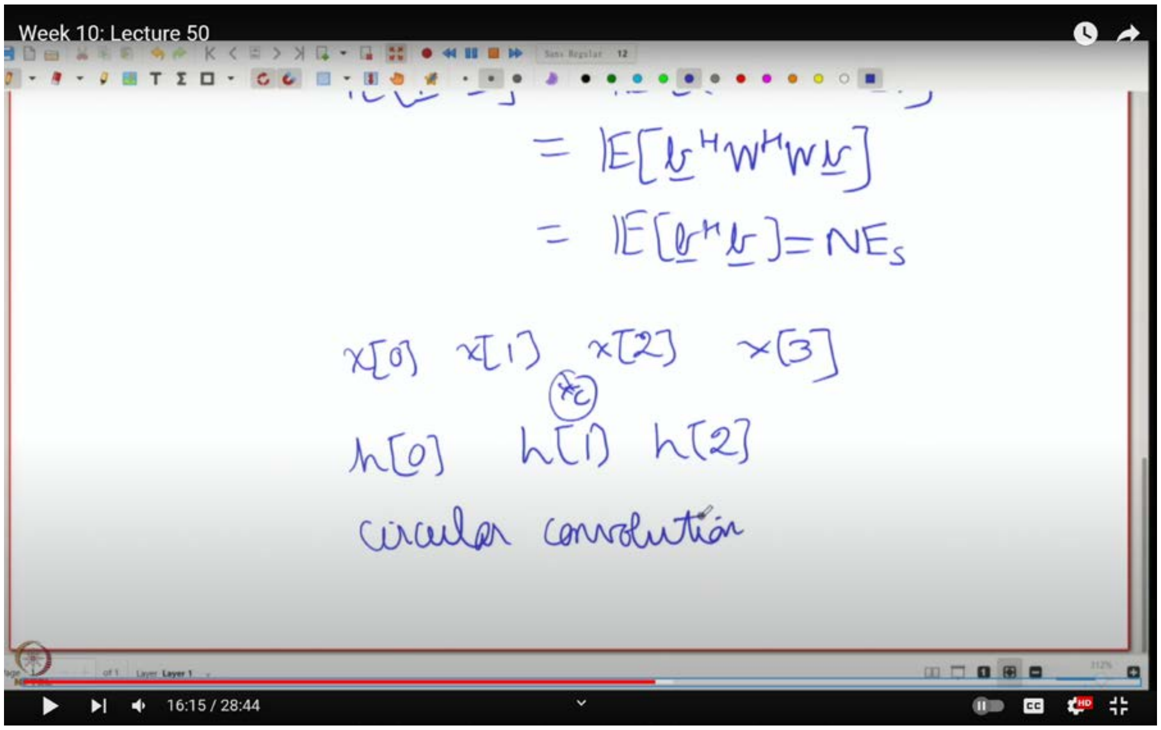
- Cyclic prefix produces circular convolution, but overhead

36

For now, just remember that, under ideal conditions, the DFT and IDFT framework allows you to perfectly recover your data with the same bandwidth usage, as the channel is divided into k or n narrowband subchannels or subcarriers.

If you recall from your Digital Signal Processing (DSP) studies, consider two sequences x[n] and h[n]. Let their N-point DFTs be denoted by X[k] and H[k], respectively. When we say "N-point DFT," it refers to the summation $X[k] = \sum x[n]e^{j2\pi kn/N}$. These DFTs, X[k] and H[k], are defined for values of k ranging from 0 to N-1.

(Refer Slide Time: 16:15)



$$= \mathbb{E}[\underline{b}^H W^H W \underline{b}]$$

$$= \mathbb{E}[\underline{b}^H \underline{b}] = NE_s$$

$$x[0] \quad x[1] \quad x[2] \quad x[3]$$

$$\circledast_C$$

$$h[0] \quad h[1] \quad h[2]$$

circular convolution

Now, if we denote Y[k] as the product of H[k] and X[k], i.e., $Y[k] = H[k] \times X[k]$, the sequence that corresponds to this multiplication is not the regular linear convolution but rather a circular convolution. That’s why I’ve placed a "C" to indicate circular convolution. What this means is that, unlike the traditional convolution in the time domain, when you multiply the DFTs of two sequences, the inverse of that multiplication will result in a sequence that is circularly convolved in the time domain.

