



**PPTs ON**

# **DIGITAL SIGNAL PROCESSING**

**(AEC012)**

**III B.Tech(ECE) VI semester**

**(IARE-R16) (2019-20)**

**Department of Electronics & Communication Engineering**

**By**

**Dr. S China Venkateswarlu, Professor-ECE.**

**Dr. G Manisha , Associate Professor, ECE**

**Ms. S Sushma, Assistant Professor, ECE**

**Mr. K Chaitanya, Assistant Professor, ECE**

## CO's

## Course outcomes

- | CO's | Course outcomes   |
|------|---|
| CO1  | Interpret, represent and process discrete/digital signals and systems.                                    |
| CO2  | Thorough understanding of time domain and frequency domain analysis of discrete time signals and systems. |
| CO3  | To apply DFT for the analysis of digital signals & systems.   |
| CO4  | Ability to design & analyze DSP systems like FIR and IIR Filter.  |
| CO5  | Design multi rate signal processing of signals through systems..  |



## UNIT-I

# REVIEW OF DISCRETE TIME SIGNALS AND SYSTEMS

## CLO's

## Course Learning Outcome

- |      |   |
|------|---|
| CLO1 | Understand how digital to analog (D/A) and analog to digital (A/D) converters operate on a signal and be able to model these operations mathematically..  |
| CLO2 | Define simple non-periodic discrete-time sequences such as the impulse and unit step, and perform time shifting and time-reversal operations on such sequences..  |
| CLO3 | Given the difference equation of a discrete-time system to demonstrate linearity, time-invariance, causality and stability, and hence show whether or not a given system belongs to the important class of causal, LTI systems. |



# Signals

- The term *signal* is generally applied to something that conveys information.
- *Signals* may, for example, convey information about the state or behavior of a physical system.
- As another class of examples, signals are synthesized for the purpose of communicating information between humans or between humans and machines.
- Although signals can be represented in many ways, in all cases, the information is contained in some pattern of variations.
- Signals are represented mathematically as functions of one or more independent variables

# What is a Signal?

- ⦿ A signal is a pattern of variation of some form
- ⦿ Signals are variables that carry information

**Examples of signal include:**

**Electrical signals**

- Voltages and currents in a circuit

**Acoustic signals**

- Acoustic pressure (sound) over time

**Mechanical signals**

- Velocity of a car over time

**Video signals**

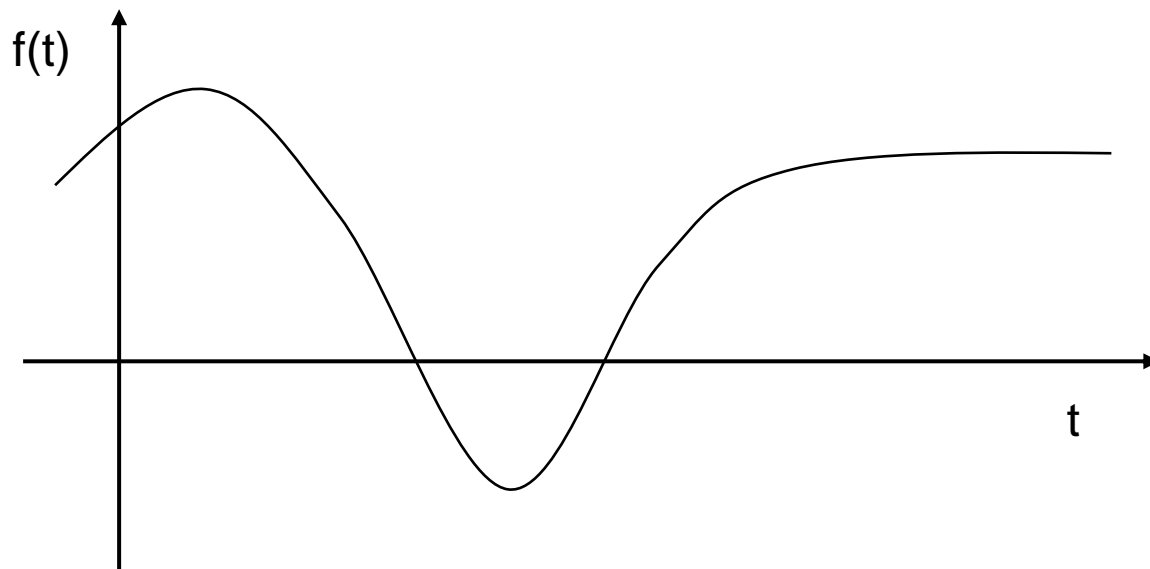
- Intensity level of a pixel (camera, video) over time

# How is a Signal Represented?

Mathematically, signals are represented as a function of one or more independent variables.

For instance a black & white video signal intensity is dependent on  $x, y$  coordinates and time  $t$   $f(x, y, t)$

On this course, we shall be exclusively concerned with signals that are a function of a single variable: time



# Signal Processing

- ◎ Humans are the most advanced signal processors
  - speech and pattern recognition, speech synthesis,...
- ◎ We encounter many types of signals in various applications
  - Electrical signals: voltage, current, magnetic and electric fields,...
  - Mechanical signals: velocity, force, displacement,...
  - Acoustic signals: sound, vibration,...
  - Other signals: pressure, temperature,...
- ◎ Most real-world signals are analog
  - They are continuous in time and amplitude
  - Convert to voltage or currents using sensors and transducers
- ◎ Analog circuits process these signals using
  - Resistors, Capacitors, Inductors, Amplifiers,...
- ◎ Analog signal processing examples
  - Audio processing in FM radios
  - Video processing in traditional TV sets

# Signal Properties

On this course, we shall be particularly interested in signals with certain properties:

Periodic signals: a signal is periodic if it repeats itself after a fixed period  $T$ ,  
i.e.  $x(t) = x(t+T)$  for all  $t$ . A  $\sin(t)$  signal is periodic.

Even and odd signals: a signal is even if  $x(-t) = x(t)$  (i.e. it can be reflected in the axis at zero). A signal is odd if  $x(-t) = -x(t)$ . Examples are  $\cos(t)$  and  $\sin(t)$  signals, respectively.

Exponential and sinusoidal signals: a signal is (real) exponential if it can be represented as  $x(t) = Ce^{at}$ . A signal is (complex) exponential if it can be represented in the same form but  $C$  and  $a$  are complex numbers.

Step and pulse signals: A pulse signal is one which is nearly completely zero, apart from a short spike,  $d(t)$ . A step signal is zero up to a certain time, and then a constant value after that time,  $u(t)$ .

These properties define a large class of tractable, useful signals and will be further considered in the coming lectures

# Limitations of Analog Signal Processing

- ⊙ Accuracy limitations due to
  - Component tolerances
  - Undesired nonlinearities
- ⊙ Limited repeatability due to
  - Tolerances
  - Changes in environmental conditions
    - Temperature
    - Vibration
- ⊙ Sensitivity to electrical noise
- ⊙ Limited dynamic range for voltage and currents
- ⊙ Inflexibility to changes
- ⊙ Difficulty of implementing certain operations
  - Nonlinear operations
  - Time-varying operations
- ⊙ Difficulty of storing information

# Signal-processing systems

- Signal-processing systems may be classified along the same lines as signals.
- That is, **continuous-time systems** are systems for which both the input and the output are continuous-time signals, and
- **discrete-time systems** are those for which both the input and the output are discrete-time signals.
- Similarly, a **digital system** is a system for which both the input and the output are digital signals.
- **Digital signal processing**, then, deals with the transformation of signals that are discrete in both amplitude and time

# Digital Signal Processing

- ◎ Represent signals by a sequence of numbers
  - Sampling or analog-to-digital conversions
- ◎ Perform processing on these numbers with a digital processor
  - Digital signal processing
- ◎ Reconstruct analog signal from processed numbers
  - Reconstruction or digital-to-analog conversion



- Analog input – analog output
  - Digital recording of music
- Analog input – digital output
  - Touch tone phone dialing
- Digital input – analog output
  - Text to speech
- Digital input – digital output
  - Compression of a file on computer

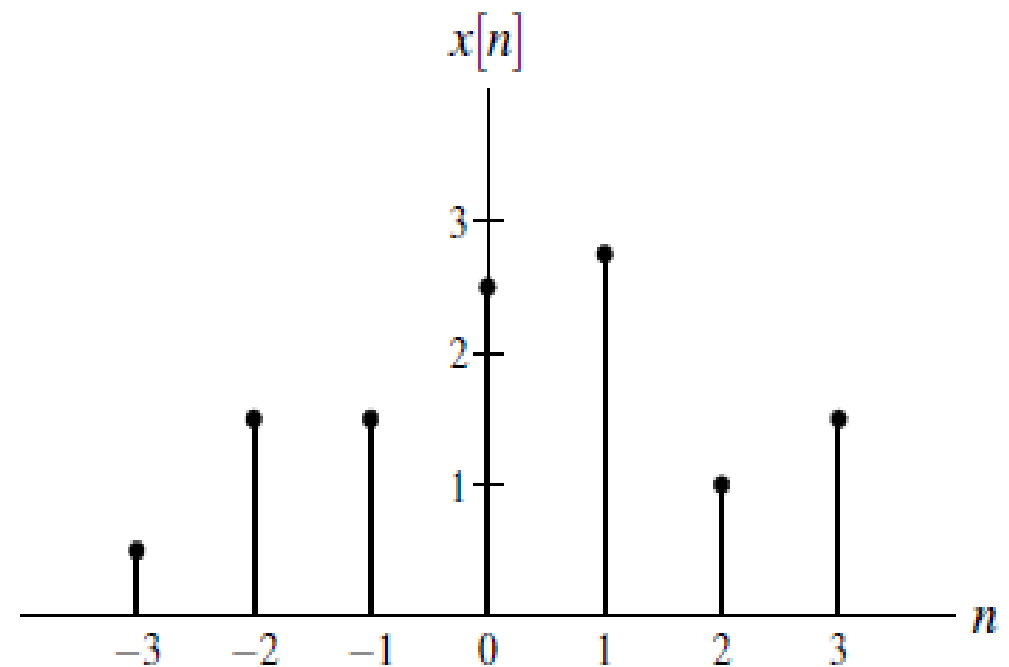
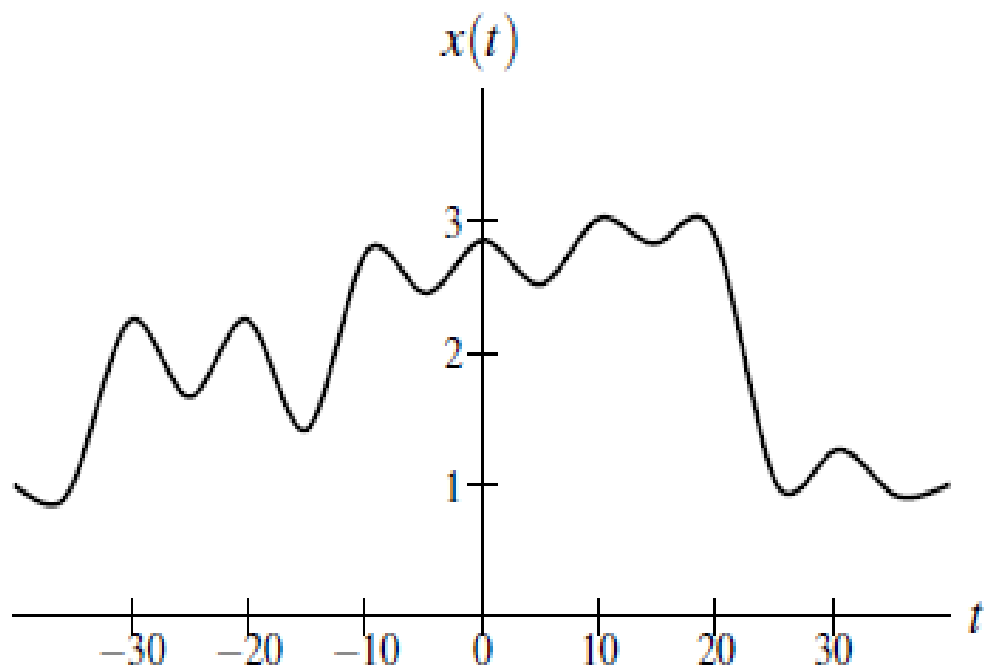


- signal: function of one or more variables that conveys information about some (usually physical) phenomenon
- for function  $f(t_1, t_2, \dots, t_n)$ , each of  $\{t_k\}$  is called independent variable, function value itself referred to as dependent variable
- examples of signals:
  - voltage or current in electronic circuit
  - position, velocity, and acceleration of object
  - forces or torques in mechanical system
  - flow rates of liquids or gases in chemical process
  - digital image, digital video, digital audio
  - stock market index

# Classifications of Signal

- number of independent variables (i.e., dimensionality)
  - one: one-dimensional (e.g., audio)
  - more than one: multi-dimensional (e.g., image)
- continuous or discrete independent variables
  - continuous: continuous-time (e.g., voltage waveform)
  - discrete: discrete-time (e.g., stock market index)
- continuous or discrete dependent variable
  - continuous: continuous-valued (e.g., voltage waveform)
  - discrete: discrete-valued (e.g., digital image)
- continuous-valued continuous-time: analog (e.g., voltage waveform)
- discrete-valued discrete-time: digital (e.g., digital audio)

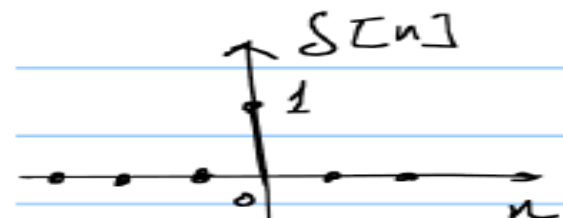
- sequence: discrete-time signal
- independent variables enclosed in parentheses for continuous-time signal (e.g.,  $x(t)$ ) and brackets for discrete-time signal (e.g.,  $x[n]$ )



# Basic Discrete Time Signals

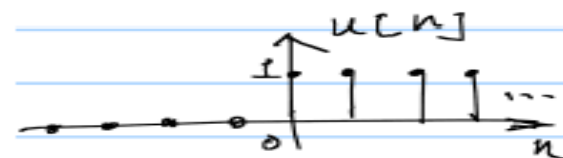
## 1 unit pulse (unit sample)

$$\delta[n] = \begin{cases} 1 & n = 0 \\ 0 & \text{otherwise} \end{cases}$$



## 2 unit step

$$u[n] = \begin{cases} 1 & n \geq 0 \\ 0 & \text{otherwise} \end{cases}$$



## 3 Sinusoids and complex exponentials

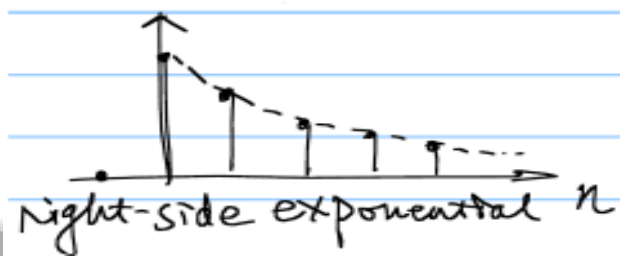
$$x_1[n] = A \cos(\omega_0 n + \theta)$$

$$x_2[n] = a e^{j\omega_0 n}$$

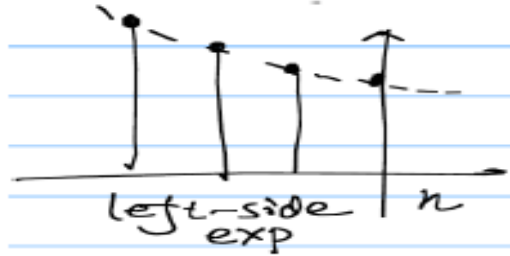
- $x_2[n]$  has real and imaginary parts; known as a single-frequency signal.

## 4 Exponentials

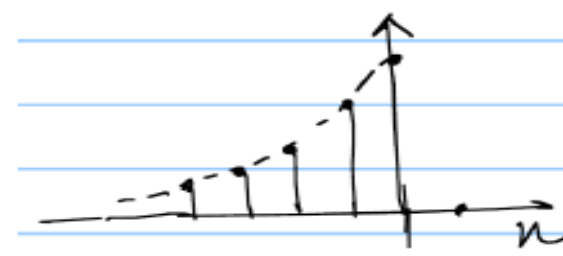
$$x[n] = a^n u[n] \quad (0 < a < 1)$$



$$x[n] = a^n u[-n]$$



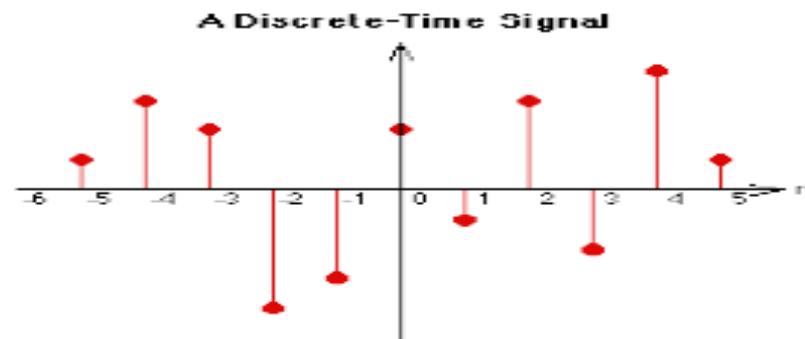
$$x[n] = a^{-n} u[-n]$$



## Discrete Time signals

The signals, which are defined at discrete times are known as discrete signals. Therefore, every independent variable has distinct value. Thus, they are represented as sequence of numbers.

Although speech and video signals have the privilege to be represented in both continuous and discrete time format; under certain circumstances, they are identical. Amplitudes also show discrete characteristics. Perfect example of this is a digital signal; whose amplitude and time both are discrete.



The figure above depicts a discrete signal's discrete amplitude characteristic over a period of time. Mathematically, these types of signals can be formularized as;

$$x=\{x[n]\}, -\infty < n < \infty$$

Where, n is an integer.

It is a sequence of numbers x, where nth number in the sequence is represented as x[n].

# Signals

- Signal: a continuous-time signal  $x(t)$  (discrete-time signal  $x[n]$ ) is a function of an independent continuous variable  $t$  (discrete variable  $n$ ).
- Elementary continuous-time signals:
  1.  $x(t) = e^{s_0 t}$ ,  $s_0 = \sigma_0 + j\omega_0$  (complex exponential)
  2.  $x(t) = e^{j\omega_0 t}$ ,  $s_0 = j\omega_0$  (periodic complex exponential)
  3.  $x(t) = e^{\sigma_0 t}$ ,  $s_0 = \sigma_0$  (real exponential)
  4.  $x(t) = \cos \omega_0 t = \text{Re}\{e^{j\omega_0 t}\}$  (sinusoidal signals)
  5. impulse function:  $\delta(t)$
  6. unit function:  $u(t)$
  7. ramp function:  $r(t)$

- Elementary discrete-time signals:
  1.  $x[n] = z_0^n, z_0 = r_0 e^{j\omega_0}$  (complex exponential)
  2.  $x[n] = e^{j\Omega_0 n}, z_0 = e^{j\Omega_0}$  (periodic complex exponential)
  3.  $x[n] = r_0^n, z_0 = r_0$  (real exponential)
  4.  $x[n] = \cos \Omega_0 n = \text{Re}\{e^{j\Omega_0 n}\}$  (sinusoidal signals)
  5. impulse function:  $\delta[n]$
  6. unit function:  $u[n]$
  7. ramp function:  $r[n]$
- We will treat continuous-time and discrete-time signals separately but in parallel.

## Classification of signals

1. continuous-time  $x(t)$  vs. discrete-time  $x[n]$

- Usually a discrete-time signal  $x[n]$  is obtained from a continuous time signal  $x(t)$  by sampling:

$$x[n] = x(nT), \quad n = 0, \pm 1, \pm 2 \dots \text{ for some fixed } T.$$

2. even vs. odd signals

- even (real):  $x(-t) = x(t)$
- odd (real):  $x(-t) = -x(t)$
- symmetric (complex):  $x(-t) = x^*(t)$
- anti-symmetric (complex):  $x(-t) = -x^*(t)$



Any signal  $x(t)$  can be decompose into the even part  $x_e(t)$  and the odd part  $x_o(t)$  by:

$$x(t) = \frac{1}{2}[x(t) + x(-t)] + \frac{1}{2}[x(t) - x(-t)],$$

where

$$x_e(t) = \frac{1}{2}[x(t) + x(-t)] \text{ and } x_o(t) = \frac{1}{2}[x(t) - x(-t)]$$

- It is easy to check that  $x_e(t) = x_e(-t)$  ,  $x_o(t) = -x_o(t)$ .

### 3. periodic vs. aperiodic signals

- A signal  $x(t)$  ( $x[n]$ ) is called a periodic signal if there exist real number  $T$  (integer  $N$ ) such that:

$$x(t + T) = x(t) \quad (x[n + N] = x[n]).$$

- The smallest  $T_0$  ( $N_0$ ) such that :

$$x(t + T_0) = x(t) \quad (x[n + N_0] = x[n])$$

is called the (fundamental) period of  $x(t)$  ( $x[n]$ ).

- $\frac{2\pi}{T_0}$  ( $\frac{2\pi}{N_0}$ ) is called the fundamental frequency ( $\frac{rad}{sec}$ ) of  $x(t)$  ( $x[n]$ ).
- $x(t)$  ( $x[n]$ ) is called aperiodic if it is not periodic.

#### 4. deterministic vs. random

- deterministic signal  $x(t)$   
 $\Rightarrow x(t_0)$  is a number, no uncertainty
- random signal  $x(t)$   
 $\Rightarrow x(t_0)$  is a random variable (with some probability specification)  
 $x(t) = \text{random signal} = \text{random process} = \text{stochastic process}$

#### 5. energy signal vs. power signal

- for a continuous signal  $x(t)$ :  
 $E = \int_{-\infty}^{\infty} x^2(t) dt$  : energy  
 $P = \lim_{T \rightarrow \infty} \frac{1}{T} \int_{-\frac{T}{2}}^{\frac{T}{2}} x^2(t) dt$  : power  
 $= \frac{1}{T} \int_{-\frac{T}{2}}^{\frac{T}{2}} x^2(t) dt$  if  $x(t)$  is periodic with period  $T$

- for a discrete signal  $x[n]$

$$E = \sum_{n=-\infty}^{\infty} x^2[n]: \text{ energy}$$

$$P = \lim_{N \rightarrow \infty} \frac{1}{2N} \sum_{n=-N}^{N-1} x^2[n]: \text{ power}$$

$$= \frac{1}{N} \sum_{n=0}^{N-1} x^2[n] \quad \text{periodic with period } N$$

- $x(t)$  ( $x[n]$ ) is an energy signal

$$\text{if } 0 < E < \infty$$

or is a power signal

$$\text{if } 0 < P < \infty$$

- A signal  $x(t)$  ( $x[n]$ ) can not be an energy signal and a power signal simultaneously.

# Pros and Cons of DSP

## ◎ Pros

- Accuracy can be controlled by choosing word length
- Repeatable
- Sensitivity to electrical noise is minimal
- Dynamic range can be controlled using floating point numbers
- Flexibility can be achieved with software implementations
- Non-linear and time-varying operations are easier to implement
- Digital storage is cheap
- Digital information can be encrypted for security
- Price/performance and reduced time-to-market

## ◎ Cons

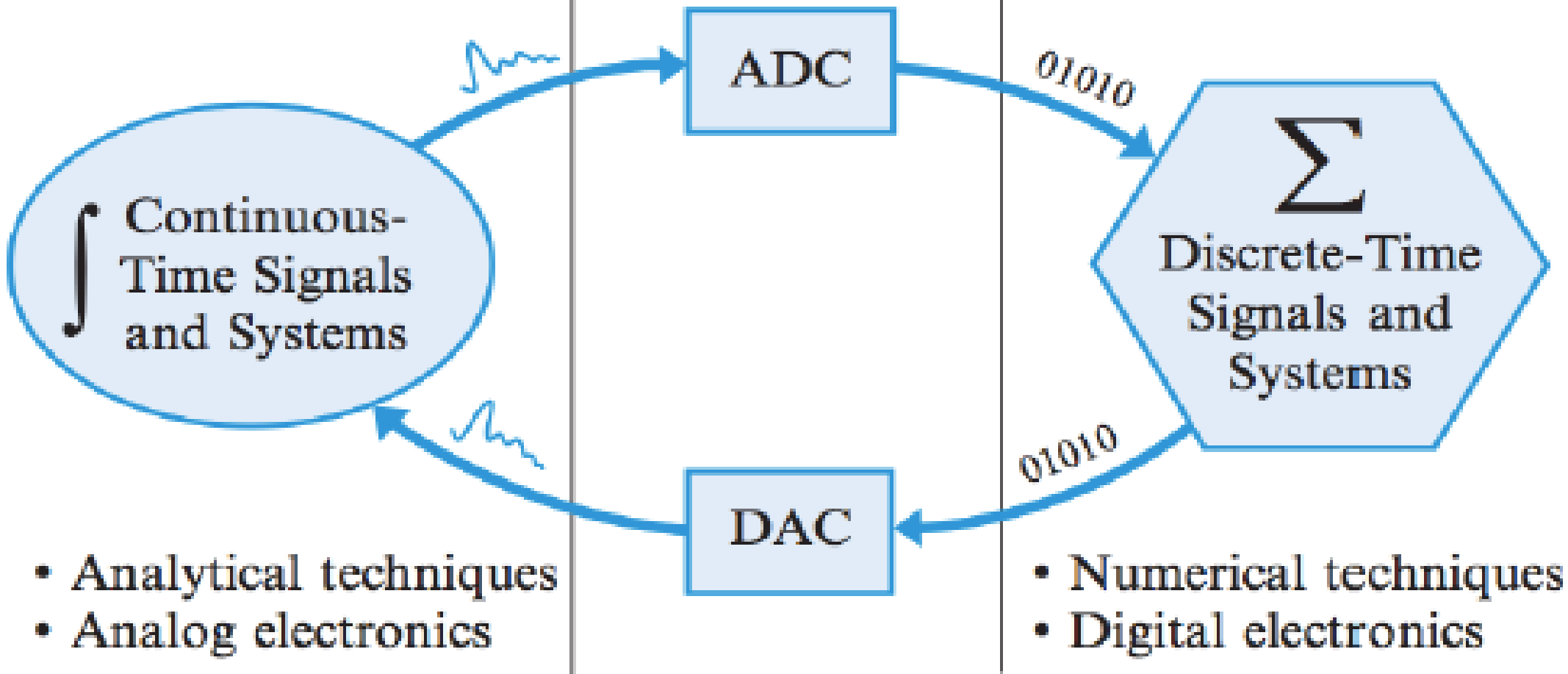
- Sampling causes loss of information
- A/D and D/A requires mixed-signal hardware
- Limited speed of processors
- Quantization and round-off errors

# Analog, digital, mixed signal processing

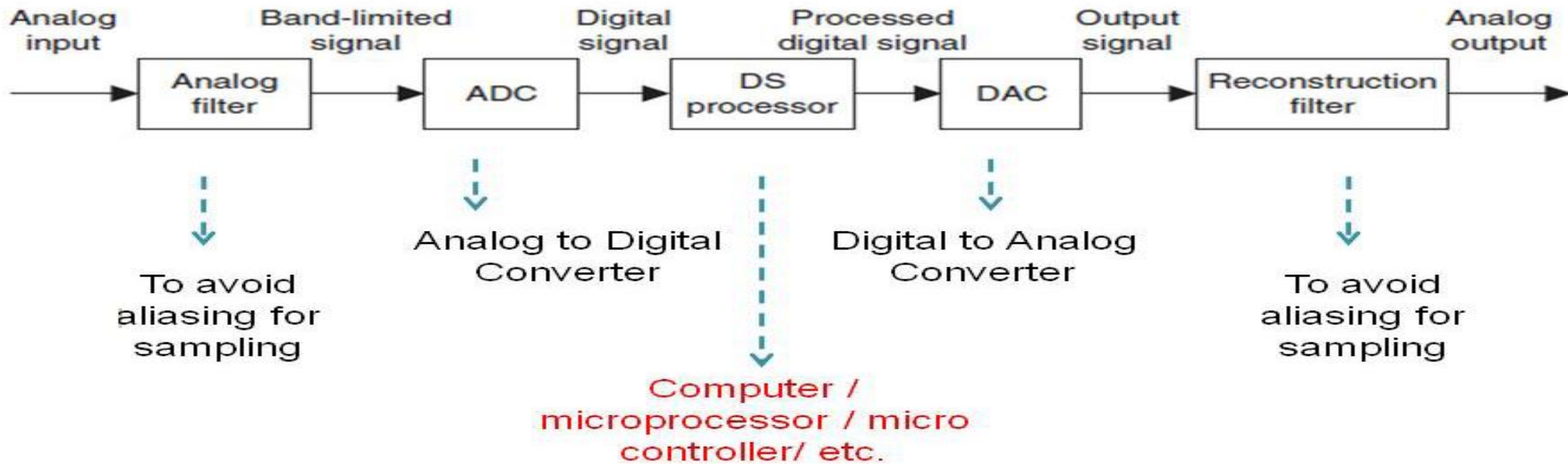
Analog  
Systems

Interface  
Systems

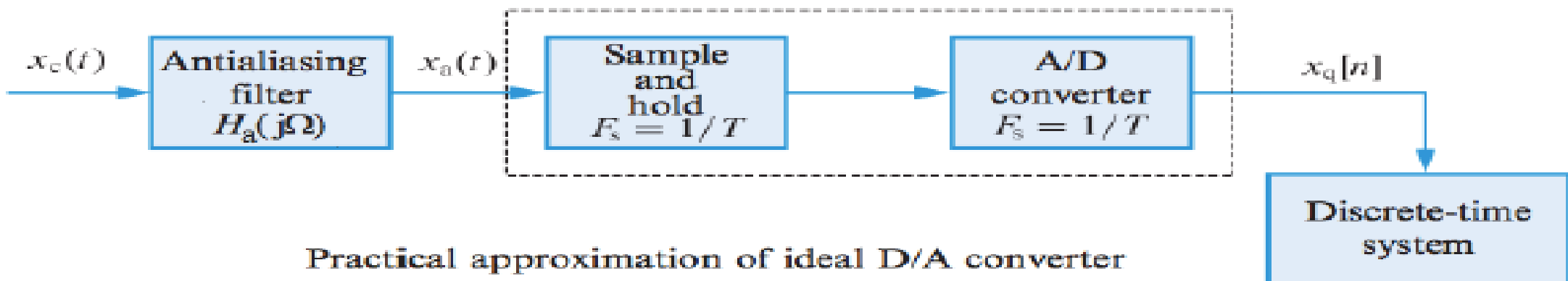
Digital  
Systems



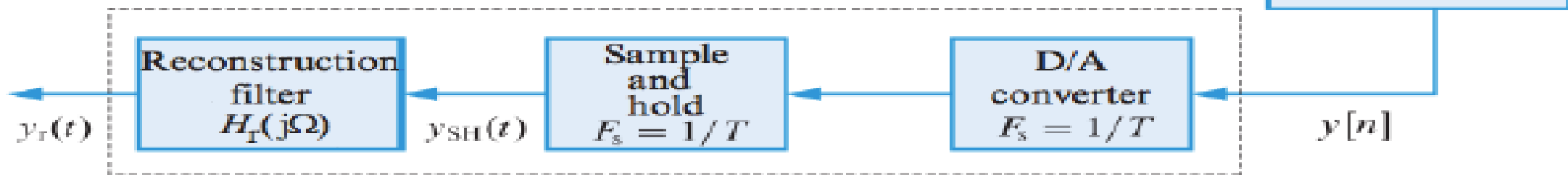
# Digital Signal Processing



## Practical approximation of ideal A/D converter

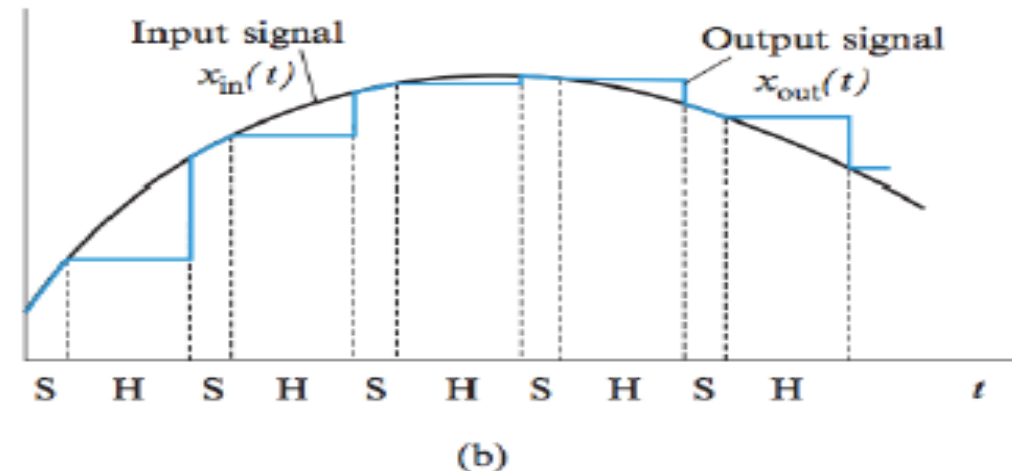
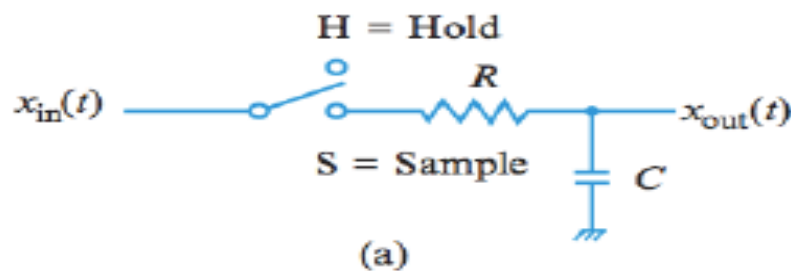


## Practical approximation of ideal D/A converter



# Sampling and reconstruction

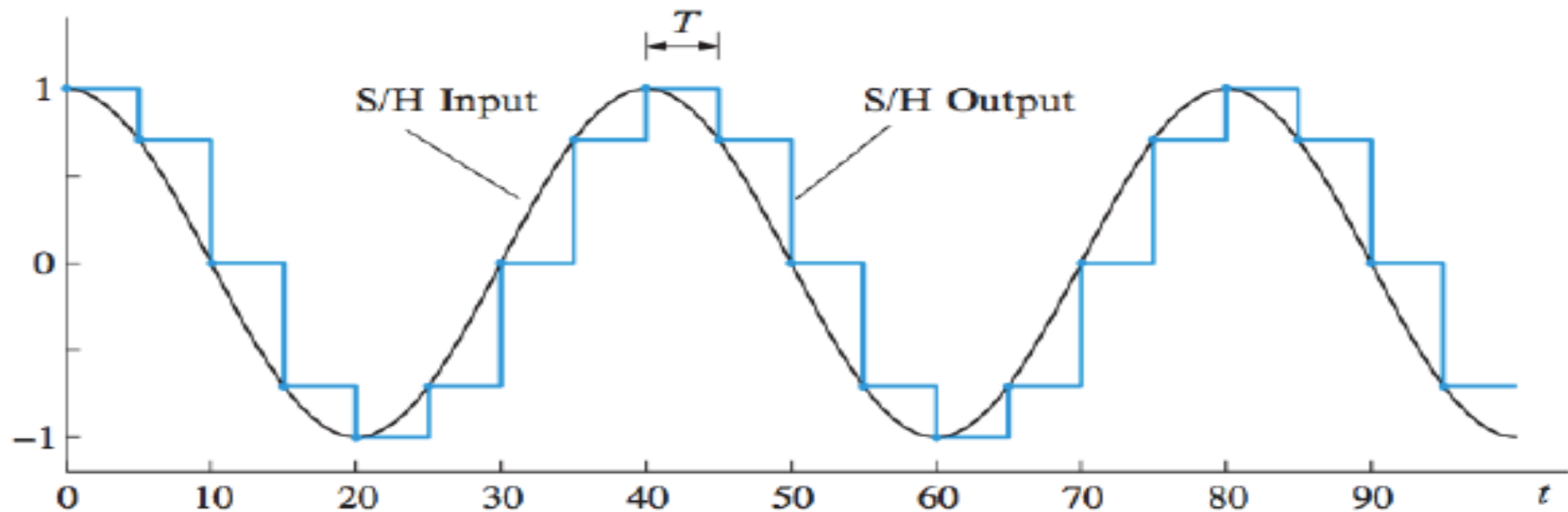
- The main function of the *low-pass antialiasing filter* is to band-limit the input signal to the folding frequency without distortion.
- It should be noted that even if the signal is band-limited, there is always wide-band additive noise which will be folded back to create aliasing.
- When an analog voltage is connected directly to an ADC, the conversion process can be adversely affected if the voltage is changing during the conversion time.
- The quality of the conversion process can be improved by using a *sample-and-hold (S/H) circuit*.





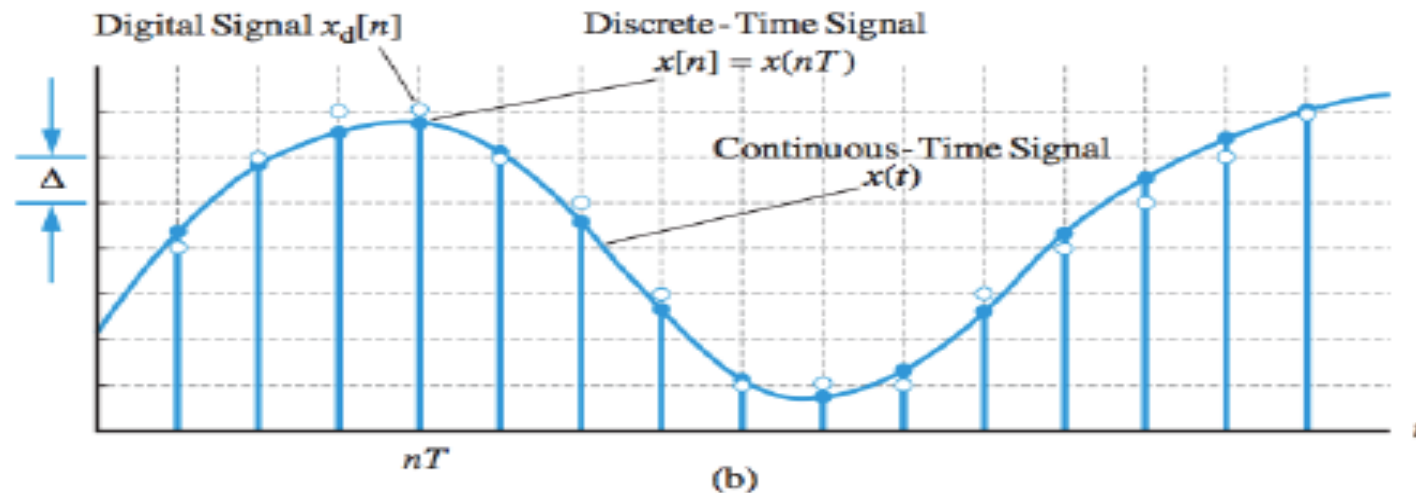
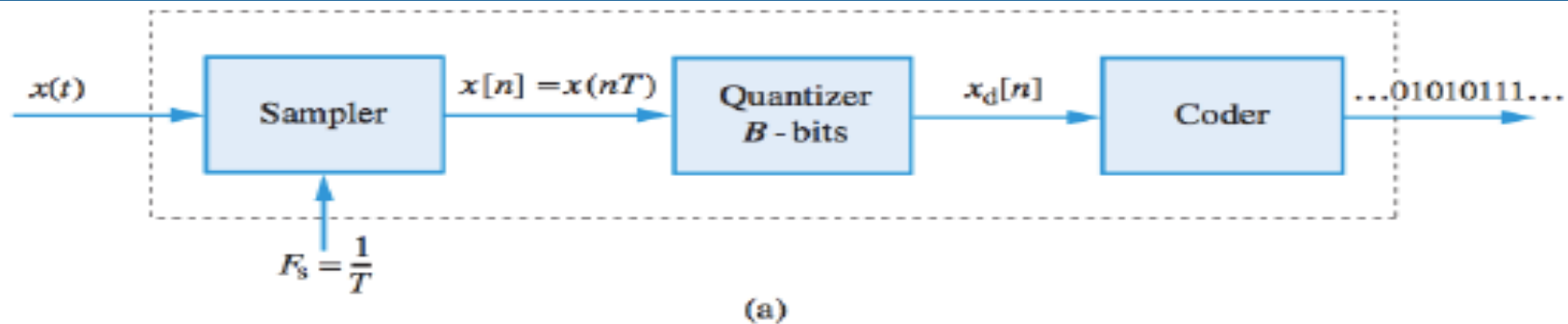
# Sample and hold (S/H)circuit

- Since the sampling operation is performed by the S/H circuit, the role of S/H is to sample  $x_c(t)$  as instantaneously as possible and to hold the sample value as constant as possible until the next sample.
- Thus, the output of the S/H circuit can be modelled as a staircase waveform where each sample value is held constant until the acquisition of the next sample.



- Note that the S/H system is linear but time-varying.

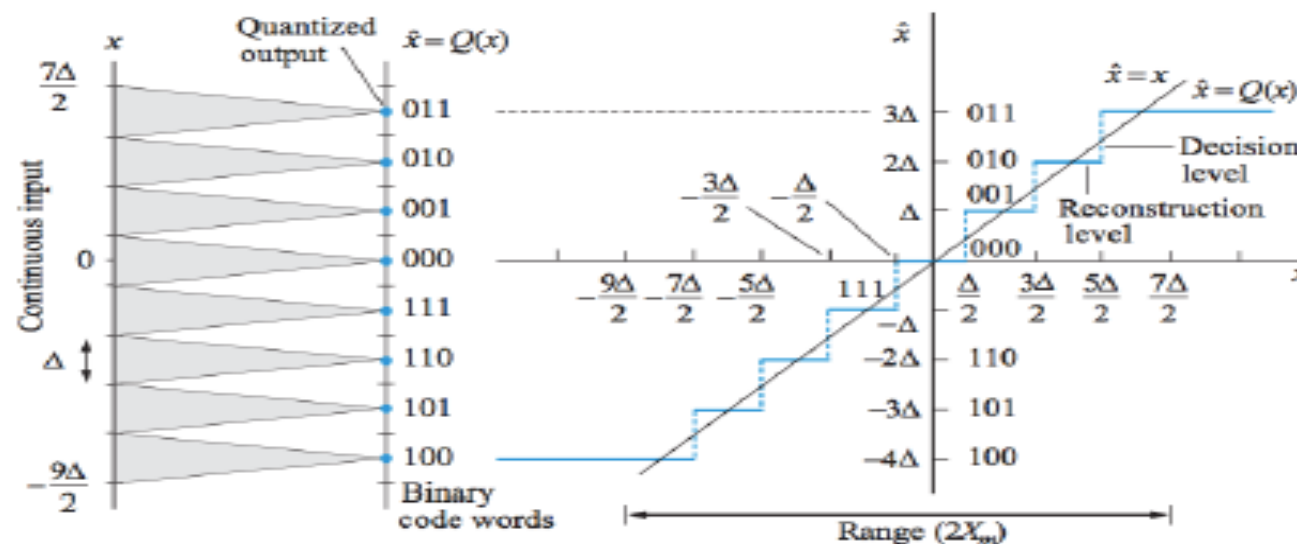
# A/D converter



- *Quantization* converts a continuous-amplitude signal  $x(t)$  into a discrete-amplitude signal  $x_d[n]$ .
- In theory, we are dealing with discrete-time signals; in practice, we are dealing with digital signals.

