Digital Signal Processing

BTEC-502

UNIT-1

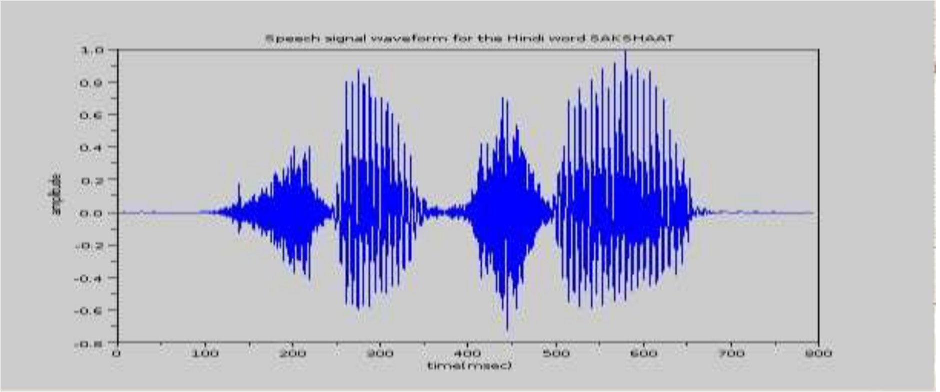
- Introduction to DSP
- Advantages & Disadvantages of DSP
- Applications of DSP
- Basic Concept of Signal & Systems
- Convolution & Correlation
- DFT & its Properties
- Linear Filtering Methods
- FFT Algorithms

Digital Signal Processing

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



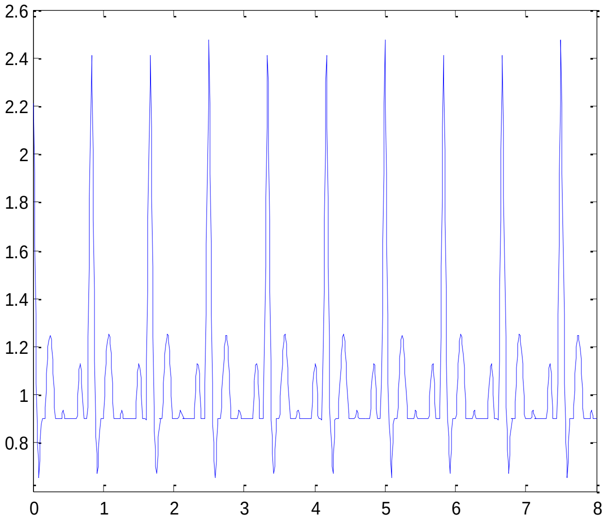
- Biomedical signals:-   (heart signals, brain signals)

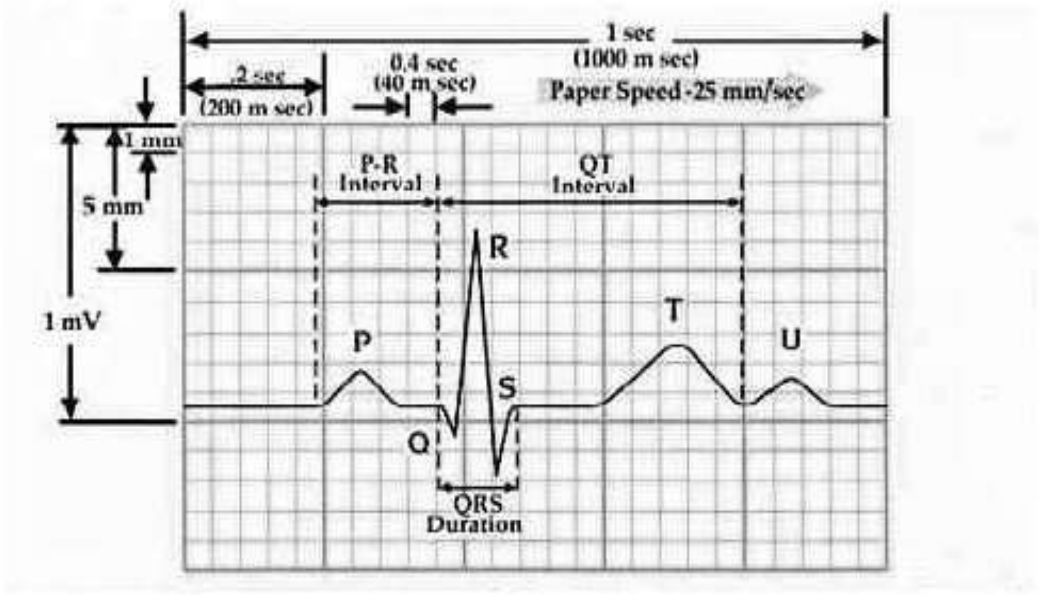
- Sound and music, as reproduced by the compact disc player, Video and image,

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

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.





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

- To processes 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 low pass 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 to 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.