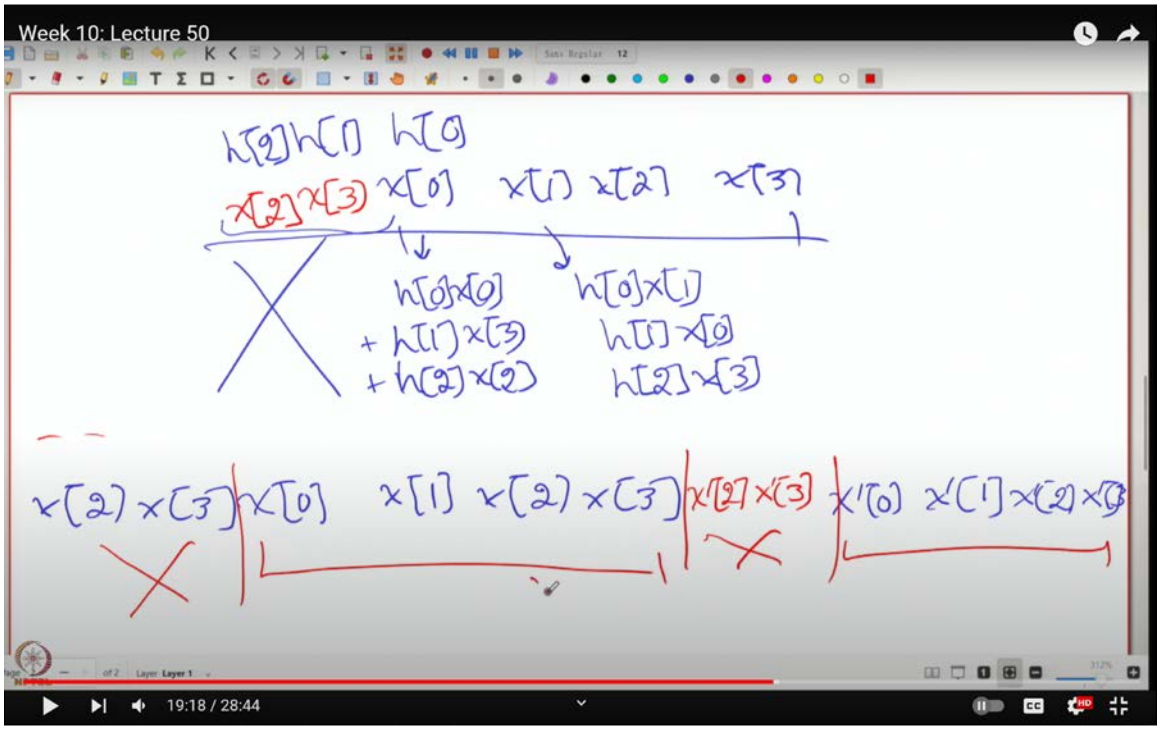
In the case of the Discrete-Time Fourier Transform (DTFT), when you multiply the DTFTs of two sequences, the inverse of this operation results in a linear convolution of the two sequences in the time domain. However, with the DFT, the multiplication of the DFTs results in circular convolution, not linear convolution. In terms of notation, your resulting sequence Y[n] would be represented as:

$$Y[n] = \sum_{l=0}^{N-1} x[l]h[n - l \, modN].$$

Alternatively, you could swap h[n] and x[n], and the outcome remains the same due to the properties of convolution.

This is an important distinction because circular convolution is not equivalent to linear convolution. Now, let's consider this within the context of OFDM. In OFDM, after applying the IDFT to your signal, the result then undergoes convolution with the channel.