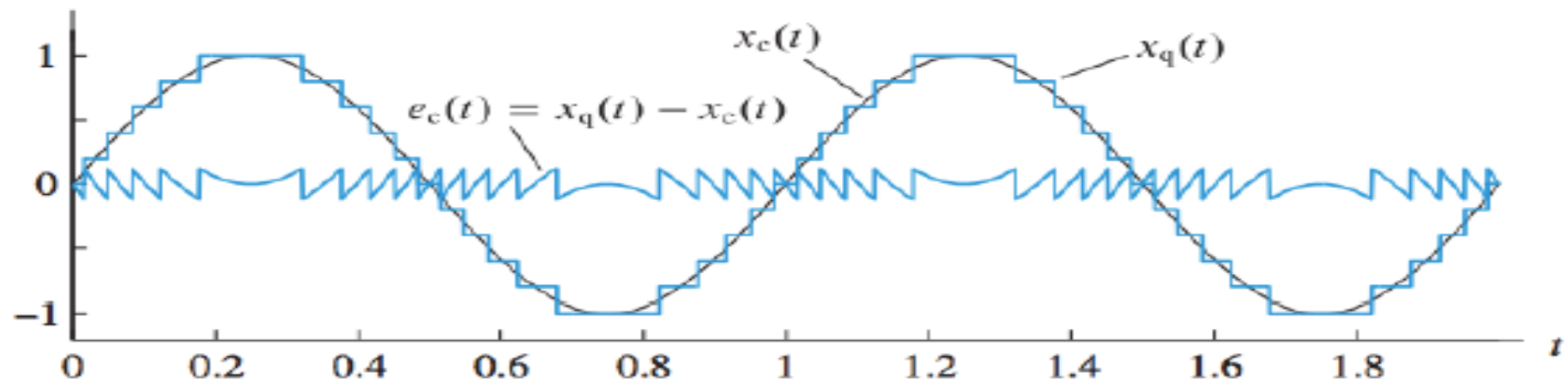
# A/D converter

- The major difference between ideal and practical conversion is that an ADC generates sample values that are known with finite precision.
- The ADC is the device in which both quantization and binary coding of the sampled signal take place.
- A  $B$ -bit quantizer can represent  $2^B$  different numbers.
- If the input amplitude range is divided into  $K$  quantization intervals of equal width  $\Delta$  (*quantization step*) and the output levels are uniformly spaced, the resulting quantizer is called *uniform*.



# Quantization noise

- The two major types of error introduced by an ADC are aliasing error and *quantization error*.



- Since quantization is a nonlinear operation, analysis of quantization error is done using statistical techniques.
- If there is a large number of small quantization intervals, the signal  $x_c(t)$  can be assumed to be approximately linear between quantization levels. In this case:

$$e_c(t) \triangleq x_q(t) - x_c(t) = \frac{\Delta}{2\tau}t, \quad -\tau \leq t \leq \tau$$

- Then the mean squared quantization error power is

$$P_Q = \frac{1}{2\tau} \int_{-\tau}^{\tau} |e_c(t)|^2 dt = \frac{\Delta^2}{12}$$

# D/A conversion

- A band-limited signal can be reconstructed from a sequence of samples using the ideal DAC described by

$$x_r(t) = \sum_n x[n] g_{BL}(t - nT) = \sum_n x[n] \text{sinc}(t/T - n)$$

- A system that implements the above formula, for an arbitrary function  $g_r(t)$ , is known as a *practical digital-to-analog converter (DAC)*.
- The function  $g_r(t)$  is also known as the *characteristic pulse* of a DAC. At each sample time  $t = nT$ , the converter generates a pulse  $g_r(t - nT)$  scaled by  $x[n]$ .
- In particular, the *switch-and-hold* DAC performs the following operation

$$x_{SH}(t) = \sum_n x_q[n] g_{SH}(t - nT)$$

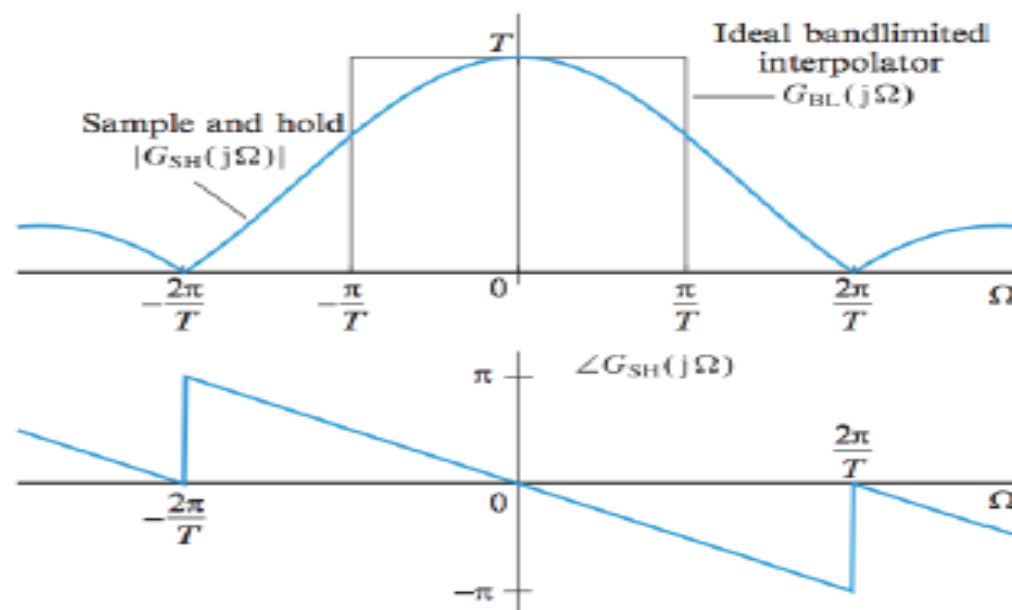
where

$$g_{SH}(t) = \begin{cases} 1, & 0 \leq t \leq T \\ 0, & \text{otherwise} \end{cases} \iff G_{SH}(j\Omega) = \frac{2 \sin(\Omega T / 2)}{\Omega} e^{-j\Omega T / 2}$$



# D/A conversion

- The S/H circuit cannot completely eliminate the spectral replicas introduced by the sampling process.
- Moreover, it introduces amplitude distortion in the Nyquist band  $|\Omega| < \pi/T$ .



- To compensate for the effects of the S/H circuit, we can use an analog post-filter  $H_r(j\Omega)$  so that  $G_{SH}(j\Omega) H_r(j\Omega) = G_{BL}(j\Omega)$ :

$$H_r(j\Omega) = \begin{cases} \frac{\Omega T/2}{\sin(\Omega T/2)} e^{j\Omega T/2}, & |\Omega| < \pi/T \\ 0, & \text{otherwise} \end{cases}$$

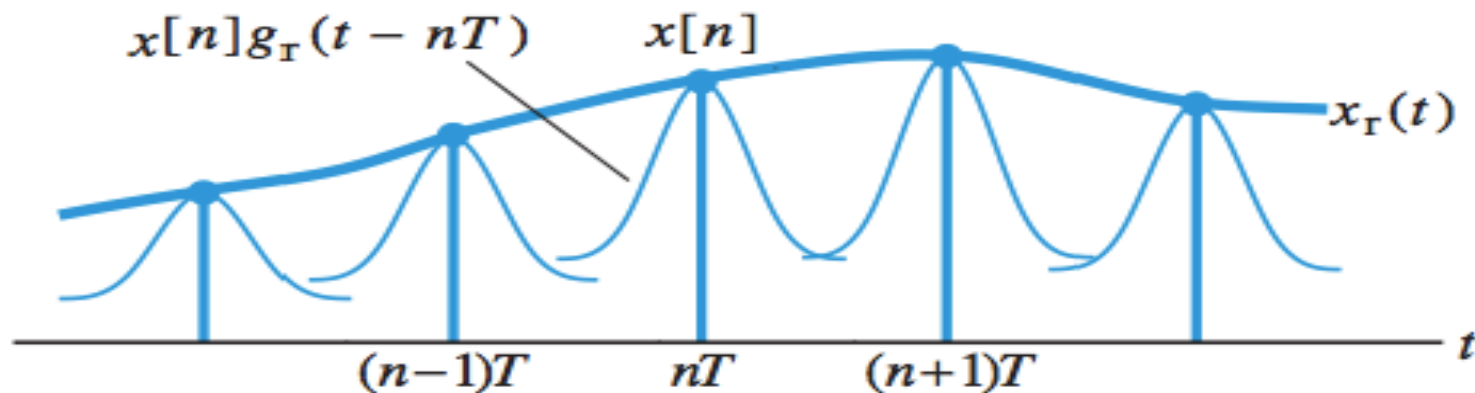
# Reconstruction

- A general formula that describes a broad class of reconstruction processes is given by

$$x_r(t) = \sum_n x[n]g_r(t - nT)$$

where  $g_r(t)$  is an interpolating reconstruction function.

- The process of fitting a continuous function to a set of samples is known as an *interpolation*.



- Thus, if the interpolation function has duration greater than or equal to  $T$ , the addition of the overlapping copies “fills the gaps” between samples.

# Reconstruction

- In the Fourier domain, the interpolation formula becomes

$$X_r(j\Omega) = \sum_n x[n] G_r(j\Omega) e^{-j\Omega n T} = G_r(j\Omega) \underbrace{\sum_n x[n] e^{-j\Omega n T}}_{X(e^{j\Omega T})}$$

- Consequently, we obtain

$$X_r(j\Omega) = G_r(j\Omega) X(e^{j\Omega T})$$

- Specifically, if we choose  $g_r(t)$  so that

$$G_r(j\Omega) \triangleq G_{BL}(j\Omega) = \begin{cases} T, & |\Omega| \leq \Omega_s/2 \\ 0, & |\Omega| > \Omega_s/2 \end{cases}$$

then  $X_r(j\Omega) = X_c(j\Omega)$  and, therefore,  $x_r(t) = x_c(t)$ .



# Reconstruction

- Evaluating the inverse Fourier transform of  $G_{BL}(j\Omega)$ , we obtain

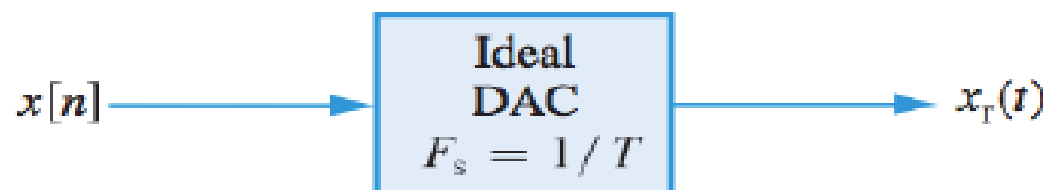
$$g_r(t) \triangleq g_{BL}(t) = \frac{\sin(\pi t/T)}{\pi t/T} = \text{sinc}(t/T)$$

- In this case we obtain:

The ideal interpolation formula

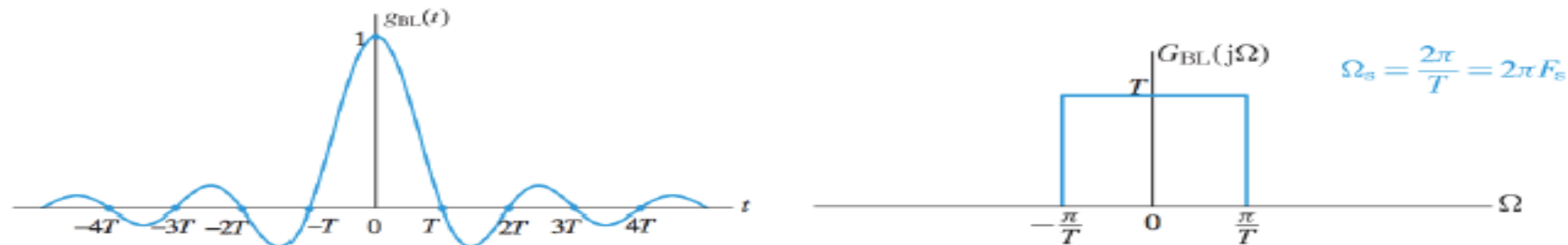
$$x_c(t) = \sum_n x[n] \frac{\sin[\pi(t - nT)/T]}{\pi(t - nT)/T}$$

- The system used to implement the ideal interpolation is known as an *ideal DAC*.

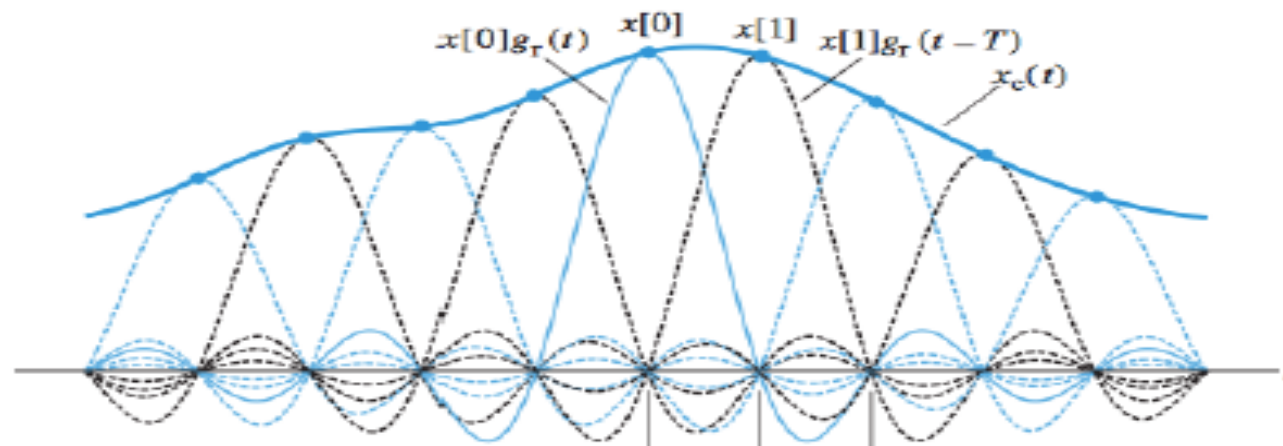


# Reconstruction

- To understand the meaning and implications of the ideal interpolation we look more closely at the sinc function  $g_{BL}(t)$ .



- We note that  $g_{BL}(t) = 0$  at all  $t = nT$ , except at  $t = 0$  where  $g_{BL}(t) = 1$ . Thus, it is always true that  $x_r(nT) = x_c(nT)$  regardless of whether aliasing occurred during sampling.



# Signals

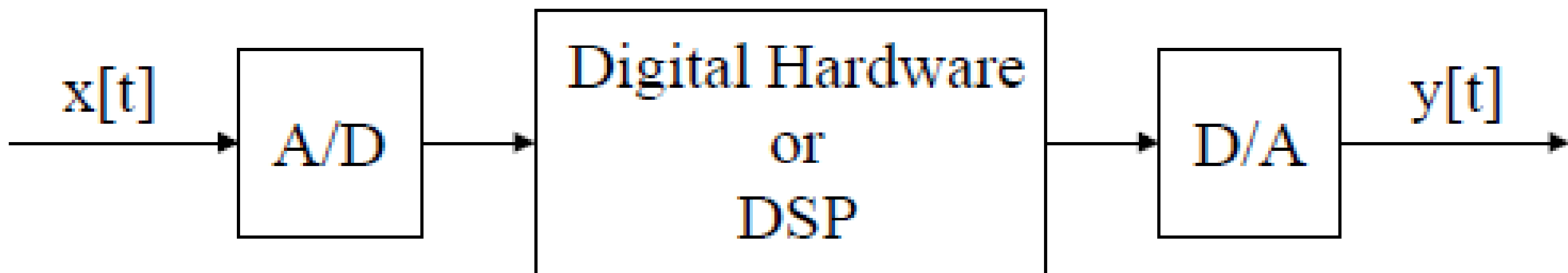
- ⦿ *Continuous-time* signals are functions of a real argument  $x(t)$  where  $t$  can take any real value  
 $x(t)$  may be 0 for a given range of values of  $t$
- ⦿ *Discrete-time signals* are functions of an argument that takes values from a discrete set  
 $x[n]$  where  $n \in \{\dots-3,-2,-1,0,1,2,3\dots\}$   
Integer index  $n$  instead of time  $t$  for discrete-time systems  
 $x$  may be an array of values (multi channel signal)
- ⦿ Values for  $x$  may be real or complex

### What is a system?

- a system is a device (machine), process or algorithm which has (multiple) inputs and (multiple) outputs. Digital systems are implemented using either software or digital circuitry. Analog (electrical) systems are implemented using circuit devices such as resistors, operational amplifiers and capacitors. This class is concerned only with the development of digital systems.

The basic elements of a DSP system are:

- ◆ A/D converter (sampler, quantizer and anti-aliasing filter)
- ◆ DSP
- ◆ D/A converter (hold and anti-imaging filter)
- ◆  $\Sigma\Delta$  modulator (if not A/D and D/A) – This is called the “all digital” solution



# Discrete-time Signals and Systems



- ⦿ *Continuous-time signals* are defined over a continuum of times and thus are represented by a continuous independent variable.
- ⦿ *Discrete-time signals* are defined at discrete times and thus the independent variable has discrete values.
- ⦿ *Analog signals* are those for which both time and amplitude are continuous.
- ⦿ *Digital signals* are those for which both time and amplitude are discrete.

# Signal Types

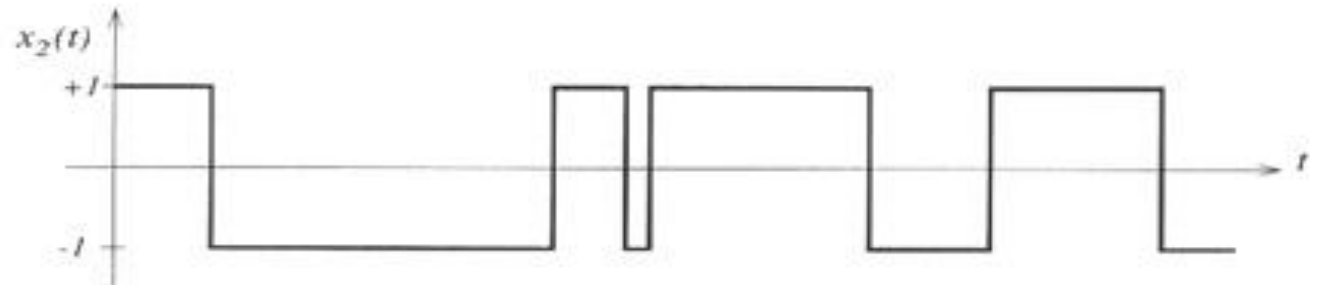
- ⊙ Analog signals: continuous in time and amplitude
  - *Example: voltage, current, temperature,...*
- ⊙ Digital signals: discrete both in time and amplitude
  - *Example: attendance of this class, digitizes analog signals,...*
- ⊙ Discrete-time signals: discrete in time, continuous in amplitude
  - *Example: hourly change of temperature*
- ⊙ Theory of digital signals would be too complicated
  - *Requires inclusion of nonlinearities into theory*
- ⊙ Theory is based on discrete-time continuous-amplitude signals
  - *Most convenient to develop theory*
  - *Good enough approximation to practice with some care*
- ⊙ In practice we mostly process digital signals on processors
  - *Need to take into account finite precision effects*

# Signal Types

Continuous time –  
Continuous amplitude



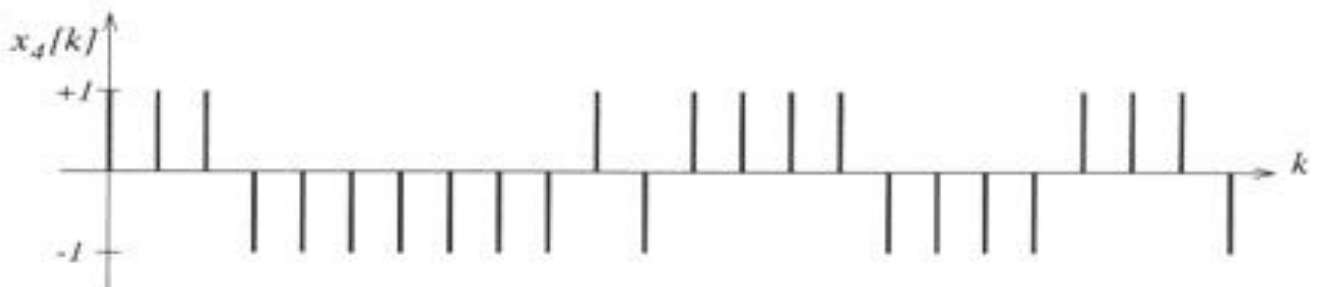
Continuous time –  
Discrete amplitude



Discrete time –  
Continuous amplitude



Discrete time –  
Discrete amplitude



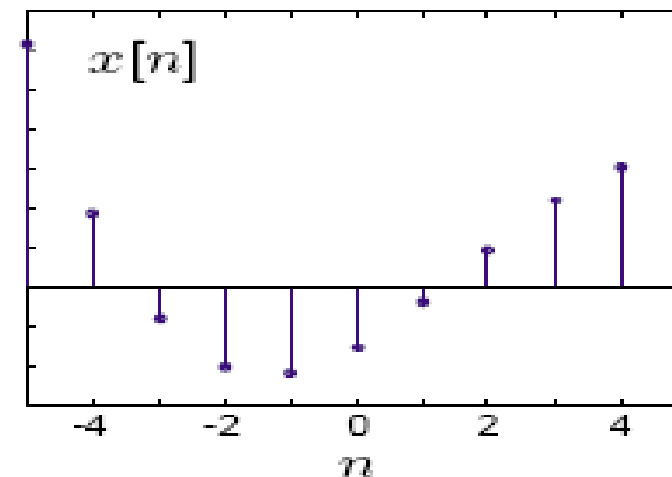
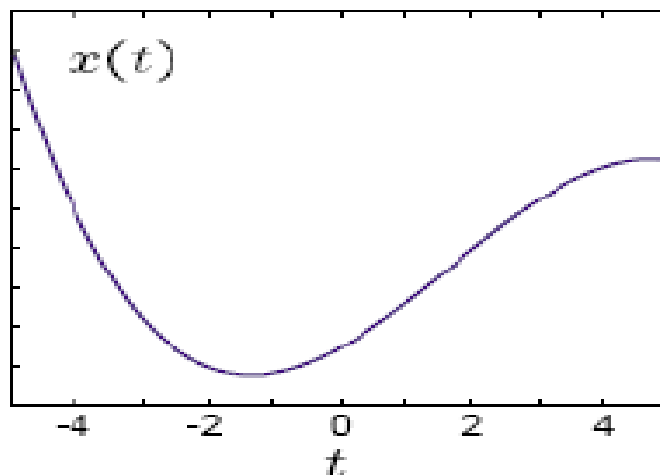
# Signal Basics

- **Continuous time (CT) and discrete time (DT) signals**

CT signals take on real or complex values as a function of an independent variable that ranges over the real numbers and are denoted as  $x(t)$ .

DT signals take on real or complex values as a function of an independent variable that ranges over the integers and are denoted as  $x[n]$ .

Note the subtle use of parentheses and square brackets to distinguish between CT and DT signals.





# Discrete-Time Signals

- **Sampling** is the acquisition of the values of a continuous-time signal at discrete points in time
- $x(t)$  is a continuous-time signal,  $x[n]$  is a discrete-time signal

$x[n] = x(nT_s)$  where  $T_s$  is the time between samples

# Discrete Time Exponential and Sinusoidal Signals

- DT signals can be defined in a manner analogous to their continuous-time counter part

$$x[n] = A \sin (2\pi n/N_0 + \theta)$$

$$= A \sin (2\pi F_0 n + \theta)$$

Discrete Time Sinusoidal Signal

$$x[n] = a^n$$

Discrete Time Exponential Signal

$n$  = the discrete time

$A$  = amplitude

$\theta$  = phase shifting radians,

$N_0$  = Discrete Period of the wave

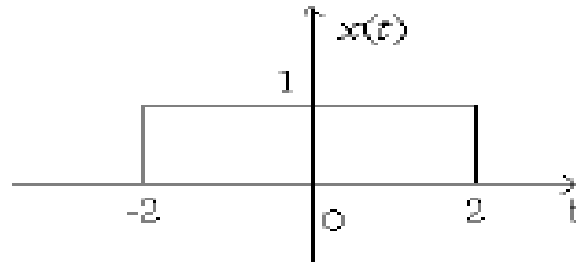
$1/N_0 = F_0 = \Omega_0/2\pi$  = Discrete Frequency

# Operations of Signals

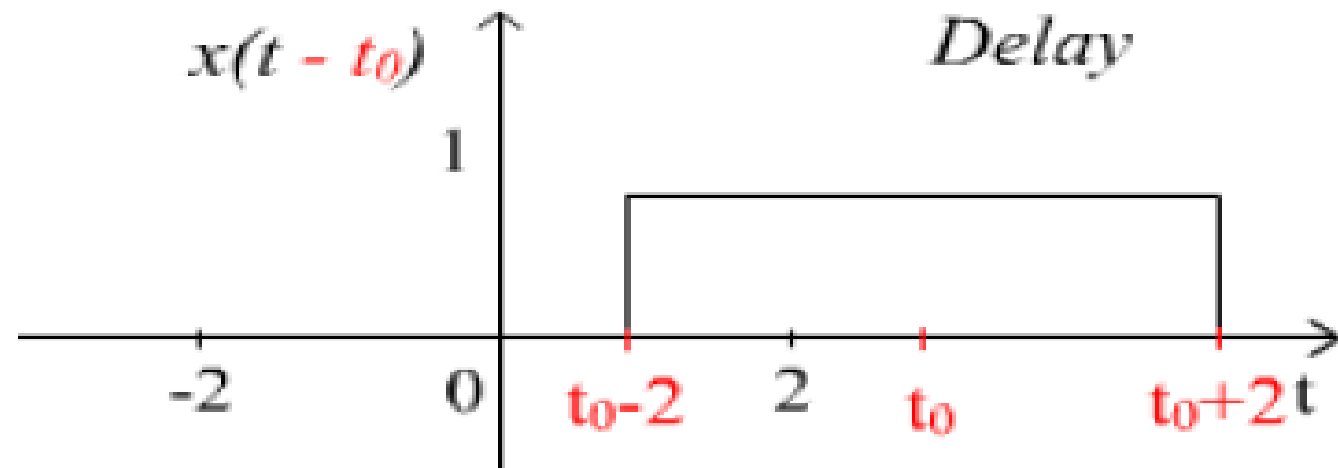
- Sometime a given mathematical function may completely describe a signal .
- Different operations are required for different purposes of arbitrary signals.
- The operations on signals can be
  - Time Shifting
  - Time Scaling
  - Time Inversion or Time Folding

# Time Shifting

- The original signal  $x(t)$  is shifted by an amount  $t_0$ .

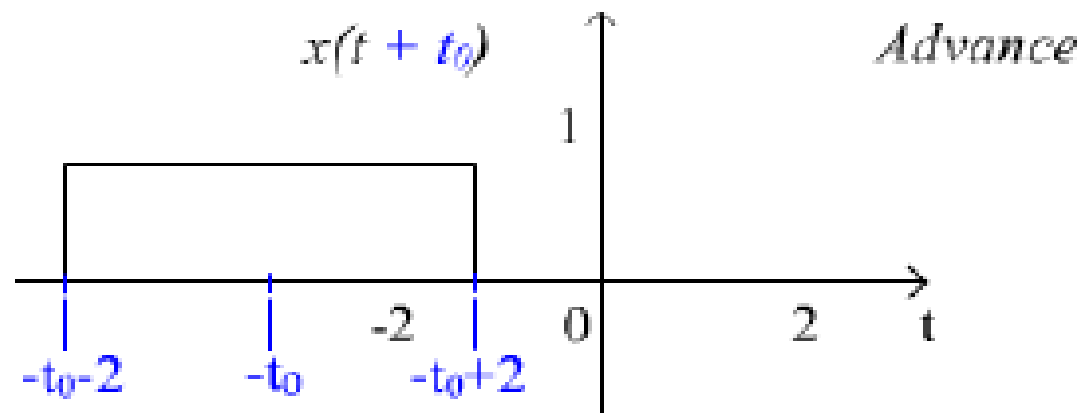


- $X(t) \rightarrow X(t-t_0) \rightarrow$  Signal Delayed  $\rightarrow$  Shift to the right



# Time Shifting Contd.

- $X(t) \rightarrow X(t+t_0) \rightarrow$  Signal Advanced  $\rightarrow$  Shift to the left



# Discrete-time Signals and Systems



- ⦿ ***Continuous-time signals*** are defined over a continuum of times and thus are represented by a continuous independent variable.
- ⦿ ***Discrete-time signals*** are defined at discrete times and thus the independent variable has discrete values.
- ⦿ ***Analog signals*** are those for which both time and amplitude are continuous.
- ⦿ ***Digital signals*** are those for which both time and amplitude are discrete.

# Impulse Response

For DT systems, the answer is surprisingly simple: All we need to know is the *impulse response* (denoted by  $h[n]$ ) which is the response to a unit impulse input

$$\delta[n] \triangleq \begin{cases} 1 & \text{if } n = 0 \\ 0 & \text{if } n \neq 0. \end{cases}$$

As an aside, we also define here the *unit step function*

$$u[n] \triangleq \begin{cases} 1 & \text{if } n \geq 0 \\ 0 & \text{if } n < 0. \end{cases}$$

The reason one only needs the impulse response is that we can write any signal  $x[n]$  as a linear combination of the unit impulse function and its time-shifts:

$$x[n] = \sum_{k=-\infty}^{\infty} x[k]\delta[n - k]$$

where  $x[k]$  are coefficients and  $\delta[n - k]$  is a time shift of  $\delta[n]$ . Mathematically, this is equivalent to noting that the canonical unit vectors (i.e.,  $\{\delta[n - k]\}_{k \in \mathbb{Z}}$ ) form a basis for the space of complex sequences with bounded entries (i.e.,  $\ell_{\infty}$ ).

The surprising conclusion is that the output of an LTI system is given by the “convolution” sum

$$y[n] = x[n] * h[n] \triangleq \sum_{k=-\infty}^{\infty} x[k]h[n - k]$$

Observation: If we know the unit impulse response  $h[n]$  of a LTI system, we can compute the output  $y[n]$  of an arbitrary input  $x[n]$  as  $y[n] = x[n] * h[n]$ . In this sense, a LTI system is fully determined by its unit impulse response.

Visualizing the calculation of convolution sum:

Step 1: Choose a value of  $n$  and consider it fixed.

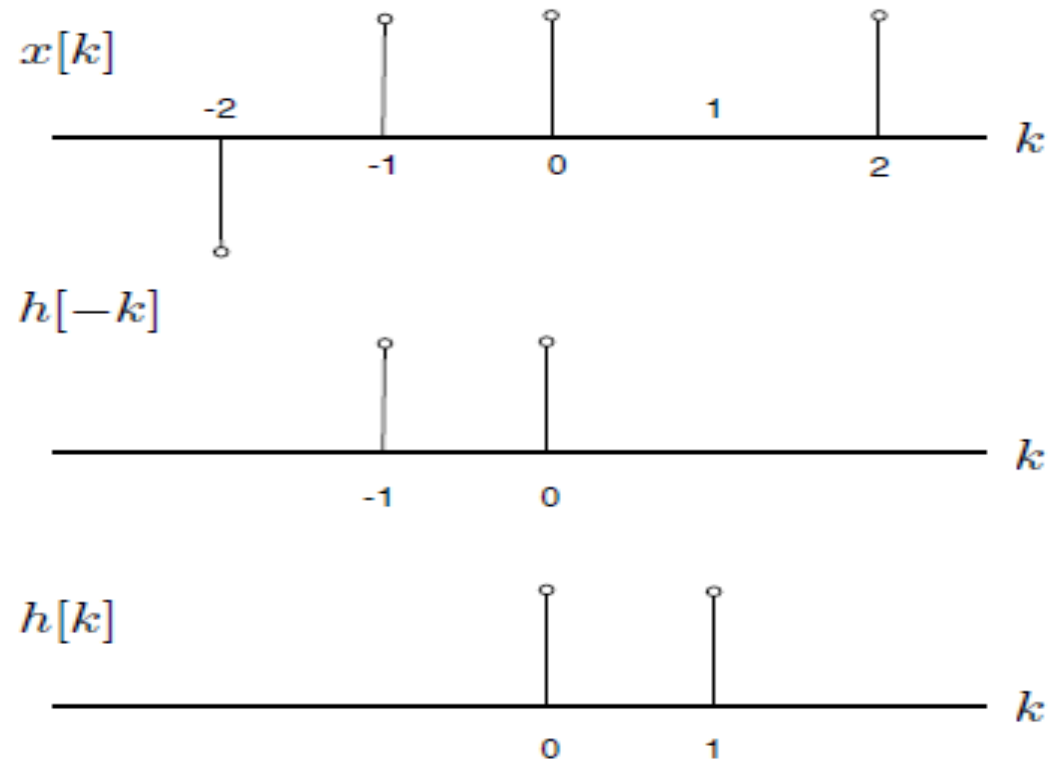
Step 2: Plot  $x[k]$  as a function of  $k$ .

Step 3: Plot the function  $h[n - k]$  (as a function of  $k$ ) by first flipping  $h[k]$  and then shift to the right by  $n$  (if  $n$  is negative, this means a shift to the left by  $|n|$ ).

Step 4: Compute the intermediate signal  $w_n[k] \triangleq x[k]h[n - k]$  via pointwise multiplication and then sum this signal to obtain the result  $y[n]$ .



To compute  $y[n + 1]$ , one can compute  $h[n + 1 - k]$  simply by shifting  $h[n - k]$  to the right by sample. Then, answer is computed by repeating Step 4.



# LCCDE

A *linear constant-coefficient difference equation* (LCCDE),

$$\sum_{k=0}^N a_k y[n-k] = \sum_{m=0}^M b_m x[n-m],$$

defines the relationship between an input sequence  $x[n]$  and an output sequence  $y[n]$ . If an LTI system is causal, then it can be described by a LCCDE.

## Example

Consider the DT system described by the LCCDE

$$y[n] - \frac{1}{2}y[n-1] = x[n].$$

We assume the system is initially at rest (i.e., causal), which is defined mathematically as

$$x[k] = 0 \text{ for all integer } k \leq k_0 \quad \implies \quad y[k] = 0 \text{ for all integer } k \leq k_0.$$

It turns out that a system is LTI if it is described by a LCCDE and it is initially at rest. In this case, we can first figure out the unit impulse response of the system  $h[n]$  and then use the convolution sum to calculate the response to  $u[n]$ . It is not too hard to verify that  $y[n] = \left(\frac{1}{2}\right)^n u[n]$  satisfies the above LCCDE for input  $x[n] = \delta[n]$ . Therefore, we say that its impulse response is

$$h[n] = \left(\frac{1}{2}\right)^n u[n].$$

Now, we can calculate the output associated with the input  $x[n] = u[n]$ , which is known as the *step response* of the system. By the convolution sum, the output  $y[n]$  corresponding to the input  $x[n] = u[n]$  is given by

$$\begin{aligned}
 y[n] &= u[n] * h[n] \\
 &= \left( \sum_{k=0}^{\infty} \delta[n-k] \right) * h[n] \\
 &= \sum_{k=0}^{\infty} h[n-k].
 \end{aligned}$$

Thus, we find that

$$\begin{aligned}
 y[0] &= 1 \\
 y[1] &= \frac{1}{2} + 1 = \frac{3}{2} \\
 y[2] &= \left(\frac{1}{2}\right)^2 + \frac{1}{2} + 1 = \frac{7}{4} \\
 &\vdots
 \end{aligned}$$

$$y[n] = \left(\frac{1}{2}\right)^n + \left(\frac{1}{2}\right)^{n-1} + \dots + \frac{1}{2} + 1 = \frac{1 - \frac{1}{2}\left(\frac{1}{2}\right)^n}{1 - \frac{1}{2}} = 2 - \left(\frac{1}{2}\right)^n$$

for  $n \geq 0$  and  $h[n] = 0$  for  $n \leq -1$ .

# FIR and IIR Systems

- A causal  $N$ -th order finite impulse response (FIR) system can have its transfer function written as  $H(z) = \sum_{n=0}^N h[n]z^{-n}$
- A causal LTI system that is not FIR is said to be IIR (infinite impulse response).

e.g. exponential signal  $h[n] = a^n u[n]$ :  
 its corresponding  $H(z) = \frac{1}{1-az^{-1}}$ .

## Based on Impulse Response Length -

- If the impulse response  $h[n]$  is of finite length, i.e.,  $h[n] = 0$  for  $n < N_1$  and  $n > N_2$ ,  $N_1 < N_2$  then it is known as a **finite impulse response (FIR)** discrete-time system
- The convolution sum description here is

$$y[n] = \sum_{k=N_1}^{N_2} h[k]x[n-k]$$

- The output  $y[n]$  of an FIR LTI discrete-time system can be computed directly from the convolution sum as it is a finite sum of products (e.g., moving-average filter & interpolator)

## Based on the Output Calculation Process -

- **Nonrecursive System** - Here the output can be calculated sequentially, knowing only the present and past input samples
- **Recursive System** - Here the output computation involves past output samples in addition to the present and past input samples

## Based on the Coefficients -

- **Real Discrete-Time System** - The impulse response samples are real valued
- **Complex Discrete-Time System** - The impulse response samples are complex valued

# Problem:

Given a sequence,

$$h(k) = \begin{cases} 3, & k = 0,1 \\ 1, & k = 2,3 \\ 0 & \text{elsewhere} \end{cases}$$

where  $k$  is the time index or sample number,

- Sketch the sequence  $h(k)$  and reversed sequence  $h(-k)$ .
- Sketch the shifted sequences  $h(k+3)$  and  $h(-k-2)$ .

$$k > 0, h(-k) = 0$$

$$k = 0, h(-0) = h(0) = 3$$

$$k = -1, h(-k) = h(-(-1)) = h(1) = 3$$

$$k = -2, h(-k) = h(-(-2)) = h(2) = 1$$

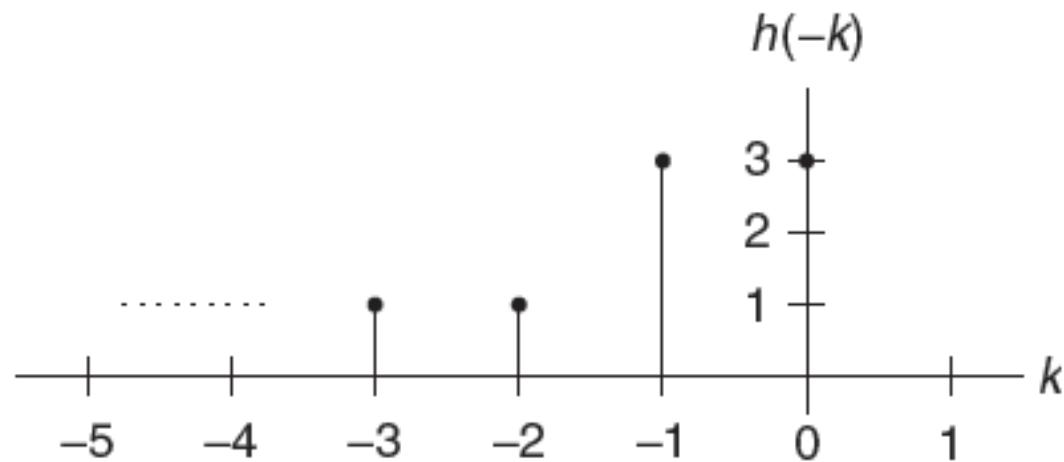
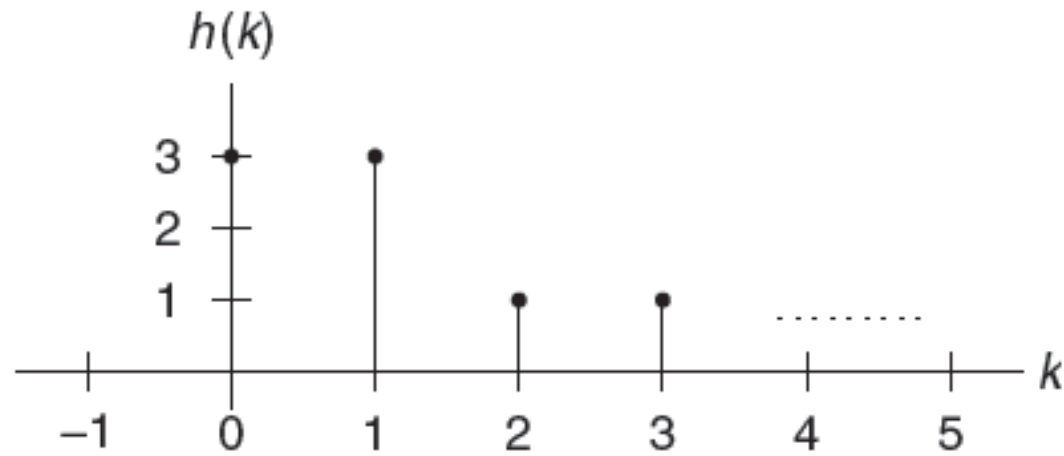
$$k = -3, h(-k) = h(-(-3)) = h(3) = 1$$

One can verify that  $k \leq -4, h(-k) = 0$ . Then the reversed sequence  $h(-k)$  is shown as the second plot in Figure

As shown in the sketches,  $h(-k)$  is just a mirror image of the original sequence  $h(k)$ .



a. Since  $h(k)$  is defined, we plot it in Figure . . . . . Next, we need to find the reversed sequence  $h(-k)$ . We examine the following for



b. Based on the definition of the original sequence, we know that  $h(0) = h(1) = 3$ ,  $h(2) = h(3) = 1$ , and the others are zeros. The time indices correspond to the following:

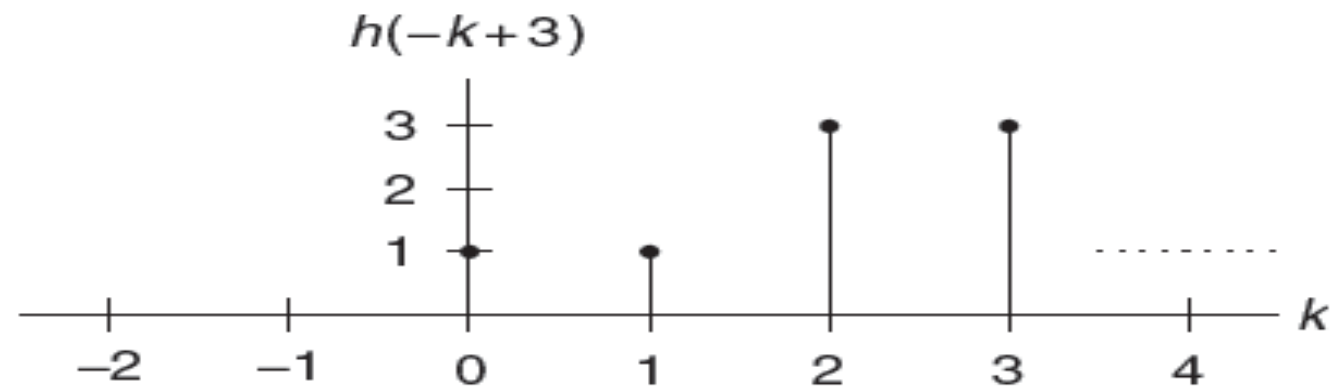
$$-k + 3 = 0, k = 3$$

$$-k + 3 = 1, k = 2$$

$$-k + 3 = 2, k = 1$$

$$-k + 3 = 3, k = 0.$$

Thus we can sketch  $h(-k + 3)$ , as shown in Figure



**Plot of the sequence  $h(-k+3)$  in Example**

Similarly,  $h(-k-2)$  is yielded in Figure



# The convolution sum and Methods of evaluating the convolution sum



- ⦿ convolution is an operation (integration or summation, for continuous and discrete time, respectively) that relates the output of a linear and time-invariant (LTI) system to its input and its impulse response.
- ⦿ Convolution is one of the primary concepts of linear system theory. It gives the answer to the problem of finding the system zero-state response due to any input—the most important problem for linear systems.
- ⦿ The main convolution theorem states that the response of a system at rest (zero initial conditions) due to any input is the convolution of that input and the system impulse response.