- Round off 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

1.Image Processing

2.Instrumentation/Control

3.Speech/Audio

4.Military

5. Telecommunications

6.Biomedical

7.Consumer applications

Introduction
to
Signal And Systems

1. Continuous time signals (CT signals)
2. Discrete time signals (DT signals)
3. Elementary Signals
4. Classification of CT and DT
5. Description of continuous time and discrete time systems

Signal:

A Function of one or more independent Variables Which contains some information is called as signal
For Eg. Music, Speech, Picture & Video

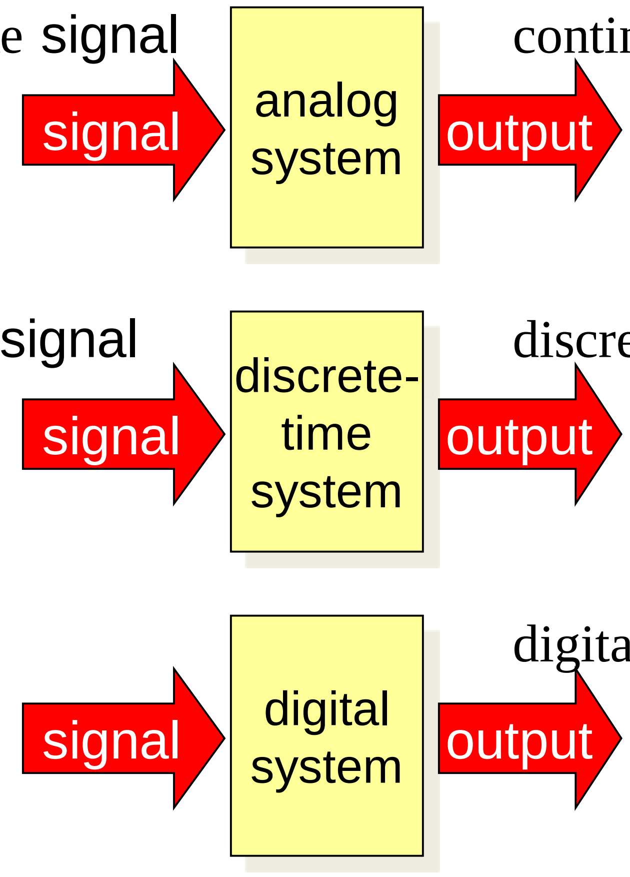
Signal

1. One Dimensional
2. Multi Dimensional

Classification Of Signals

1. Continuous Time Signal
2. Discrete Time Signal

Signal Process Systems



continuous-time signal

signal

analog system

output

continuous-time signal

discrete-time signal

signal

discrete-time system

output

discrete-time signal

digital signal

signal

digital system

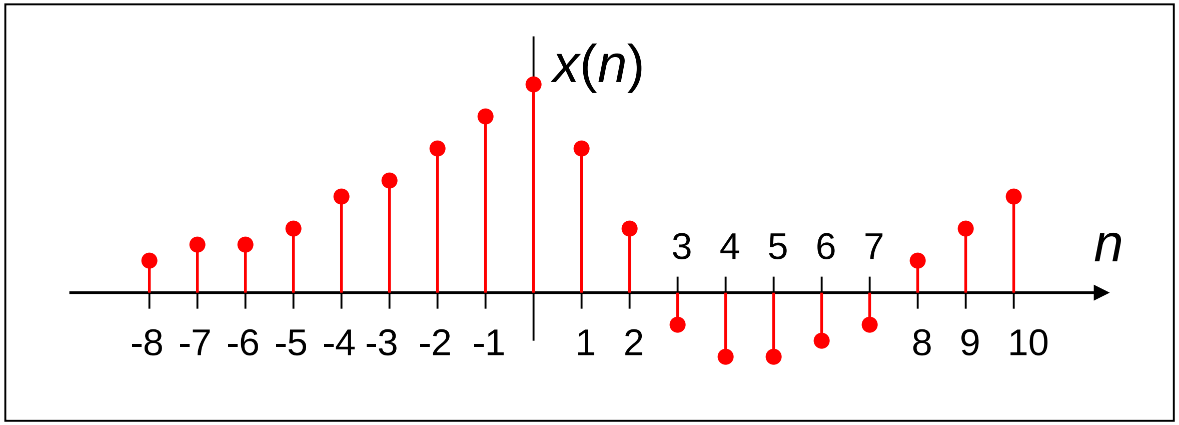
output

digital signal

Representation by a Sequence

- Discrete-time system theory
  - Concerned with processing signals that are represented by *sequences*.