(Refer Slide Time: 19:18)


Week 10: Lecture 50
h[2] h[1] h[0]
x[2] x[3] x[0] x[1] x[2] x[3]
h[0]x[0] + h[1]x[3] + h[2]x[2]
h[0]x[1] h[1]x[0] h[2]x[3]
x[2] x[3] x[0] x[1] x[2] x[3] x'[2] x'[3] x'[0] x'[1] x'[2] x'[3]
19:18 / 28:44

Here's where the issue arises: in order to leverage single-tap equalization methods, such as zero-forcing or MMSE equalizers, you want the relationship in the frequency domain to be straightforward. Specifically, if your channel response h is known and your received signal y is known, then you can compute the transmitted signal x by performing $\frac{y}{h}$. In the case of a zero-forcing equalizer, you would just divide y by h. Similarly, for MMSE equalization, you would divide by some function of h. This approach works perfectly when you have circular convolution.

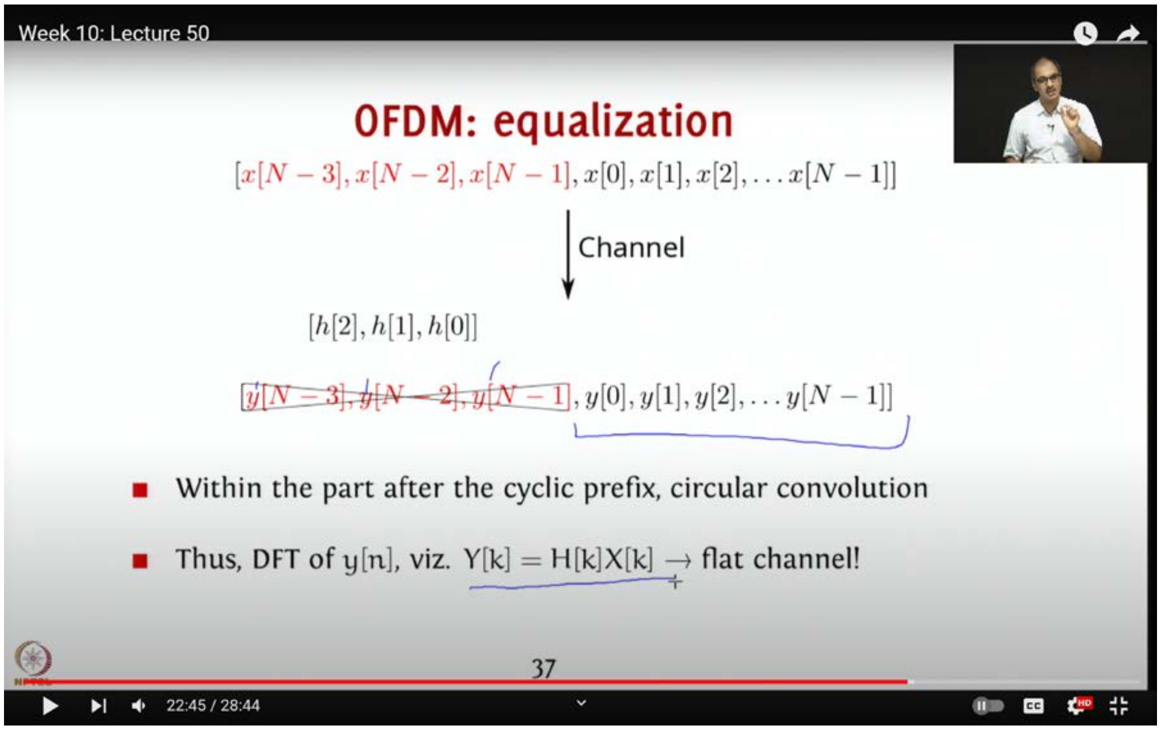
However, the complication is that practical channels perform linear convolution, not circular convolution. This is the key challenge. To exploit the DFT and IDFT properties, where you want $Y[k] = X[k] \times H[k]$ to hold true. You need to address this discrepancy.