# Representation of Discrete-Time Signals



- We assume Discrete-Time LTI systems

$$x[n] = \sum_{k=-\infty}^{\infty} x[k] \delta[n-k]$$

- The signal  $X[n]$  can be represented using **unit sample function** or **unit impulse function**:  $\delta[n]$

- Remember: 
$$x[n] \delta[n-k] = \begin{cases} x[k], n=k \\ 0, else \end{cases}$$

- Notations:

$$x_0[n] = x[n] \delta[n-0] = x[0] \delta[n-0] = x[0], n=0$$

$$x_1[n] = x[n] \delta[n-1] = x[1] \delta[n-1] = x[1], n=1$$

notes

# Convolution for Discrete-Time Systems

- LTI system response can be described using:



- For **time-invariant**:  $\delta[n-k] \rightarrow h[n-k]$

Impulse Response of a System

- For a **linear system**:  $x[k]\delta[n-k] \rightarrow x[k]h[n-k]$

$$x[n] = \sum_{k=-\infty}^{\infty} x[k]\delta[n-k]$$

- Remember:

- Thus, for LTI:  $x[n] = \sum_{k=-\infty}^{\infty} x[k]\delta[n-k] \rightarrow y[n] = \sum_{k=-\infty}^{\infty} x[k]h[n-k] = x[n] * h[n]$

- We call this the **convolution sum**

$$y[n] = \sum_{k=-\infty}^{\infty} x[k]h[n-k] = x[n] * h[n]$$

$$y[n] = \sum_{k=-\infty}^{\infty} h[k]x[n-k] = h[n] * x[n]$$

- Remember:  $h[n] * \delta[n-n_0] = h[n-n_0] * \delta[n] = h[n-n_0]$

# Convolution for Discrete-Important Properties



- By definition

$$y[n] = h[n] * \delta[n] = h[n]$$

- Remember (due to time-invariance property):

$$h[n] * \delta[n - n_0] = h[n - n_0] * \delta[n] = h[n - n_0]$$

- Multiplication

$$\delta[n]g[n - n_0] = \delta[n]g[-n_0]$$

# The convolution sum and Methods of evaluating the convolution sum



## Convolution Sum.

- ✓  $x[n]$  is a signal as a weighted sum of basis function; time-shift version of the unit impulse signal.  $x[k]$  represents a specific value of the signal  $x[n]$  at time  $k$ .

$$x[n] = \sum_{k=-\infty}^{\infty} x[k] \delta[n-k]$$

- ✓ The output of the LTI system  $y[n]$  is given by a weighted sum of time-shifted impulse response.  $h[n]$  is the impulse response of LTI system H.

$$y[n] = \sum_{k=-\infty}^{\infty} x[k] h[n-k]$$

- ✓ The **convolution** of two discrete-time signals  $y[n]$  and  $h[n]$  is denoted as

$$x[n] * h[n] = \sum_{k=-\infty}^{\infty} x[k] h[n-k]$$



## Steps for Convolution Computation.

**Step 1:** Plot  $x$  and  $h$  versus  $k$  since the convolution sum is on  $k$ .

**Step 2:** Flip  $h[k]$  around the vertical axis to obtain  $h[-k]$ .

**Step 3:** Shift  $h[-k]$  by  $n$  to obtain  $h[n-k]$ .

**Step 4:** Multiply to obtain  $x[k] h[n-k]$ .

**Step 5:** Sum on  $k$  to compute

**Step 6:** Index  $n$  and repeat Step 3-6.

$$y[n] = x[n] * h[n] = \sum_{k=-\infty}^{\infty} x[k] h[n-k]$$

## Representation of an arbitrary sequence

Any signal  $x(n)$  can be represented as weighted sum of impulses as given below

$$x(n) = \sum_{k=-\infty}^{\infty} x(k)\delta(n-k)$$

The response of the system for unit sample input is called impulse response of the system  $h(n)$

$$\begin{aligned} y(n) &= \mathcal{T}[x(n)] = \mathcal{T}\left[\sum_{k=-\infty}^{\infty} x(k)\delta(n-k)\right] \\ &= \sum_{k=-\infty}^{\infty} x(k)\mathcal{T}[\delta(n-k)] \\ &= \sum_{k=-\infty}^{\infty} x(k)h(n,k) \end{aligned}$$

By time invariant property, we have

$$y(n) = \sum_{k=-\infty}^{\infty} x(k)h(n-k)$$

The above equation is called **convolution sum**.

Some of the properties of convolution are commutative law, associative law and distributive law.

## Correlation of two sequences

It is basically used to compare two signals. It is the measure of similarity between two signals. Some of the applications are communication systems, radar, sonar etc.

The cross correlation of two sequences  $x(n)$  and  $y(n)$  is given by

$$r_{xy}(l) = \sum_{n=-\infty}^{\infty} x(n)y(n-l) \quad l = 0, \pm 1, \pm 2, \dots$$

One of the important properties of cross correlation is given by

$$r_{xy}(l) = r_{yx}(-l)$$

The auto correlation of the signal  $x(n)$  is given by

$$r_{xx}(l) = \sum_{n=-\infty}^{\infty} x(n)x(n-l)$$

## Linear time invariant systems characterized by constant coefficient difference equation

The response of the first order difference equation is given by

$$y(n) = a^{n+1}y(-1) + \sum_{k=0}^n a^k x(n-k) \quad n \geq 0$$

The first part contain initial condition  $y(-1)$  of the system, the second part contains input  $x(n)$  of the system.

The response of the system when it is in relaxed state at  $n=0$  or  $y(-1)=0$  is called **zero state response** of the system or **forced response**.

$$y_{zs}(n) = \sum_{k=0}^n a^k x(n-k) \quad n \geq 0$$

The output of the system at zero input condition  $x(n)=0$  is called **zero input response** of the system or **natural response**.

The impulse response of the system is given by zero state response of the system

$$\begin{aligned}
 y_{zs}(n) &= \sum_{k=0}^n a^k \delta(n - k) \\
 &= a^n \quad n \geq 0
 \end{aligned}$$

The total response of the system is equal to sum of natural response and forced responses.

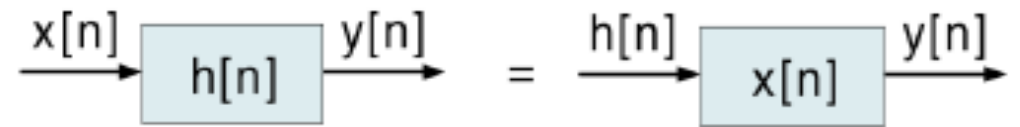
# Properties of Convolution

- ◎ **Commutative , Associative , Distributive**

# Properties of Convolution

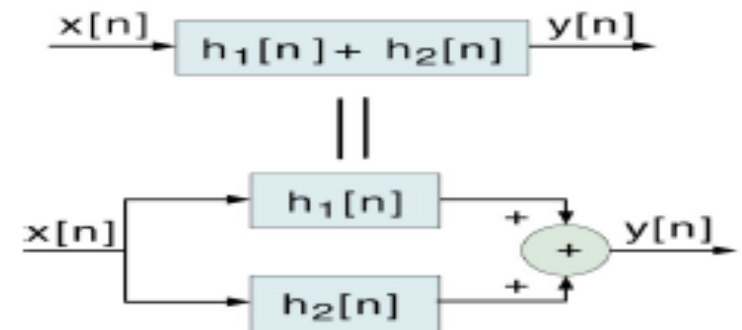
## Commutative Property

$$x[n] * h[n] = h[n] * x[n]$$



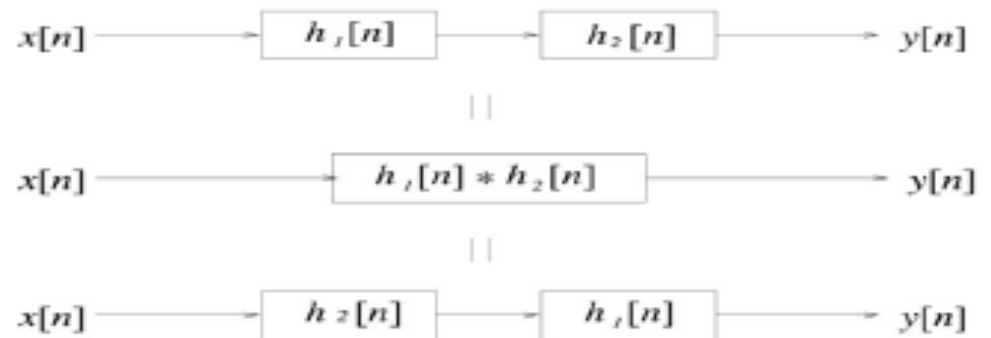
## Distributive Property

$$x[n] * (h_1[n] + h_2[n]) = (x[n] * h_1[n]) + (x[n] * h_2[n])$$



## Associative Property

$$x[n] * h_1[n] * h_2[n] = (x[n] * h_1[n]) * h_2[n] = (x[n] * h_2[n]) * h_1[n]$$



# The Commutative Property

$$f_1[k] * f_2[k] = f_2[k] * f_1[k]$$

This can be proved as follow:

$$\begin{aligned}
 f_1[k] * f_2[k] &= \sum_{m=-\infty}^{\infty} f_1[m]f_2[k - m] \\
 &= - \sum_{w=\infty}^{-\infty} f_1[w - k]f_2[w], \quad w = k - m \\
 &= \sum_{w=-\infty}^{\infty} f_2[w]f_1[w - k] \\
 &= f_2[k] * f_1[k]
 \end{aligned}$$



## The Distributive Property

$$f_1[k] * (f_2[k] + f_3[k]) = f_1[k] * f_2[k] + f_1[k] * f_3[k]$$

The proof is as follow:

$$\begin{aligned} f_1[k] * (f_2[k] + f_3[k]) &= \sum_{m=-\infty}^{\infty} f_1[m] (f_2[k - m] + f_3[k - m]) \\ &= \sum_{m=-\infty}^{\infty} f_1[m] f_2[k - m] + \sum_{m=-\infty}^{\infty} f_1[m] f_3[k - m] \\ &= f_1[k] * f_2[k] + f_1[k] * f_3[k] \end{aligned}$$

## The Associative Property

$$f_1[k] * (f_2[k] * f_3[k]) = (f_1[k] * f_2[k]) * f_3[k]$$

The proof is as follow:

$$\begin{aligned} f_1[k] * (f_2[k] * f_3[k]) &= \sum_{m_1=-\infty}^{\infty} f_1[m_1] (f_2[k - m_1] * f_3[k - m_1]) \\ &= \sum_{m_1=-\infty}^{\infty} f_1[m_1] \sum_{m_2=-\infty}^{\infty} f_2[m_2] f_3[k - m_1 - m_2] \\ &= \sum_{m_1=-\infty}^{\infty} \sum_{m_2=-\infty}^{\infty} f_1[\lambda - m_2] f_2[m_2] f_3[k - \lambda] \end{aligned}$$

,where  $\lambda = m_1 + m_2$ .

Then we have

$$\begin{aligned} f_1[k] * (f_2[k] * f_3[k]) &= \sum_{m_1=-\infty}^{\infty} \sum_{m_2=-\infty}^{\infty} f_1[\lambda - m_2] f_2[m_2] f_3[k - \lambda] \\ &= (f_1[k] * f_2[k]) * f_3[k] \end{aligned}$$

## The Convolution with an Impulse

$$f[k] * \delta[k] = \sum_{m=-\infty}^{\infty} f[m] \delta[k - m]$$

Since  $\delta[k - m] = 1$ , if  $k - m = 0$  or  $m = k$ , then

$$f[k] * \delta[k] = f[k].$$

## The shifting Property

If

$$f_1[k] * f_2[k] = c[k]$$

then

$$\begin{aligned} f_1[k] * f_2[k - n] &= f_1[k] * f_2[k] * \delta[k - n] \\ &= c[k] * \delta[k - n] = c[k - n] \end{aligned}$$

$$\begin{aligned} f_1[k - n] * f_2[k] &= f_1[k] * \delta[k - n] * f_2[k] \\ &= f_1[k] * f_2[k] * \delta[k - n] \\ &= c[k] * \delta[k - n] = c[k - n] \end{aligned}$$

$$\begin{aligned} f_1[k - n] * f_2[k - l] &= f_1[k] * \delta[k - n] * f_2[k] * \delta[k - l] \\ &= c[k] * \delta[k - n] * \delta[k - l] = c[k - n - l] \end{aligned}$$

## Convolution Sum: Analytical Method

Determine  $c[k] = f[k] * g[k]$  for

$$f[k] = (0.8)^k u[k] \quad \text{and} \quad g[k] = (0.3)^k u[k]$$

we have

$$c[k] = \sum_{m=0}^k f[m]g[k-m]$$

since both signals are causal.

$$c[k] = \begin{cases} \sum_{m=0}^k (0.8)^m (0.3)^{k-m} & k \geq 0 \\ 0 & k < 0 \end{cases}$$

$$\begin{aligned} c[k] &= (0.3)^k \sum_{m=0}^k \left(\frac{0.8}{0.3}\right)^m u[k] = (0.3)^k \frac{(0.8)^{k+1} - (0.3)^{k+1}}{(0.3)^k (0.8 - 0.3)} u[k] \\ &= 2 \left[ (0.8)^{k+1} - (0.3)^{k+1} \right] u[k] \end{aligned}$$

Find the zero-state response  $y[k]$  of an LTID system described by the equation

$$y[k + 2] - 0.6y[k + 1] - 0.16y[k] = 5f[k + 2]$$

if the input  $f[k] = 4^{-k}u[k]$  and  $h[k] = [(-0.2)^k + 4(0.8)^k] u[k]$ .

We have

$$\begin{aligned} y_i[k] &= f[k] * h[k] \\ &= (4)^{-k}u[k] * [(-0.2)^k u[k] + 4(0.8)^k u[k]] \\ &= (4)^{-k}u[k] * (-0.2)^k u[k] + (4)^{-k}u[k] * 4(0.8)^k u[k] \\ &= (0.25)^k u[k] * (-0.2)^k u[k] + 4(0.25)^k u[k] * (0.8)^k u[k] \end{aligned}$$

Using Pair 4 from the convolution sum table:

$$y[k] = \left[ \frac{(0.25)^{k+1} - (-0.2)^{k+1}}{0.25 - (-0.2)} + 4 \frac{(0.25)^{k+1} - (0.8)^{k+1}}{0.25 - 0.8} \right] u[k]$$

$$\begin{aligned}
 y[k] &= \left( 2.22 \left[ (0.25)^{k+1} - (-0.2)^{k+1} \right] - 7.27 \left[ (0.25)^{k+1} - (0.8)^{k+1} \right] \right) u[k] \\
 &= \left[ -5.05(0.25)^{k+1} - 2.22(-0.2)^{k+1} + 7.27(0.8)^{k+1} \right] u[k]
 \end{aligned}$$

Recognizing that

$$\gamma^{k+1} = \gamma(\gamma)^k$$

We can express  $y[k]$  as

$$\begin{aligned}
 y[k] &= \left[ -1.26(0.25)^k + 0.444(-0.2)^k + 5.81(0.8)^k \right] u[k] \\
 &= \left[ -1.26(4)^{-k} + 0.444(-0.2)^k + 5.81(0.8)^k \right] u[k]
 \end{aligned}$$

# Graphical Procedure for Convolution Sum

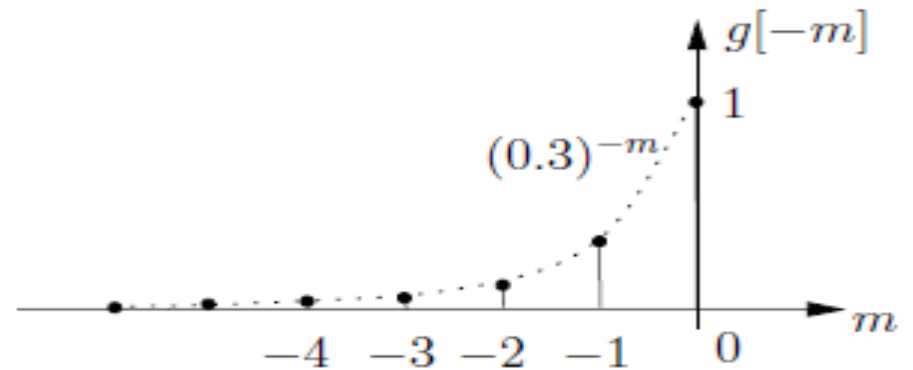
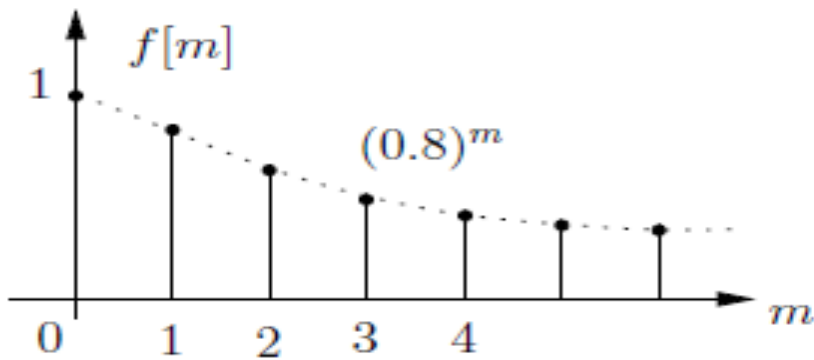
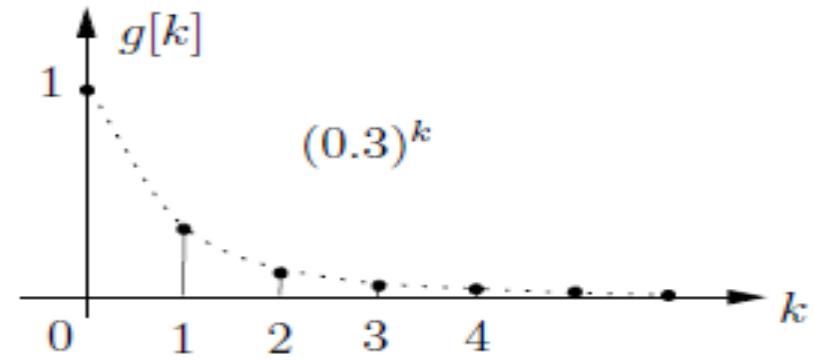
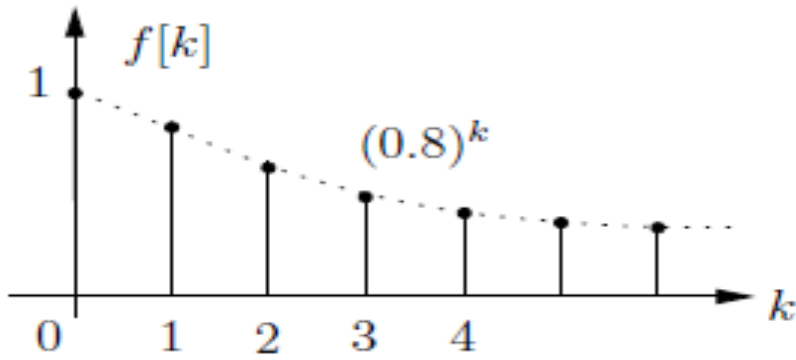
The convolution sum of causal signals  $f[k]$  and  $g[k]$  is given by

$$c[k] = \sum_{m=0}^k f[k]g[k - m]$$

- Invert  $g[m]$  about the vertical axis ( $m = 0$ ) to obtain  $g[-m]$ .
- Time shift  $g[-m]$  by  $k$  units to obtain  $g[k - m]$ . For  $k > 0$ , the shift is to the right (delay); for  $k < 0$ , the shift is to the left (advance).
- Next we multiply  $f[m]$  and  $g[k - m]$  and add all the products to obtain  $c[k]$ . The procedure is repeated to each value of  $k$  over the range  $-\infty$  to  $\infty$ .

# Problem:

Find  $c[k] = f[k] * g[k]$ , where  $f[k]$  and  $g[k]$  are depicted in the Figures.



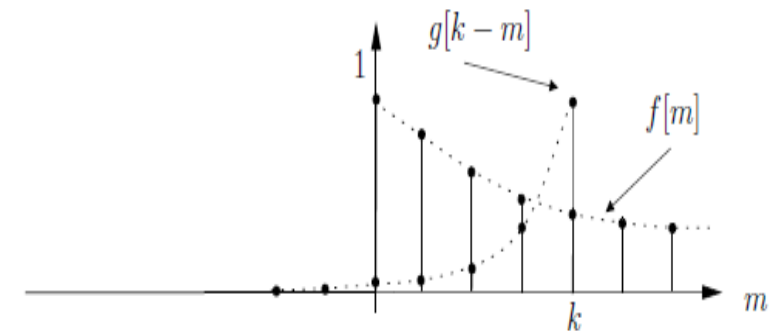
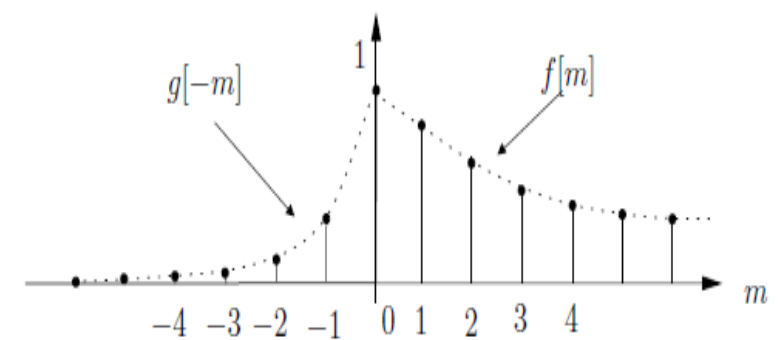


Therefore

$$\begin{aligned}c[k] &= \sum_{m=0}^k f[m]g[k-m] \\&= \sum_{m=0}^k (0.8)^m (0.3)^{k-m} \\&= (0.3)^k \sum_{m=0}^k \left(\frac{0.8}{0.3}\right)^m \\&= 2 \left[ (0.8)^{k+1} - (0.3)^{k+1} \right], \quad k \geq 0\end{aligned}$$

For  $k < 0$ , there is no overlap between  $f[m]$  and  $g[k-m]$ , so that  $c[k] = 0 \quad k < 0$  and

$$c[k] = 2 \left[ (0.8)^{k+1} - (0.3)^{k+1} \right] u[k].$$



The two functions  $f[m]$  and  $g[k-m]$  overlap over the interval  $0 \leq m \leq k$ .

Using the sliding tape method, convolve the two sequences  $f[k]$  and  $g[k]$ .

- write the sequences  $f[k]$  and  $g[k]$  in the slots of two tapes
- leave the  $f$  tape stationary (to correspond to  $f[m]$ ). The  $g[-m]$  tape is obtained by time inverting the  $g[m]$
- shift the inverted tape by  $k$  slots, multiply values on two tapes in adjacent slots, and add all the products to find  $c[k]$ .

For the case of  $k = 0$ ,

$$c[0] = 0 \times 1 = 0$$

For  $k = 1$

$$c[1] = (0 \times 1) + (1 \times 1) = 1$$

Similarly,

$$c[2] = (0 \times 1) + (1 \times 1) + (2 \times 1) = 3$$

$$c[3] = (0 \times 1) + (1 \times 1) + (2 \times 1) + (3 \times 1) = 6$$

$$c[4] = (0 \times 1) + (1 \times 1) + (2 \times 1) + (3 \times 1) + (4 \times 1) = 10$$

$$c[5] = (0 \times 1) + (1 \times 1) + (2 \times 1) + (3 \times 1) + (4 \times 1) + (5 \times 1) = 15$$

$$c[6] = (0 \times 1) + (1 \times 1) + (2 \times 1) + (3 \times 1) + (4 \times 1) + (5 \times 1) = 15$$

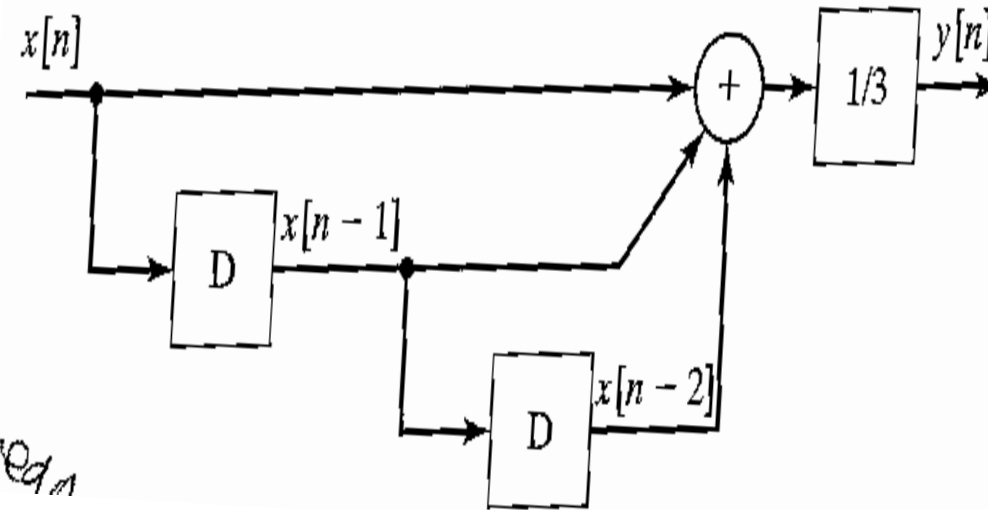
⋮



# Problems on Convolution Sum

1. Given the following block diagram

- Find the difference equation
- Find the impulse response:  $h[n]$ ; plot  $h[n]$
- Is this an FIR (finite impulse response) or IIR system?
- Given  $x[1]=3$ ,  $x[2]=4.5$ ,  $x[3]=6$ , Plot  $y[n]$  vs.  $n$
- Plot  $y[n]$  vs.  $n$  using Matlab



• Difference equation  $y[n] = \frac{1}{3}(x[n] + x[n-1] + x[n-2])$

• To find  $h[n]$  we assume  $x[n]=\delta[n]$ , thus  $y[n]=h[n]$

$$y[n] = h[n] = \frac{1}{3}(\delta[n] + \delta[n-1] + \delta[n-2])$$

• Thus:  $h[0]=h[1]=h[2]=1/3$

• Since  $h[n]$  is finite, the system is FIR

• In terms of inputs:  $y[n] = \sum_{k=-\infty}^{\infty} x[k]h[n-k] = x[n] * h[n]$

$$y[n] = \sum_{k=-\infty}^{\infty} h[k]x[n-k] = h[n] * x[n]$$

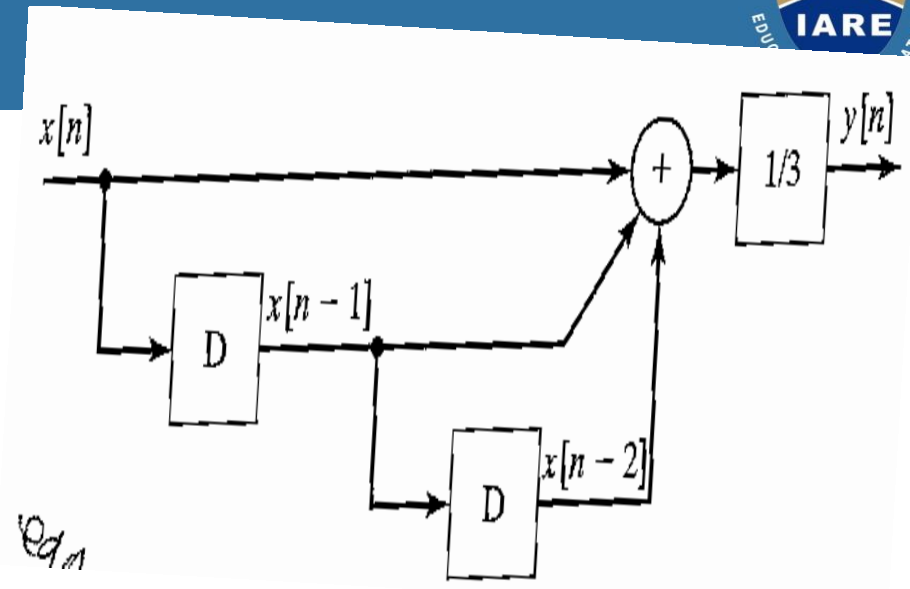
$$= \dots + x[n-3]h[3] + x[n-2]h[2] + x[n-1]h[1] + x[n-0]h[0] + x[n+1]h[-1] + \dots$$

$$= x[n-2]h[2] + x[n-1]h[1] + x[n-0]h[0]$$

**Figure: FIR system contains finite number of nonzero terms**

# Problem 1. - cont.

- Given the following block diagram
  - Find the difference equation
  - Find the impulse response:  $h[n]$ ; plot  $h[n]$
  - Is this an FIR (finite impulse response) or IIR system?
  - Given  $x[1]=3$ ,  $x[2]=4.5$ ,  $x[3]=6$ , Plot  $y[n]$  vs.  $n$
  - Plot  $y[n]$  vs.  $n$  using Matlab



- In terms of inputs:

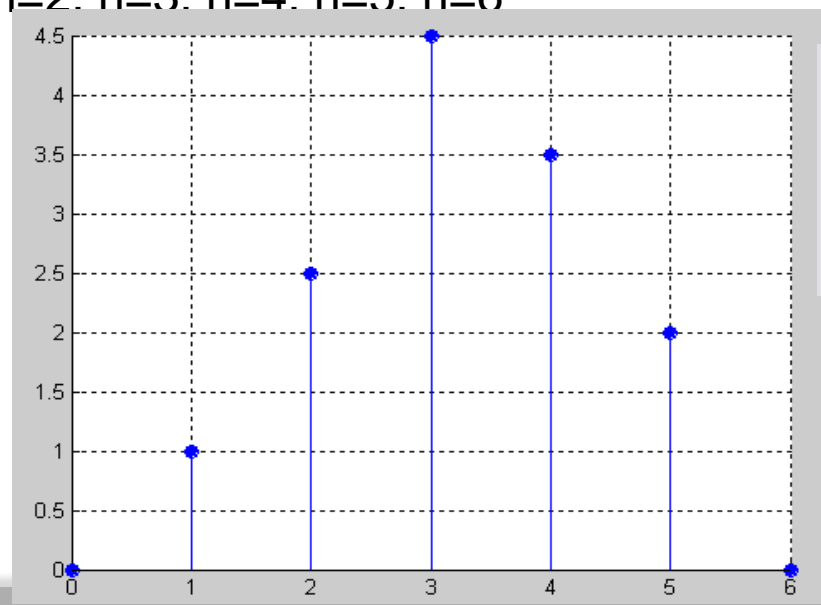
$$y[n] = \dots + x[n-3]h[3] + x[n-2]h[2] + x[n-1]h[1] + x[n-0]h[0] + x[n+1]h[-1] + \dots$$

$$= x[n-2]h[2] + x[n-1]h[1] + x[n-0]h[0]$$

Try for different values of n

- Calculate for  $n=0$ ,  $n=1$ ,  $n=2$ ,  $n=3$ ,  $n=4$ ,  $n=5$ ,  $n=6$

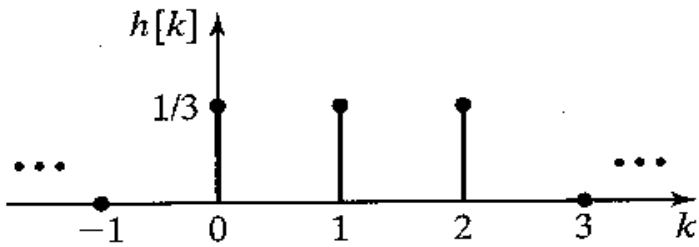
- $n=0$ ;  $y[0]=0$
- $n=1$ ;  $y[1]=1$
- $n=2$ ;  $y[2]=2.5$
- $n=3$ ;  $y[3]=4.5$
- $n=4$ ;  $y[4]=3.5$
- $n=5$ ;  $y[5]=2$
- $n=6$ ;  $y[6]=0$



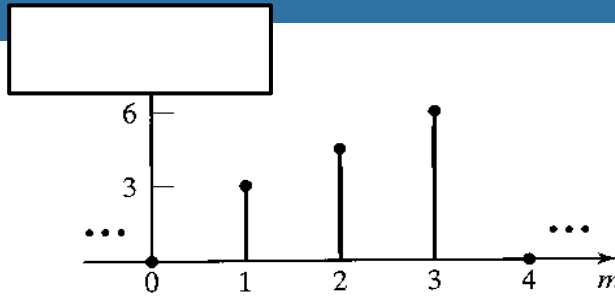
```

1 - n=0:6;
2 - x=[0 3 4.5 6];
3 - h=[1/3 1/3 1/3 0];
4 - y=conv(x,h)
5 - stem(n, y, 'fill'), grid
    
```

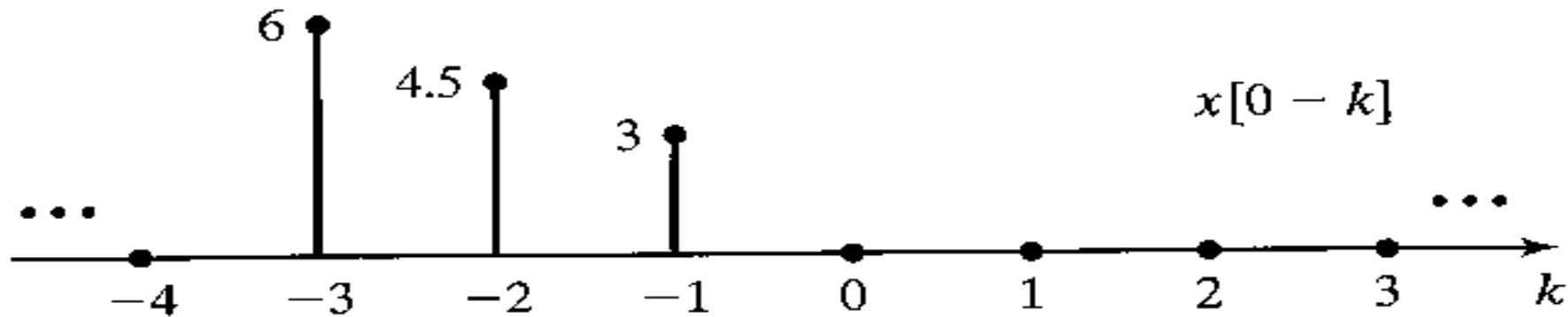
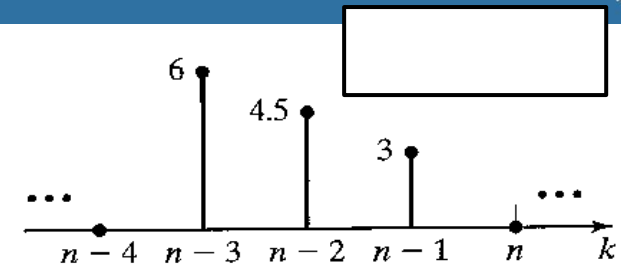
# Example – cont. (Graphical Representation)



$$h[0]=h[1]=h[2]=1/3$$



$$x[1]=3, x[2]=4.5, x[3]=6$$



$$y[n] = \dots + x[n-3]h[3] + x[n-2]h[2] + x[n-1]h[1] + x[n-0]h[0] + x[n+1]h[-1] + \dots$$

$$= x[n-2]h[2] + x[n-1]h[1] + x[n-0]h[0]$$

# Basic Structures for IIR Systems



- ⦿ **Direct Forms**
- ⦿ **Cascade Form**
- ⦿ **Parallel Form**
- ⦿ **Feedback in IIR Systems**



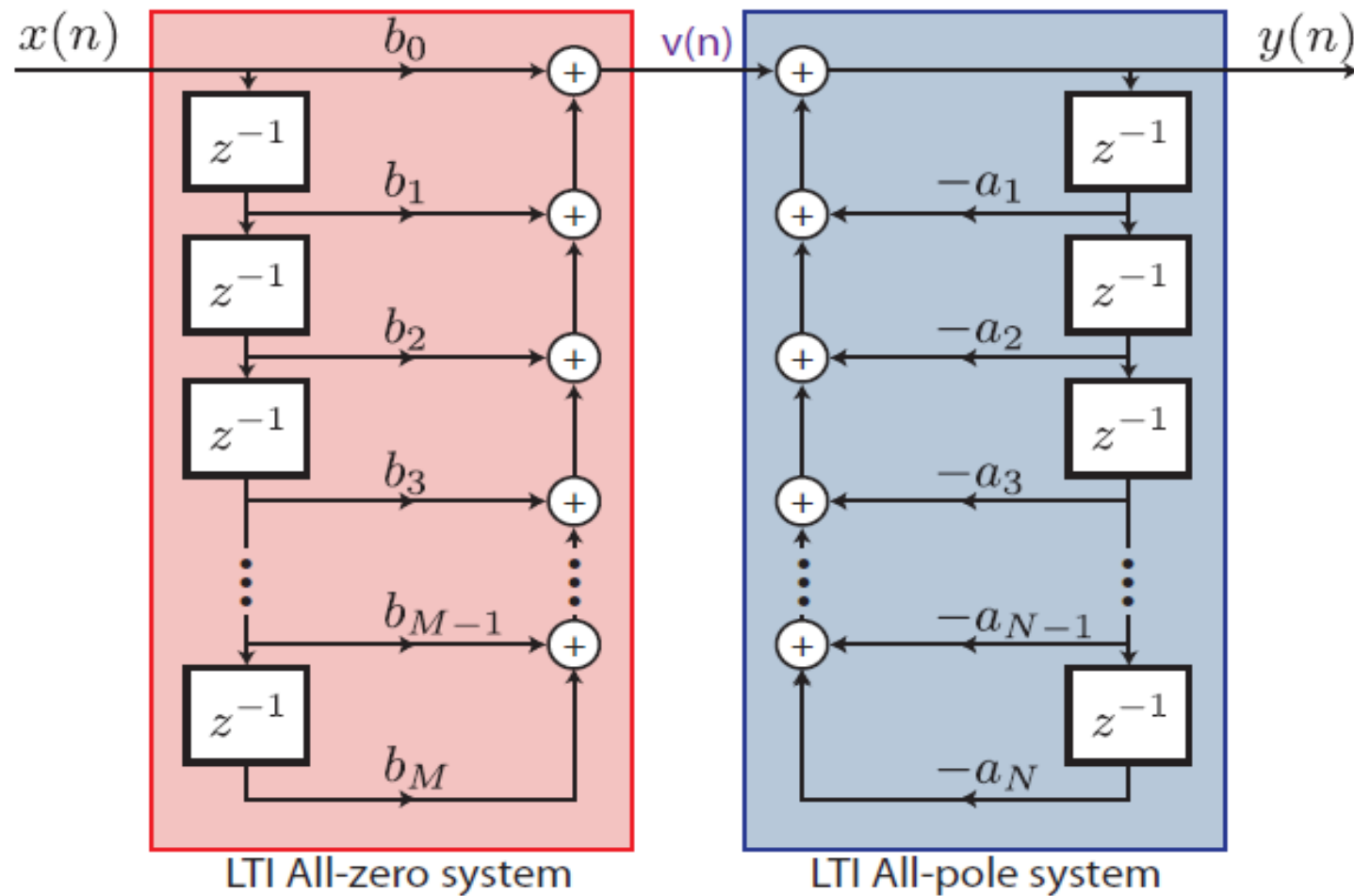
# Basic Structures for IIR Systems

## ◎ Direct Forms

$$y[n] - \sum_{k=1}^N a_k y[n-k] = \sum_{k=0}^M b_k x[n-k]$$

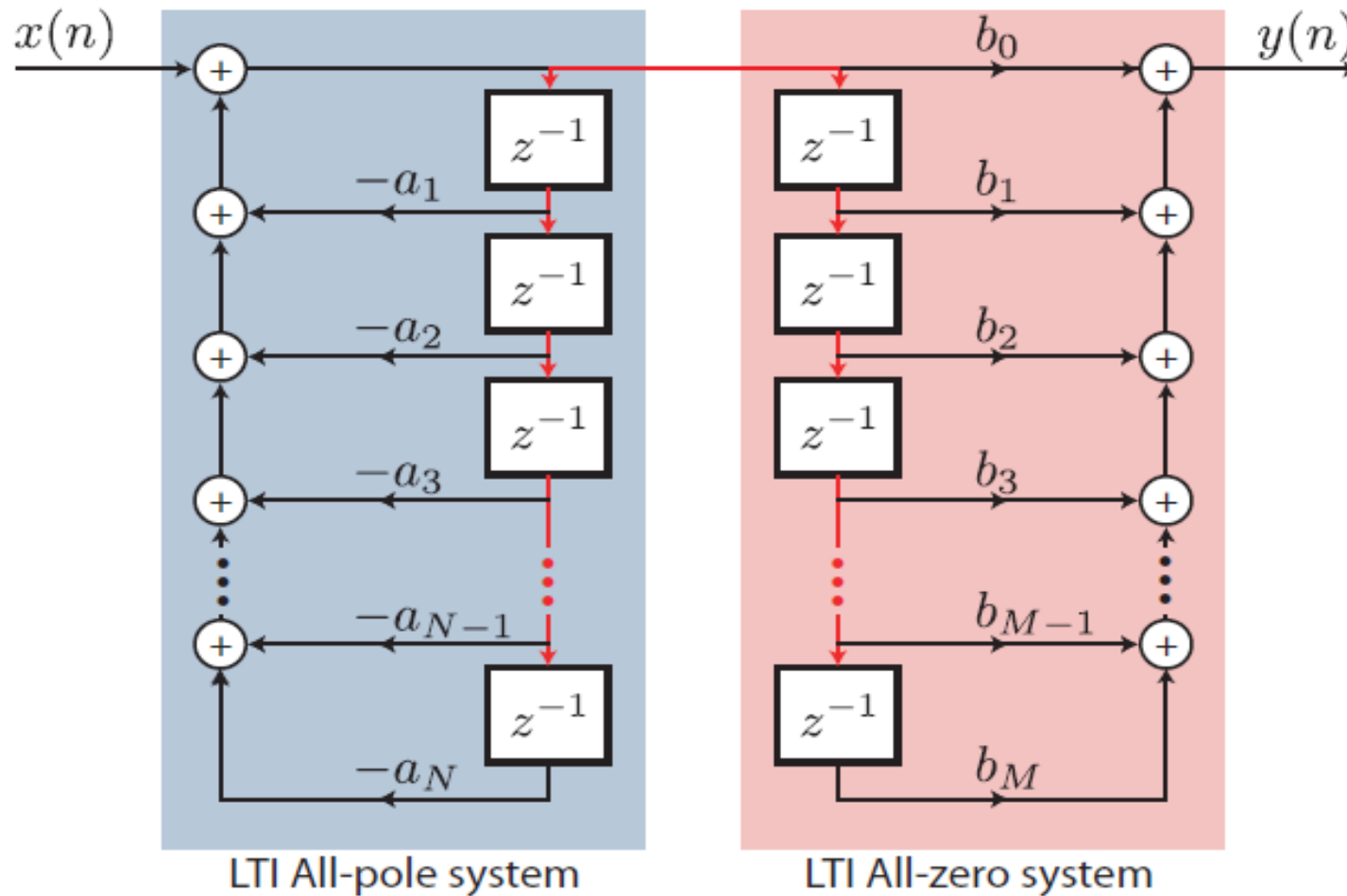
$$H(z) = \frac{\sum_{k=0}^M b_k z^{-k}}{1 - \sum_{k=1}^N a_k z^{-k}}$$

# Direct Form I (M = N)



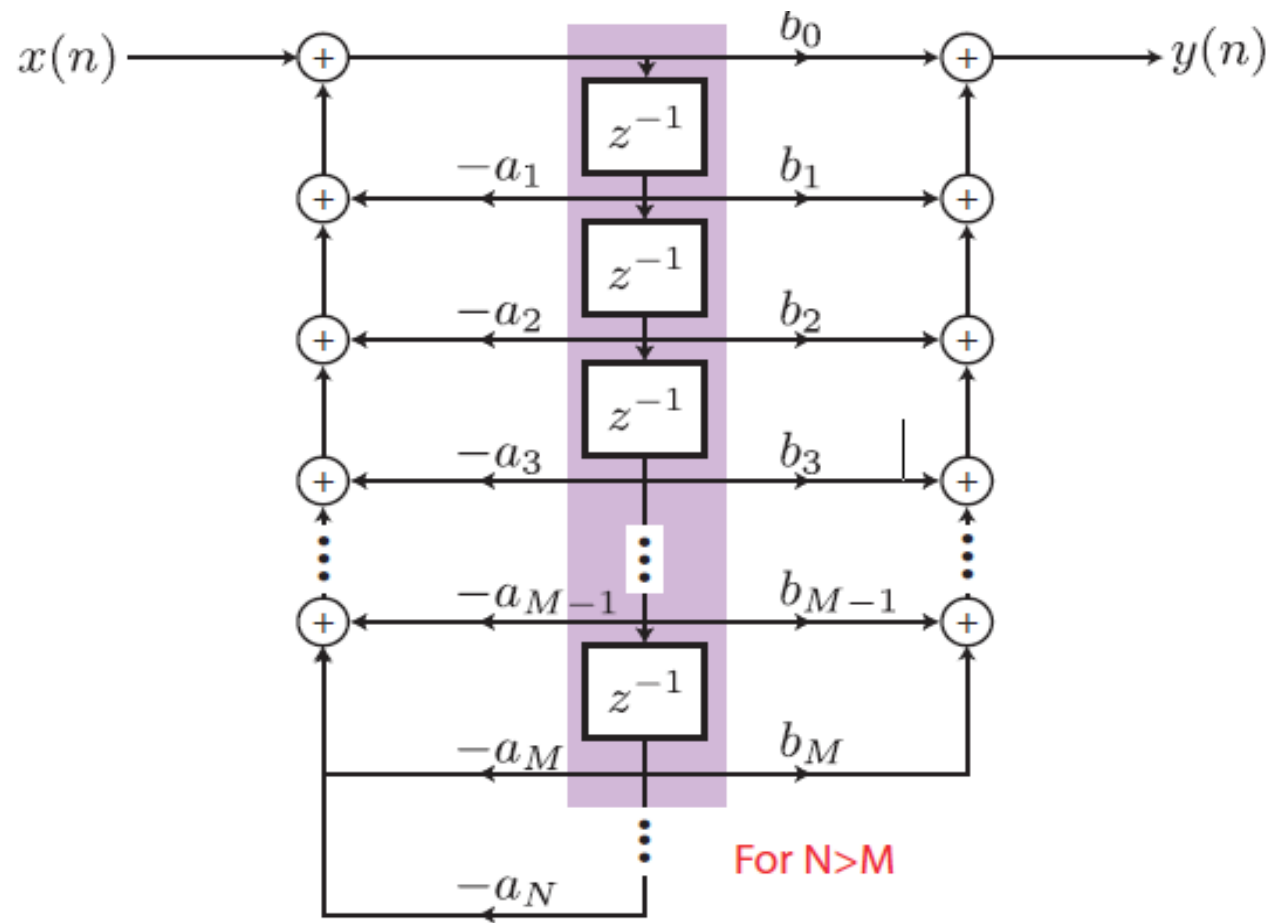
Requires:  $M + N + 1$  multiplications,  $M + N$  additions,  $M + N$  memory locations