$$x = \{x(n)\}, \qquad -\infty < n < \infty$$



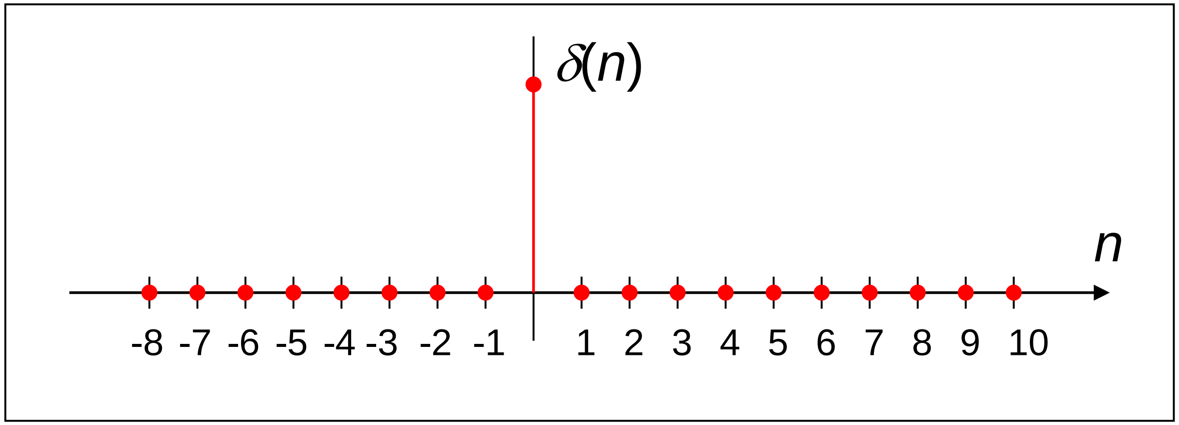
Elementary Signals

- Unit Impulse Signal
- Unit Step Signal
- Exponential Signal
- Sinusoidal Signal

Important Signals

- Unit-sample sequence δ(*n*)
- Sometime call δ(*n*)
  - a *discrete-time impulse*; or
  - an *impulse*

$$\delta(n) = \begin{cases} 1 & n = 0 \\ 0 & n \neq 0 \end{cases}$$



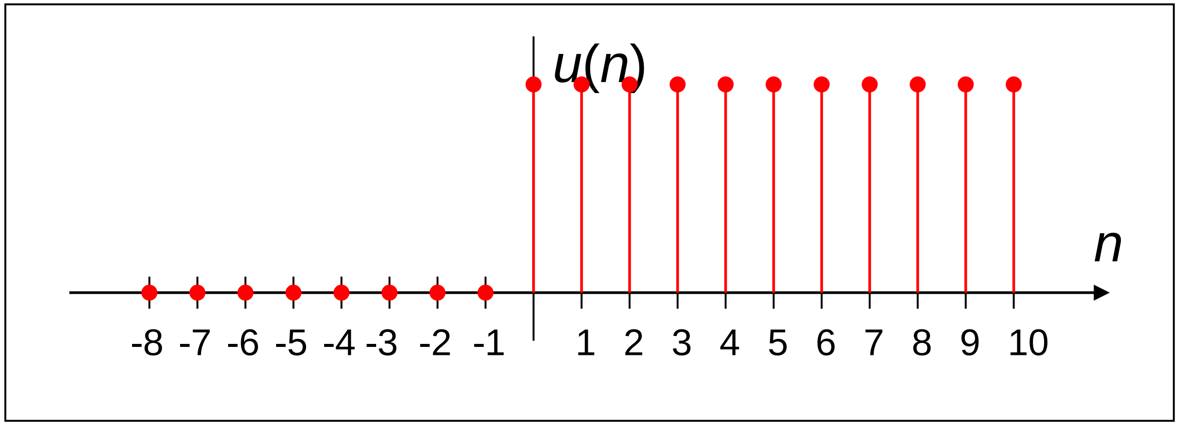
Important Signals

- Unit-step sequence *u*(*n*)

$$u(n) = \begin{cases} 1 & n \geq 0 \\ 0 & n < 0 \end{cases}$$

- Fact:

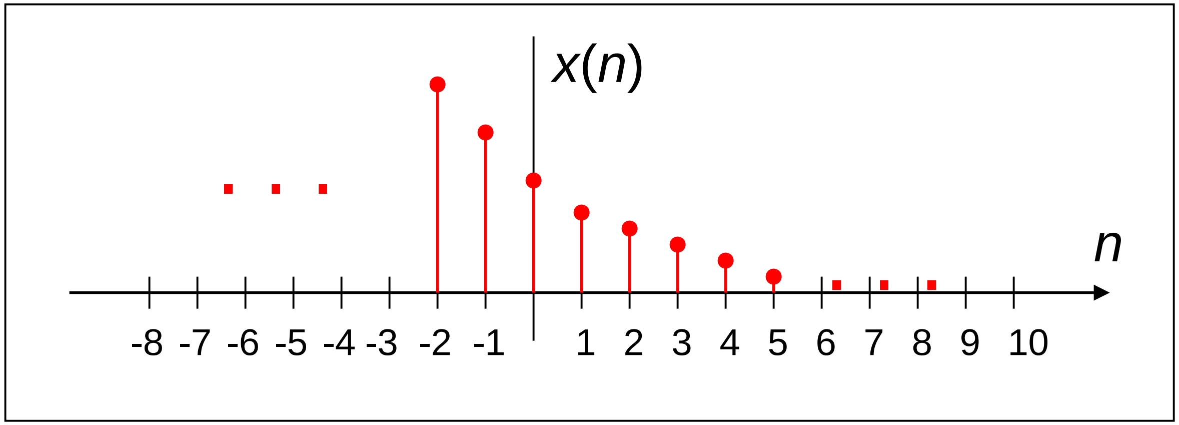
$$\delta(n) = u(n) - u(n-1)$$



Important Signals

- Real exponential sequence

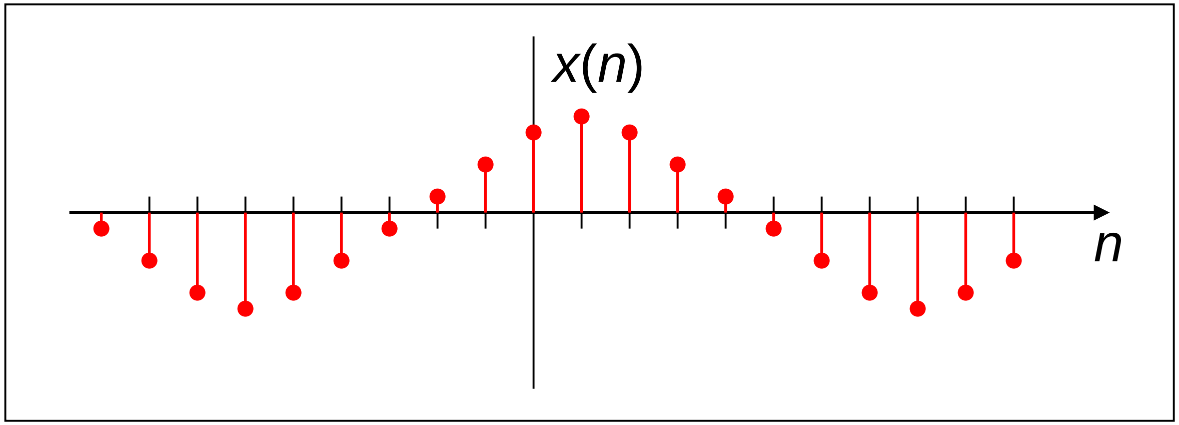
$$\boxed{x(n) = a^n}$$



Important Signal

- Sinusoidal sequence

$$\boxed{x(n) = A\cos(n\omega_0 + \phi)}$$



Classification of Signals

- Deterministic and Random
- Periodic and Non Periodic
- Even and Odd Signals
- Energy and Power Signal

Important Signal

- A sequence *x*(*n*) is defined to be periodic with period *N* if

$$\boxed{x(n) = x(n+N) \quad \text{for all } N}$$

- Example: consider $x(n) = e^{j\omega_0 n}$

$$x(n) = e^{j\omega_0 n} = e^{j\omega_0 (n+N)} = e^{j\omega_0 N} e^{j\omega_0 n} = x(n+N)$$

⇒ $\omega_0 N = 2k\pi$ ⇒ $N = \dfrac{2k\pi}{\omega_0}$ ⇒ $\dfrac{2\pi}{\omega_0}$ must be a rational number

Energy of a Sequence

- Energy of a sequence is defined by

$$\boxed{E = \sum_{n=-\infty}^{n=\infty} | x(n) |^2}$$

Operations on Sequences

- Sum

$$x + y = \{x(n) + y(n)\}$$

- Product

$$x \cdot y = \{x(n)y(n)\}$$

- Multiplicati

$$\alpha x = \{\alpha x(n)\}$$

- Shift

$$y(n) = x(n - n_0)$$

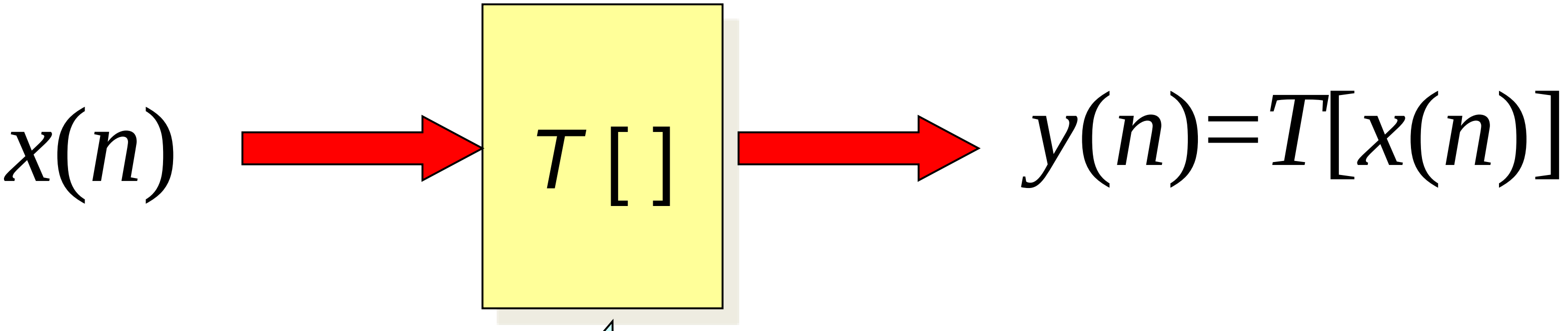
Systems

Set of Elementary or Functional blocks which are connected together and produces an output in response to an input signal. The response or output of the system depends upon transfer function of the system

Classification of Systems

1. Dynamic and Static
2. Time Variant and invariant
3. Linear and non- linear
4. Causal and non Causal
5. Stable and un- stable