The natural behavior of channels doesn't provide us with circular convolution, it performs linear convolution instead. So, how can we address this mismatch? One clever solution is to artificially induce circular convolution by incorporating what's known as a cyclic prefix. Here's the idea: imagine we take a block of symbols, say $b_0, b_1, b_2, \dots, b_{n-1}$, and after applying the inverse DFT (IDFT), we obtain the corresponding time-domain sequence $x_0, x_1, \dots, x_{n-1}$. This sequence is often referred to as your "x-vector." To this x-vector, we then prepend a cyclic prefix, consisting of the last few samples of the sequence, say $x_{n-3}, x_{n-2}, x_{n-1}$.

What does this cyclic prefix achieve? Let's analyze it in detail. Suppose we have a sequence x_0, x_1, x_2, x_3, and we wish to convolve this with a channel represented by coefficients h_0, h_1, h_2. If we want to mimic circular convolution, the cyclic prefix enables us to trick the system into doing just that.

Here's how: when you perform the circular convolution, instead of the regular linear convolution where h_0 multiplies x_0, circular convolution adds terms like $h_2 \times x_3$ and $h_1 \times x_2$ from earlier samples due to the wrap-around effect. Normally, this wouldn't happen naturally with linear convolution, but by adding the cyclic prefix, we essentially extend the sequence and shift this wrap-around behavior into the cyclic prefix portion, so that the actual convolution within the desired segment of the signal behaves like circular convolution.

(Refer Slide Time: 22:45)



Week 10: Lecture 50

OFDM: equalization

$$[x[N-3], x[N-2], x[N-1], x[0], x[1], x[2], \dots x[N-1]]$$

Channel

$$[h[2], h[1], h[0]]$$

$$[\cancel{y[N-3], y[N-2], y[N-1]}, \underbrace{y[0], y[1], y[2], \dots y[N-1]]}$$

- Within the part after the cyclic prefix, circular convolution
- Thus, DFT of y[n], viz. $\underline{Y[k] = H[k]X[k]} \rightarrow$ flat channel!

37

22:45 / 28:44

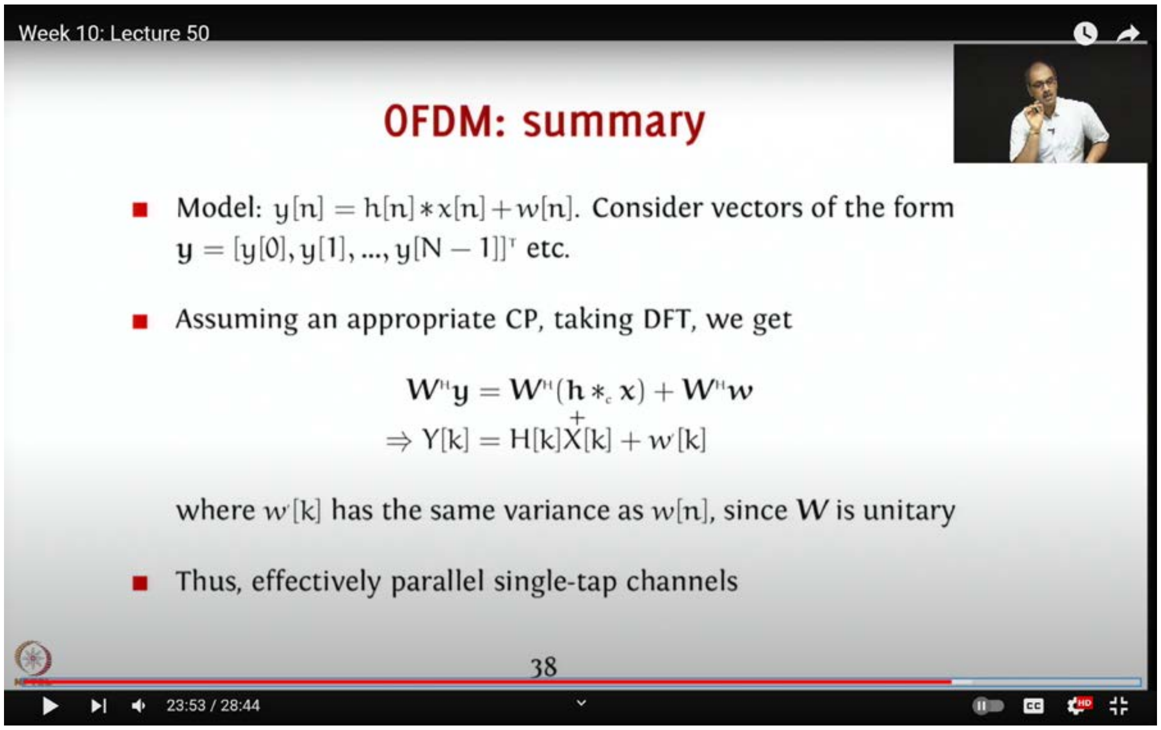
For example, suppose you have x_0, x_1, x_2, x_3 and you perform a convolution with h_0, h_1, h_2. In linear convolution, h_0 would only interact with x_0, but circular convolution incorporates terms like $h_2 \times x_3$ from the tail end of the sequence. By appending $x_{n-3}, x_{n-2}, x_{n-1}$ in front of the sequence as a cyclic prefix, we ensure that this wrap-around behavior occurs in a controlled manner, allowing the convolution within the sequence's main body to behave as if it were circular.