# Direct Form II (M = N)



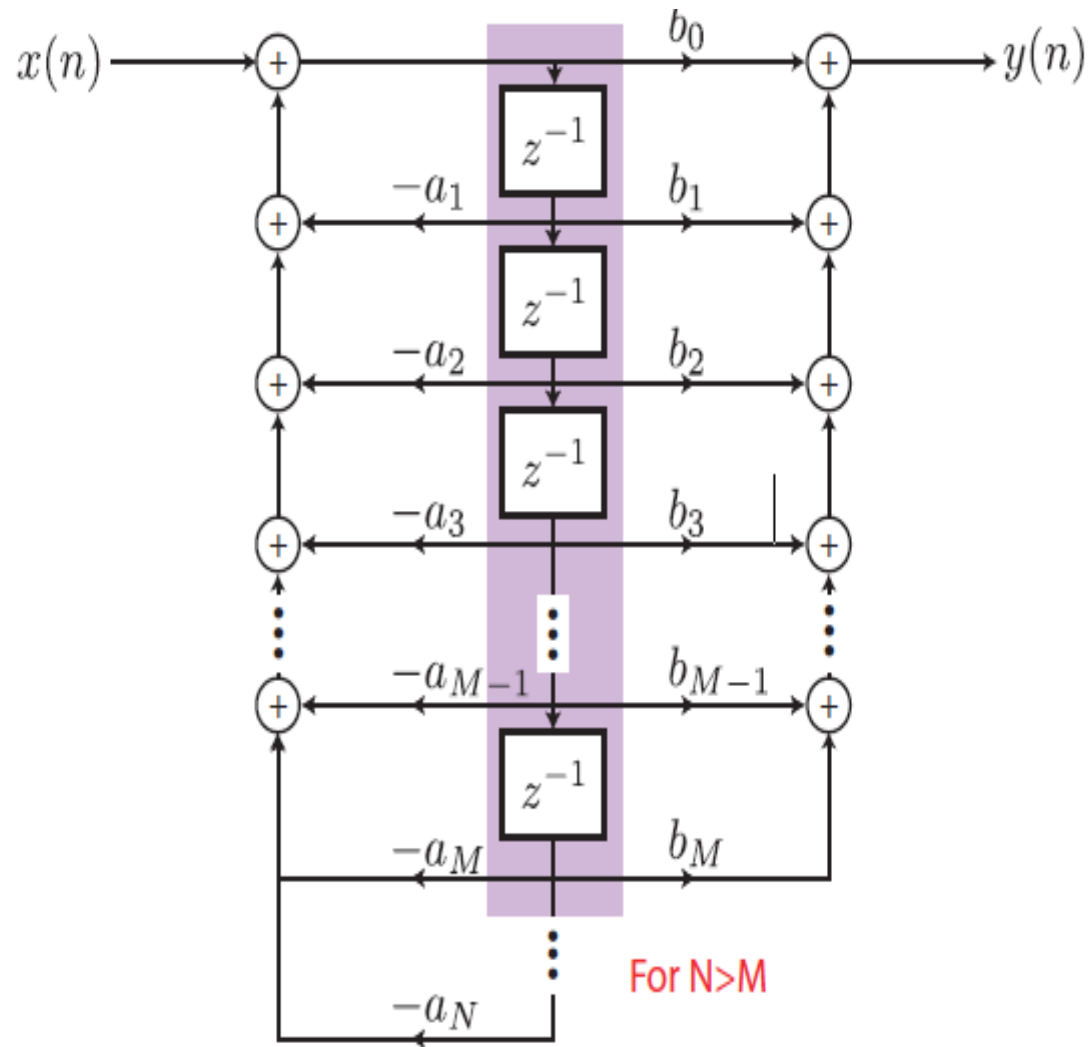
Requires:  $M + N + 1$  multiplications,  $M + N$  additions,  $M + N$  memory locations

# Direct Form II



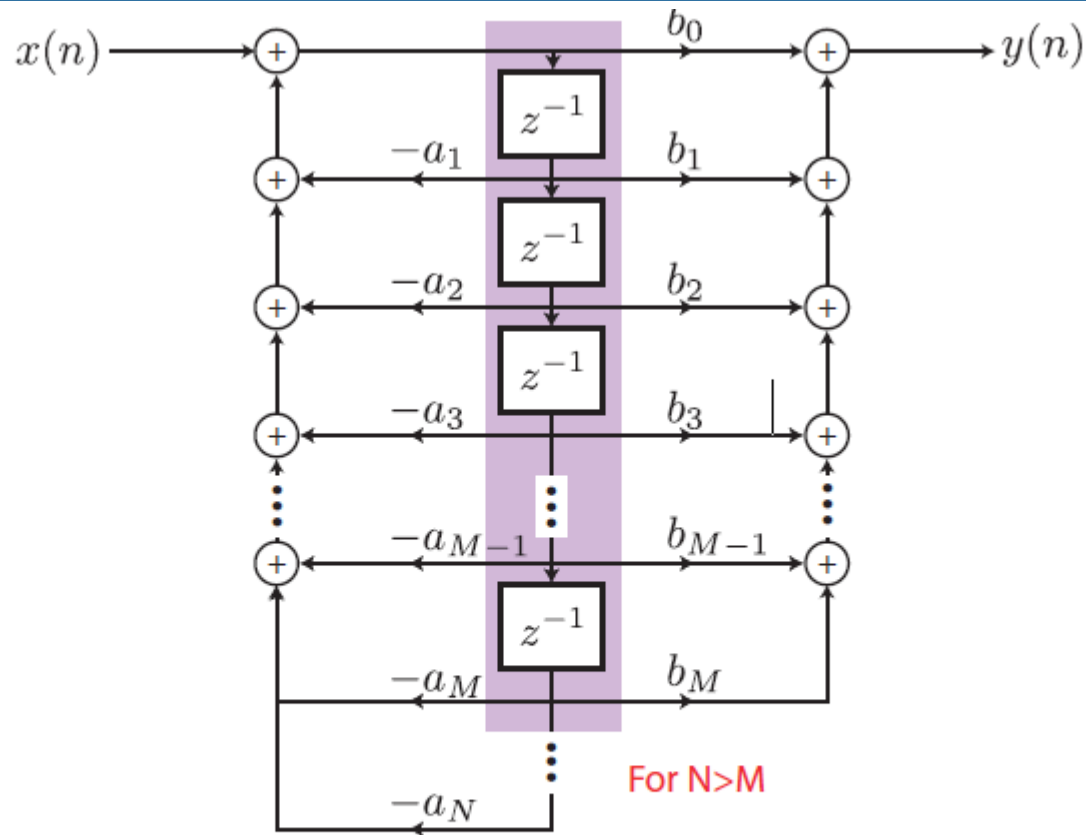
Requires:  $M + N + 1$  multiplications,  $M + N$  additions,  $\max(M, N)$  memory locations

# Direct Form II



Requires:  $M + N + 1$  multiplications,  $M + N$  additions,  $\max(M, N)$  memory locations

# Direct Form II



Requires:  $M + N + 1$  multiplications,  $M + N$  additions,  $\max(M, N)$  memory locations



## ◎ Cascade Form

$$H(z) = A \frac{\prod_{k=1}^{M_1} (1 - g_k z^{-1}) \prod_{k=1}^{M_2} (1 - h_k z^{-1})(1 - h_k^* z^{-1})}{\prod_{k=1}^{N_1} (1 - c_k z^{-1}) \prod_{k=1}^{N_2} (1 - d_k z^{-1})(1 - d_k^* z^{-1})}$$

where  $M = M_1 + 2M_2$  and  $N = N_1 + 2N_2$ .

- ◎ A modular structure that is advantageous for many types of implementations is obtained by combining pairs of real factors and complex conjugate pairs into second-order factors.

$$H(z) = \prod_{k=1}^{N_s} \frac{b_{0k} + b_{1k} z^{-1} + b_{2k} z^{-2}}{1 - a_{1k} z^{-1} - a_{2k} z^{-2}}$$

where  $N_s$  is the largest integer contained in  $(N+1)/2$ .

# Basic Structures for IIR Systems

## Parallel Form

$$H(z) = \sum_{k=0}^{N_P} C_k z^{-k} + \sum_{k=1}^{N_1} \frac{A_k}{1 - c_k z^{-1}} + \sum_{k=1}^{N_2} \frac{B_k (1 - e_k z^{-1})}{(1 - d_k z^{-1})(1 - d_k^* z^{-1})}$$

where  $N = N_1 + 2N_2$ . If  $M \leq N$ , then  $N_P = M - N$ ; otherwise, the first summation in right hand side of equation above is not included.

## Alternatively, the real poles of $H(z)$ can be grouped in pairs :

$$H(z) = \sum_{k=0}^{N_P} C_k z^{-k} + \sum_{k=1}^{N_S} \frac{e_{0k} + e_{1k} z^{-1}}{1 - a_{1k} z^{-1} - a_{2k} z^{-2}}$$

where  $N_S$  is the largest integer contained in  $(N+1)/2$ , and if  $N_P = M - N$  is negative, the first sum is not present.



# Basic Structures for FIR Systems

- **Direct Form**

- It is also referred to as a *tapped delay line* structure or a *transversal filter* structure.

- **Transposed Form**

- **Cascade Form**

$$H(z) = \sum_{n=0}^M h[n]z^{-n} = \prod_{k=1}^{M_S} (b_{0k} + b_{1k}z^{-1} + b_{2k}z^{-2})$$

where  $M_S$  is the largest integer contained in  $(M + 1)/2$ . If  $M$  is odd, one of coefficients  $b_{2k}$  will be zero.

# Direct Form

- For causal FIR system, the system function has only zeros (except for poles at  $z = 0$ ) with the difference equation:

$$y[n] = \sum_{k=0}^M b_k x[n-k]$$

- It can be interpreted as the discrete convolution of  $x[n]$  with the impulse response

$$h[n] = \begin{cases} b_n & , n = 0, 1, \dots, M, \\ 0 & , \text{otherwise.} \end{cases}$$



**UNIT- II**  
**DISCRETE FOURIER TRANSFORM AND EFFICIENT COMPUTATION**

CLO's	Course Learning Outcome
CLO4	Given the impulse response of a causal LTI system, show whether or not the system is bounded-input/bounded-output (BIBO) stable...
CLO5	Perform time, frequency and Z-transform analysis on signals.
CLO6	From a linear difference equation of a causal LTI system, draw the Direct Form I and Direct Form II filter realizations.

- ⊙ The DFT pair was given as

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

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

- ⊙ Baseline for computational complexity:

- Each DFT coefficient requires
  - N complex multiplications
  - N-1 complex additions
- All N DFT coefficients require
  - N<sup>2</sup> complex multiplications
  - N(N-1) complex additions

- ⊙ Complexity in terms of real operations

- 4N<sup>2</sup> real multiplications
- 2N(N-1) real additions

- ◎ Most fast methods are based on symmetry properties
  - Conjugate symmetry
  - Periodicity in n and k

$$e^{-j(2\pi/N)k(N-n)} = e^{-j(2\pi/N)kN} e^{-j(2\pi/N)k(-n)} = e^{j(2\pi/N)kn}$$

$$e^{-j(2\pi/N)kn} = e^{-j(2\pi/N)k(n+N)} = e^{j(2\pi/N)(k+N)n}$$

# The Goertzel Algorithm

- ⊙ Makes use of the periodicity  $e^{j(2\pi/N)Nk} = e^{j2\pi k} = 1$

- ⊙ Multiply DFT equation with this factor

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

- ⊙ Define 
$$y_k[n] = \sum_{r=-\infty}^{\infty} x[r] e^{j(2\pi/N)k(n-r)} u[n-r]$$

- ⊙ With this definition and using  $x[n]=0$  for  $n<0$  and  $n>N-1$

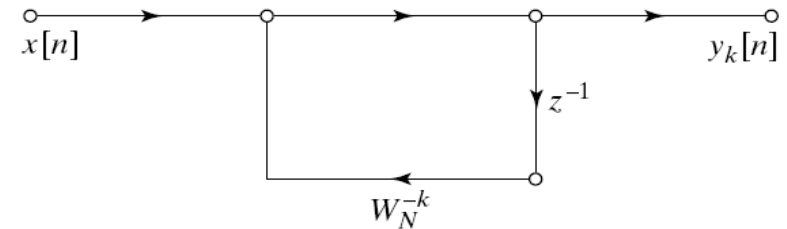
$$X[k] = y_k[n] \Big|_{n=N}$$

- ⊙  $X[k]$  can be viewed as the output of a filter to the input  $x[n]$ 
  - Impulse response of filter:  $e^{j(2\pi/N)kn} u[n]$
  - $X[k]$  is the output of the filter at time  $n=N$

# The Goertzel Filter

## Goertzel Filter

$$H_k(z) = \frac{1}{1 - e^{j\frac{2\pi}{N}k} z^{-1}}$$



## Computational complexity

- 4N real multiplications
- 2N real additions
- Slightly less efficient than the direct method

## Multiply both numerator and denominator

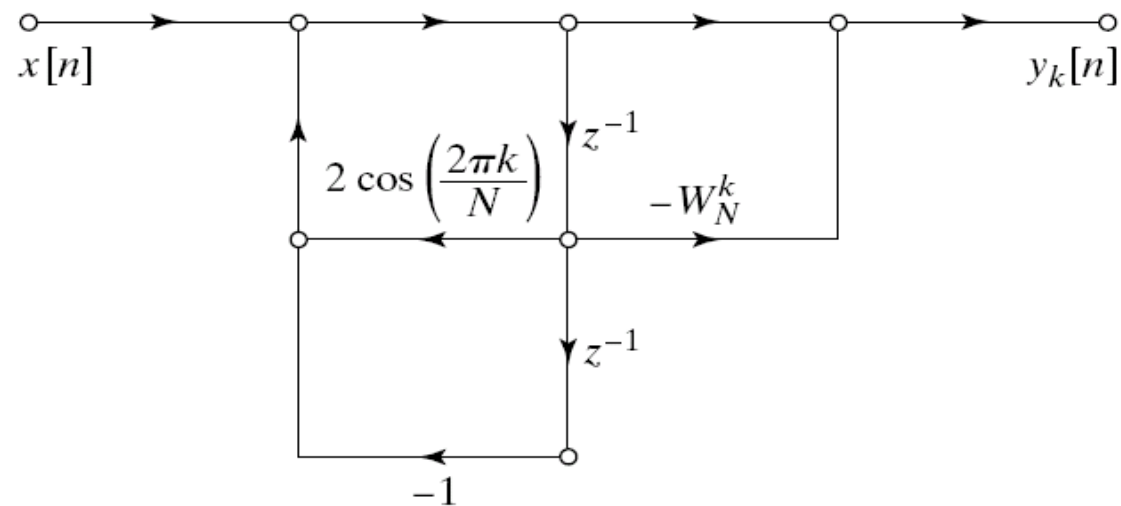
$$H_k(z) = \frac{1 - e^{-j\frac{2\pi}{N}k} z^{-1}}{\left(1 - e^{j\frac{2\pi}{N}k} z^{-1}\right)\left(1 - e^{-j\frac{2\pi}{N}k} z^{-1}\right)} = \frac{1 - e^{-j\frac{2\pi}{N}k} z^{-1}}{1 - 2\cos\frac{2\pi k}{N} z^{-1} + z^{-2}}$$



# Second Order Goertzel Filter

- Second order Goertzel Filter

$$H_k(z) = \frac{1 - e^{-j\frac{2\pi}{N}k} z^{-1}}{1 - 2\cos\left(\frac{2\pi k}{N}\right) z^{-1} + z^{-2}}$$



- Complexity for one DFT coefficient

- Poles: 2N real multiplications and 2N real additions
- Zeros: Need to be implemented only once
  - 4 real multiplications and 4 real additions

- Complexity for all DFT coefficients

- Each pole is used for two DFT coefficients
  - Approximately  $N^2$  real multiplications and  $2N^2$  real additions

- Do not need to evaluate all N DFT coefficients

- Goertzel Algorithm is more efficient than FFT if
  - less than M DFT coefficients are needed
  - $M < \log_2 N$

# Decimation-In-Time FFT Algorithms

- ⊙ Makes use of both symmetry and periodicity
- ⊙ Consider special case of N an integer power of 2
- ⊙ Separate  $x[n]$  into two sequence of length  $N/2$ 
  - Even indexed samples in the first sequence
  - Odd indexed samples in the other sequence

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

- ⊙ Substitute variables  $n=2r$  for  $n$  even and  $n=2r+1$  for odd

$$\begin{aligned} X[k] &= \sum_{r=0}^{N/2-1} x[2r]W_N^{2rk} + \sum_{r=0}^{N/2-1} x[2r+1]W_N^{(2r+1)k} \\ &= \sum_{r=0}^{N/2-1} x[2r]W_{N/2}^{rk} + W_N^k \sum_{r=0}^{N/2-1} x[2r+1]W_{N/2}^{rk} \\ &= G[k] + W_N^k H[k] \end{aligned}$$

- ⊙  $G[k]$  and  $H[k]$  are the  $N/2$ -point DFT's of each subsequence

# 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

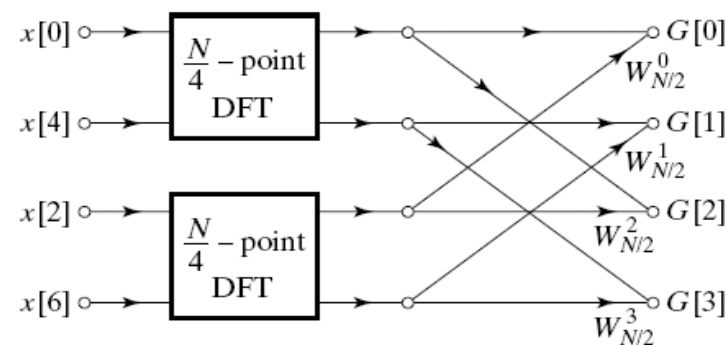
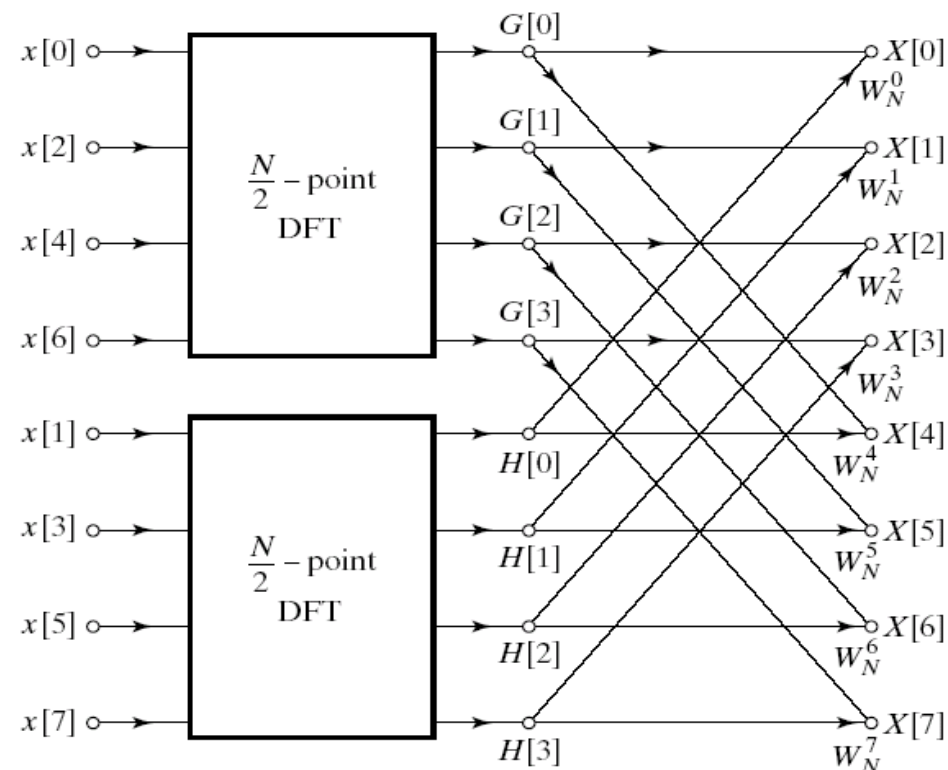
- $N$  complex multiplications
- $N$  complex additions

- Total complexity

- $N^2/2 + N$  complex multiplications
- $N^2/2 + N$  complex additions
- More efficient than direct DFT

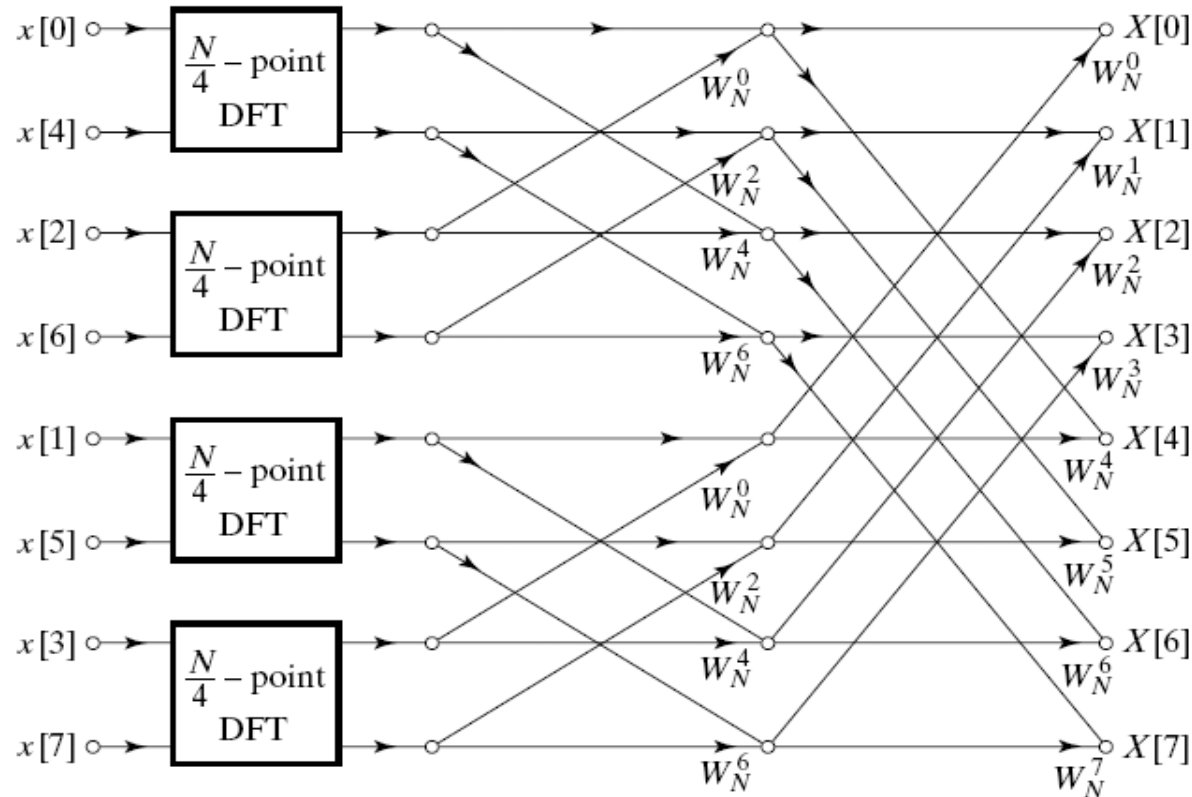
- Repeat same process

- Divide  $N/2$ -point DFTs into
- Two  $N/4$ -point DFTs
- Combine outputs

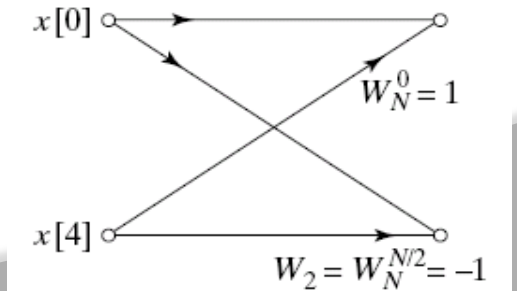


# Decimation In Time Cont'd

- After two steps of decimation in time

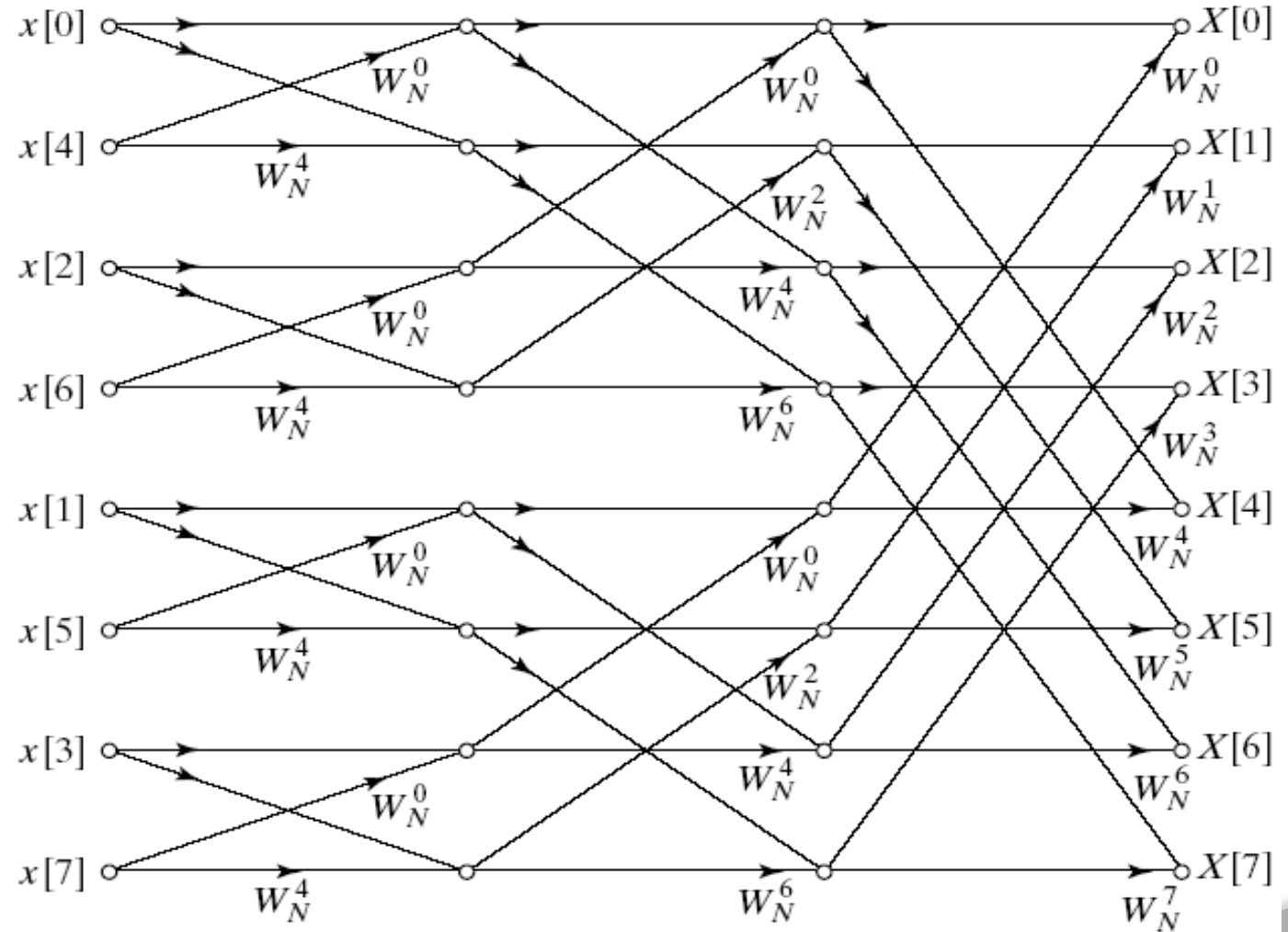


- Repeat until we're left with two-point DFT's



# Decimation-In-Time FFT Algorithm

- Final flow graph for 8-point decimation in time

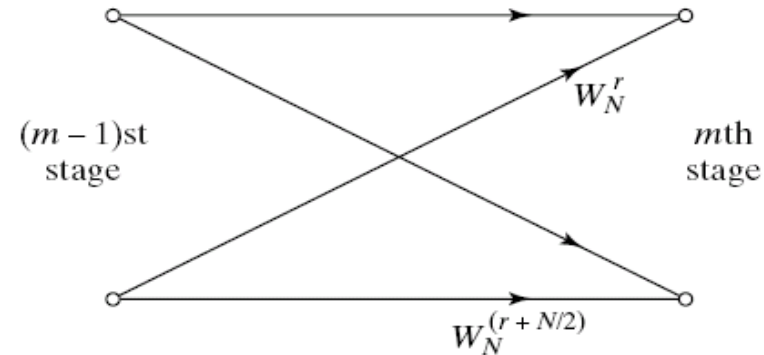


- Complexity:

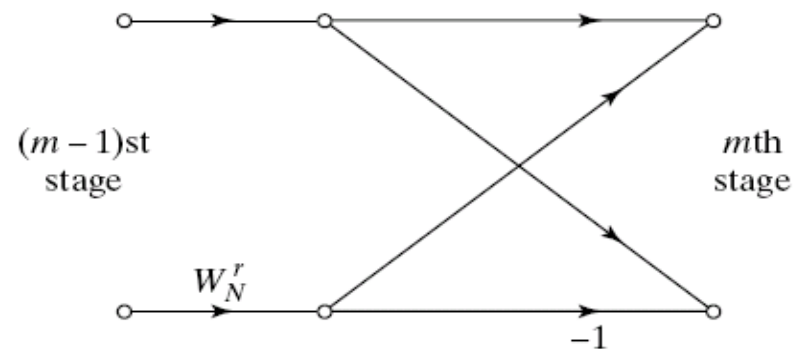
- $N \log_2 N$  complex multiplications and additions

# Butterfly Computation

- 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

# In-Place Computation

- ⊙ 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

$$X_0[0] = x[0] \leftrightarrow X_0[000] = x[000]$$

$$X_0[1] = x[4] \leftrightarrow X_0[001] = x[100]$$

$$X_0[2] = x[2] \leftrightarrow X_0[010] = x[010]$$

$$X_0[3] = x[6] \leftrightarrow X_0[011] = x[110]$$

$$X_0[4] = x[1] \leftrightarrow X_0[100] = x[001]$$

$$X_0[5] = x[5] \leftrightarrow X_0[101] = x[101]$$

$$X_0[6] = x[3] \leftrightarrow X_0[110] = x[011]$$

$$X_0[7] = x[7] \leftrightarrow X_0[111] = x[111]$$

# Decimation-In-Frequency FFT Algorithm

- ◎ 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

$$X[2r] = \sum_{n=0}^{N/2-1} x[n]W_N^{n2r} + \sum_{n=0}^{N/2-1} x[n+N/2]W_N^{(n+N/2)2r} = \sum_{n=0}^{N/2-1} (x[n] + x[n+N/2])W_{N/2}^{nr}$$

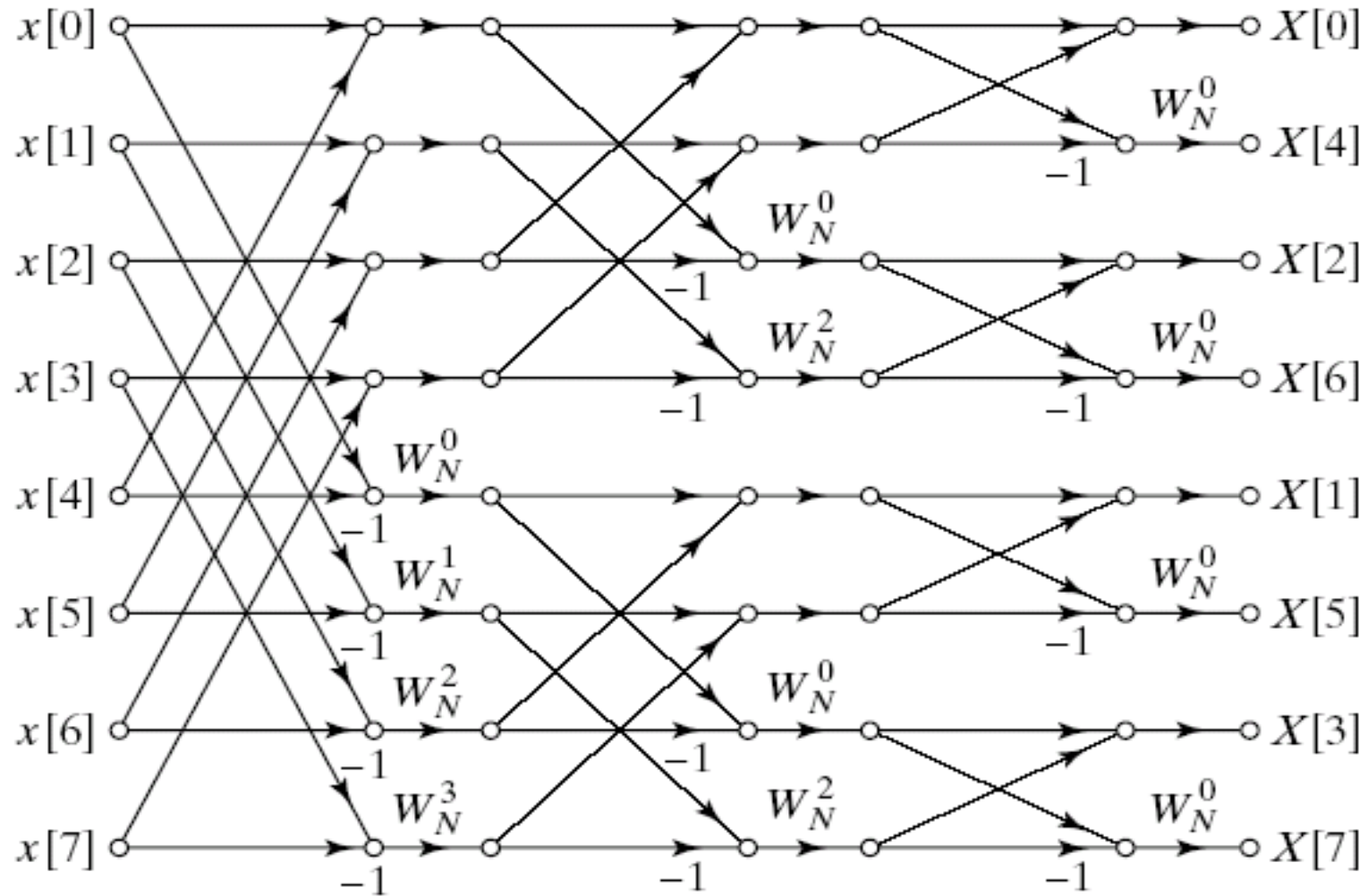
- ◎ Similarly for odd-numbered frequencies