Systems



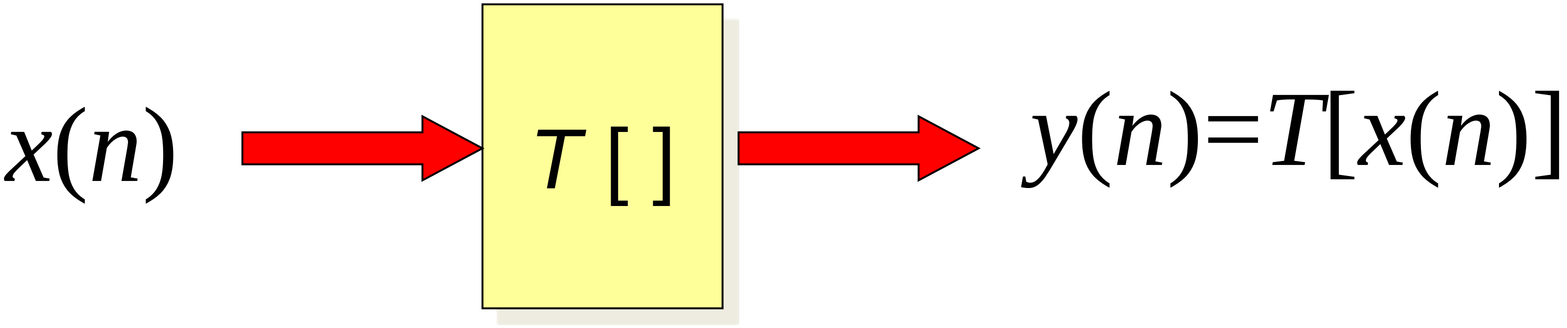
$x(n)$

$T[\,]$

$y(n)=T[x(n)]$

Mathematically modeled as a unique *transformation* or

Linear Systems



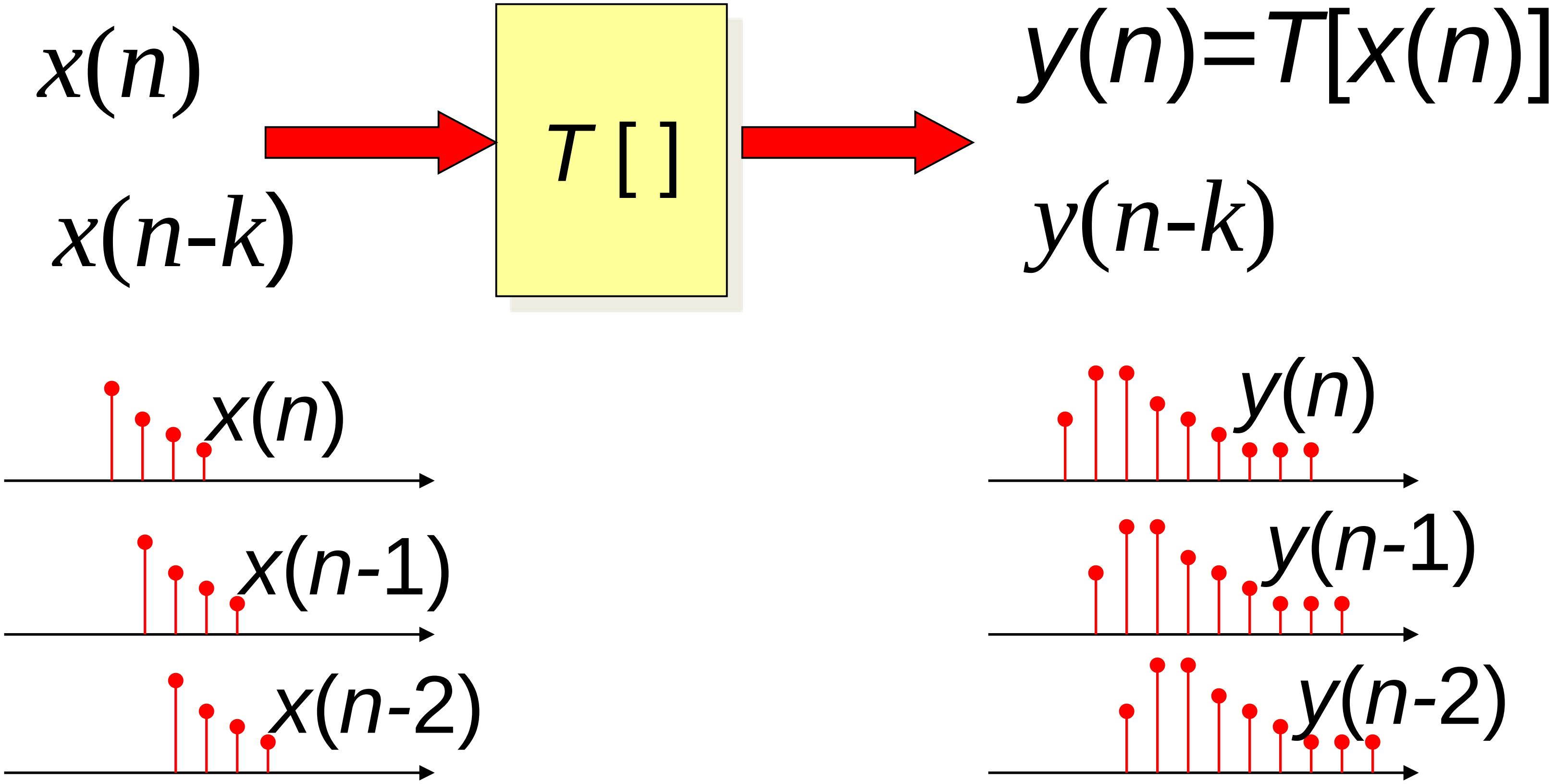
$x(n)$

$T[\;]$

$y(n)=T[x(n)]$

$$\boxed{T[ax_1(n)+bx_2(n)]=aT[x_1(n)]+bT[x_2(n)]}$$

Shift-Invariant Systems



$x(n)$

$x(n-k)$

$T[\,]$

$y(n)=T[x(n)]$

$y(n-k)$

$x(n)$

$x(n-1)$

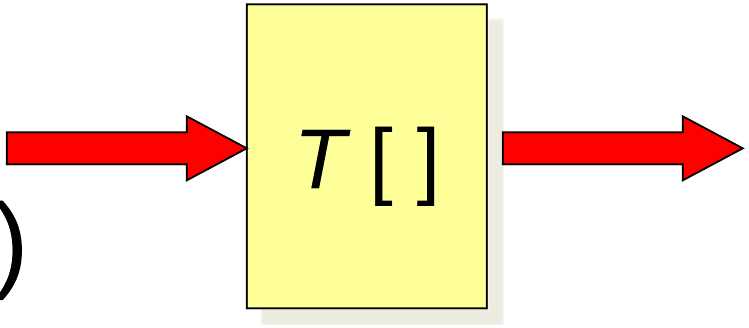
$x(n-2)$

$y(n)$

$y(n-1)$

$y(n-2)$