To see this in action, imagine we align h_0, h_1, h_2 against x_0, x_1, x_2, x_3, perform the convolution, and examine the outputs in the region of interest. Due to the cyclic prefix, we get the desired circular convolution results in the center of the sequence. For instance, instead of just $h_0 \times x_0$, we now get contributions like $h_1 \times x_3$ and $h_2 \times x_2$, exactly as circular convolution would require. Similarly, as you shift, the convolution behaves as needed, and you can ignore the unwanted terms introduced by the linear convolution at the prefix and suffix.

Thus, by adding this cyclic prefix to the OFDM symbols, we effectively mimic circular convolution within the main part of the sequence, even though the underlying channel naturally performs linear convolution. For example, suppose we have two groups of symbols, x_0, x_1, x_2, x_3 followed by another block of symbols x'_0, x'_1, x'_2, x'_3. To induce circular convolution, we prepend a portion of each sequence to itself, so x_2 and x_3 are prepended before x_0, x_1, x_2, x_3, and similarly, x'_2 and x'_3 are prepended before the next sequence.

As the convolution occurs with the channel, h_0, h_1, h_2 moves across the sequence, and by carefully discarding the results at the cyclic prefix region, we retain the desired behavior of circular convolution in the main sequence, ensuring the OFDM signal can be effectively processed.

(Refer Slide Time: 23:53)



OFDM: summary

- Model: $y[n] = h[n] * x[n] + w[n]$. Consider vectors of the form $\mathbf{y} = [y[0], y[1], ..., y[N-1]]^T$ etc.
- Assuming an appropriate CP, taking DFT, we get

$$\mathbf{W}^H \mathbf{y} = \mathbf{W}^H(\mathbf{h} *_c \mathbf{x}) + \mathbf{W}^H \mathbf{w}$$
$$\Rightarrow Y[k] = H[k]X[k] + w'[k]$$

where $w'[k]$ has the same variance as $w[n]$, since $\mathbf{W}$ is unitary

- Thus, effectively parallel single-tap channels

38

The key here is that by using a cyclic prefix, we are able to achieve circular convolution, even though the natural tendency of the channel is to perform linear convolution. This trick allows us to preserve the desirable properties of circular convolution, but it comes with a cost. Specifically, to implement this, we need to prepend a portion of data to the beginning of each block, which creates overhead. Essentially, you are repeating some of the symbols. It’s not a free benefit, but a trade-off.

For instance, in OFDM, if you utilize a cyclic prefix, let’s say with three symbols, then instead of sending n symbols in n time instances, you are now sending n symbols in n+3 time instances. So, this overhead reduces the system's efficiency slightly. The efficiency of the system, therefore, can be thought of as $\frac{n}{n+3}$, which reflects the fact that the data rate has decreased.