$$X[2r+1] = \sum_{n=0}^{N/2-1} (x[n] - x[n+N/2])W_{N/2}^{n(2r+1)}$$



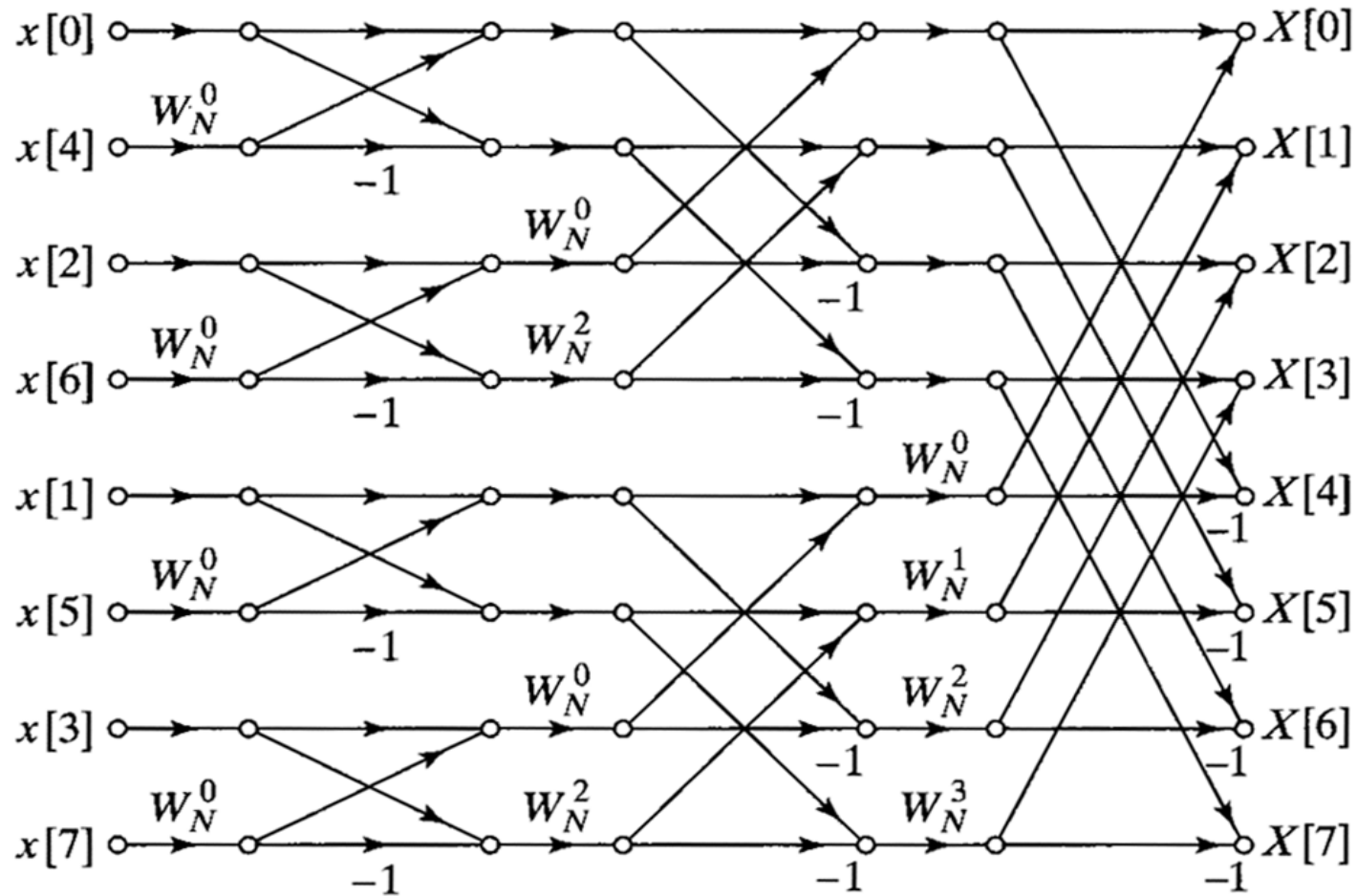
# Decimation-In-Frequency FFT Algorithm

- Final flow graph for 8-point decimation in frequency



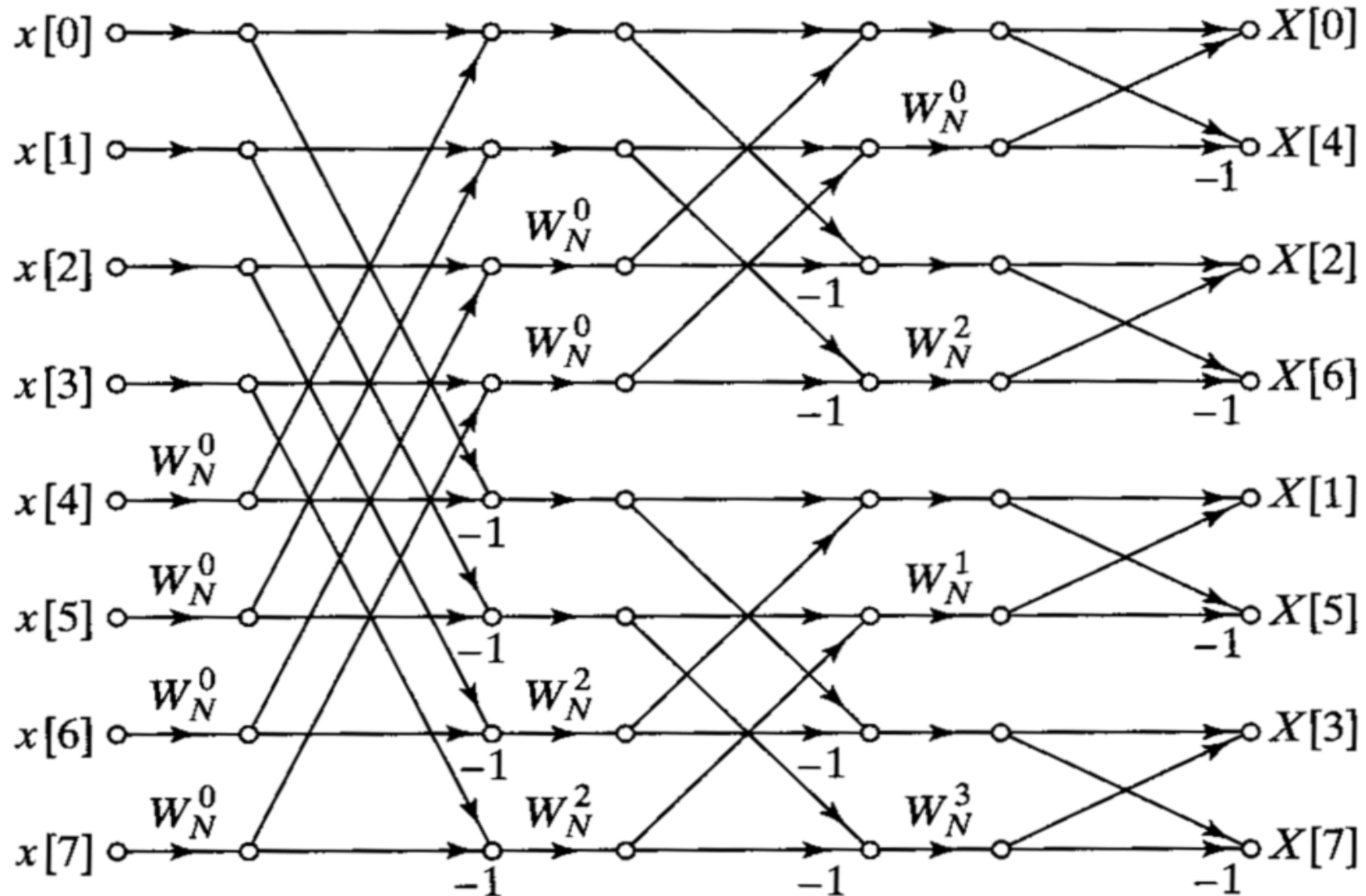
# Decimation-In-Frequency FFT Algorithm

- ⦿ DIT structure with input bit-reversed, output natural



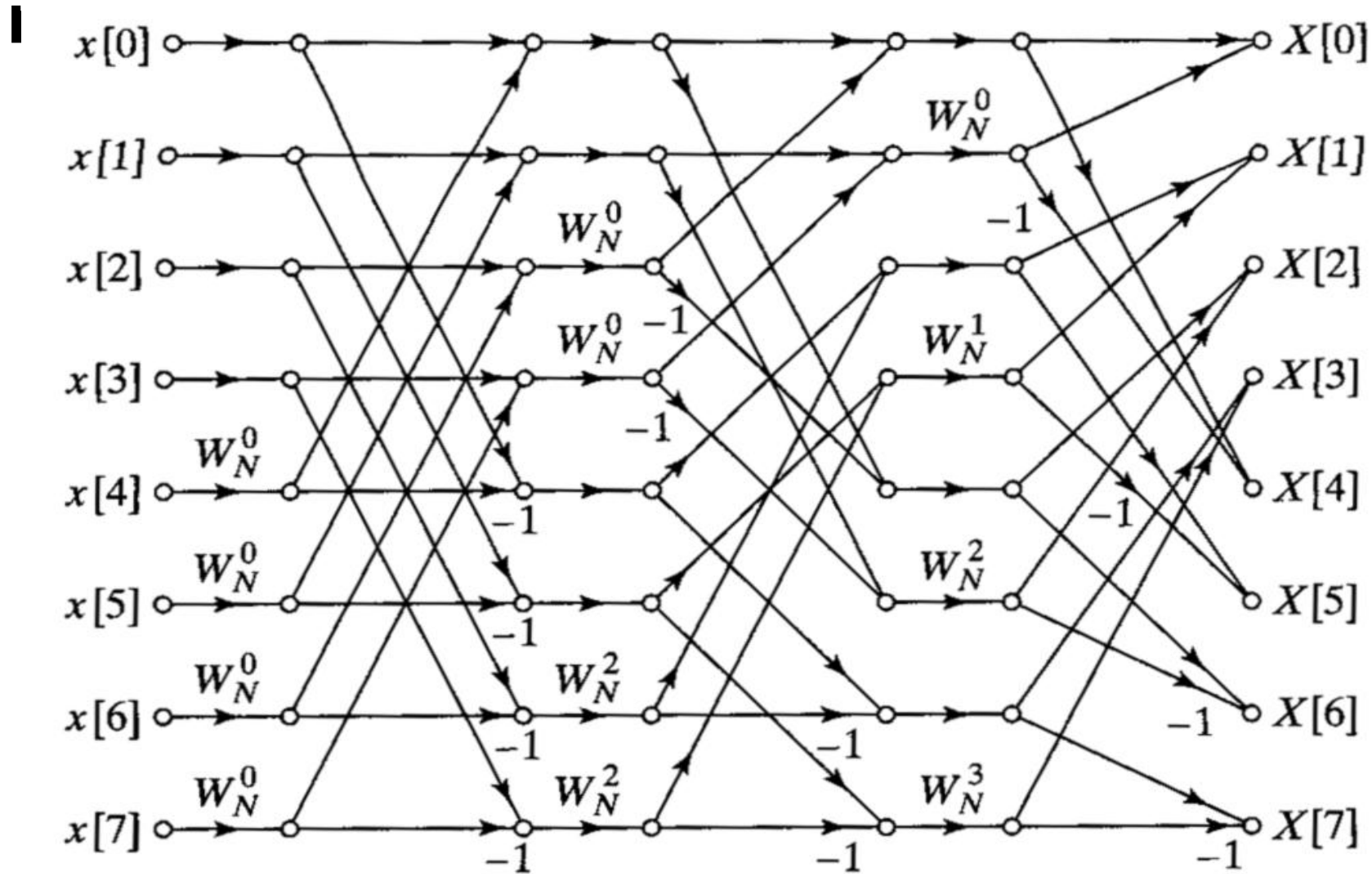
# Decimation-In-Frequency FFT Algorithm

- ⊙ DIT structure with input natural, output bit-reversed



# Decimation-In-Frequency FFT Algorithm

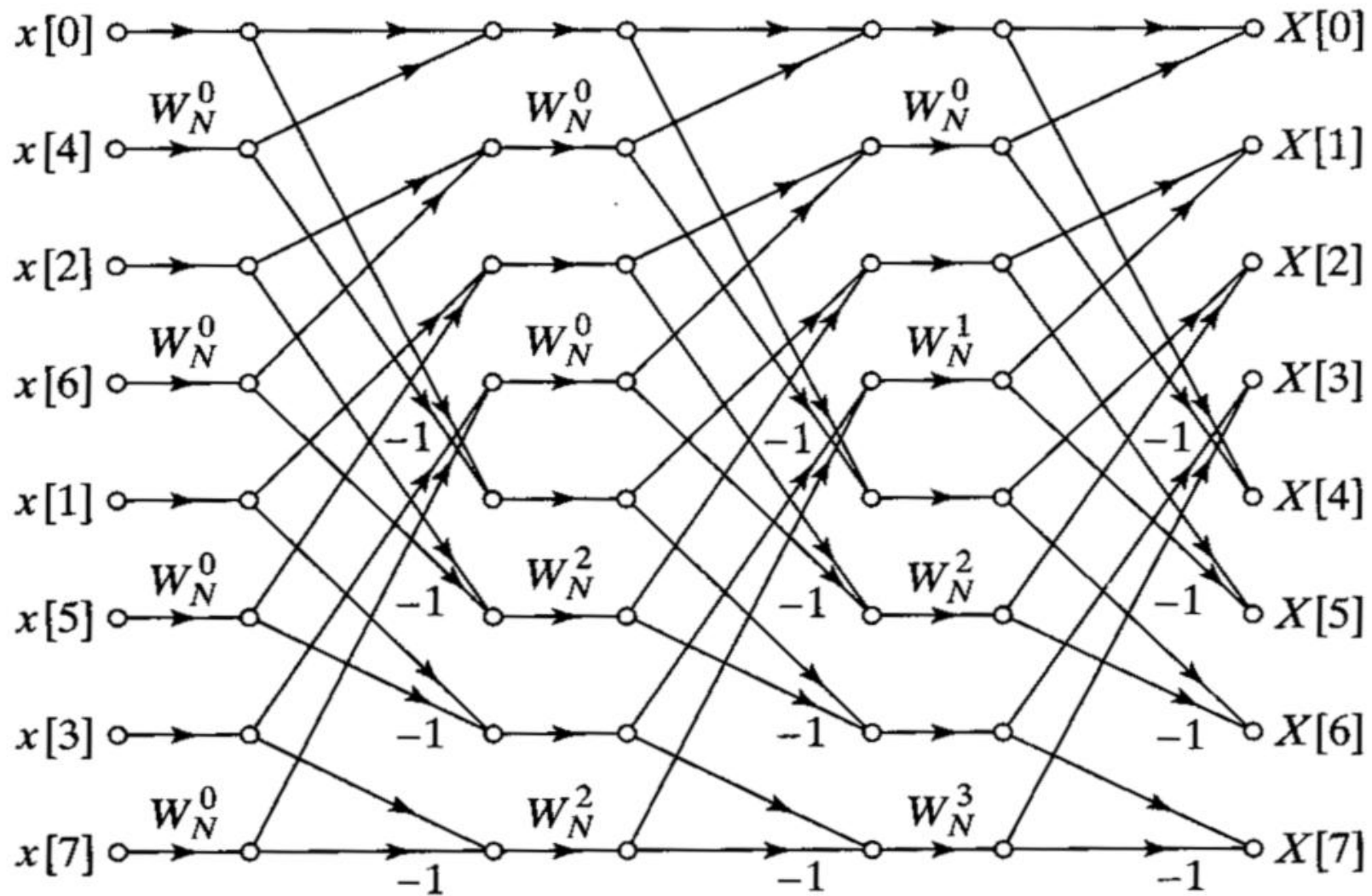
## ⦿ DIT structure with both input and output





# Decimation-In-Frequency FFT Algorithm

- ◎ DIT structure with same structure for each stage



- ◎ **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**

# 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 subroutine

# 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  $1/N$
- ⊙ This method has the advantage that the internal FFT code is undisturbed; it is widely used.



- ⦿ **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**

# Decimation-In-Frequency FFT Algorithm

- Consider the original DFT equation ....

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

- Separate the first half and the second half of time samples:

$$\begin{aligned} X[k] &= \sum_{n=0}^{(N/2)-1} x[n]W_N^{nk} + \sum_{n=N/2}^{N-1} x[n]W_N^{nk} \\ &= \sum_{n=0}^{(N/2)-1} x[n]W_N^{nk} + W_N^{(N/2)k} \sum_{n=0}^{(N/2)-1} x[n+(N/2)]W_N^{nk} \\ &= \sum_{n=0}^{(N/2)-1} \left[ x[n] + (-1)^k x[n+(N/2)] \right] W_N^{nk} \end{aligned}$$

- Note that these are **not**  $N/2$ -point DFTs

# Decimation-In-Frequency FFT Algorithm

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

⊙ **For  $k$  even, let  $k = 2r$**

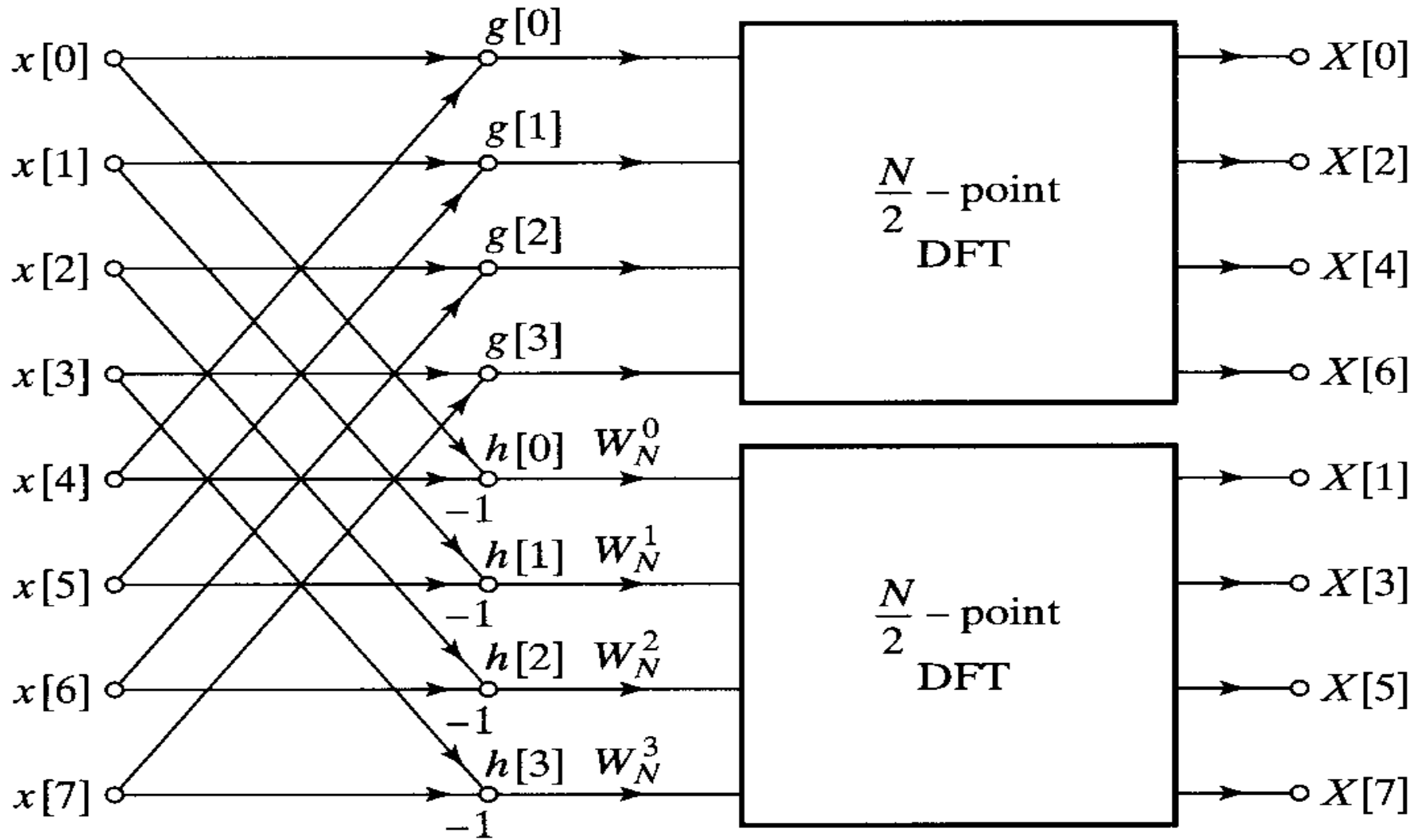
$$X[k] = \sum_{n=0}^{(N/2)-1} \left[ x[n] + (-1)^{2r} x[n + (N/2)] \right] W_N^{n2r} = \sum_{n=0}^{(N/2)-1} \left[ x[n] + x[n + (N/2)] \right] W_{N/2}^{nr}$$

⊙ **For  $k$  odd, let  $k = 2r + 1$**

$$\begin{aligned} X[k] &= \sum_{n=0}^{(N/2)-1} \left[ x[n] + (-1)^{2r} (-1) x[n + (N/2)] \right] W_N^{n(2r+1)} \\ &= \sum_{n=0}^{(N/2)-1} \left[ x[n] - x[n + (N/2)] \right] W_N^n W_{N/2}^{nr} \end{aligned}$$

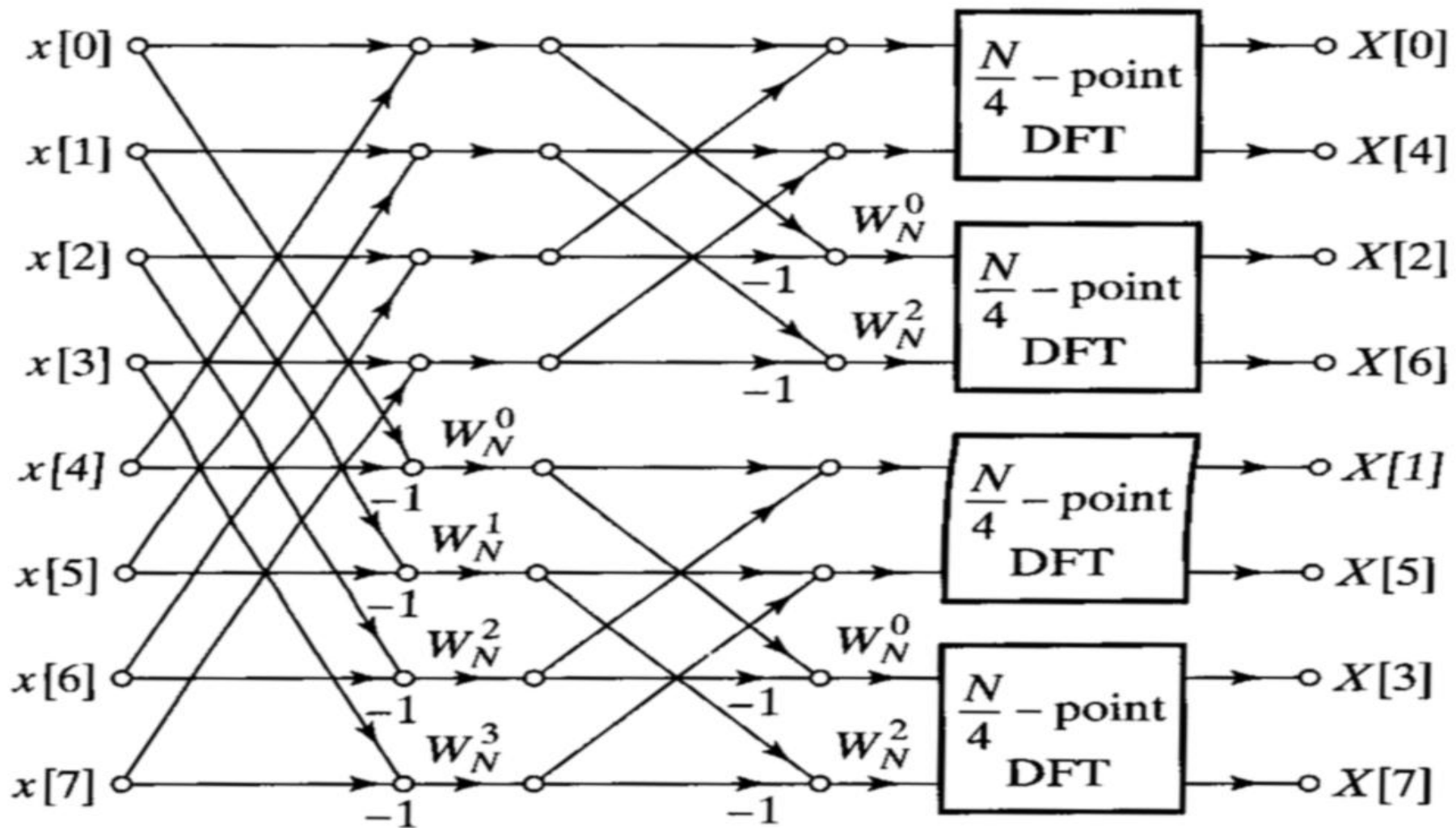
⊙ **These expressions are the  $N/2$ -point DFTs of  $x[n] + x[n + (N/2)]$  and  $x[n] - x[n + (N/2)]$**

# Decimation-In-Frequency FFT Algorithm



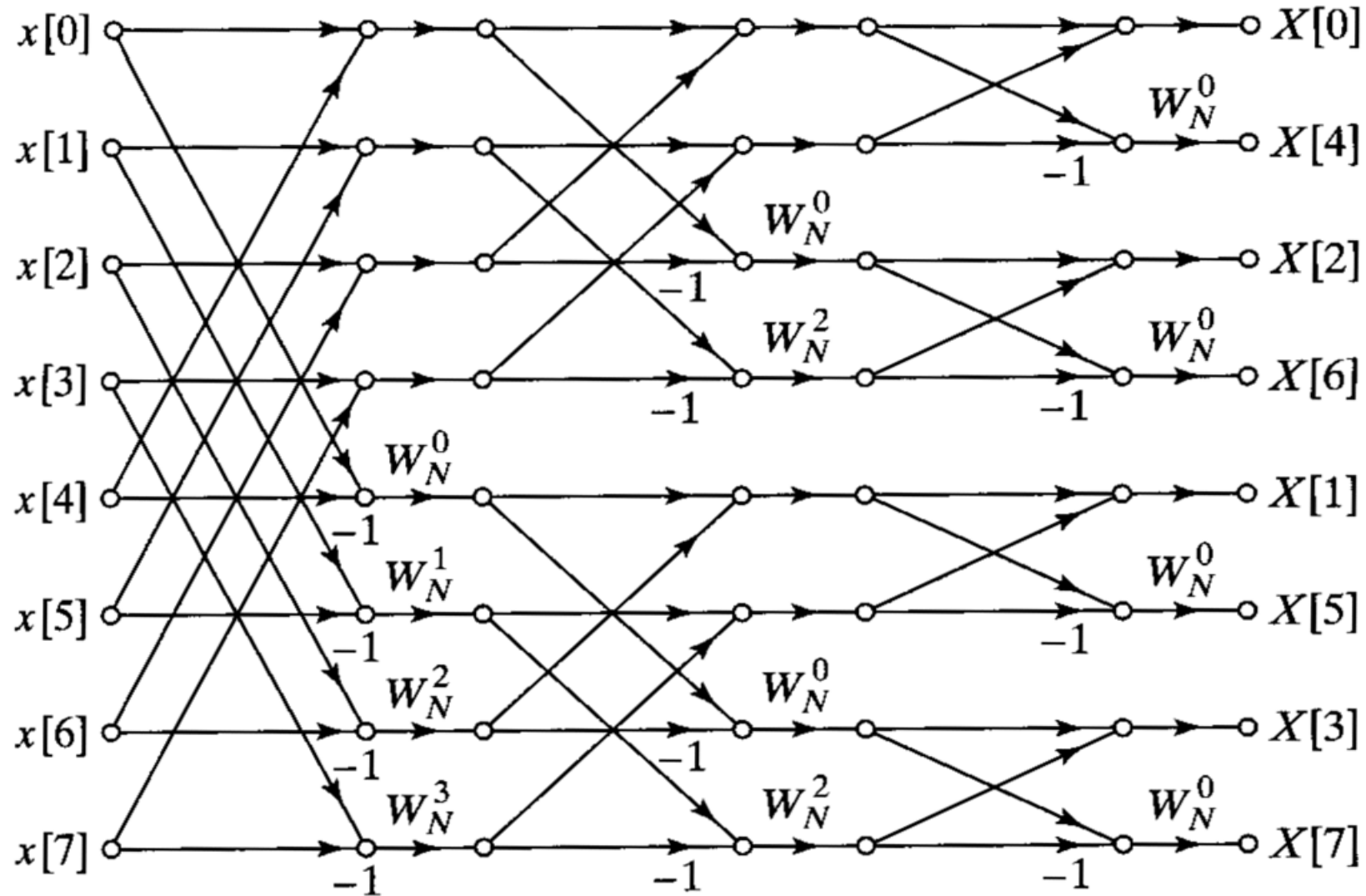
# Decimation-In-Frequency FFT Algorithm

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



# Decimation-In-Frequency FFT Algorithm

... and replacing the  $N/4$ -point DFTs by butterflies we obtain

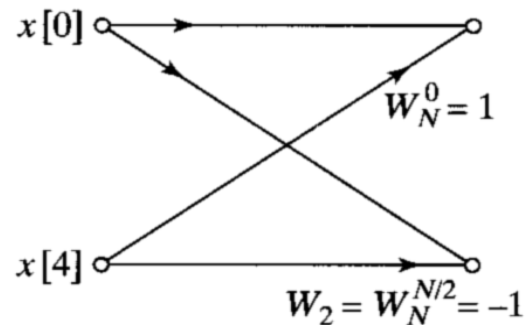


The DIF FFT is the *transpose* of the DIT FFT

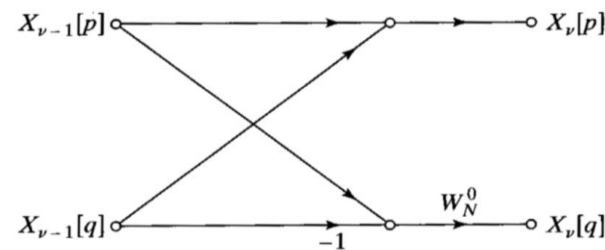
◎ **To obtain flowgraph transposes:**

- Reverse direction of flowgraph arrows
- Interchange input(s) and output(s)

◎ **DIT butterfly:**



**DIF butterfly:**



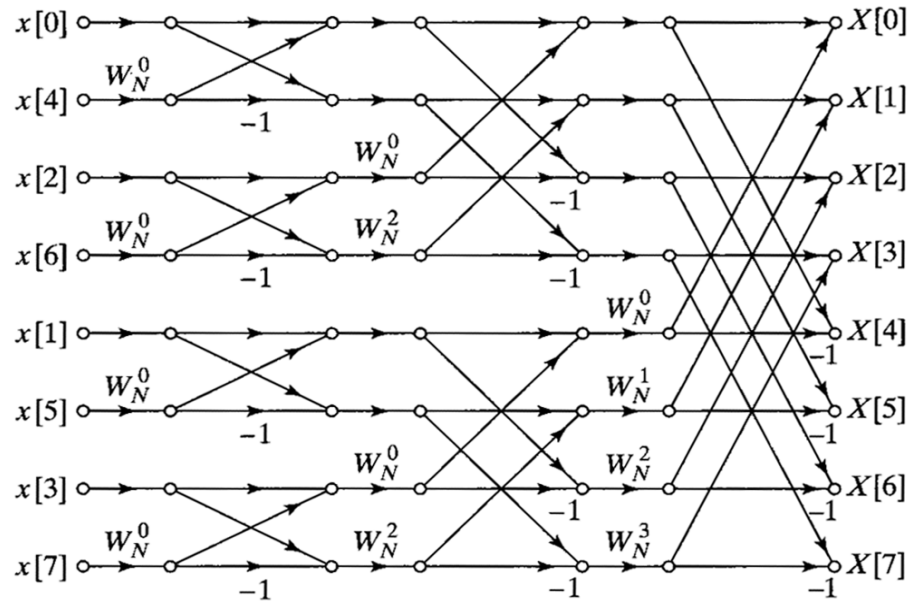
◎ **Comment:**

- We will revisit transposed forms again in our discussion of filter implementation

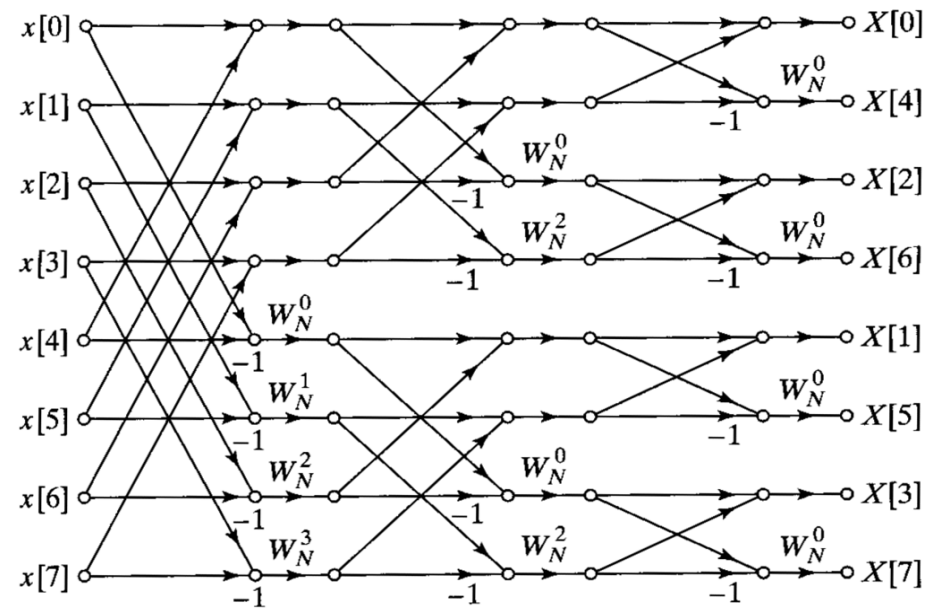
The DIF FFT is the *transpose* of the DIT FFT

Comparing DIT and DIF structures:

DIT FFT structure:



DIF FFT structure:

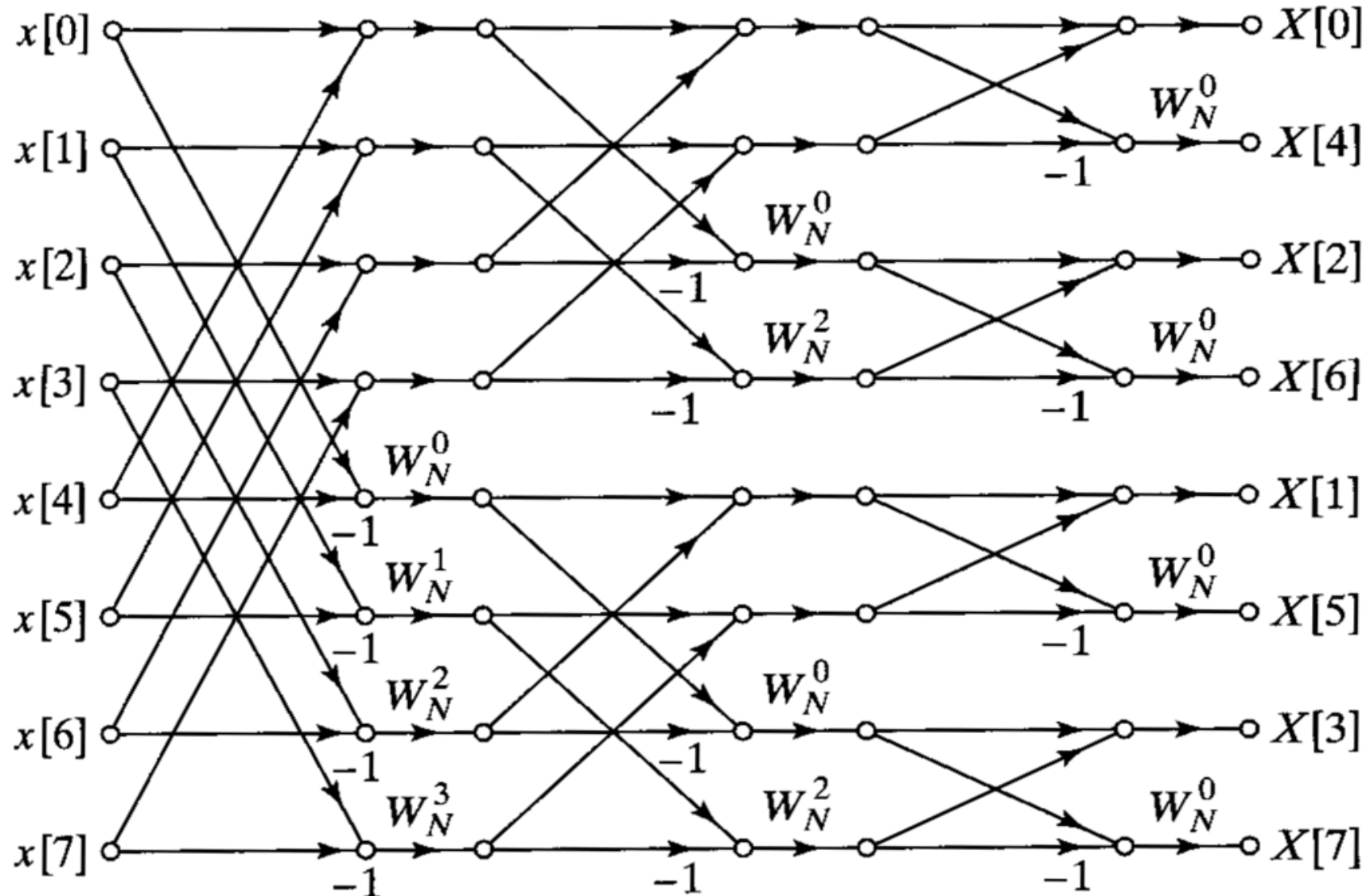


Alternate forms for DIF FFTs are similar to those of DIT FFTs



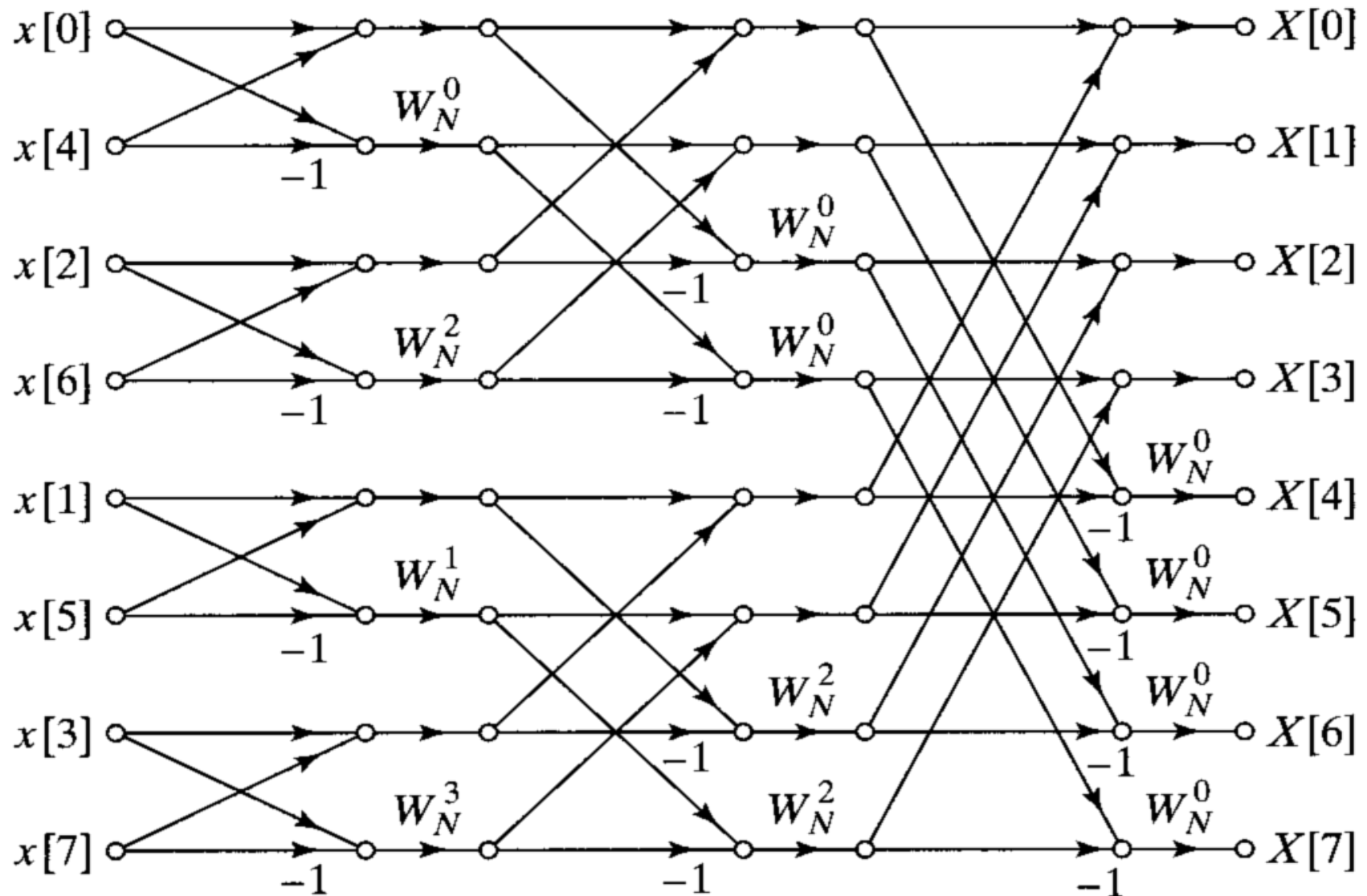
# Alternate DIF FFT structures

- ⊙ DIF structure with input natural, output bit-reversed



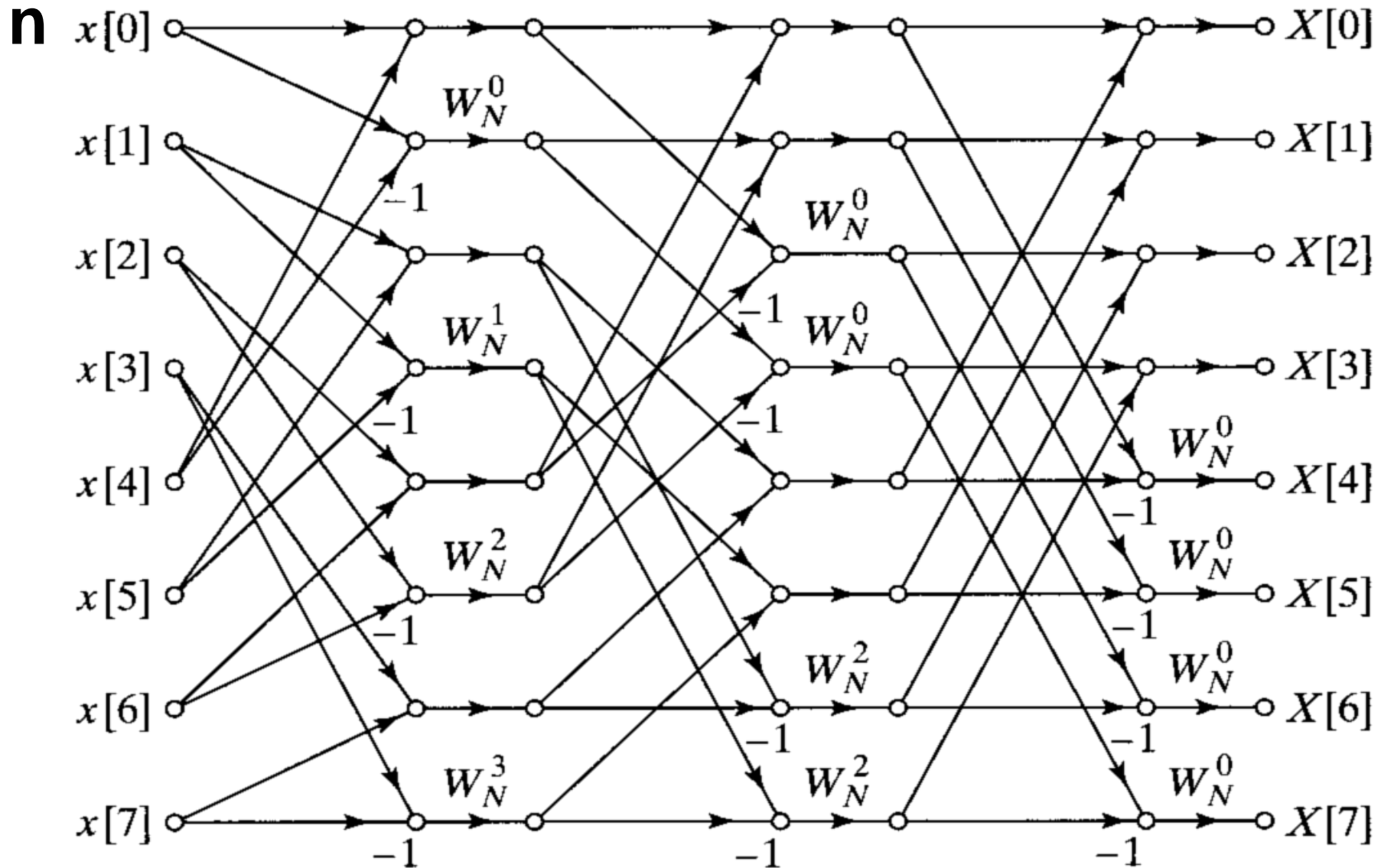
# Alternate DIF FFT structures

- ⊙ DIF structure with input bit-reversed, output natural



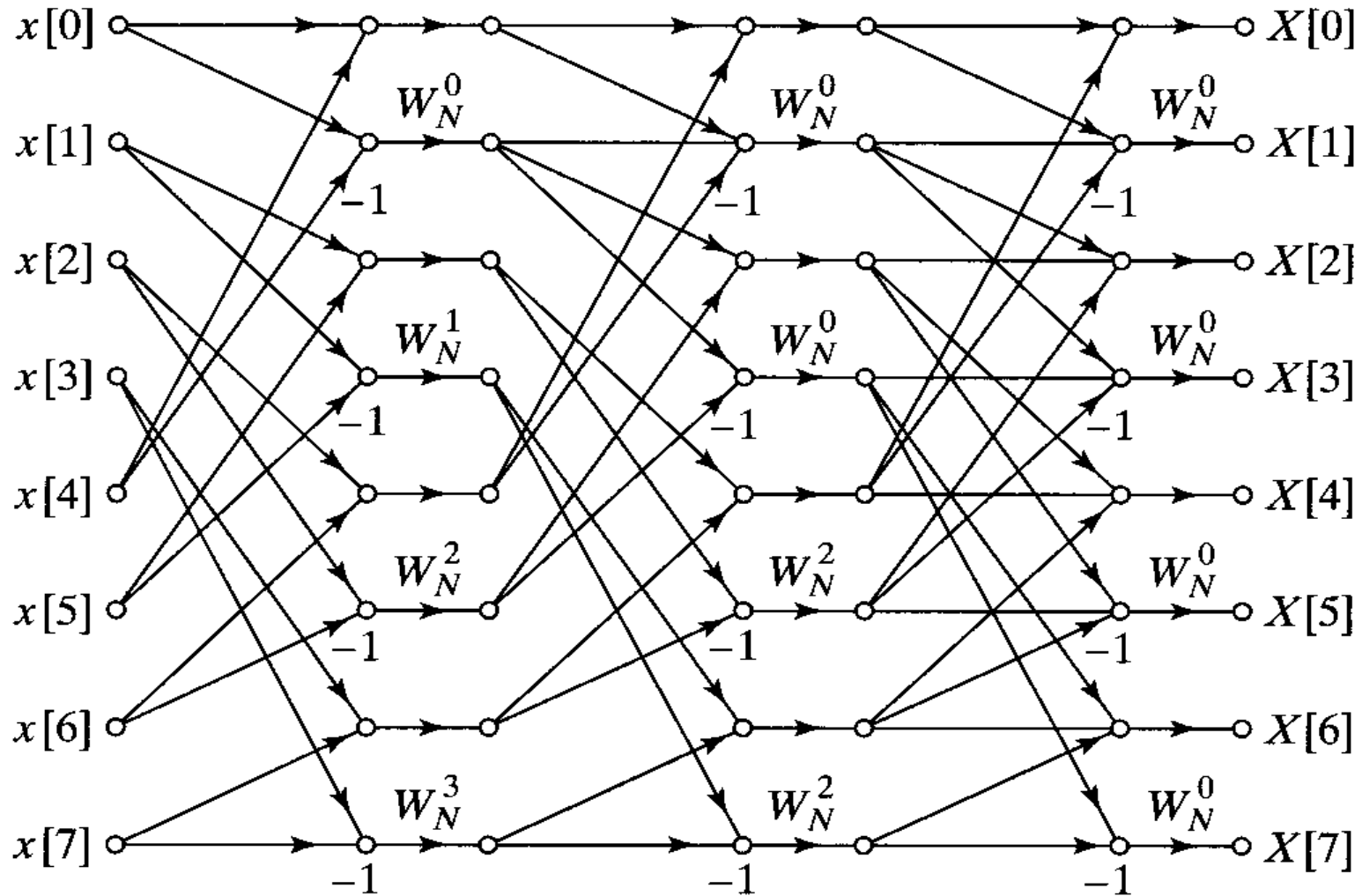
# Alternate DIF FFT structures

## ⦿ DIF structure with both input and output



# Alternate DIF FFT structures

- ⦿ DIF structure with same structure for each stage



# FFT structures for other DFT sizes

- ⊙ Can we do anything when the DFT size  $N$  is not an integer power of 2 (the non-radix 2 case)?
- ⊙ Yes! Consider a value of  $N$  that is not a power of 2, but that still is highly factorable

... Let  $N = p_1 p_2 p_3 p_4 \dots p_v$ ;  $q_1 = N / p_1$ ,  $q_2 = N / p_1 p_2$ , etc.