Shift-Invariant Systems



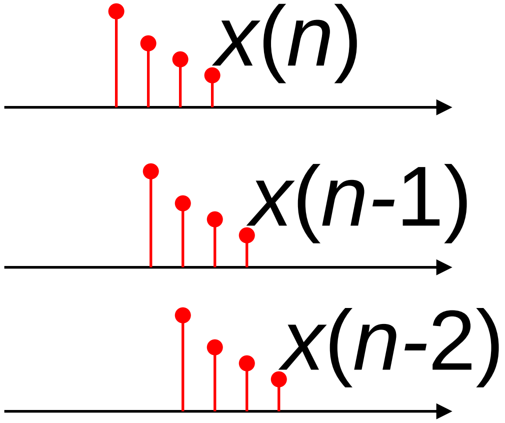
$x(n)$

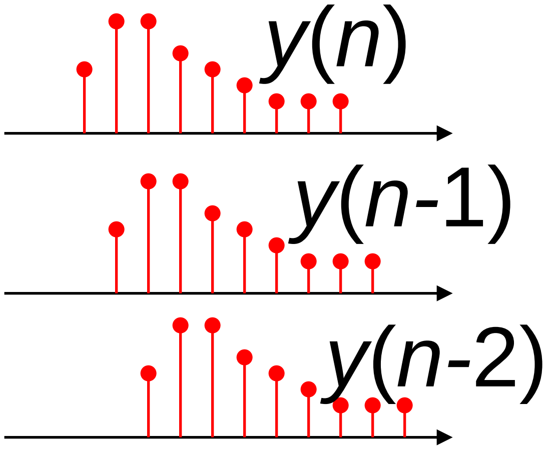
$x(n-k)$

$T[\,]$

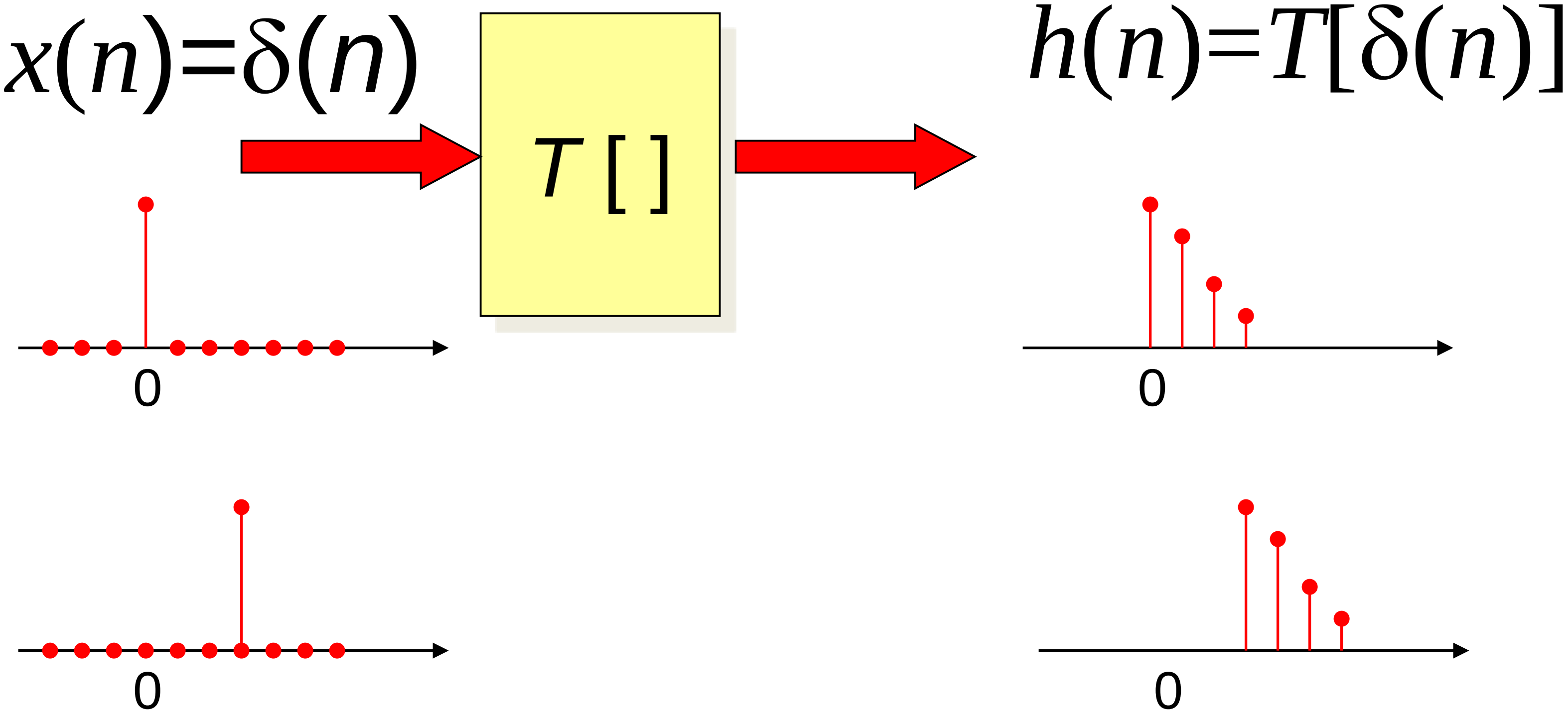
$y(n)=T[x(n)]$

$y(n-k)$





Impulse Response

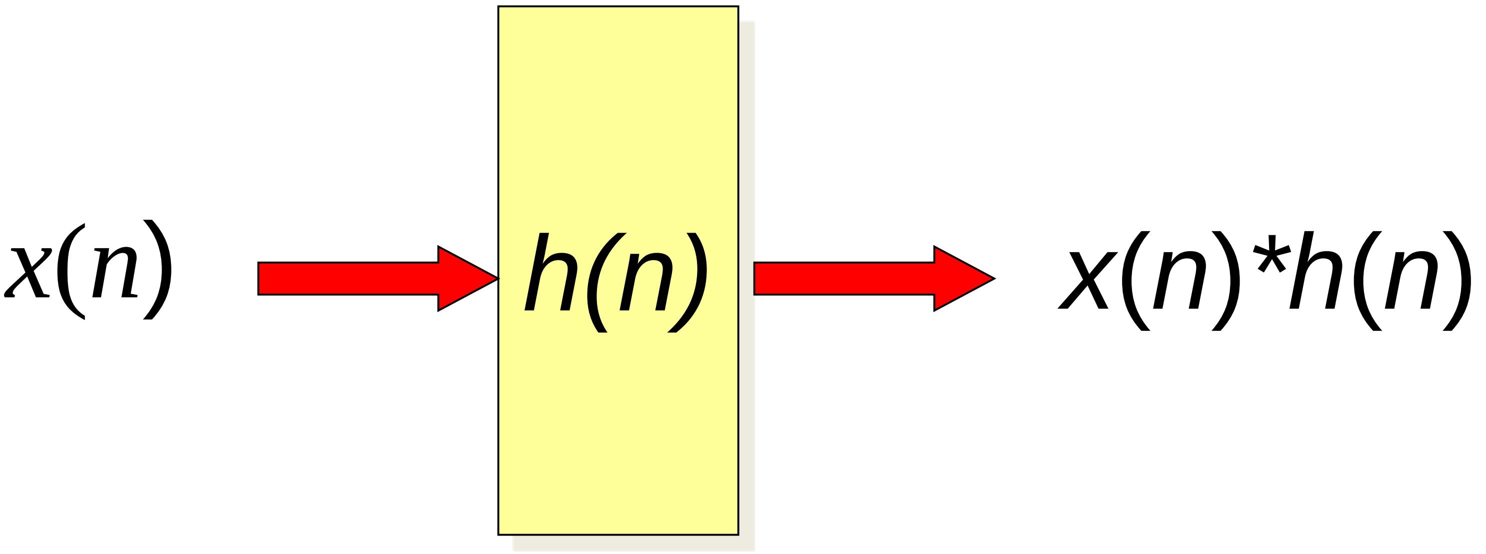


$x(n)=\delta(n)$

$T[\ ]$

$h(n)=T[\delta(n)]$

Characterize a System



Key DSP Operations

1. Convolution
2. Correlation
3. Digital Filtering
4. Discrete Transformation
5. Modulation

Convolution

Convolution is one of the most frequently used operations in DSP. Specially in digital filtering applications where two finite and causal sequences x[n] and h[n] of lengths N_1 and N_2 are convolved

$$y[n] = h[n] \otimes x[n] = \sum_{k=-\infty}^{\infty} h[k]x[n-k] = \sum_{k=0}^{\infty} h[k]x[n-k]$$

where, n = 0,1,…….,(M-1) and M = N_1 + N_2 -1

This is a multiply and accumulate operation and DSP device manufacturers have developed signal processors that perform this action.