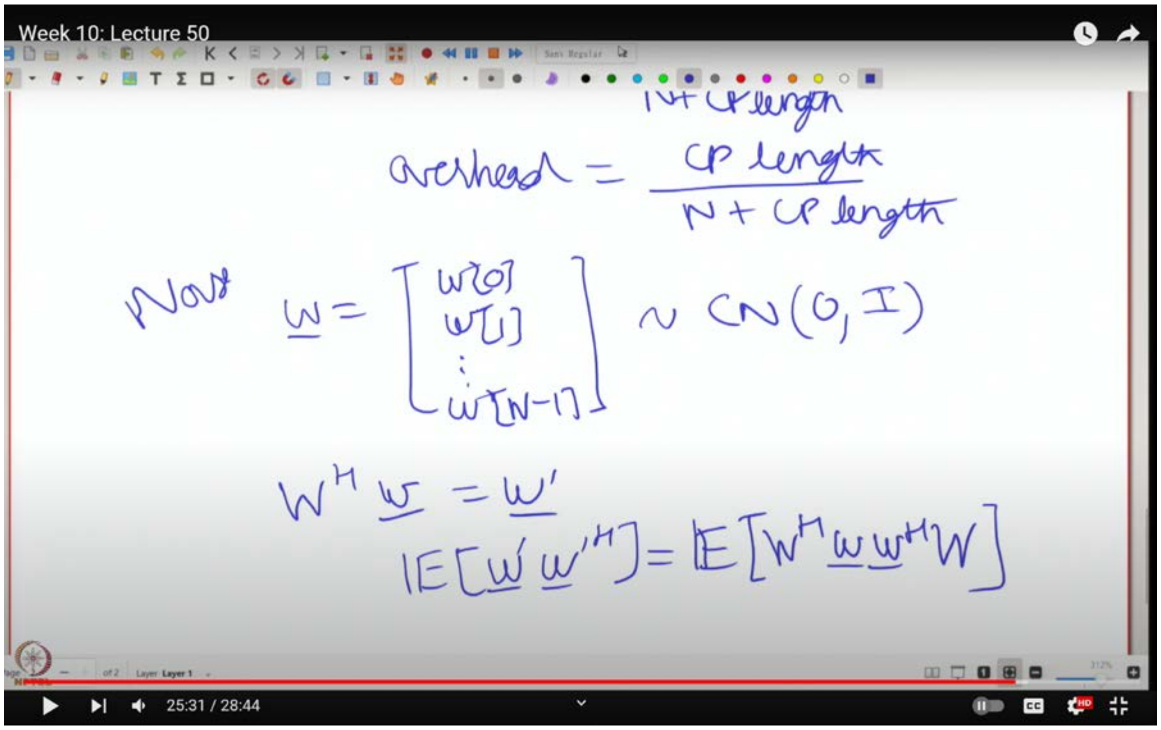
Now, because of this trade-off, when designing an OFDM system, the size of the FFT block, n, is typically chosen to be as large as possible while still meeting the desired data rate. At the same time, the length of the cyclic prefix is selected based on the expected channel length in terms of taps. For example, if you have a channel with three taps H_0, H_1, and H_2, the cyclic prefix needs to be at least two samples long. However, designers will often choose a slightly longer cyclic prefix to accommodate for unforeseen conditions, such as environments where the channel has more taps than anticipated. You wouldn’t want the system to fail simply because it encounters a channel with more taps than expected.

This overhead due to the cyclic prefix can be quantified as well. We can define an "OFDM efficiency," which is not a standard term, but it is useful for understanding the trade-off. The efficiency is given by $\frac{n}{n+\text{cp length}}$, where "cp length" represents the length of the cyclic prefix. Alternatively, we can think of the overhead as a percentage, and naturally, the goal is to have n much larger than the cyclic prefix length.

Of course, the cp length should be chosen to match the length of the channel, typically, it should be the length of the channel minus one tap. So, the principle is that by using the cyclic prefix, even after you convolve with the channel, you can safely discard the portion of the signal that results from the cyclic prefix.