- ⊙ Then let

$$\begin{aligned}
 X[k] &= \sum_{n=0}^{N-1} x[n] W_N^{nk} \\
 &= \sum_{r=0}^{q_1-1} x[p_1 r] W_N^{p_1 r k} + \sum_{r=0}^{q_1-1} x[p_1 r + 1] W_N^{(p_1 r + 1)k} + \sum_{r=0}^{q_1-1} x[p_1 r + 2] W_N^{(p_1 r + 2)k} + \dots
 \end{aligned}$$

- ⊙ An arbitrary term of the sum on the previous panel is

$$\begin{aligned} & \sum_{r=0}^{q_1-1} x[p_1 r + l] W_N^{(p_1 r + l)k} \\ &= \sum_{r=0}^{q_1-1} x[p_1 r + l] W_N^{p_1 r k} W_N^{lk} = W_N^{lk} \sum_{r=0}^{q_1-1} x[p_1 r + l] W_{q_1}^{rk} \end{aligned}$$

- ⊙ This is, of course, a DFT of size  $q_1$  of points spaced by  $p_1$

- ⊙ In general, for the first decomposition we use

$$X[k] = \sum_{l=0}^{p_1-1} W_N^{lk} \sum_{r=0}^{q_1-1} x[p_1r + l] W_{q_1}^{rk}$$

- ⊙ **Comments:**

- This procedure can be repeated for subsequent factors of  $N$
- The amount of computational savings depends on the extent to which  $N$  is “composite”, able to be factored into small integers
- Generally the smallest factors possible used, with the exception of some use of radix-4 and radix-8 FFTs

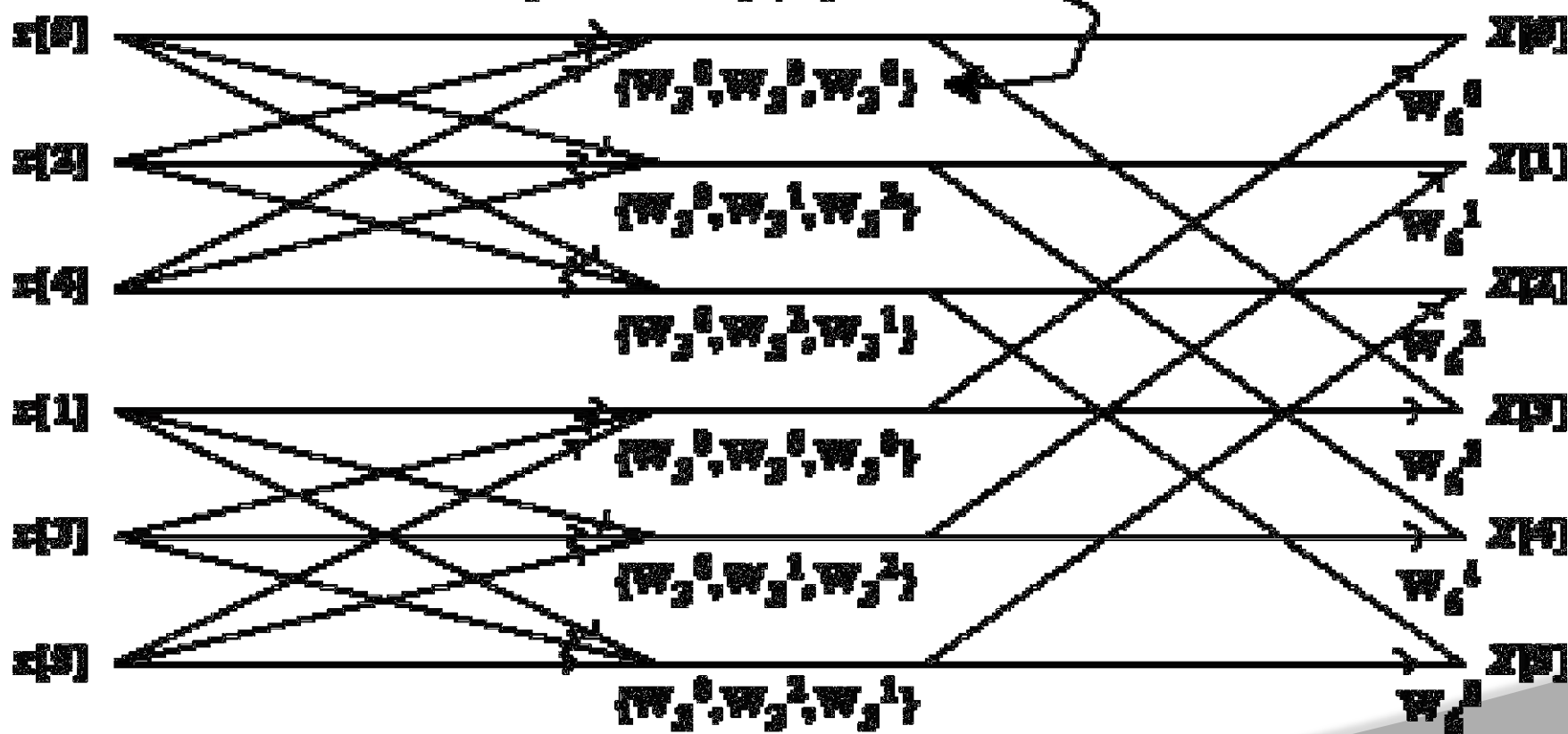


# Example: the 6-point DIT FFT

⊙  $P_1 = 2; P_2 = 3;$

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

Twiddle factors for 3-point butterflies, top to bottom





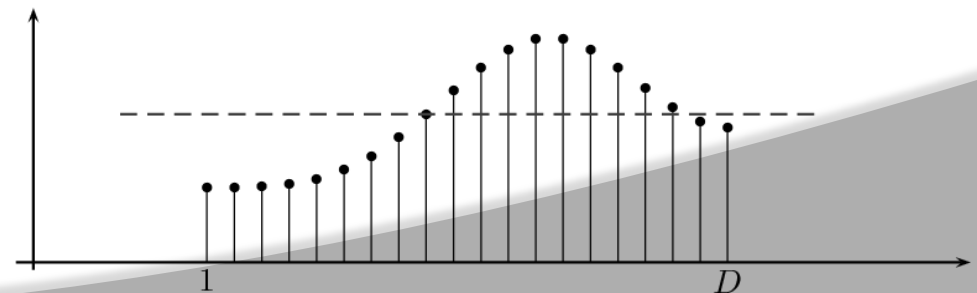
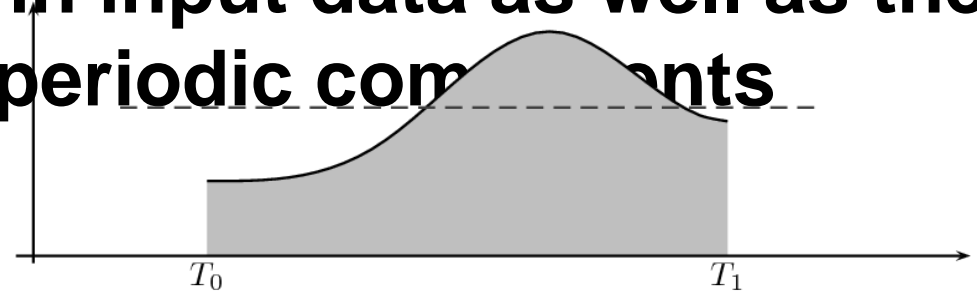
# Discrete Fourier Transform (DFT): Overview

## What?

- Converts a sampled function from time domain to frequency domain

## Use?

- DFTs reveal periodicities in input data as well as the relative strengths of any periodic components



# Discrete Fourier Transform (DFT): Mathematical Interpretation



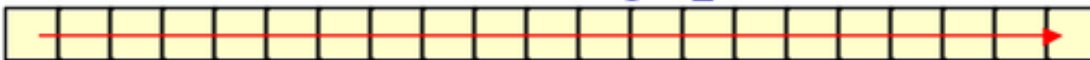
- Equation: 
$$X_k = \sum_{n=0}^{N-1} x_n \cdot e^{-i2\pi \frac{k}{N} n}$$
- Interpretation:
  - $x_n$  is original continuous time signal
  - $N$  is the frequency of the sampling rate which denotes the number of values to transform
  - Result is a sequence of magnitudes at  $N$  frequencies
- Runtime Complexity
  - To transform  $N$  values, the DFT performs  $N$  complex multiplications of  $x_n$  with  $e^{-\frac{i2\pi kn}{N}}$  and  $N-1$  additions of the resulting values
  - $O(N^2)$

# FFT: Cooley-Tukey algorithm

- ◎ **Most common FFT algorithm**
  - Divide and conquer algorithm
  
- ◎ **Methodology**
  - Breaks up DFT of  $N$  samples into  $N=N_1N_2$
  
- ◎ **Benefit**
  - Can be combined with any other DFT algorithm
  - What Matlab fft function does for optimization

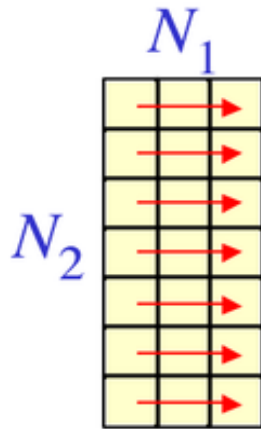
# FFT: Cooley-Tukey algorithm Methodology

1d DFT of size  $N$ :  $N = N_1 N_2$



$\approx$  2d DFT of size  $N_1 \times N_2$

reinterpret 1d inputs:

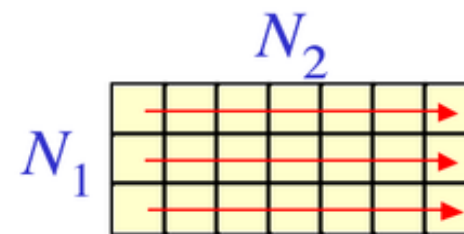


$\longrightarrow$  = contiguous

first DFT columns, size  $N_2$   
(non-contiguous)

multiply by  $N$  “twiddle factors”

*transpose*



finally, DFT columns, size  $N_1$   
(non-contiguous)

# Fast Fourier Transform (FFT):

- ⦿ Also called chirp z-transform algorithm
- ⦿ Methodology
  - Expresses DFT as a convolution
- ⦿ Benefit
  - Computes DFT of arbitrary sizes
  - Can be used to compute more general transforms
- ⦿ Tradeoff
  - Only  $O(N \log_2 N)$  complexity for prime-sized DFTs

# Fast Fourier Transform (FFT): Bluestein Methodology



$$X_k = \sum_{n=0}^{N-1} x_n \cdot e^{-i2\pi \frac{k}{N} n}$$

- Original DFT:

- Let  $nk = \frac{-(k-n)^2}{2} + \frac{n^2}{2} + \frac{k^2}{2}$

$$X_k = e^{-\frac{\pi i}{N} k^2} \sum_{n=0}^{N-1} \left( x_n e^{-\frac{\pi i}{N} n^2} \right) e^{\frac{\pi i}{N} (k-n)^2}$$

- Substitution:

- Convolution of 2 sequences:

$$a_n = x_n e^{-\frac{\pi i}{N} n^2}$$
$$b_n = e^{\frac{\pi i}{N} n^2}$$

# Goertzel Algorithm

## Use

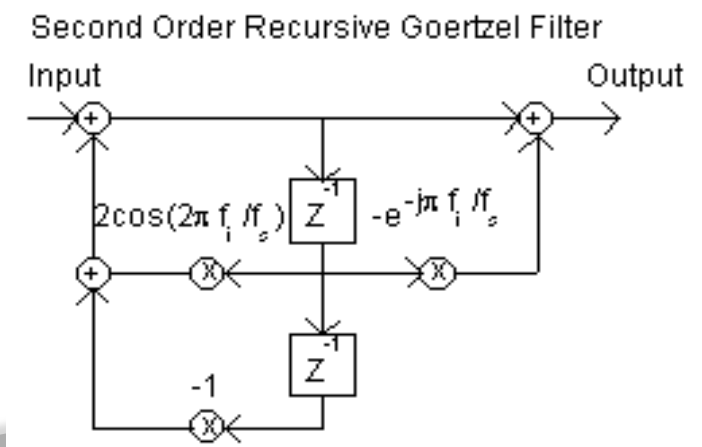
- Evaluation of individual terms of DFT
- Generally used for computing a small number of selected frequency components

## Methodology

- Calculation takes the form of an Infinite Impulse Response (IIR) band-pass filter
- Coefficients of DFT found by comparing signal energy before and after filter to see where in the frequency range it lies

$$H_{f_i}(z) = \frac{1 - e^{-j2\pi f_i/f_s} z^{-1}}{1 - 2 \cos(2\pi f_i/f_s) z^{-1} + z^{-2}}$$

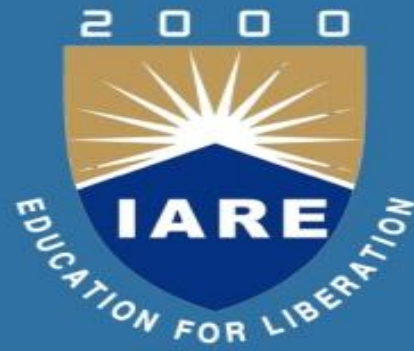
Where  $f_i$  is desired frequency and  $f_s$  is sampling frequency



# Goertzel Algorithm

- ⊙ Tradeoff
  - $O(NM)$ ,  $N$  is number of DFT terms,  $M$  is the set of DFT terms to calculate
- ⊙ Benefit
  - Simple structure of algorithm makes it well suited to small processors
  - More efficient than FFT for small number of frequencies ( if  $M < \log_2 N$ )
- ⊙ Applications
  - Used to recognize DTMF tones produced by buttons on telephone keypad
  - Call progress (dial tone, busy)



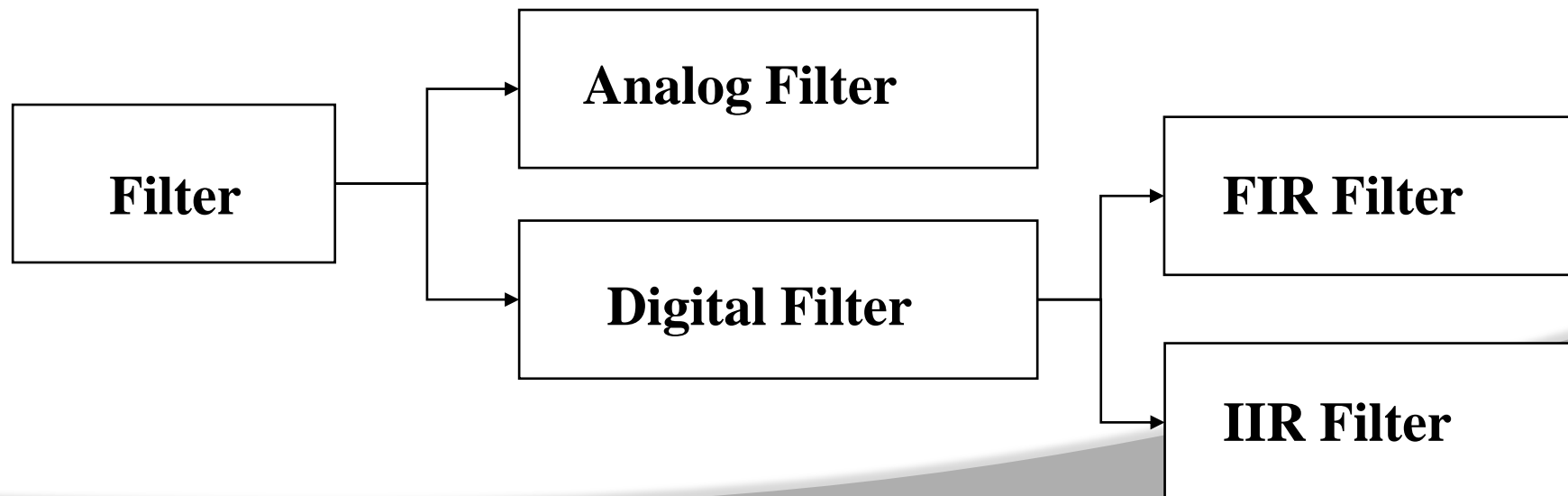


**UNIT- III**  
**STRUCUTRE OF IIR FILTERS**

CLO's	Course Learning Outcome
CLO7	Knowing the poles and zeros of a transfer function, make a rough sketch of the gain response.
CLO8	Define the Discrete Fourier Transform (DFT) and the inverse DFT (IDFT) of length $N$ .
CLO9	Understand the inter-relationship between DFT and various transforms.
CLO10	Understand the significance of various filter structures and effects of round-off errors.
CLO11	Understand the fast computation of DFT and appreciate the FFT Processing.

# Introduction

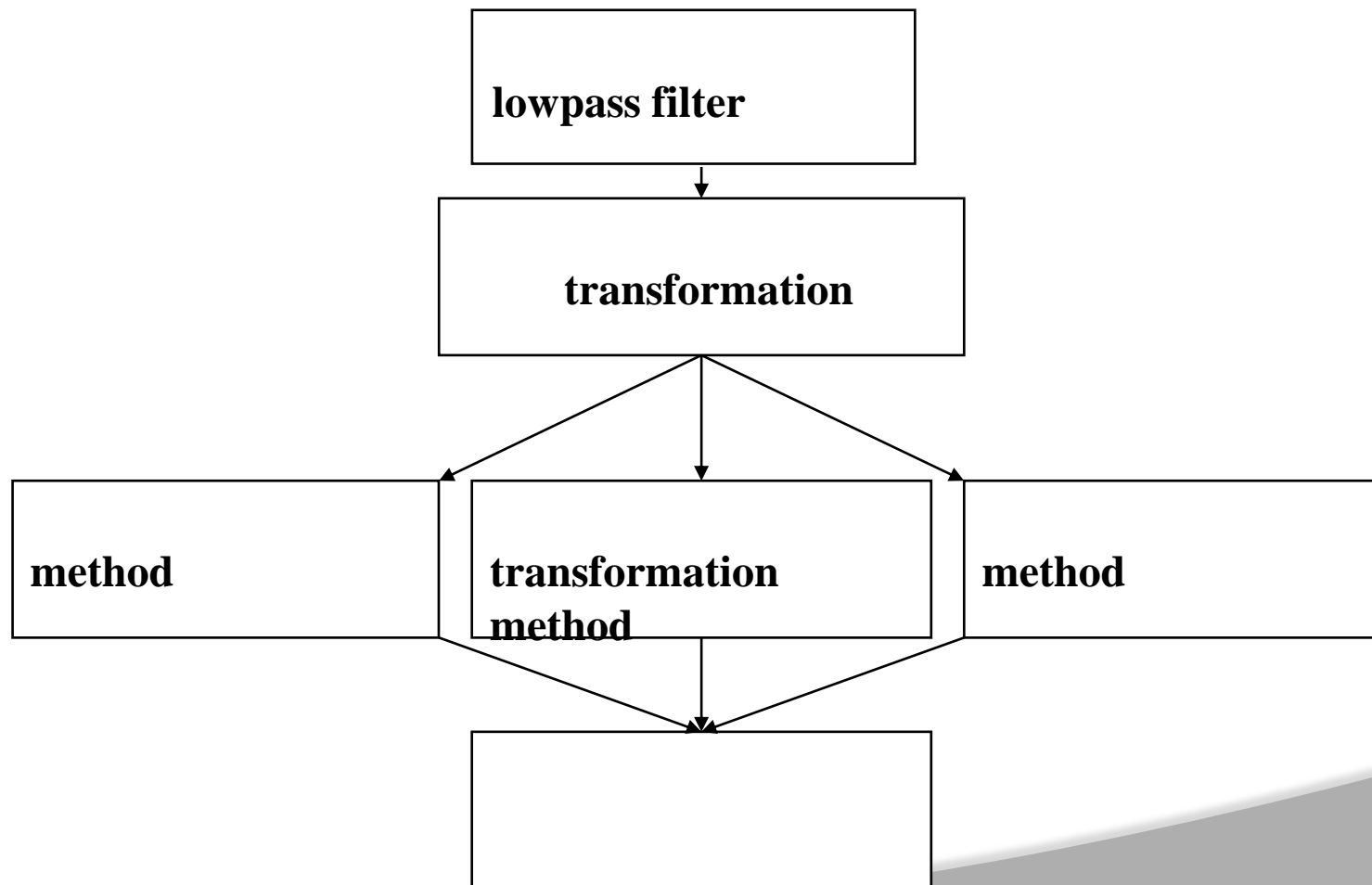
- ⦿ Basic filter classification
- ⦿ We put emphasis on the digital filter now, and will introduce to the design method of the FIR filter and IIR filter respectively.



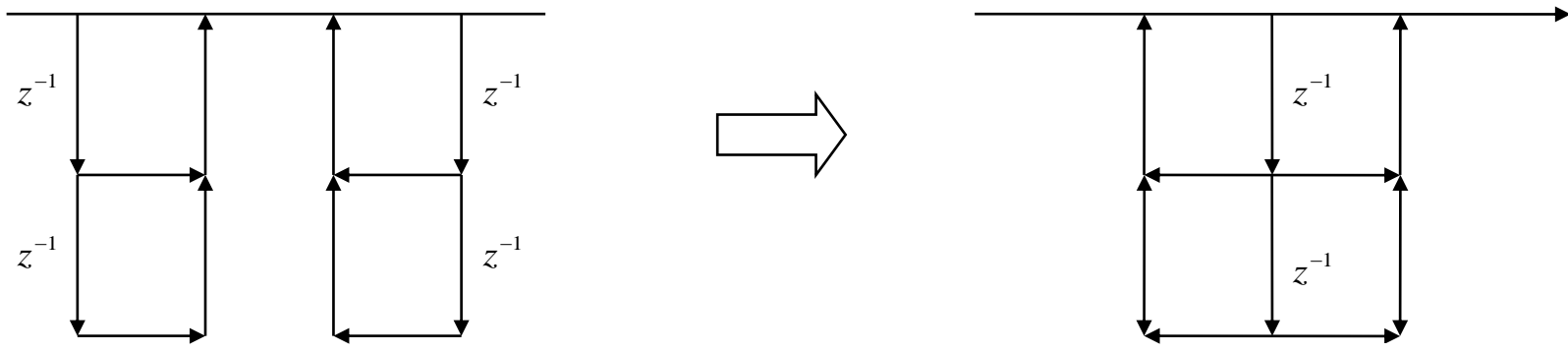
- ◎ IIR is the infinite impulse response abbreviation.
- ◎ Digital filters by the accumulator, the multiplier, and it constitutes IIR filter the way, generally may divide into three kinds, respectively is Direct form, Cascade form, and Parallel form.

- ◎ IIR filter design methods include the impulse invariance, bilinear transformation, and step invariance.
- ◎ We must emphasize at impulse invariance and bilinear transformation.

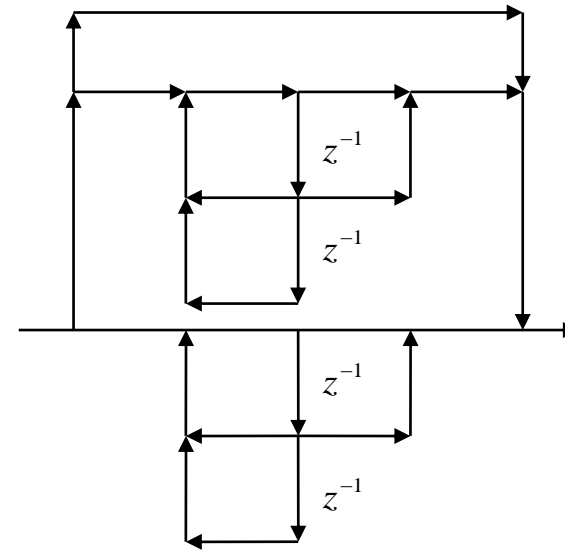
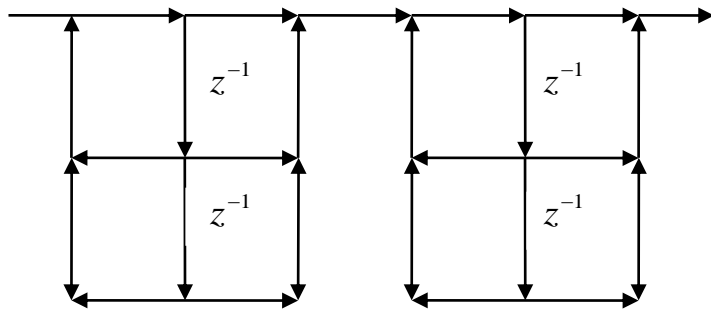
## ◎ IIR filter design methods



# ◎ The structures of IIR filter



◎ The structures of IIR filter

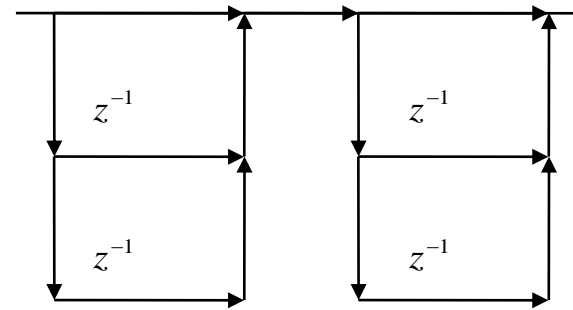
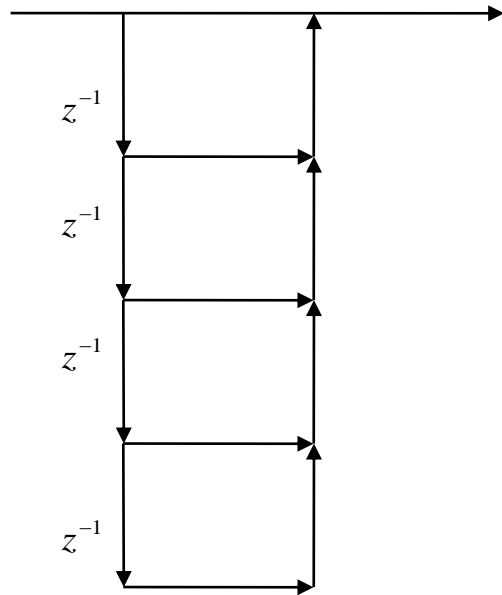




- ⦿ FIR is the finite impulse response abbreviation, because its design construction has not returned to the part which gives.
- ⦿ Its construction generally uses Direct form and Cascade form.

- ① FIR filter design methods include the window function, frequency sampling, minimize the maximal error, and MSE.
- ① We must emphasize at window function, frequency sampling, and MSE.

# ◎ The structures of FIR filter



# IIR Filter Design by Impulse invariance method



- ⦿ The most straightforward of these is the impulse invariance transformation
- ⦿ Let  $h_c(t)$  be the impulse response corresponding to  $H_c(s)$ , and define the continuous to discrete time transformation by setting
- ⦿ We sample the continuous time impulse response to produce the discrete time filter

$$h(n) = h_c(nT)$$

- ◎ The frequency response  $H'(\omega)$  is the Fourier transform of the continuous time function

$$h_c^*(t) = \sum_{n=-\infty}^{\infty} h_c(nT)\delta(t - nT)$$

and hence

$$H'(\omega) = \frac{1}{T} \sum_{k=-\infty}^{\infty} H_c \left[ j\left(\omega - k \frac{2\pi}{T}\right) \right]$$

# IIR Filter Design by Impulse invariance method

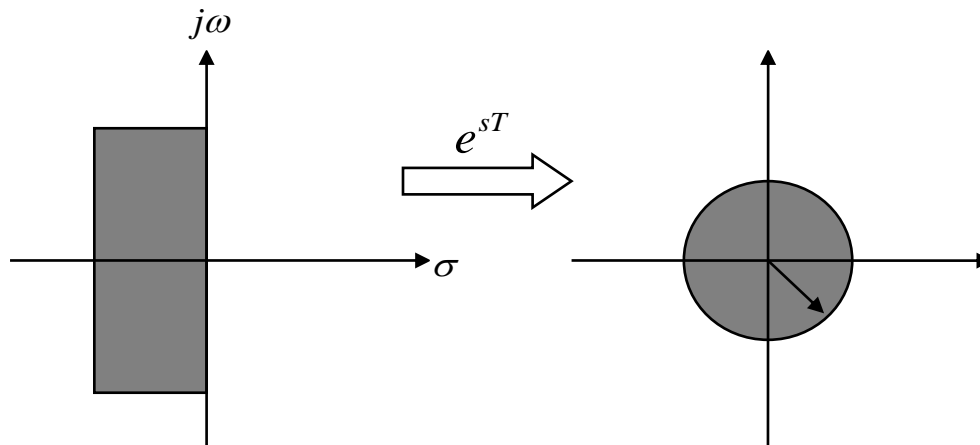
- ◎ The system function is

$$H(z) \Big|_{z=e^{sT}} = \frac{1}{T} \sum_{k=-\infty}^{\infty} H_c \left[ s - jk \frac{2\pi}{T} \right]$$

- ◎ It is the many-to-one transformation from the s plane to the z plane.

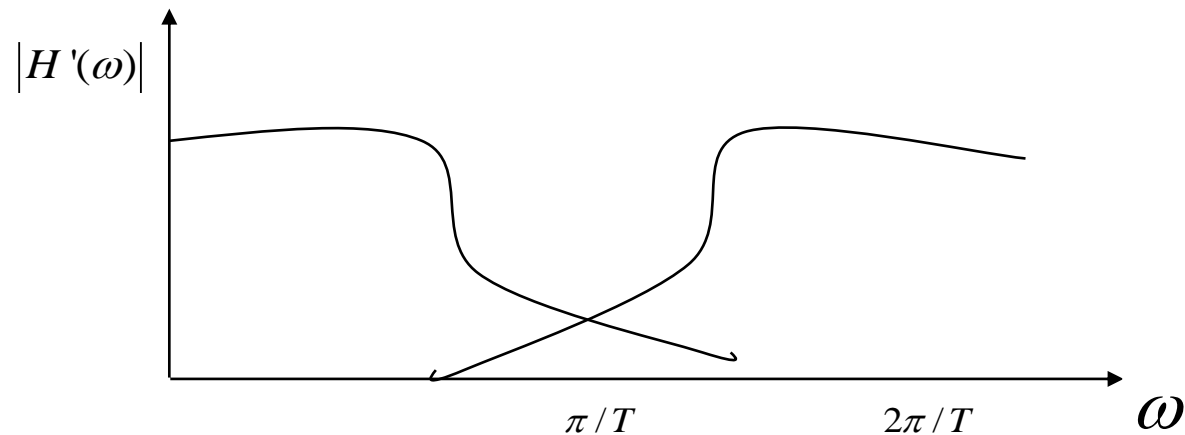
# IIR Filter Design by Impulse invariance method

- ⦿ The impulse invariance transformation does map the  $j\omega$ -axis and the left-half  $s$  plane into the unit circle and its interior, respectively



# IIR Filter Design by Impulse invariance method

- $H'(\omega)$  is an aliased version of  $H_c(j\omega)$



- The stop-band characteristics are maintained adequately in the discrete time frequency response only if the aliased tails of  $H_c(j\omega)$  are sufficiently small.



# IIR Filter Design by Impulse invariance method



- ⦿ The Butterworth and Chebyshev-I lowpass designs are more appropriate for impulse invariant transformation than are the Chebyshev-II and elliptic designs.
- ⦿ This transformation cannot be applied directly to highpass and bandstop designs.

# IIR Filter Design by Impulse invariance method

- ⦿  $H_c(s)$  is expanded a partial fraction expansion to produce

$$H_c(s) = \sum_{k=1}^N \frac{A_k}{s - s_k}$$

- ⦿ We have assumed that there are no multiple poles

- ⦿ And thus  $h_c(t) = \sum_{k=1}^N A_k e^{s_k t} u(t)$        $h(n) = \sum_{k=1}^N A_k e^{s_k nT} u(n)$

$$H(z) = \sum_{k=1}^N \frac{A_k}{1 - e^{s_k T} z^{-1}}$$

# IIR Filter Design by Impulse invariance method

## ◎ Example:

$$H_c(s) = \frac{s + a}{(s + a)^2 + b^2}$$

Expanding in a partial fraction

expansion, it produce

$$H_c(s) = \frac{1/2}{s + a + jb} + \frac{1/2}{s + a - jb}$$

The impulse invariant transformation yields a discrete time design with the

system function

$$H(z) = \frac{1/2}{1 - e^{(-a+jb)T} z^{-1}} + \frac{1/2}{1 - e^{(-a-jb)T} z^{-1}}$$

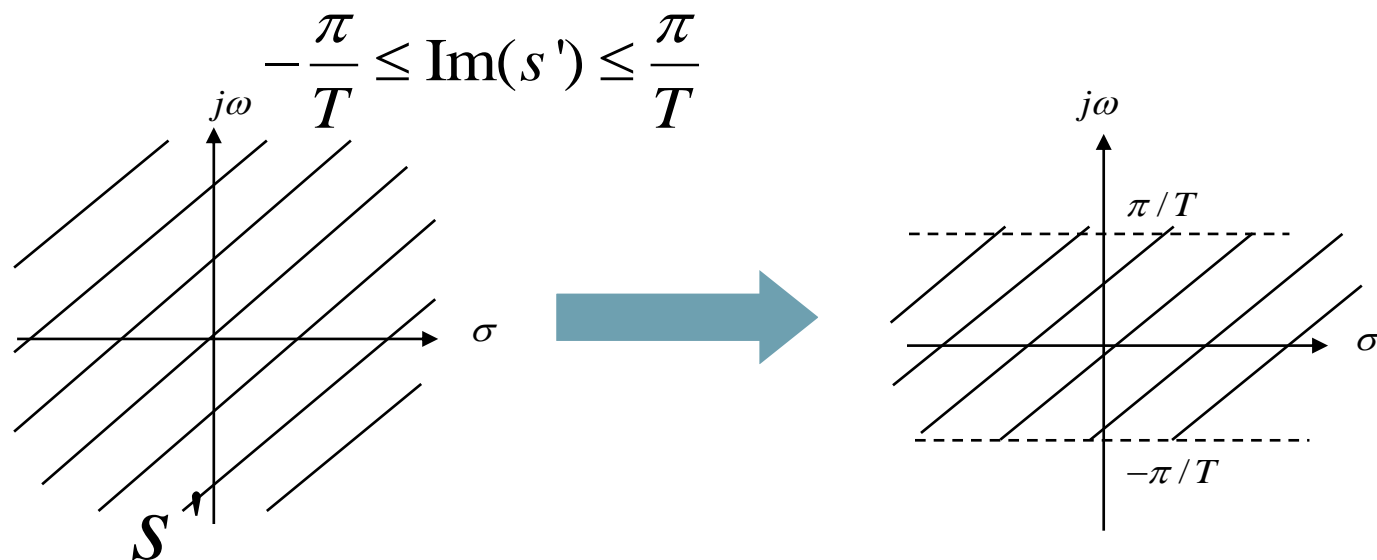
# IIR Filter Design by Bilinear transformation method



- ◎ The most generally useful is the bilinear transformation.
- ◎ To avoid aliasing of the frequency response as encountered with the impulse invariance transformation.
- ◎ We need a one-to-one mapping from the  $s$  plane to the  $z$  plane.
- ◎ The problem with the transformation is many-to-one.

$$z = e^{sT}$$

- We could first use a one-to-one transformation from  $s$  to  $s'$ , which compresses the entire  $s$  plane into the strip



- Then  $z = e^{s'T}$  could be transformed to  $z$  by

with no effect from aliasing.

- The transformation from  $s$  to  $s'$  is given by

$$s' = \frac{2}{T} \tanh^{-1} \left( \frac{sT}{2} \right)$$

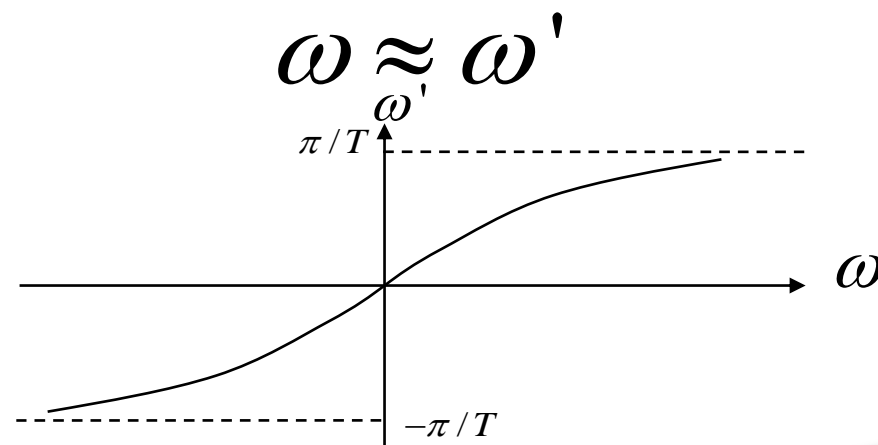
- The characteristic of this transformation is seen most readily from its effect on the axis.

- Substituting  $s = j\omega$  and  $s' = j\omega'$ , we obtain

$$\omega' = \frac{2}{T} \tan^{-1} \left( \frac{\omega T}{2} \right)$$

# IIR Filter Design by Bilinear transformation method

- ⦿ The  $\omega$  axis is compressed into the interval for  $(-\frac{\pi}{T}, \frac{\pi}{T})$  in a one-to-one method
- ⦿ The relationship between  $\omega'$  and  $\omega$  is nonlinear, but it is approximately linear at small  $\omega'$ .



# IIR Filter Design by Bilinear transformation method

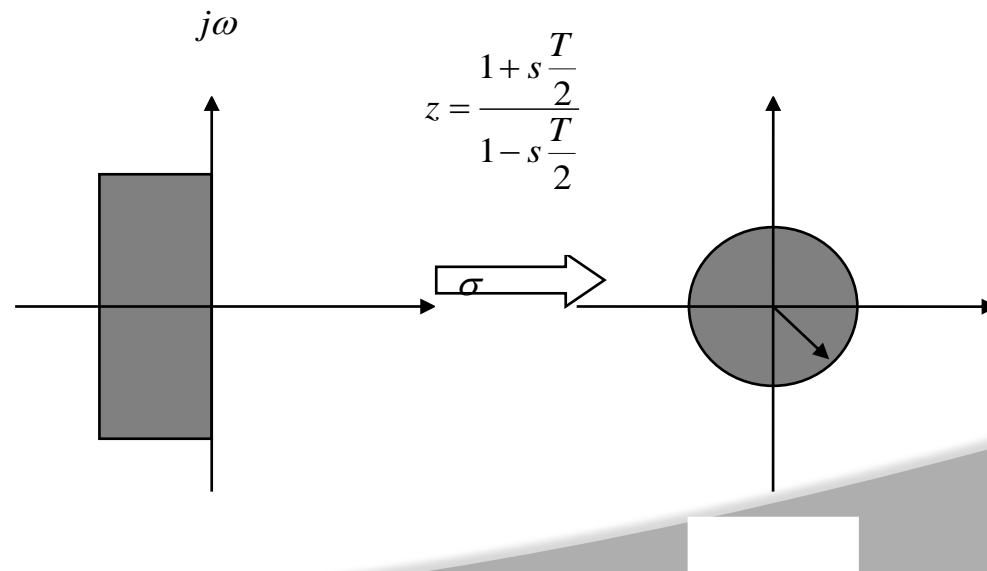
- The desired transformation  $s$  to  $z$  is now obtained by inverting to produce

$$s' = \frac{2}{T} \tanh^{-1}\left(\frac{sT}{2}\right)$$

$$s = \frac{2}{T} \tanh\left(\frac{s'T}{2}\right)$$
- And setting  $s' = \left(\frac{1}{T}\right) \ln z$  which yields

$$s = \frac{2}{T} \tanh\left(\frac{\ln z}{2}\right)$$

$$= \frac{2}{T} \left(\frac{1 - z^{-1}}{1 + z^{-1}}\right)$$





# IIR Filter Design by Bilinear transformation method

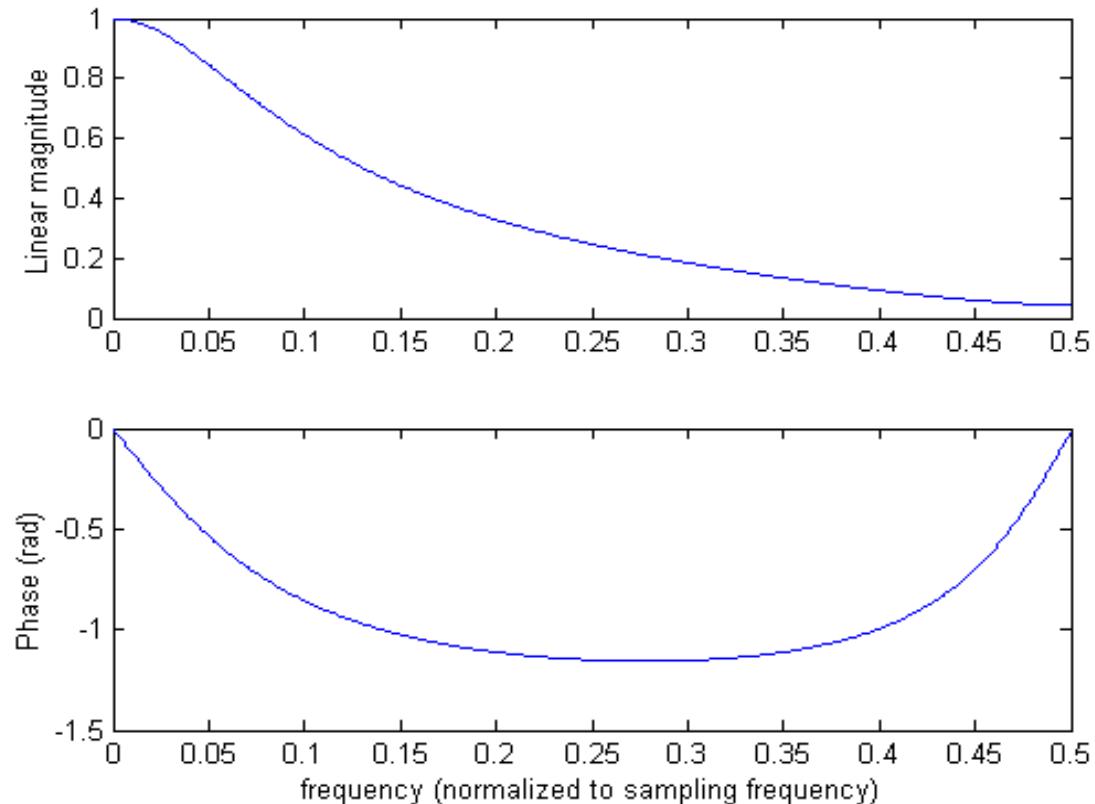
- ⦿ The discrete-time filter design is obtained from the continuous-time design by means of the bilinear transformation

$$H(z) = H_c(s) \Big|_{s=(2/T)(1-z^{-1})/(1+z^{-1})}$$

- ⦿ Unlike the impulse invariant transformation, the bilinear transformation is one-to-one, and invertible.

# Z Transfer Function

- We obtain the transfer function by evaluation of the z transform on the unit circle

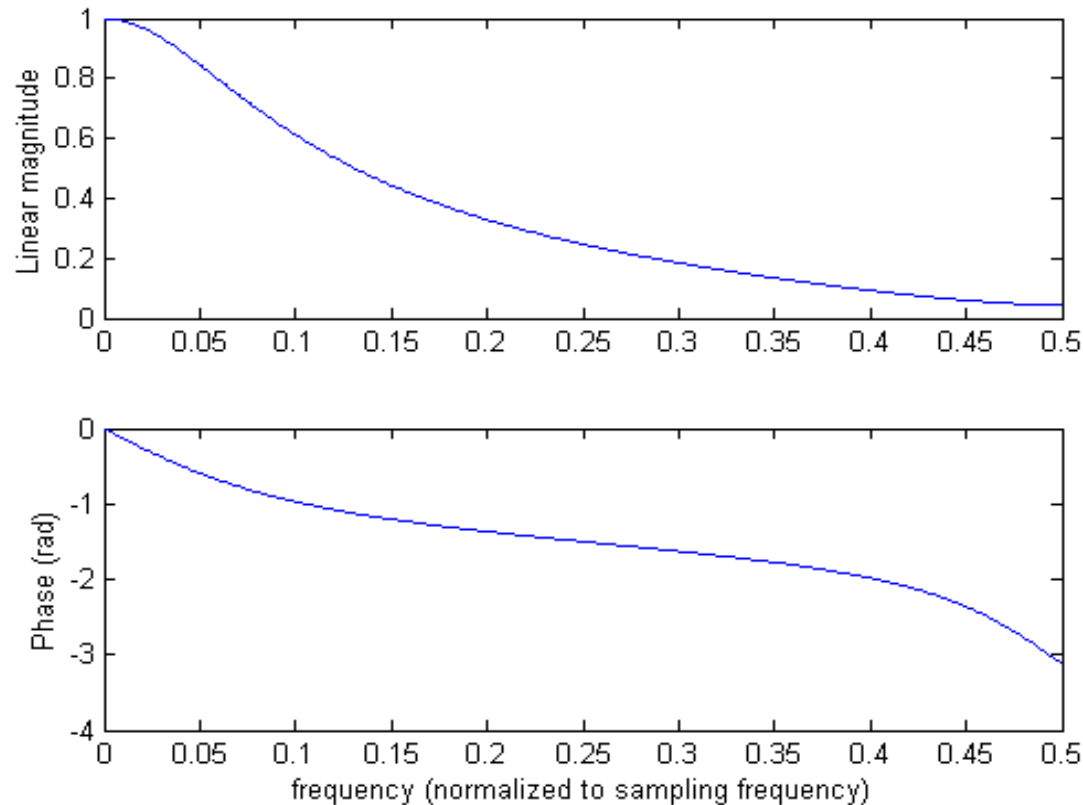


- We can see that it is a minimum phase filter (the phase comes back at 0 at  $F_e/2$ ) because the zero of the filter is inside the unit circle.

# Z Transfer Function

- If we change the zero  $z_1$  to  $1/z_1$ , we get the same magnitude transfer function (up to a scale factor)

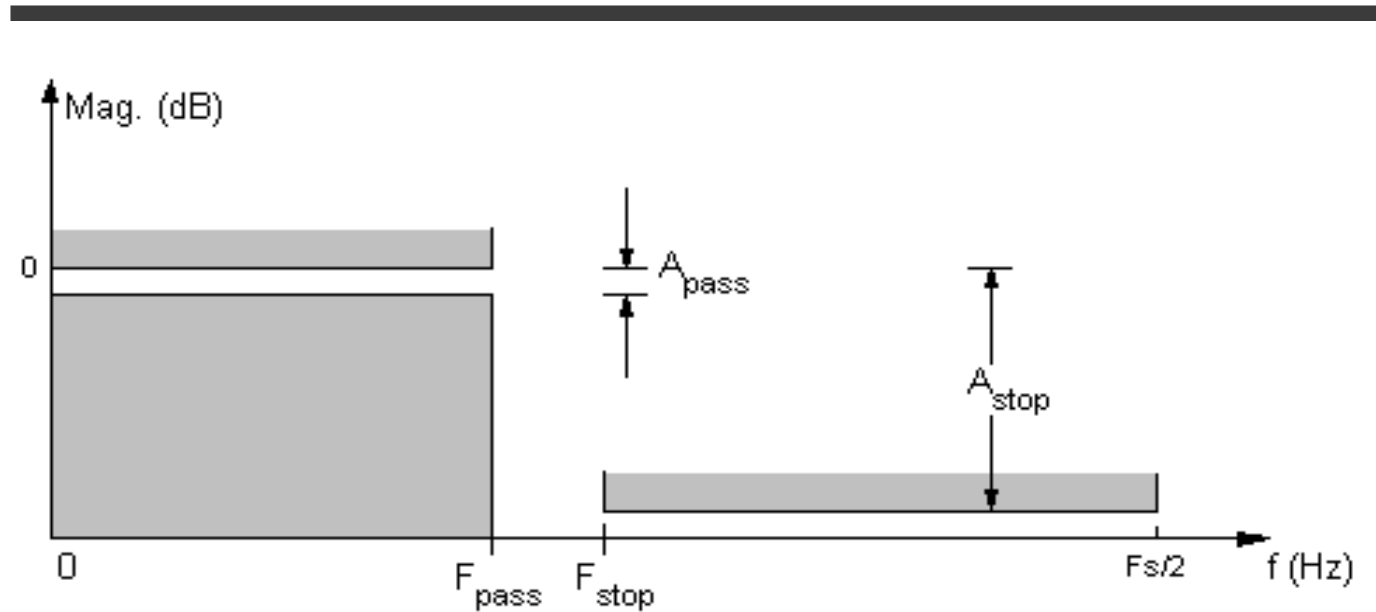
...



- But a maximum phase filter (the phase goes to  $-\pi$  at  $F_e/2$ ) because now, the zero lies outside the unit circle.

# IIR Filter Synthesis

- Starting from frequency specifications (here low pass filter):



- $F_{pass}$  : passband end frequency,
- $F_{stop}$  : stopband start frequency,
- $A_{pass}$  : maximum passband ripple,
- $A_{stop}$  : minimum stopband attenuation.

- Analog prototype with analog to digital transformation (bilinear transform) :
  - Digital to analog frequency specification transformation using prewarping
  - Analog filter prototype
  - Analog transfer function to digital transfer function transformation using bilinear transform.
- Direct digital method : Yule Walker
  - Try to find the recursive filter of order  $N$  which is as close as possible to the frequency specifications using the least square optimization method.

# IIR Filters Synthesis



- Characteristics frequencies ( $F_p$ ,  $F_a$ ) of the target specifications have to be warped.
- This warped specifications is used to compute an analog prototype using approximation functions :
  - Butterworth
  - Chebyshev I
  - Chebyshev II
  - Elliptic
- Then the analog prototype is transformed into a digital filter that matches target frequency specification thanks to Bilinear Transform (BT) (this cancels the warping introduced at the first step).



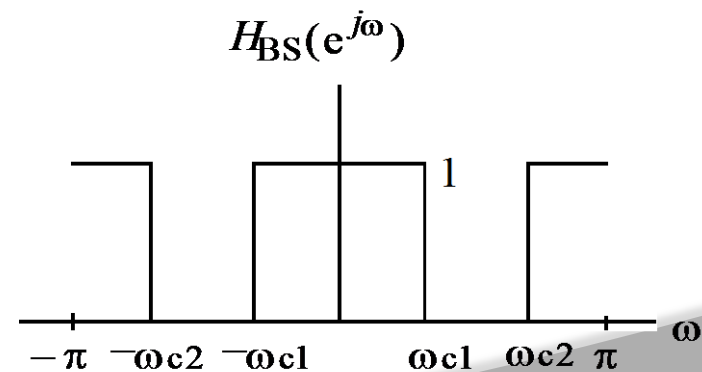
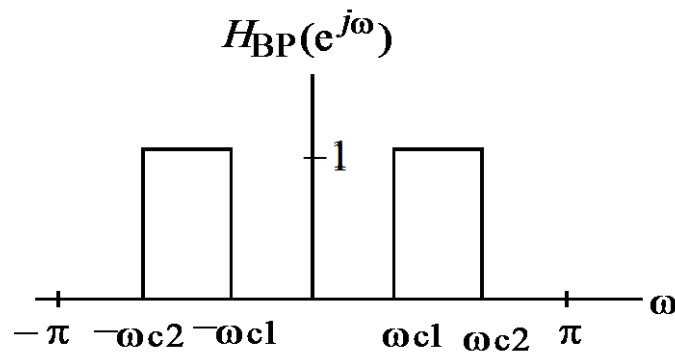
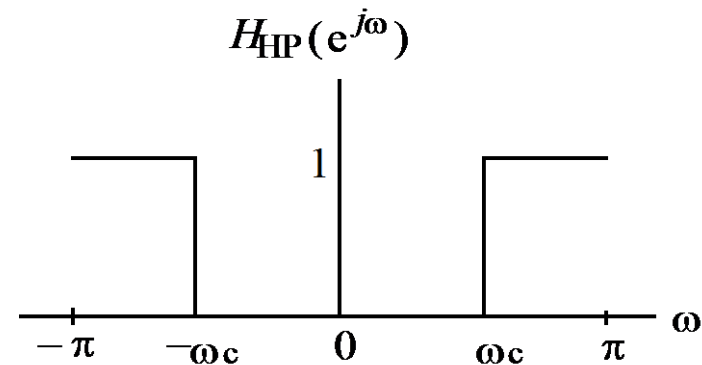
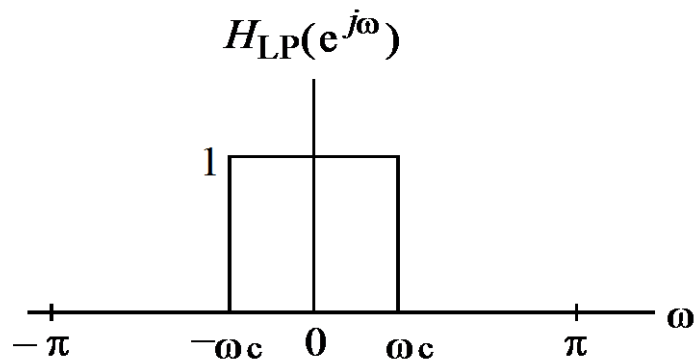
**UNIT- IV**  
**SYMMETRIC AND ANTISYMMETRIC FIR FILTERS**

CLO's	Course Learning Outcome
CLO12	Design of finite impulse response (FIR) filters for a given specification.
CLO13	Compare the characteristics of IIR and FIR filters.
CLO14	Design of infinite impulse response (IIR) filters for a given specification.



# Digital Filter Specifications

- Only the magnitude approximation problem
- Four basic types of ideal filters with magnitude responses as shown below (Piecewise flat)



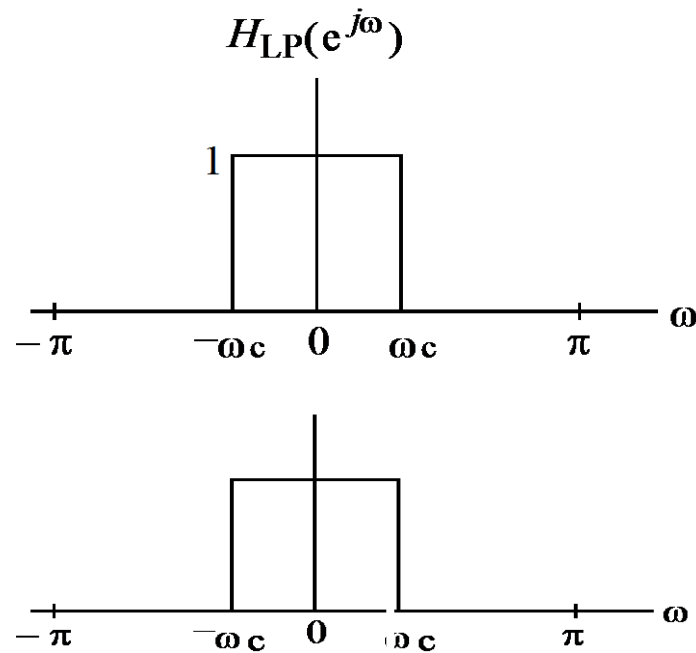
# Digital Filter Specifications

- ⦿ **These filters are unrealisable because (one of the following is sufficient)**
  - **their impulse responses infinitely long non-causal**
  - **Their amplitude responses cannot be equal to a constant over a band of frequencies**

**Another perspective that provides some understanding can be obtained by looking at the ideal amplitude squared.**

# Digital Filter Specifications

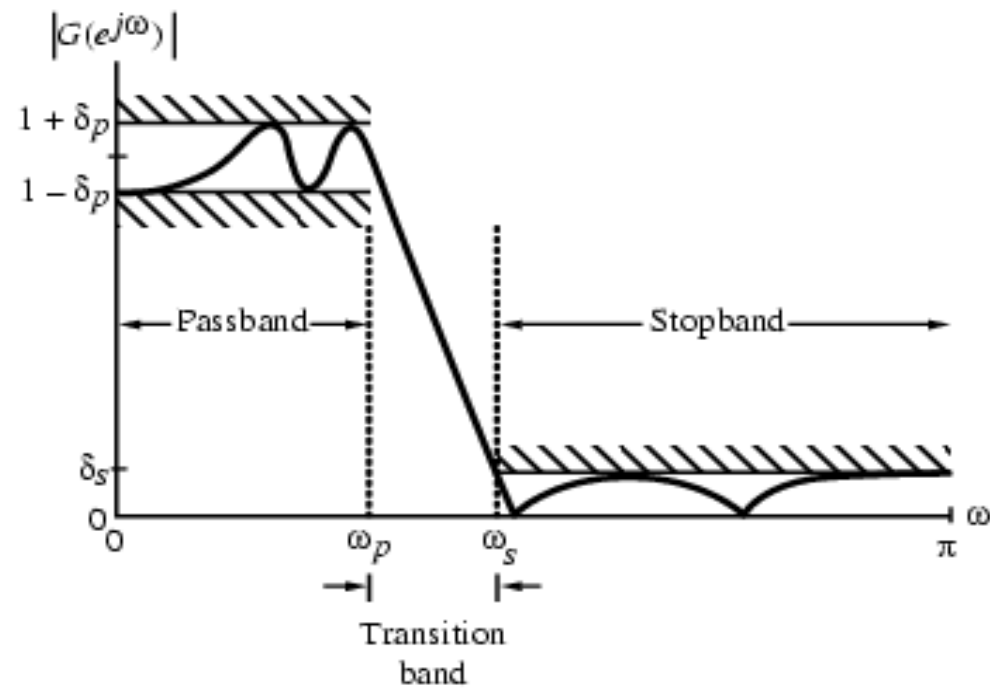
- Consider the ideal LP response squared (same as actual LP response)



- ⦿ The realizable squared amplitude response transfer function (and its differential) is continuous in  $\omega$  Such functions
  - if IIR can be infinite at point but around that point cannot be zero.
  - if FIR cannot be infinite anywhere.
- ⦿ Hence previous differential of ideal response is unrealizable

- ⦿ A realisable response would effectively need to have an approximation of the *delta functions* in the differential
- ⦿ This is a necessary condition

- For example the magnitude response of a digital lowpass filter may be given as indicated below



◎ **In the passband**  $0 \leq \omega \leq \omega_p$  **we require that**  
 $|G(e^{j\omega})| \cong 1$  **with a deviation**  $\pm \delta_p$

$$1 - \delta_p \leq |G(e^{j\omega})| \leq 1 + \delta_p, \quad |\omega| \leq \omega_p$$

◎ **In the stopband**  $\omega_s \leq \omega \leq \pi$  **we require that**  
 $|G(e^{j\omega})| \cong 0$  **with a deviation**  $\delta_s$

$$|G(e^{j\omega})| \leq \delta_s, \quad \omega_s \leq |\omega| \leq \pi$$

## Filter specification parameters

- ⦿  $\omega_p$  - **passband edge frequency**
- ⦿  $\omega_s$  - **stopband edge frequency**
- ⦿  $\delta_p$  - **peak ripple value in the passband**
- ⦿  $\delta_s$  - **peak ripple value in the stopband**



- Practical specifications are often given in terms of loss function (in dB)



- Peak passband ripple

$$G(\omega) = -20 \log_{10} |G(e^{j\omega})| \text{ dB}$$

- Minimum stopband attenuation

$$\alpha_p = -20 \log_{10} (1 - \delta_p) \text{ dB}$$

$$\alpha_s = -20 \log_{10} (\delta_s)$$

- ⦿ **In practice, passband edge frequency  $F_p$  and stopband edge frequency  $F_s$  are specified in Hz**
- ⦿ **For digital filter design, normalized bandedge frequencies need to be computed from specifications in Hz using**

$$\omega_p = \frac{\Omega_p}{F_T} = \frac{2\pi F_p}{F_T} = 2\pi F_p T$$

$$\omega_s = \frac{\Omega_s}{F_T} = \frac{2\pi F_s}{F_T} = 2\pi F_s T$$

- ⊙ **Example** - **Let** 7 kHz, 25 kHz, and 3 kHz, and
- kHz
- ⊙ **Then**

$$F_p = 7$$

$$F_s = 3$$

$$F_T = 25$$

$$\omega_p = \frac{2\pi(7 \times 10^3)}{25 \times 10^3} = 0.56\pi$$

$$\omega_s = \frac{2\pi(3 \times 10^3)}{25 \times 10^3} = 0.24\pi$$

# Selection of Filter Type

- ⦿ The transfer function  $H(z)$  meeting the specifications must be a causal transfer function

- ⦿ For IIR real digital filter the transfer function is a real rational function of  $z^{-1}$

$$H(z) = \frac{p_0 + p_1 z^{-1} + p_2 z^{-2} + \dots + p_M z^{-M}}{d_0 + d_1 z^{-1} + d_2 z^{-2} + \dots + d_N z^{-N}}$$

- ⦿  $H(z)$  must be stable and of lowest order  $N$  or  $M$  for reduced computational complexity

- ◎ **FIR real digital filter transfer function is a polynomial in  $z^{-1}$  (order  $N$ ) with real coefficients**

$$H(z) = \sum_{n=0}^N h[n] z^{-n}$$

- ◎ **For reduced computational complexity, degree  $N$  of  $H(z)$  must be as small as possible**
- ◎ **If a linear phase is desired then we must have:**
- ◎ **(More on this later)**

$$h[n] = \pm h[N - n]$$

# Selection of Filter Type

- ⦿ Advantages in using an FIR filter -
  - (1) Can be designed with exact linear phase
  - (2) Filter structure always stable with quantised coefficients
- ⦿ Disadvantages in using an FIR filter - Order of an FIR filter is considerably higher than that of an equivalent IIR filter meeting the same specifications; this leads to higher computational complexity for FIR

# Finite Impulse Response Filters

- ⦿ The transfer function is given by

$$H(z) = \sum_{n=0}^{N-1} h(n).z^{-n}$$

- ⦿ The length of Impulse Response is N
- ⦿ All poles are at  $z = 0$
- ⦿ Zeros can be placed anywhere on the z-plane

# FIR: Linear phase

- ⦿ Thus for linear phase the second term in the fundamental phase relationship must be identically zero for all index values.
- ⦿ Hence
- ⦿ 1) the maximum phase factor has zeros which are the inverses of the those of the minimum phase factor
- ⦿ 2) the phase response is linear with group delay (normalised) equal to the number of zeros outside the unit circle



## FIR: Linear phase

- It follows that zeros of linear phase FIR transfer functions not on the circumference of the unit circle occur in the form

$$[\rho_i e^{\pm j\theta_i}]^{\pm 1}$$

◎ **For Linear Phase t.f. (order  $N-1$ )**

◎ 
$$h(n) = \pm h(N-1-n)$$

◎ **so that for  $N$  even:**

$$\begin{aligned}
 H(z) &= \sum_{n=0}^{N/2-1} h(n).z^{-n} \pm \sum_{n=N/2}^{N-1} h(n).z^{-n} \\
 &= \sum_{n=0}^{N/2-1} h(n).z^{-n} \pm \sum_{n=0}^{N/2-1} h(N-1-n).z^{-(N-1-n)} \\
 &= \sum_{n=0}^{N/2-1} h(n) \left[ z^{-n} \pm z^{-m} \right] \quad m = N-1-n
 \end{aligned}$$

⊙ for  $N$  odd:

$$H(z) = \sum_{n=0}^{\frac{N-1}{2}-1} h(n) \cdot [z^{-n} \pm z^{-m}] + h\left(\frac{N-1}{2}\right) z^{-\left(\frac{N-1}{2}\right)}$$

⊙ I) On  $C: |z|=1$  we have for  $N$  even, and  
+ve sign

$$H(e^{j\omega T}) = e^{-j\omega T \left(\frac{N-1}{2}\right)} \cdot \sum_{n=0}^{\frac{N}{2}-1} 2h(n) \cdot \cos\left(\omega T \left(n - \frac{N-1}{2}\right)\right)$$

◎ II) While for –ve sign

$$H(e^{j\omega T}) = e^{-j\omega T \left( \frac{N-1}{2} \right)} \cdot \sum_{n=0}^{N/2-1} j2h(n) \cdot \sin \left( \omega T \left( n - \frac{N-1}{2} \right) \right)$$

◎ [Note: antisymmetric case adds  $\pi/2$  rads to phase, with discontinuity at  $\omega = 0$  ]

◎ III) For N odd with +ve sign

$$H(e^{j\omega T}) = e^{-j\omega T \left[ \frac{N-1}{2} \right]} \left\{ h \left( \frac{N-1}{2} \right) + \sum_{n=0}^{\frac{N-3}{2}} 2h(n) \cdot \cos \left[ \omega T \left( n - \frac{N-1}{2} \right) \right] \right\}$$

◎ **IV) While with a –ve sign**

$$H(e^{j\omega T}) = e^{-j\omega T \left[ \frac{N-1}{2} \right]} \left\{ \sum_{n=0}^{\frac{N-3}{2}} 2j \cdot h(n) \cdot \sin \left[ \omega T \left( n - \frac{N-1}{2} \right) \right] \right\}$$

◎ **[Notice that for the antisymmetric case to have linear phase we require**

$$h\left(\frac{N-1}{2}\right) = 0.$$

**The phase discontinuity is as for  $N$  even]**

- ◎ **The cases most commonly used in filter design are (I) and (III), for which the amplitude characteristic can be written as a polynomial in**

$$\cos \frac{\omega T}{2}$$

# Design of FIR filters: Windows

- (i) Start with ideal infinite duration  $\{h(n)\}$
- (ii) Truncate to finite length. (This produces unwanted ripples increasing in height near discontinuity.)
- (iii) Modify to  $\tilde{h}(n) = h(n).w(n)$   
Weight  $w(n)$  is the window

# Windows

## Commonly used windows

- Rectangular  $1 - \frac{2|n|}{N}$
- Bartlett  $1 + \cos\left(\frac{2\pi n}{N}\right)$
- Hann  $0.54 + 0.46 \cos\left(\frac{2\pi n}{N}\right)$
- Hamming  $0.54 + 0.46 \cos\left(\frac{2\pi n}{N}\right)$
- Blackman  $0.42 + 0.5 \cos\left(\frac{2\pi n}{N}\right) + 0.08 \cos\left(\frac{4\pi n}{N}\right)$
- Kaiser  $J_0 \left[ \beta \sqrt{1 - \left(\frac{2n}{N-1}\right)^2} \right] / J_0(\beta)$

$$|n| < \frac{N-1}{2}$$



# Kaiser window

## ◎ Kaiser window

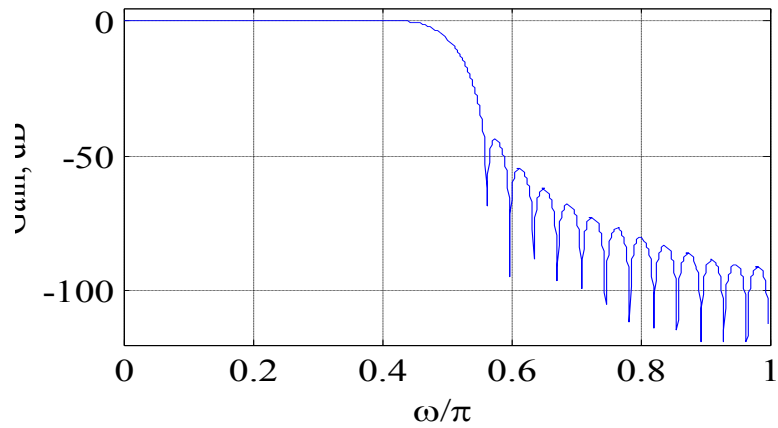
$\beta$	Transition width (Hz)	Min. stop attn dB
2.12	$1.5/N$	30
4.54	$2.9/N$	50
6.76	$4.3/N$	70
8.96	$5.7/N$	90

# Example

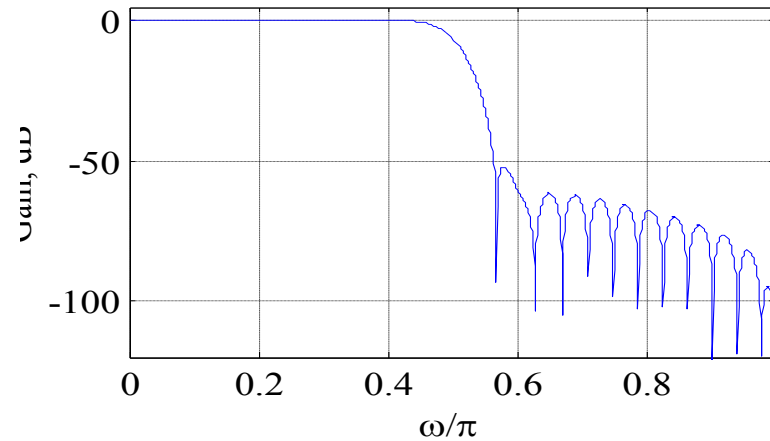
- Lowpass filter of length  $N$  and

$$\omega_c = \pi / 2$$

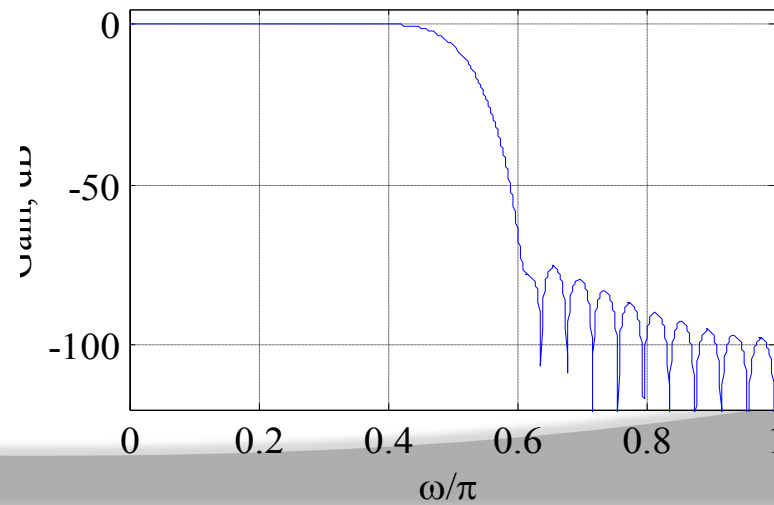
Lowpass Filter Designed Using Hann window



Lowpass Filter Designed Using Hamming window



Lowpass Filter Designed Using Blackman window



- In this approach we are given  $H(k)$  and need to find  $H(z)$
- This is an interpolation problem and the solution is given in the DFT part of the course

$$H(z) = \frac{1}{N} \sum_{k=0}^{N-1} H(k) \cdot \frac{1 - z^{-N}}{1 - e^{j\frac{2\pi}{N}k} \cdot z^{-1}}$$

- It has similar problems to the windowing approach

# Linear-Phase FIR Filter Design by Optimisation

- Amplitude response for all 4 types of linear-phase FIR filters can be expressed as

where

$$\check{H}(\omega) = Q(\omega)A(\omega)$$

$$Q(\omega) = \begin{cases} 1, & \text{for Type 1} \\ \cos(\omega/2), & \text{for Type 2} \\ \sin(\omega), & \text{for Type 3} \\ \sin(\omega/2), & \text{for Type 4} \end{cases}$$

## ⊙ Modified form of weighted error function

$$\begin{aligned} E(\omega) &= W(\omega)[Q(\omega)A(\omega) - D(\omega)] \\ &= W(\omega)Q(\omega)\left[A(\omega) - \frac{D(\omega)}{Q(\omega)}\right] \\ &= \tilde{W}(\omega)[A(\omega) - \tilde{D}(\omega)] \end{aligned}$$

where

$$\tilde{W}(\omega) = W(\omega)Q(\omega)$$

$$\tilde{D}(\omega) = D(\omega) / Q(\omega)$$

- **Optimisation Problem** - **Determine**  $\tilde{a}[k]$

**which minimise the peak absolute value**

**of**

$$E(\omega) = \tilde{W}(\omega) \left[ \sum_{k=0}^L \tilde{a}[k] \cos(\omega k) - \tilde{D}(\omega) \right]$$

**over the specified frequency bands**  $\omega \in R$

- **After**  $\tilde{a}[k]$  **has been determined,**  
**construct the original**  $A(e^{j\omega})$  **and hence**  
 **$h[n]$**

**Solution is obtained via the Alternation  
Theorem**

**The optimal solution has **equiripple**  
behaviour consistent with the total  
number of available parameters.**

**Parks and McClellan used the Remez  
algorithm to develop a procedure for  
designing linear FIR digital filters.**

# FIR Digital Filter Order Estimation

## Kaiser's Formula:

$$N \cong \frac{-20 \log_{10}(\sqrt{\delta_p \delta_s})}{14.6(\omega_s - \omega_p) / 2\pi}$$

- ⦿ **ie  $N$  is inversely proportional to transition band width and not on transition band location**



# FIR Digital Filter Order Estimation

## ◎ Hermann-Rabiner-Chan's Formula:

$$N \cong \frac{D_{\infty}(\delta_p, \delta_s) - F(\delta_p, \delta_s)[(\omega_s - \omega_p) / 2\pi]^2}{(\omega_s - \omega_p) / 2\pi}$$

where

$$D_{\infty}(\delta_p, \delta_s) = [a_1(\log_{10} \delta_p)^2 + a_2(\log_{10} \delta_p) + a_3] \log_{10} \delta_s \\ + [a_4(\log_{10} \delta_p)^2 + a_5(\log_{10} \delta_p) + a_6]$$

with  $F(\delta_p, \delta_s) = b_1 + b_2[\log_{10} \delta_p - \log_{10} \delta_s]$

$$a_1 = 0.005309, a_2 = 0.07114, a_3 = -0.4761$$

$$a_4 = 0.00266, a_5 = 0.5941, a_6 = 0.4278$$

$$b_1 = 11.01217, b_2 = 0.51244$$

# FIR Digital Filter Order Estimation

- ⊙ **Formula valid for  $\delta_p \geq \delta_s$**
- ⊙ **For  $\delta_p < \delta_s$ , formula to be used is obtained by interchanging  $\delta_p$  and  $\delta_s$**
- ⊙ **Both formulae provide only an estimate of the required filter order  $N$**
- ⊙ **If specifications are not met, increase filter order until they are met**

# FIR Digital Filter Order Estimation

- ◎ **Fred Harris' guide:**

$$N \cong \frac{A}{20(\omega_s - \omega_p) / 2\pi}$$

where **A** is the attenuation in dB

- ◎ **Then add about 10% to it**



# MODULE– V

## APPLICATIONS OF DSP

CLO's

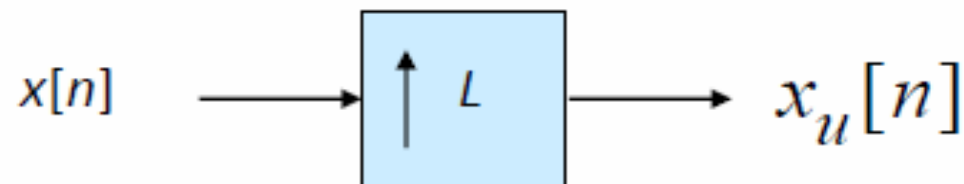
Course Learning Outcome

- |       |   |
|-------|---|
| CLO15 | Understand the tradeoffs between normal and multi rate DSP techniques and finite length word effects. |
| CLO16 | Understand the signal interpolation and decimation, and explain their operation                       |
| CLO17 | Explain the cause of limit cycles in the implementation of IIR filters.                               |

# Up-sampler

## Time-Domain Characterization

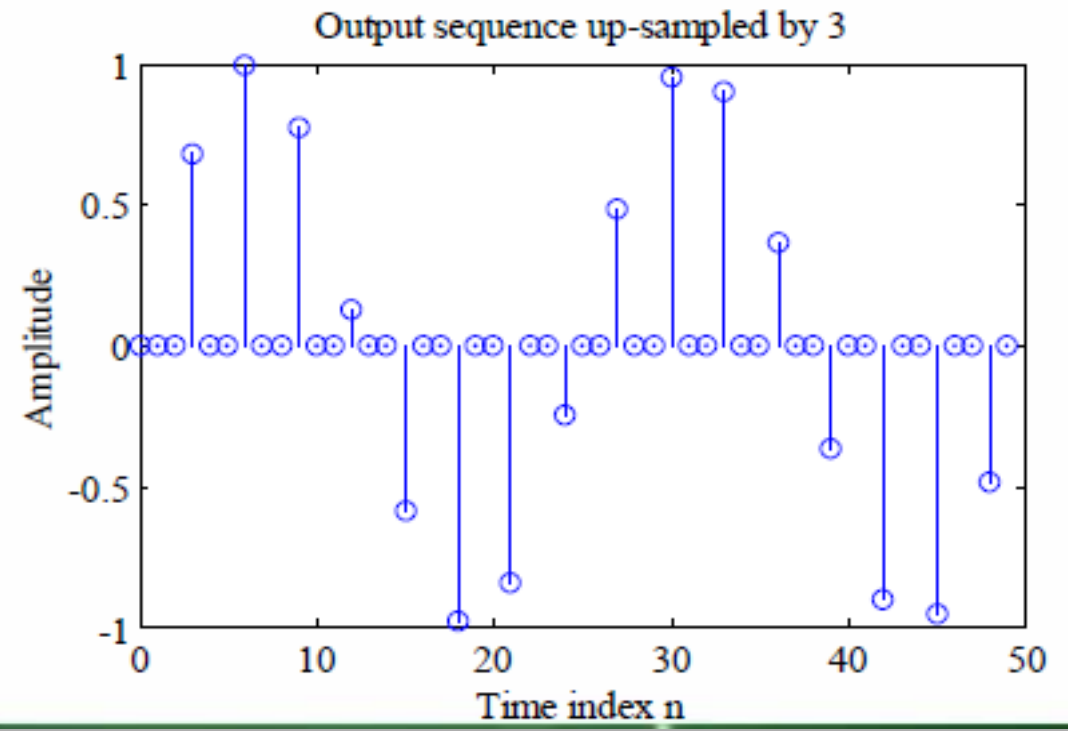
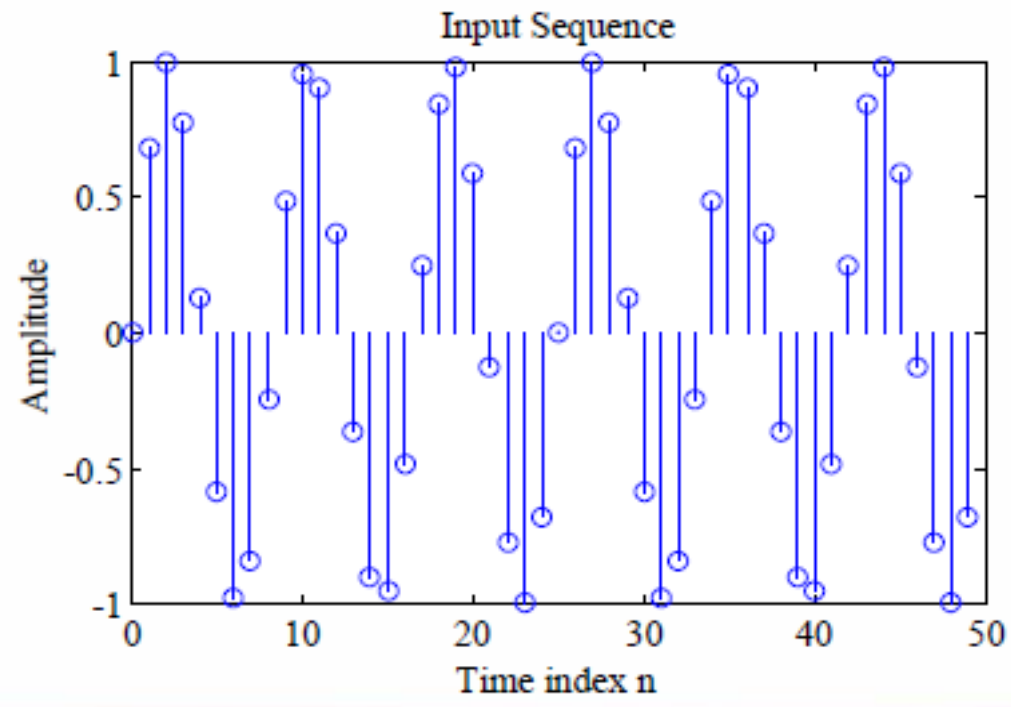
- An up-sampler with an *up-sampling factor*  $L$ , where  $L$  is a positive integer, develops an output sequence  $x_u[n]$  with a sampling rate that is  $L$  times larger than that of the input sequence  $x[n]$
- Block-diagram representation



- Up-sampling operation is implemented by inserting  $L-1$  equidistant zero-valued samples between two consecutive samples of  $x[n]$
- Input-output relation

$$x_u[n] = \begin{cases} x[n/L], & n = 0, \pm L, \pm 2L, \dots \\ 0, & \text{otherwise} \end{cases}$$

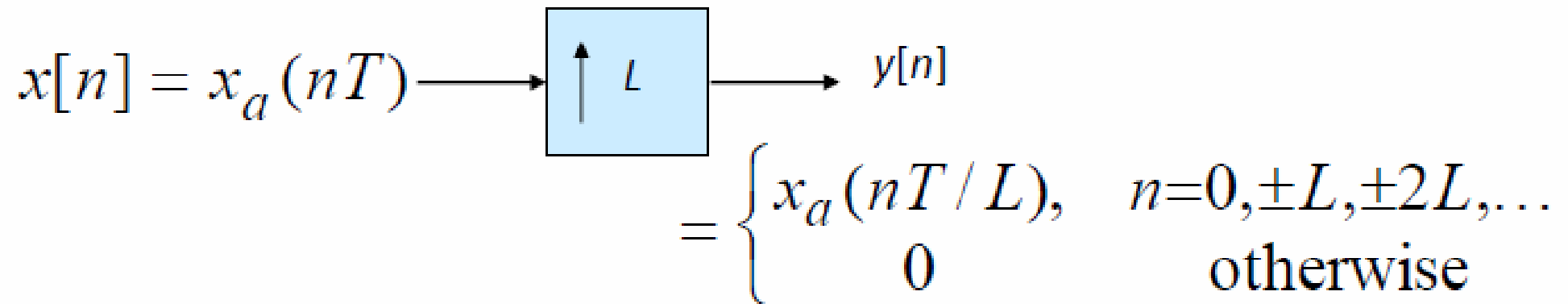
- Figure below shows the up-sampling by a factor of 3 of a sinusoidal sequence with a frequency of 0.12 Hz obtained using Program 10\_1





- In practice, the zero-valued samples inserted by the up-sampler are replaced with appropriate nonzero values using some type of filtering process
- Process is called *interpolation* and will be discussed later

- Figure below shows explicitly the time-dimensions for the up-sampler



Input sampling frequency

$$F_T = \frac{1}{T}$$

Output sampling frequency

$$F'_T = LF_T = \frac{1}{T'}$$

# Up-Sampler

## *Frequency-Domain Characterization*

- Consider first a factor-of-2 up-sampler whose input-output relation in the time-domain is given by

$$x_u[n] = \begin{cases} x[n/2], & n = 0, \pm 2, \pm 4, \dots \\ 0, & \text{otherwise} \end{cases}$$

- In terms of the  $z$ -transform, the input-output relation is then given by

$$\begin{aligned}
 X_u(z) &= \sum_{n=-\infty}^{\infty} x_u[n] z^{-n} = \sum_{\substack{n=-\infty \\ n \text{ even}}}^{\infty} x[n/2] z^{-n} \\
 &= \sum_{m=-\infty}^{\infty} x[m] z^{-2m} = X(z^2)
 \end{aligned}$$

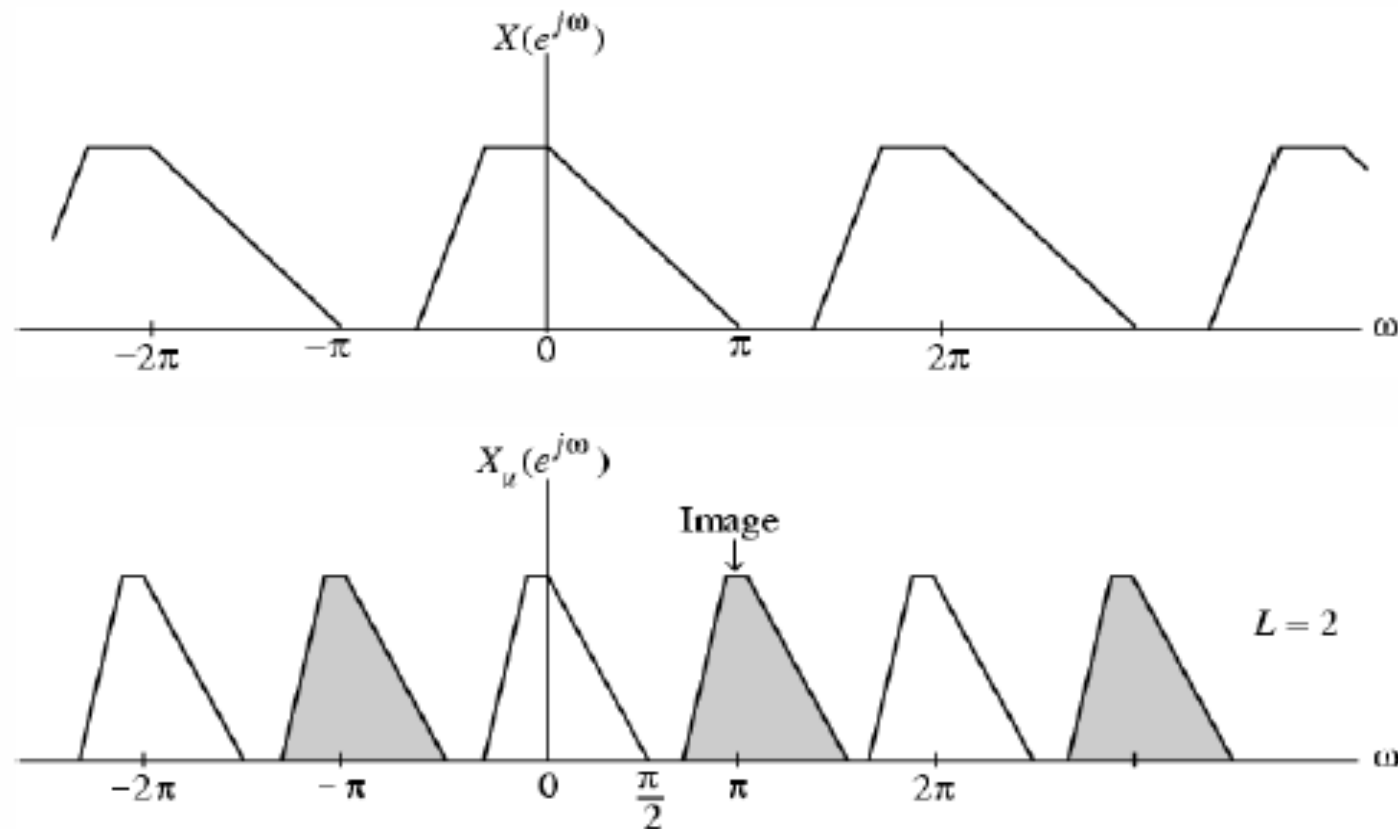
- In a similar manner, we can show that for a *factor-of-L up-sampler*

$$X_u(z) = X(z^L)$$

- On the unit circle, for  $z = e^{j\omega}$ , the input-output relation is given by

$$X_u(e^{j\omega}) = X(e^{j\omega L})$$

- Figure below shows the relation between  $X(e^{j\omega})$  and  $X_u(e^{j\omega})$  for  $L = 2$  in the case of a typical sequence  $x[n]$



- As can be seen, a factor-of-2 sampling rate expansion leads to a compression of  $X(e^{j\omega})$  by a factor of 2 and a 2-fold repetition in the baseband  $[0, 2\pi]$
- This process is called *imaging* as we get an additional “*image*” of the input spectrum

- Similarly in the case of a factor-of- $L$  sampling rate expansion, there will be  $L-1$  additional images of the input spectrum in the baseband
- Lowpass filtering of  $x_u[n]$  removes the images and in effect “fills in” the zero-valued samples in  $x_u[n]$  with interpolated sample values

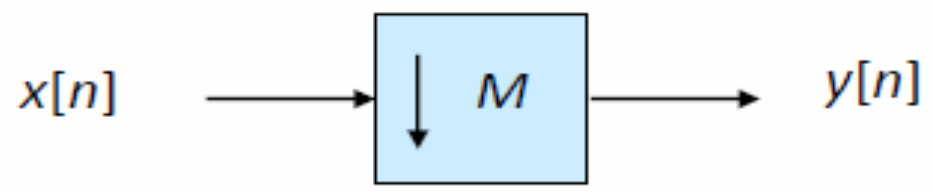
$$x_u[n]$$



# Down Sampler

## Time-Domain Characterization

- An down-sampler with a *down-sampling factor*  $M$ , where  $M$  is a positive integer, develops an output sequence  $y[n]$  with a sampling rate that is  $(1/M)$ -th of that of the input sequence  $x[n]$
- Block-diagram representation

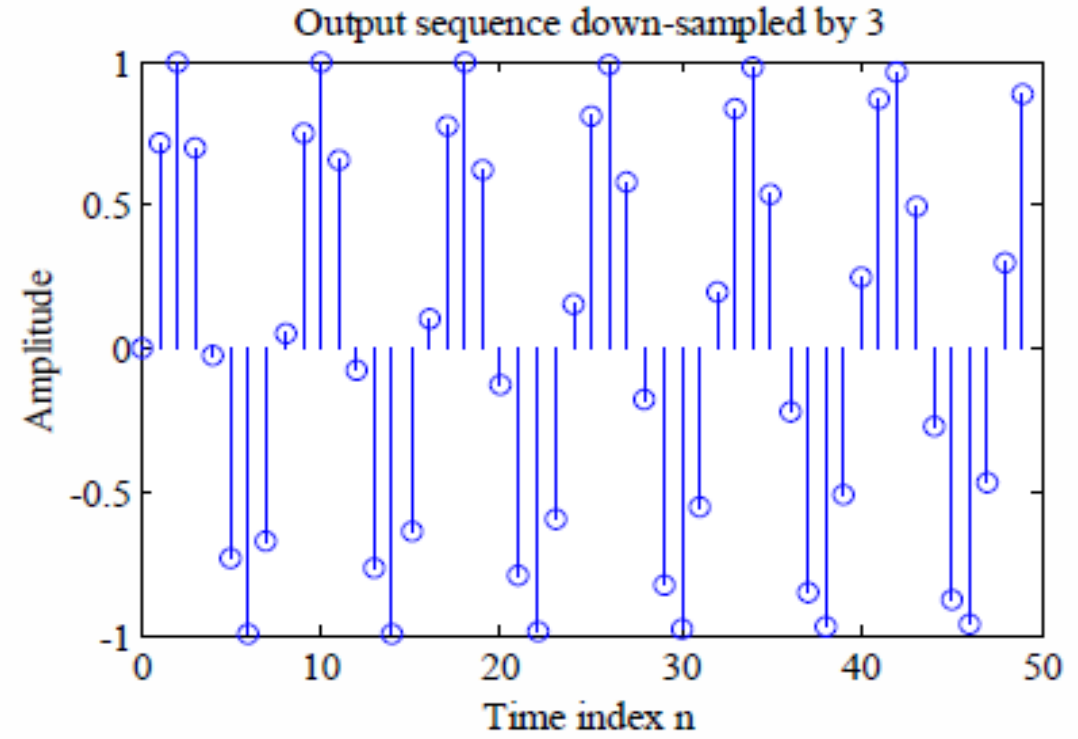
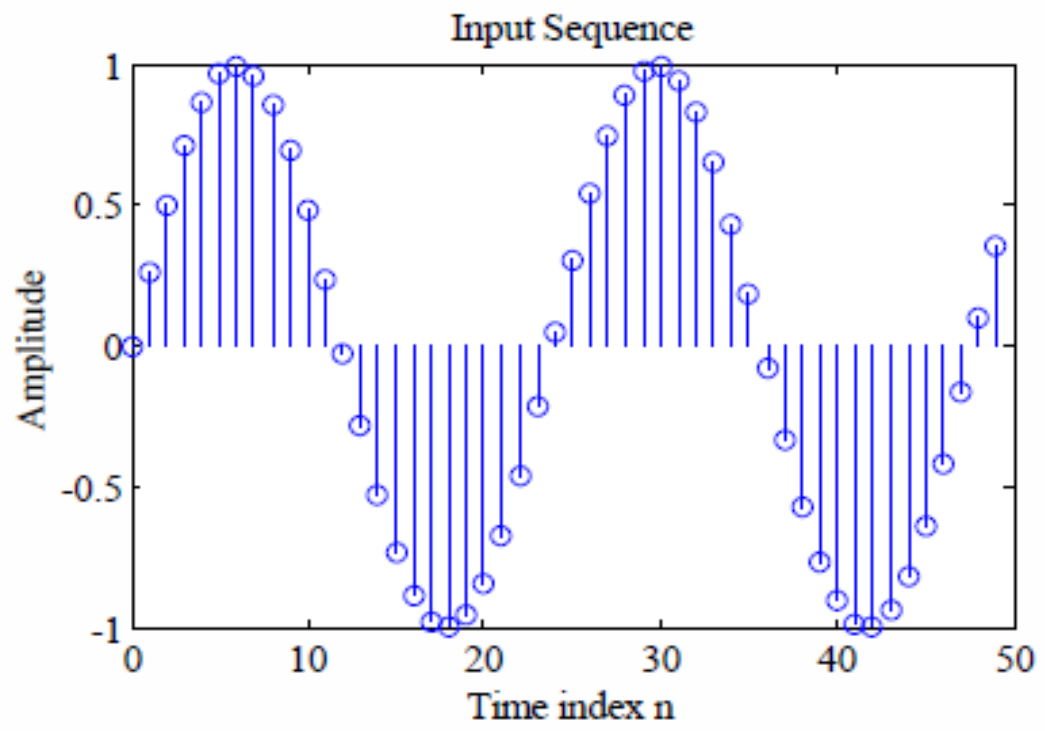