For example, let’s denote the output of the channel convolution as y'. After convolution, you would get y'_{n-2}, y'_{n-3}, etc., and you simply ignore those initial samples. The rest of the signal will be circularly convolved as intended. Therefore, the cyclic prefix effectively enables you to preserve the properties of circular convolution while avoiding the distortions that linear convolution would otherwise introduce.

(Refer Slide Time: 25:31)



N + CP length

$$\text{overhead} = \frac{\text{CP length}}{N + \text{CP length}}$$

Now $$\underline{w} = \begin{bmatrix} w[0] \\ w[1] \\ \vdots \\ w[N-1] \end{bmatrix} \sim \mathcal{CN}(0, I)$$

$$W^H \underline{w} = \underline{w}'$$

$$\mathbb{E}[\underline{w}' \underline{w}'^H] = \mathbb{E}[W^H \underline{w}\, \underline{w}^H W]$$

After the cyclic prefix has been accounted for at the receiver, what you essentially achieve is circular convolution. Now, what does that do for us? If you take the DFT of the segment after the cyclic prefix, let’s call this the "clean" portion, you will obtain $Y_k = H_k X_k$. This results in a "flat" channel, meaning the output Y_k is simply the input X_k multiplied by H_k, the channel frequency response. This is incredibly beneficial because now, equalization becomes much simpler. Whether you’re using an MMSE equalizer or a zero-forcing approach, you only need to multiply or divide by a single number H_k for each subcarrier. This simplification is the core advantage of the cyclic prefix technique.

But there’s another aspect we haven’t discussed yet: the impact of noise. Let’s tackle that now.

The overall model for the received signal includes both convolution with the channel impulse response and the addition of noise. Mathematically, it can be expressed as:

$$Y_n = h_n * X_n + W_n,$$

where W_n represents the noise. If we express the received vector y as $[y_0, y_1, \dots, y_{N-1}]^T$, and similarly for $\boldsymbol{x}$, we then include the cyclic prefix in a way that ensures the impact of the channel convolution appears as circular convolution within the region of interest. The cyclic prefix length is chosen to be long enough to encompass the channel length, effectively transforming the linear convolution into a circular one.

Now, if we use the DFT matrix W that we previously introduced, where $W^\dagger W = I$ (i.e., W is unitary), and apply it to the signal after the cyclic prefix, we get:

$$W^\dagger y = W^\dagger (h \circledast x) + W^\dagger w.$$

Taking the DFT of h $\circledast$ x gives us H_k X_k, as expected. The question then arises: what happens to the noise?

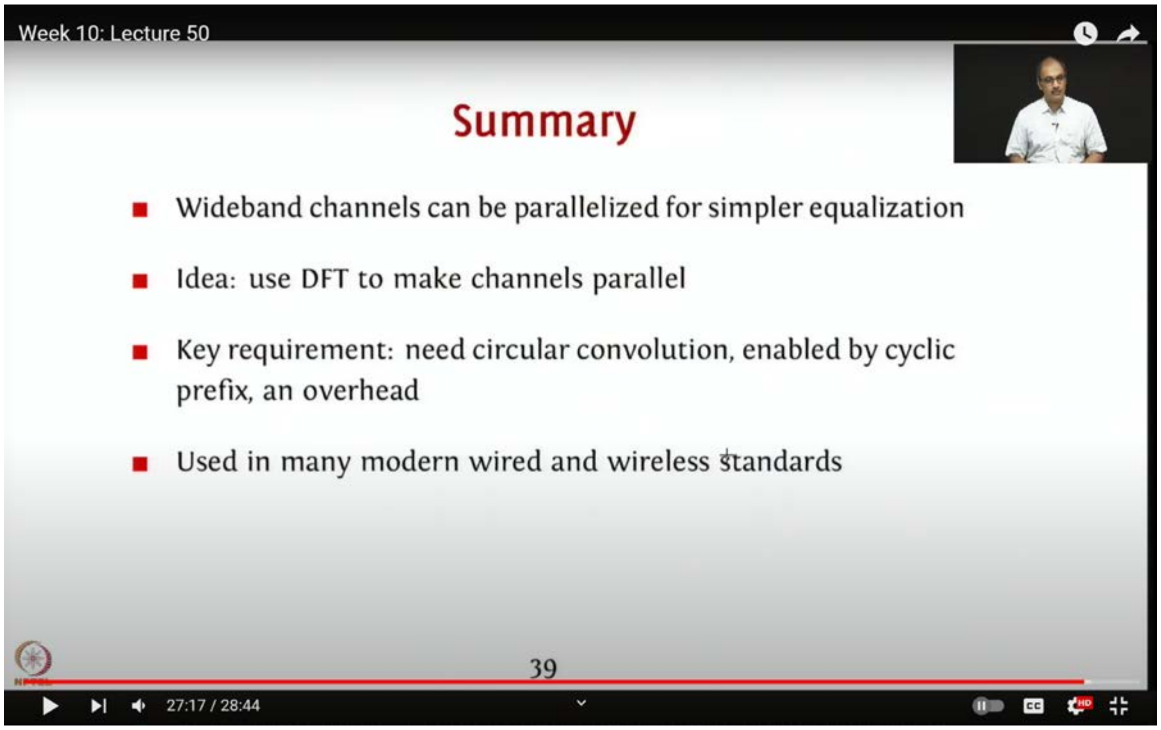
Here’s the clever part: the noise $\boldsymbol{w}$, which we assume to be complex normal with zero mean and an identity covariance matrix, maintains its distribution even after the DFT. This is due to the fact that the DFT matrix, when scaled appropriately, behaves like a rotation matrix. As a result, the noise variance remains unchanged. In other words, applying the DFT to the noise doesn't affect its statistical properties.