- Down-sampling operation is implemented by keeping every  $M$ -th sample of  $x[n]$  and removing in-between samples to generate  $y[nM-1]$

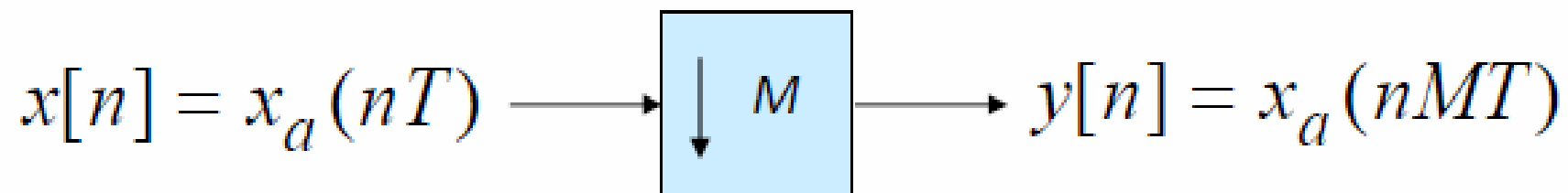
- Input-output relation

$$y[n] = x[nM]$$

- Figure below shows the down-sampling by a factor of 3 of a sinusoidal sequence of frequency 0.042 Hz obtained using Program 10\_2



- Figure below shows explicitly the time-dimensions for the down-sampler



Input sampling frequency

$$F_T = \frac{1}{T}$$

Output sampling frequency

$$F_T' = \frac{F_T}{M} = \frac{1}{T'}$$

## Frequency-Domain Characterization

- Applying the  $z$ -transform to the input-output relation of a factor-of- $M$  down-sampler

we get

$$y[n] = x[Mn]$$

$$Y(z) = \sum_{n=-\infty}^{\infty} x[Mn] z^{-n}$$

- The expression on the right-hand side cannot be directly expressed in terms of  $X(z)$

- To get around this problem, define a new sequence  $x_{\text{int}}[n]$ :

$$x_{\text{int}}[n] = \begin{cases} x[n], & n = 0, \pm M, \pm 2M, \dots \\ 0, & \text{otherwise} \end{cases}$$

- Then

$$\begin{aligned} Y(z) &= \sum_{n=-\infty}^{\infty} x[Mn]z^{-n} = \sum_{n=-\infty}^{\infty} x_{\text{int}}[Mn]z^{-n} \\ &= \sum_{k=-\infty}^{\infty} x_{\text{int}}[k]z^{-k/M} = X_{\text{int}}(z^{1/M}) \end{aligned}$$

- Now,  $x_{\text{int}}[n]$  can be formally related to  $x[n]$  through

$$x_{\text{int}}[n] = c[n] \cdot x[n]$$

where

$$c[n] = \begin{cases} 1, & n = 0, \pm M, \pm 2M, \dots \\ 0, & \text{otherwise} \end{cases}$$

- A convenient representation of  $c[n]$  is given by

$$c[n] = \frac{1}{M} \sum_{k=0}^{M-1} W_M^{kn}$$

where

$$W_M = e^{-j2\pi / M}$$

- Consider a factor-of- $M$  down-sampler defined by  $y[n] = x[nM]$
- Its output  $y_1[n]$  for an input  $x_1[n] = x[n - n_0]$  then given by

$$y_1[n] = x_1[Mn] = x[Mn - n_0]$$

- From the input-output relation of the down-sampler we obtain

$$\begin{aligned} y[n - n_0] &= x[M(n - n_0)] \\ &= x[Mn - Mn_0] \neq y_1[n] \end{aligned}$$



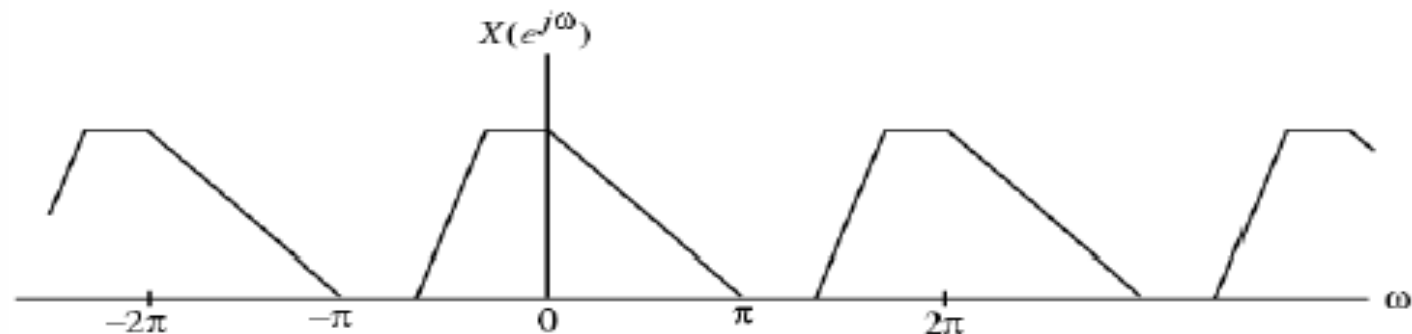
- Taking the  $z$ -transform of  $x_{\text{int}}[n] = c[n] \cdot x[n]$  and making use of

$$c[n] = \frac{1}{M} \sum_{k=0}^{M-1} W_M^{kn}$$

we arrive at

$$\begin{aligned} X_{\text{int}}(z) &= \sum_{n=-\infty}^{\infty} c[n]x[n]z^{-n} = \frac{1}{M} \sum_{n=-\infty}^{\infty} \left( \sum_{k=0}^{M-1} W_M^{kn} \right) x[n]z^{-n} \\ &= \frac{1}{M} \sum_{k=0}^{M-1} \left( \sum_{n=-\infty}^{\infty} x[n]W_M^{kn} z^{-n} \right) = \frac{1}{M} \sum_{k=0}^{M-1} X(zW_M^{-k}) \end{aligned}$$

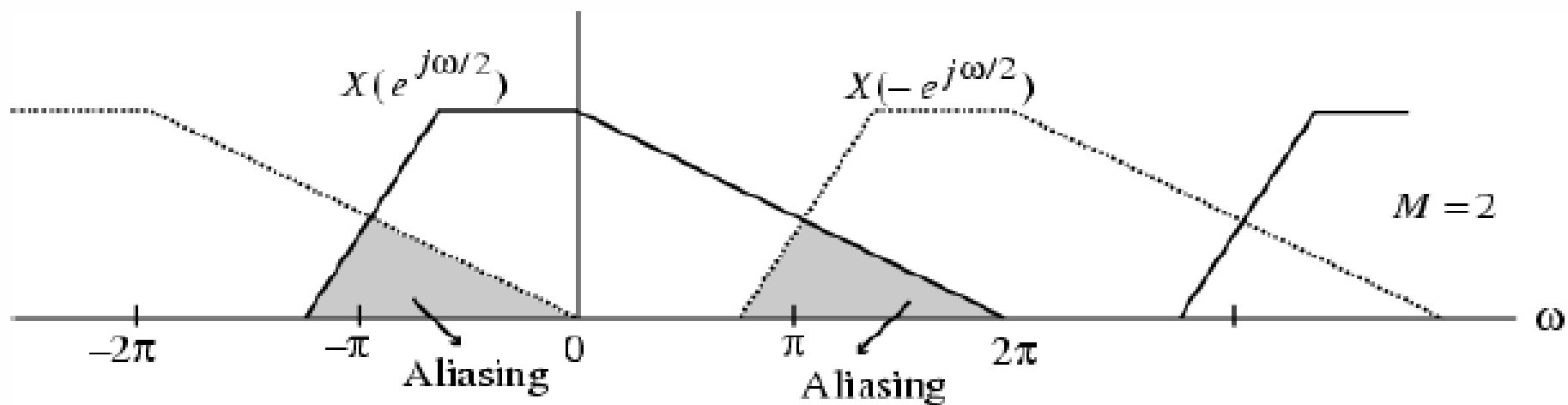
- Consider a factor-of-2 down-sampler with an input  $x[n]$  whose spectrum is as shown below



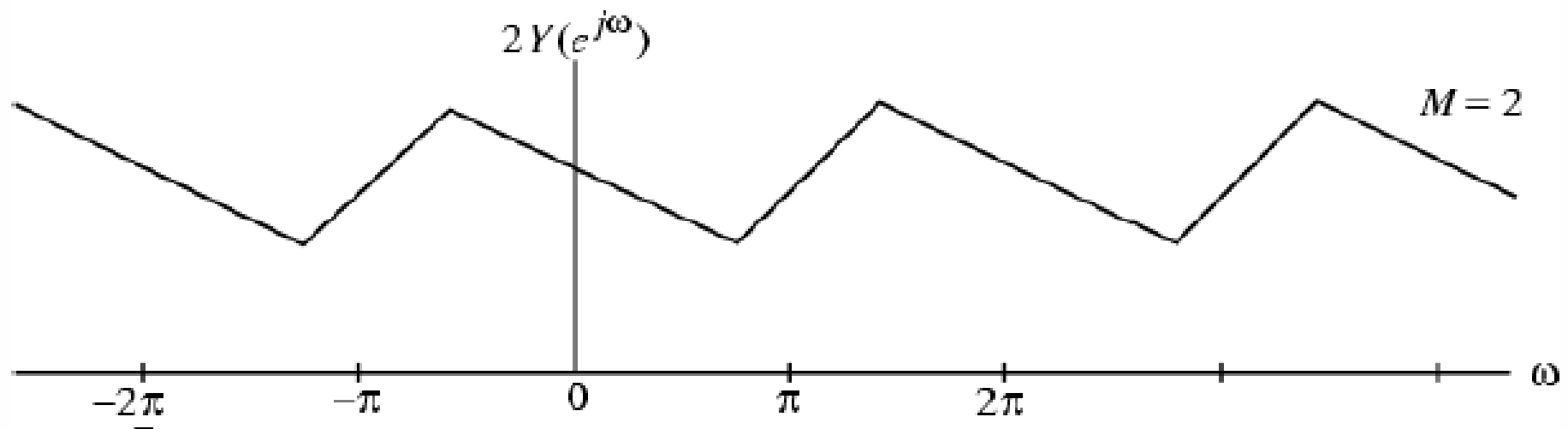
- The DTFTs of the output and the input sequences of this down-sampler are then related as

$$Y(e^{j\omega}) = \frac{1}{2} \{X(e^{j\omega/2}) + X(-e^{j\omega/2})\}$$

- Now  $X(-e^{j\omega/2}) = X(e^{j(\omega-2\pi)/2})$  implying that the second term  $X(-e^{j\omega/2})$  in the previous equation is simply obtained by shifting the first term  $X(e^{j\omega/2})$  to the right by an amount  $2\pi$  as shown below



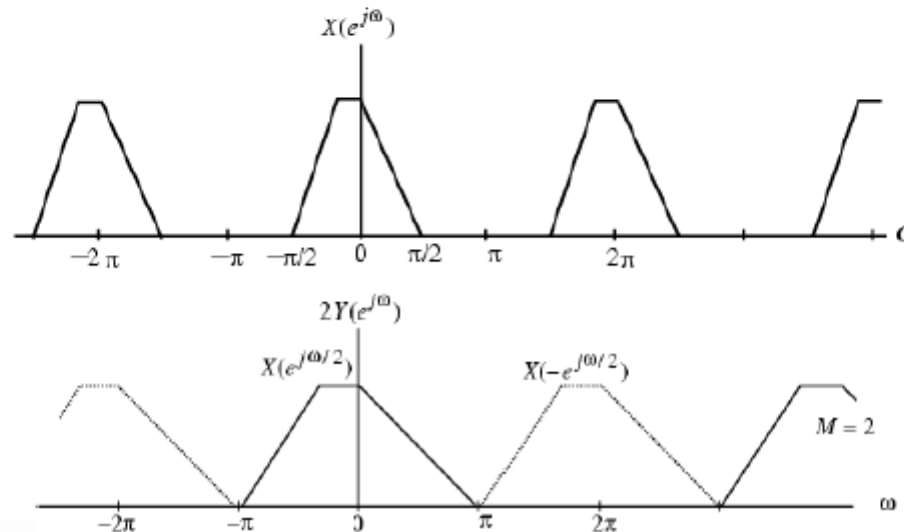
- The plots of the two terms have an overlap, and hence, in general, the original “*shape*” of  $X(e^{j\omega})$  is lost when  $x[n]$  is down-sampled as indicated below



- Aliasing is absent if and only if

$X(e^{j\omega}) = 0$  for  $|\omega| \geq \pi/M$   
as shown below for  $M = 2$

$$X(e^{j\omega}) = 0 \text{ for } |\omega| \geq \pi/2$$



# Basic Sampling Rate Alteration Deices

- Sampling periods have not been explicitly shown in the block-diagram representations of the up-sampler and the down-sampler
- This is for simplicity and the fact that the *mathematical theory of multirate systems* can be understood without bringing the sampling period  $T$  or the sampling frequency into the picture

$$F_T$$

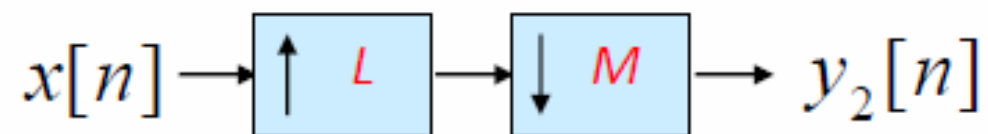
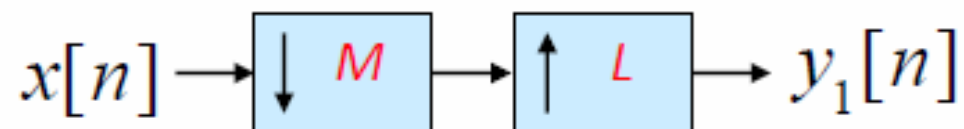
- The *up-sampler* and the *down-sampler* are *linear* but *time-varying discrete-time systems*
- We illustrate the time-varying property of a down-sampler
- The time-varying property of an up-sampler can be proved in a similar manner

# Cascade Equivalences

- A complex *multirate system* is formed by an interconnection of the up-sampler, the down-sampler, and the components of an LTI digital filter
- In many applications these devices appear in a cascade form
- An interchange of the positions of the branches in a cascade often can lead to a computationally efficient realization



- To implement a *fractional change* in the *sampling rate* we need to employ a cascade of an up-sampler and a down-sampler
- Consider the two cascade connections shown below



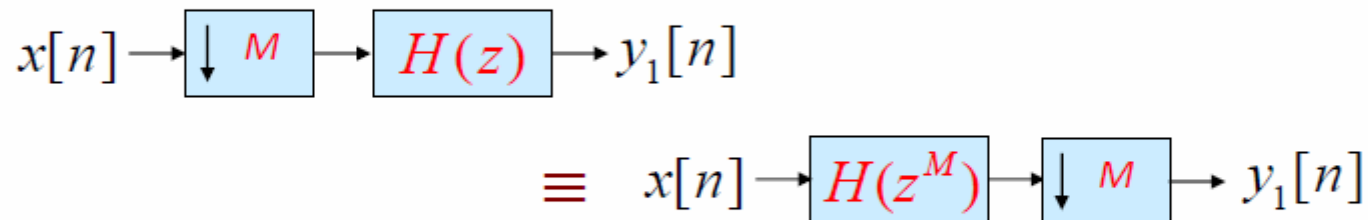
- A cascade of a factor-of- $M$  down-sampler and a factor-of- $L$  up-sampler is interchangeable with no change in the input-output relation:

*if and only if  $M$  and  $L$  are relatively prime, i.e.,  $M$  and  $L$  do not have any common factor that is an integer  $k > 1$*

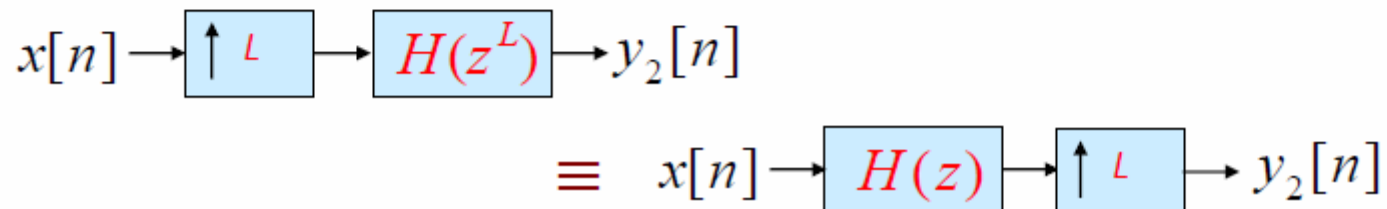
$y_1[n] = y_2[n]$

- Two other cascade equivalences are shown below

*Cascade equivalence #1*



*Cascade equivalence #2*



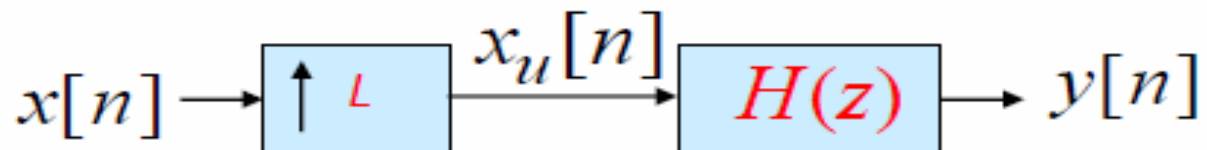
# Filters in Sampling Rate Alteration Systems

- From the *sampling theorem* it is known that a the sampling rate of a critically sampled discrete-time signal with a spectrum occupying the full Nyquist range cannot be reduced any further since such a reduction will introduce aliasing
- Hence, the bandwidth of a critically sampled signal must be reduced by *lowpass filtering* before its sampling rate is reduced by a down-sampler

- Likewise, the zero-valued samples introduced by an up-sampler must be interpolated to more appropriate values for an effective sampling rate increase
- We shall show next that this interpolation can be achieved simply by digital lowpass filtering
- We now develop the frequency response specifications of these lowpass filters

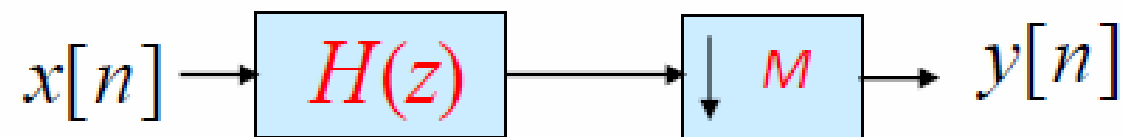
# Filter Specifications

- Since up-sampling causes periodic repetition of the basic spectrum, the unwanted images in the spectra of the up-sampled signal  $x_u[n]$  must be removed by using a lowpass filter  $H(z)$ , called the ***interpolation filter***, as indicated below



- The above system is called an ***interpolator***

- On the other hand, prior to down-sampling, the signal  $v[n]$  should be bandlimited to  $|\omega| < \pi / M$  by means of a lowpass filter, called the *decimation filter*, as indicated below to avoid aliasing caused by down-sampling



- The above system is called a *decimator*

# Interpolation Filter Specifications

- Assume  $x[n]$  has been obtained by sampling a continuous-time signal  $x_a(t)$  at the Nyquist rate
- If  $X_a(j\Omega)$  and  $X(e^{j\omega})$  denote the Fourier transforms of  $x_a(t)$  and  $x[n]$ , respectively, then it can be shown

$$X(e^{j\omega}) = \frac{1}{T_0} \sum_{k=-\infty}^{\infty} X_a\left(\frac{j\omega - j2\pi k}{T_0}\right)$$

- where  $T_0$  is the sampling period



- Since the sampling is being performed at the *Nyquist rate*, there is no overlap between the shifted spectras  $X(j\omega/T_0)$
- If we instead sample  $x_a(t)$  at a much higher rate  $T = L \cdot T_0$  yielding  $y[n]$ , its Fourier transform  $Y(e^{j\omega})$  is related to  $X_a(j\Omega)$  through

$$Y(e^{j\omega}) = \frac{1}{T} \sum_{k=-\infty}^{\infty} X_a\left(\frac{j\omega - j2\pi k}{T}\right) = \frac{L}{T_0} \sum_{k=-\infty}^{\infty} X_a\left(\frac{j\omega - j2\pi k}{T_0/L}\right)$$

- In practice, a transition band is provided to ensure the realizability and stability of the lowpass interpolation filter  $H(z)$
- Hence, the desired lowpass filter should have a stopband edge at  $\omega_s$  and a passband edge close to  $\omega_p = \pi - \omega_s$  to reduce the distortion of the spectrum of  $X[n]$

- If  $\omega_c$  is the highest frequency that needs to be preserved in  $x[n]$ , then

- Summarizing the specifications of the lowpass interpolation filter are thus given by

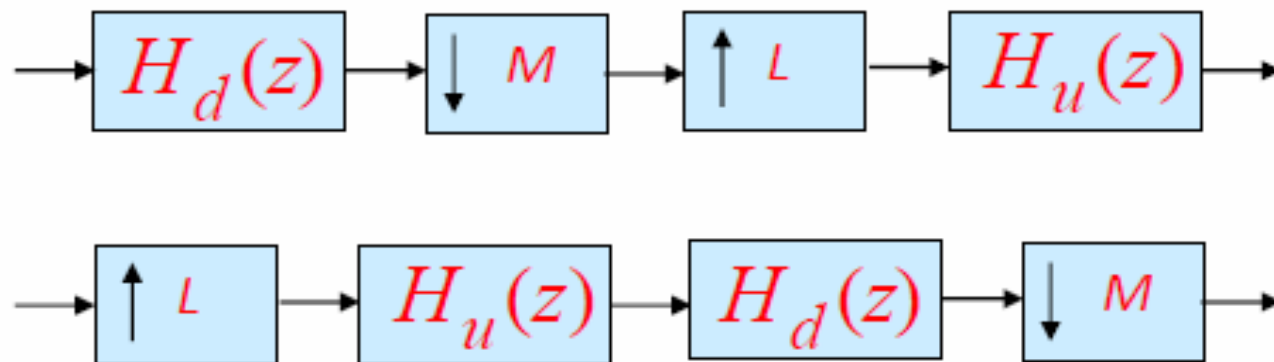
$$|H(e^{j\omega})| = \begin{cases} L, & |\omega| \leq \omega_c / L \\ 0, & \pi / L \leq |\omega| \leq \pi \end{cases}$$

# Decimation Filter Specifications

- In a similar manner, we can develop the specifications for the lowpass decimation filter that are given by

$$|H(e^{j\omega})| = \begin{cases} 1, & |\omega| \leq \omega_c / M \\ 0, & \pi / M \leq |\omega| \leq \pi \end{cases}$$

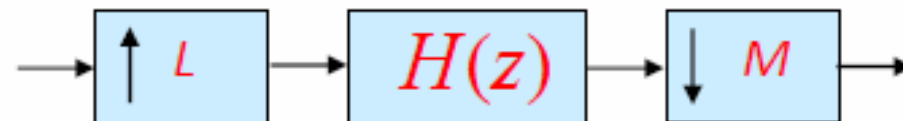
- There are two possible such cascade connections as indicated below



- The second scheme is more computationally efficient since only one of the filters,  $H_u(z)$  or  $H_d(z)$ , is adequate to serve both the interpolation and the decimation filter

- Hence, the desired configuration for the fractional sampling rate alteration is as indicated below where the lowpass filter  $H(z)$  has a stopband edge frequency given by

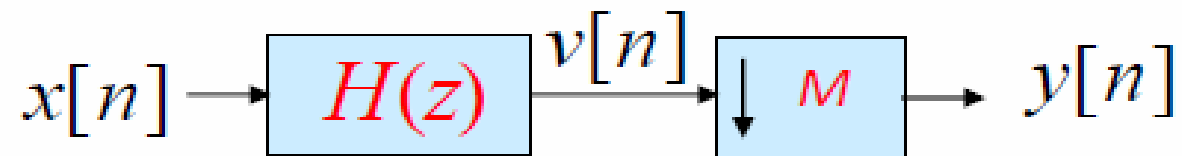
$$\omega_s = \min\left(\frac{\pi}{L}, \frac{\pi}{M}\right)$$



# Computational Requirements

- The lowpass decimation or interpolation filter can be designed either as an FIR or an IIR digital filter
- In the case of single-rate digital signal processing, *IIR digital filters* are, in general, computationally more efficient than equivalent FIR digital filters, and are therefore preferred where computational cost needs to be minimized

- This issue is not quite the same in the case of multirate digital signal processing
- To illustrate this point further, consider the factor-of- $M$  decimator shown below



- If the decimation filter  $H(z)$  is an FIR filter of length  $N$  implemented in a direct form, then

$$v[n] = \sum_{m=0}^{N-1} h[m]x[n-m]$$



- Now, the down-sampler keeps only every  $M$ -th sample of  $v[n]$  at its output
- Hence, it is sufficient to compute  $v[n]$  only for values of  $n$  that are multiples of  $M$  and skip the computations of in-between samples
- This leads to a factor of  $M$  savings in the computational complexity

- Now assume  $H(z)$  to be an IIR filter of order  $K$  with a transfer function

where

$$\frac{V(z)}{X(z)} = H(z) = \frac{P(z)}{D(z)}$$

$$P(z) = \sum_{n=0}^K p_n z^{-n}$$

$$D(z) = 1 + \sum_{n=1}^K d_n z^{-n}$$

- Its direct form implementation is given by

$$w[n] = -d_1 w[n-1] - d_2 w[n-2] - \dots - d_K w[n-K] + x[n]$$

- $v[n] = p_0 w[n] + p_1 w[n-1] + \dots + p_K w[n-K]$   
 Since  $v[n]$  is being down-sampled, it is sufficient to compute  $v[n]$  only for values of  $n$  that are integer multiples of  $M$

- However, the intermediate signal  $w[n]$  must be computed for all values of  $n$
- For example, in the computation of
$$v[M] = p_0 w[M] + p_1 w[M-1] + \dots + p_K w[M-K]$$
 $K+1$  successive values of  $w[n]$  are still required
- As a result, the savings in the computation in this case is going to be less than a factor of  $M$

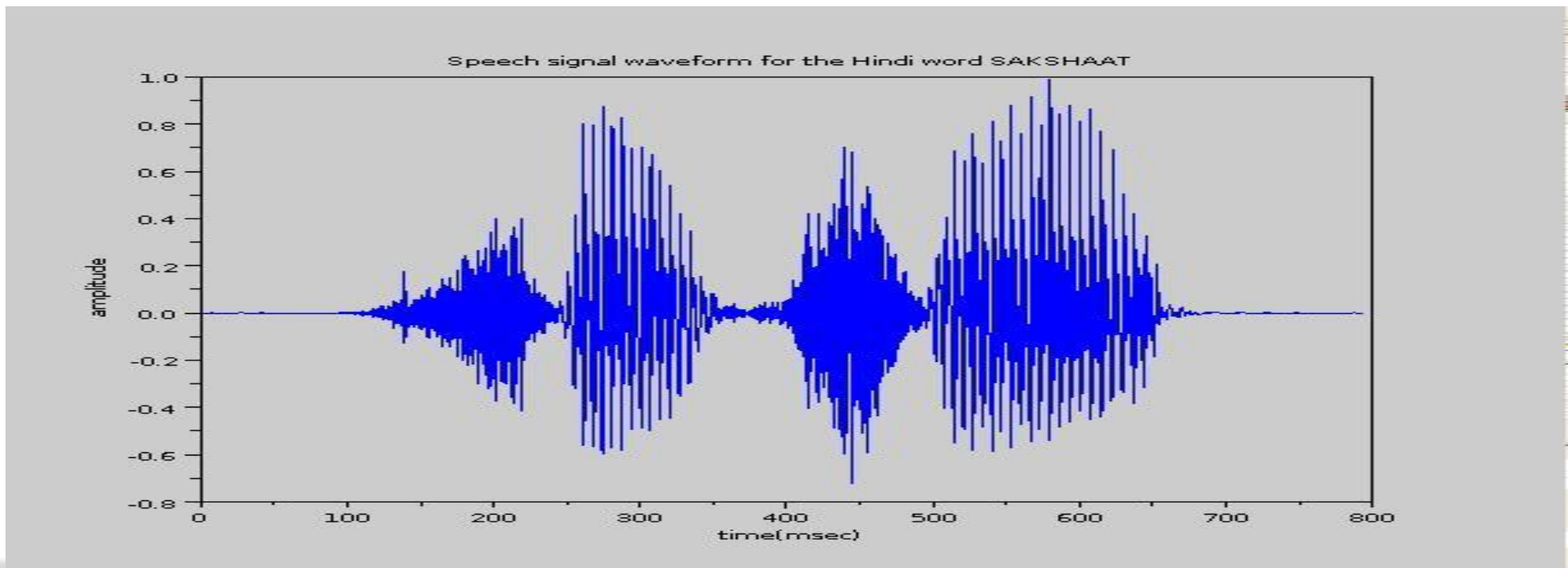
- For the case of interpolator design, very similar arguments hold
- If  $H(z)$  is an FIR interpolation filter, then the computational savings is by a factor of  $L$  (since  $v[n]$  has  $L-1$  zeros between its consecutive nonzero samples)
- On the other hand, computational savings is significantly less with IIR filters

# Digital Signal Processing And Its Benefits

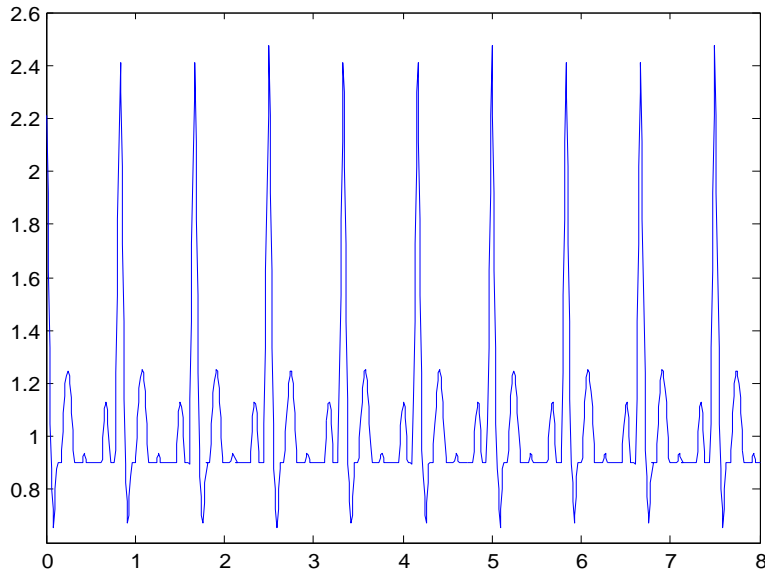
By a **signal** we mean any variable that carries or contains some kind of information that can be conveyed, displayed or manipulated.

Examples of signals of particular interest are:

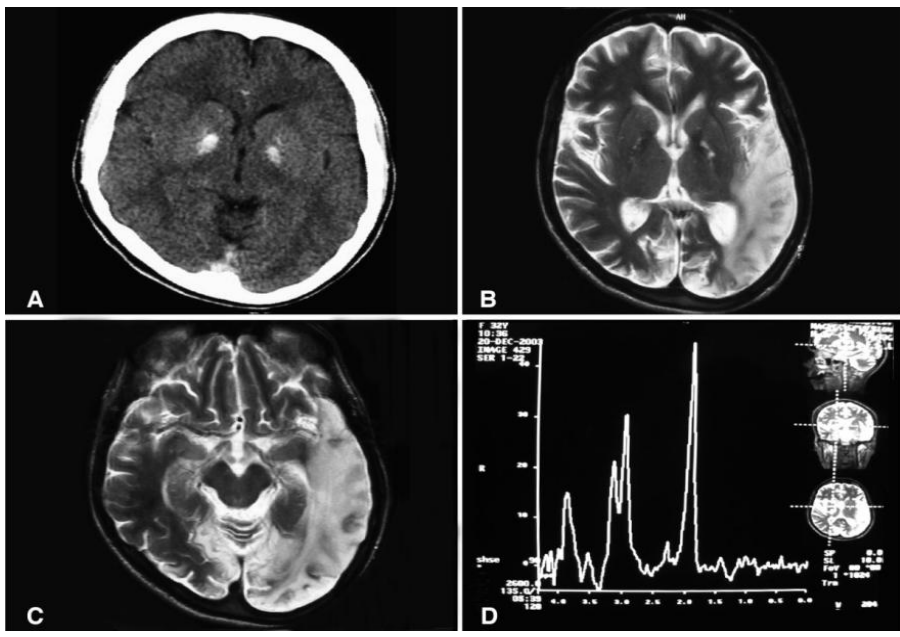
- speech, is encountered in telephony, radio, and everyday life



# biomedical signals, (heart signals, brain signals)



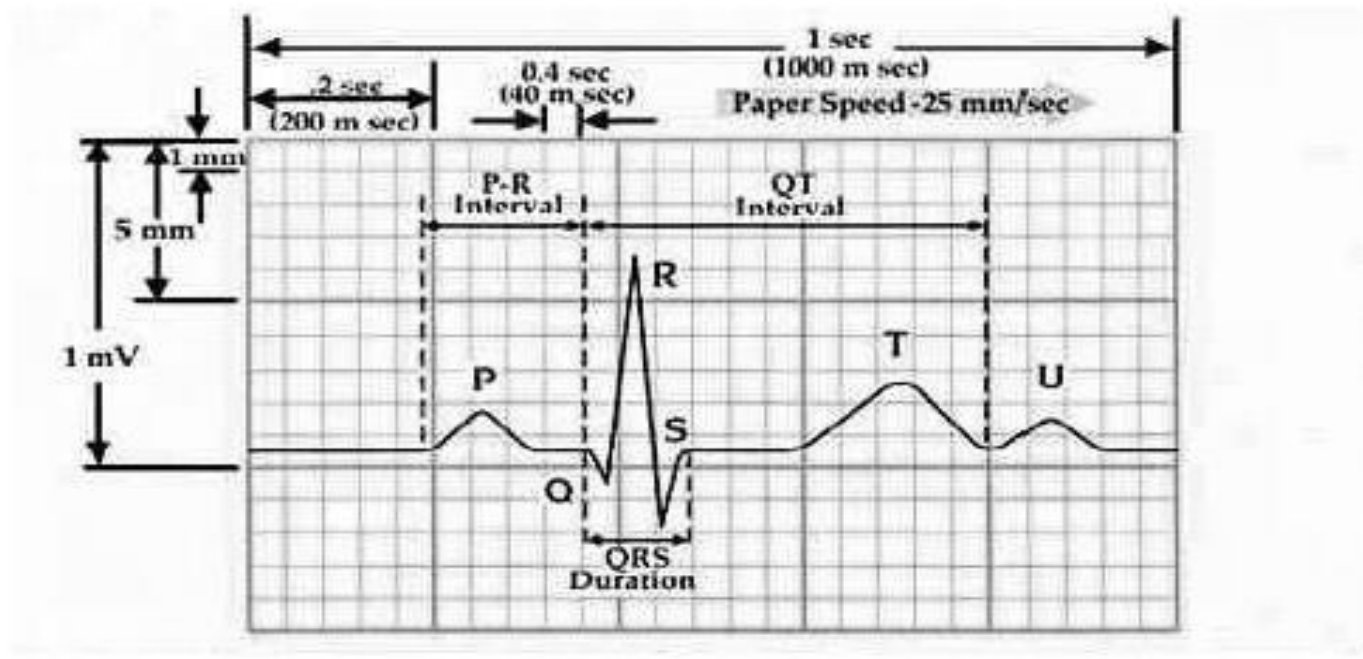
⊙ ECG



Tomography

## Significant features of ECG waveform

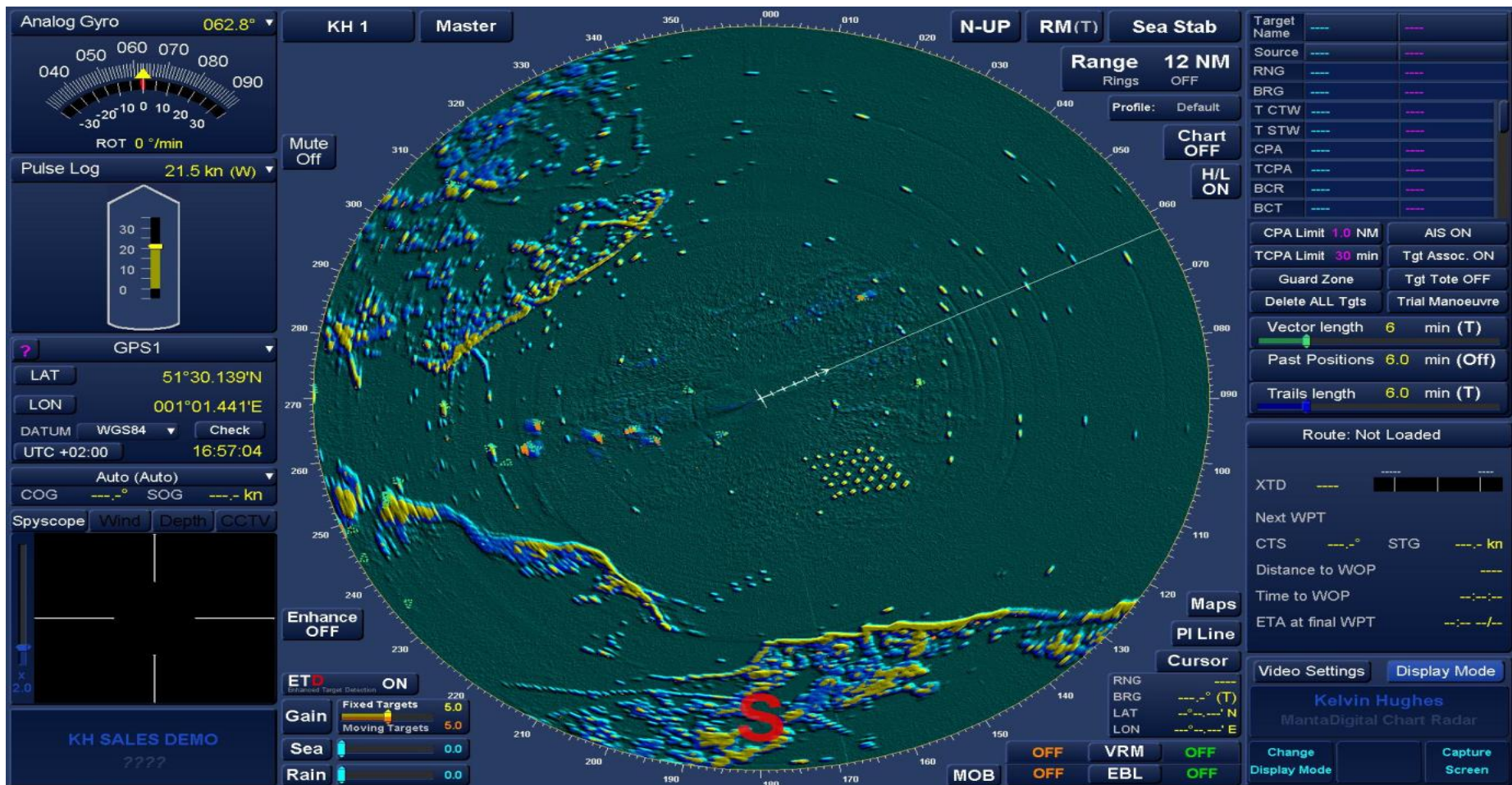
- A typical scalar electrocardiographic lead is shown in Fig. 1, where the significant features of the waveform are the P, Q, R, S, and T waves, the duration of each wave, and certain time intervals such as the P-R, S-T, and Q-T intervals.





# Sound and music, as reproduced by the compact disc player

- Video and image,
- Radar signals, which are used to determine the range and bearing of distant targets



- **Most of the signals in our environment are analog such as sound, temperature and light**



- To process these signals with a computer, we must:
  1. convert the analog signals into electrical signals, e.g., using a transducer such as a microphone to convert sound into electrical signal
  2. digitize these signals, or convert them from analog to digital, using an ADC (Analog to Digital Converter)

# Steps in Digital Signal Processing

- Analog input signal is filtered to be a band-limited signal by an input lowpass filter
- Signal is then sampled and quantized by an ADC
- Digital signal is processed by a digital circuit, often a computer or a digital signal processor
- Processed digital signal is then converted back to an analog signal by a DAC
- The resulting step waveform is converted to a smooth signal by a reconstruction filter called an anti-imaging filter

# Why do we need DSPs

- ◎ **DSP operations require a lot of multiplying and adding operations of the form:**

$$A = B * C + D$$

- ◎ **This simple equation involves a multiply and an add operation**
- ◎ **The multiply instruction of a GPP is very slow compared with the add instruction**
- ◎ **Motorola 68000 microprocessor uses**
  - 10 clock cycles for add**
  - 74 clock cycles for multiply**

- **Digital signal processors can perform the multiply and the add operation in just one clock cycle**
  - **Most DSPs have a specialized instruction that causes them to multiply, add and save the result in a single cycle**
  - **This instruction is called a MAC (Multiply, Add, and Accumulate)**

## Attraction of DSP comes from key advantages such as :

- \* **Guaranteed accuracy:** (accuracy is only determined by the number of bits used)
- \* **Perfect Reproducibility:** Identical performance from unit to unit
  - ie. A digital recording can be copied or reproduced several times with no loss in signal quality
- \* **No drift in performance with temperature and age**
- \* **Uses advances in semiconductor technology to achieve:**
  - (i) smaller size
  - (ii) lower cost
  - (iii) low power consumption
  - (iv) higher operating speed
- \* **Greater flexibility:** Reprogrammable , no need to modify the hardware
- \* **Superior performance**
  - ie. linear phase response can be achieved
  - complex adaptive filtering becomes possible

# Disadvantages of DSP

## \* Speed and Cost

**DSP techniques are limited to signals with relatively low bandwidths  
DSP designs can be expensive, especially when large bandwidth signals  
are involved.**

**ADC or DACs are either too expensive or do not have sufficient  
resolution for wide bandwidth applications.**

**\* DSP designs can be time consuming plus need the necessary resources  
(software etc)**

## \* Finite word-length problems

**If only a limited number of bits is used due to economic considerations  
serious degradation in system performance may result.**



- The use of finite precision arithmetic makes it necessary to quantize filter calculations by rounding or truncation.
- Roundoff noise is that error in the filter output that results from rounding or truncating calculations within the filter.
- As the name implies, this error looks like low-level noise at the filter output



# Application Areas

## Image Processing

Pattern recognition  
Robotic vision  
Image enhancement  
Facsimile  
animation

## Instrumentation/Control

spectrum analysis  
noise reduction  
data compression  
position and rate  
control

## Speech/Audio

speech recognition  
speech synthesis  
text to speech  
digital audio  
equalization

## Military

secure communications  
radar processing  
sonar processing  
missile guidance

## Telecommunications

Echo cancellation  
Adaptive equalization  
ADPCM trans-coders  
Spread spectrum  
Video conferencing

## Biomedical

patient monitoring  
scanners  
EEG brain mappers  
ECG Analysis  
X-Ray storage/enhancement

## Consumer applications

cellular mobile phones  
UMTS  
digital television  
digital cameras  
internet phone etc.