So, if you take the DFT of the noise, you get a noise sequence with the same distribution as before. The covariance of the transformed noise remains identical. Specifically, if we call the transformed noise $\boldsymbol{w}'$, the expectation of $w' w'^\dagger$ is:

$$E[w' w'^\dagger] = W^\dagger E[w w^\dagger] W.$$

Since w has identity covariance, this reduces to $W^{\dagger}W = I$, confirming that the covariance of the transformed noise is still the identity matrix. Thus, even though we've applied this DFT-based processing, the noise variance does not change, making the system easier to manage.

(Refer Slide Time: 27:17)



Summary

- Wideband channels can be parallelized for simpler equalization
- Idea: use DFT to make channels parallel
- Key requirement: need circular convolution, enabled by cyclic prefix, an overhead
- Used in many modern wired and wireless standards

39

Therefore, the w_k's are actually independent and identically distributed (IID) because I chose my w to be IID. They also have the same variance as w. This is the key point here. This result holds only when your noise w has a covariance structure that is a scaled version of the identity matrix, which is exactly the scenario we are considering. Hence, w'_k has the same variance as w_n and maintains the IID property just like w_n.

In effect, what we've achieved is a very straightforward single-tap scaling in the presence of additive white Gaussian noise (AWGN). This simplification makes the channel extremely easy to manage. Essentially, your OFDM system has transformed what was originally a complex convolutional filtering problem into something that resembles a very clean AWGN channel with single-tap scaling. This dramatically simplifies receiver design and makes the whole process much more tractable.

Of course, there are trade-offs involved. For instance, you had to incorporate a cyclic prefix, and as a result, there’s a somewhat unusual frequency footprint. However, the ease of implementation and the simplicity of the receiver design often outweigh these costs when considering the bigger picture.

To summarize what we’ve covered: Broadband channels can be broken down into parallel channels for simpler equalization. The idea is to divide your data into parallel frequency bands, and by using the DFT and IDFT, you can effectively achieve this parallelization. One of the key requirements to make this work is circular convolution, which is facilitated by the cyclic prefix, an overhead, but a necessary one. Due to its simplicity and effectiveness, OFDM has become the dominant modulation technique in many modern wireless communication standards. For example, Wi-Fi and LTE both rely on OFDM or its variants. Even in wired communications, such as ADSL, a variation of OFDM known as discrete multi-tone (DMT) is widely used.

In the next lecture, we will dive into a couple of practical examples of implementing OFDM using GNU Radio, which will help clarify these concepts further. Thank you.