Correlation

There are two forms of correlation :

1. Auto-correlation
2. Cross-correlation

Auto-Correlation :

The auto-correlation of a process x(t) is defined as the mean of the product $\boldsymbol{x}(t_1)\boldsymbol{x}^*(t_2)$. This function is denoted as R(t_1, t_2) or $R_x\,(t_1, t_2)$ or $R_{xx}\,(t_1, t_2)$.

$$R_x\,(t_1, t_2) = E\{\boldsymbol{x}_1(t_1)\boldsymbol{x}^*(t_2)\}$$

$$R(t_1, t_2) = \mathrm{E}\{x(t_1)x(t_2)\} = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} x_1 x_2 f(x_1, x_2; t_1, t_2)dx_1 dx_2$$

Auto-covariance:

Auto-covariance C (t_1, t_2) of a process ***x***(t) is the covarince of the random variables $\boldsymbol{x}(t_1)$ and $\boldsymbol{x}(t_2)$ and is defined as

$$C\ (t_1, t_2) = R(t_1, t_2) - \eta(t_1)\eta^*(t_2)$$

Here $\eta(t)$ = E{***x***(t)} is the mean of ***x***(t)

Cross-Correlation:

Cross-correlation of two processes ***x***(t) and ***y***(t) is given as

$$R_{xy}(t_1, t_2) = E\{x(t_1)y^*(t_2)\} = R_{yx}{}^*(t_2, t_1)$$

Cross-Covarience:

Cross-covarience of two processes ***x***(t) and ***y***(t) is given as

$$C_{xy}(t_1, t_2) = R_{xy}(t_1, t_2) - \eta_x(t_1)\eta^*{}_y(t_2)$$

If $C_{xy}(t_1, t_2)$=0 for every t1 and t2 then the two Processes are uncorrelated (independent)

For sampled signal (i.e. sampled signal), the autocorrelation is defined as either biased or unbiased defined as follows:

$$R_{xx}(m) = \frac{1}{N-|m|}\sum_{n=1}^{N-m+1} x(n)x(n+m-1) \qquad \textit{[Biased Autocorrelation]}$$

(2)

$$R_{xx}(m) = \frac{1}{N}\sum_{n=1}^{N-m+1} x(n)x(n+m-1) \qquad \textit{[Uniased Autocorrelation]}$$

for m=1,2,…,M+1

Correlation coefficient for discrete signals

Normalized version of the cross-covarience is known as the correlation coefficient and is defined as below

$$\rho_{xy}(n) = \frac{r_{xy}(n)}{\left[r_{xx}(0) r_{yy}(0)\right]^{1/2}} \qquad n = 0, \pm 1, \pm 2, \ldots$$

Where, r_{xy}(n) is an estimate of the cross-covarience

The cross-covarience is defined as

$$r_{xy}\left(n\right)=\begin{cases}\dfrac{1}{N}\sum_{k=0}^{N-n-1}x[k]y[k+n] & n=0,1,2,\ldots\\ \dfrac{1}{N}\sum_{k=0}^{N+n-1}x[k-n]y[k] & n=0,-1,-2,\ldots\end{cases}$$

$$r_{xx}(0)=\frac{1}{N}\sum_{k=0}^{N-1}\left[x[k]\right]^2 \quad , \quad r_{yy}(0)=\frac{1}{N}\sum_{k=0}^{N-1}\left[y[k]\right]^2$$

Fast Fourier Transforms

Discrete Fourier Transform

- The DFT pair was given as

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

- Baseline for computational complexity:
  - Each DFT coefficient requires
    - N complex multiplications
    - N-1 complex additions
  - All N DFT coefficients require
    - N^2 complex multiplications
    - N(N-1) complex additions
- Complexity in terms of real operations
    - $4N^2$ real multiplications $e^{-j(2\pi/N)k(N-n)} = e^{-j(2\pi/N)kN} e^{-j(2\pi/N)k(-n)} = e^{j(2\pi/N)kn}$
    - 2N(N-1) real additions $e^{-j(2\pi/N)kn} = e^{-j(2\pi/N)k(n+N)} = e^{j(2\pi/N)(k+N)n}$
- Most fast methods are based on symmetry properties
  - Conjugate symmetry

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

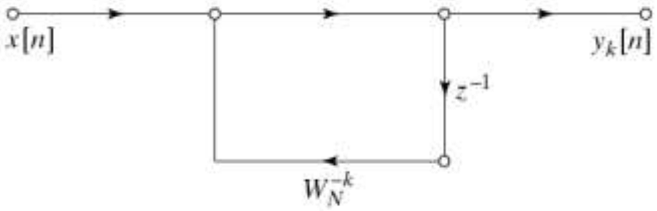
$$y_k[n] = \sum_{r=-\infty}^{\infty} x[r]e^{j(2\pi/N)k(n-r)}u[n-r]$$

- X[k] can be viewed as the output of a filter to the input x[n]
  - Impulse response of filter: $X[k] = y_k[n]\Big|_{n=N}$
  - X[k] is the output of the filter at time n=N

$$e^{j(2\pi/N)kn}u[n]$$

The Goertzel Filter

- Goertzel Filter

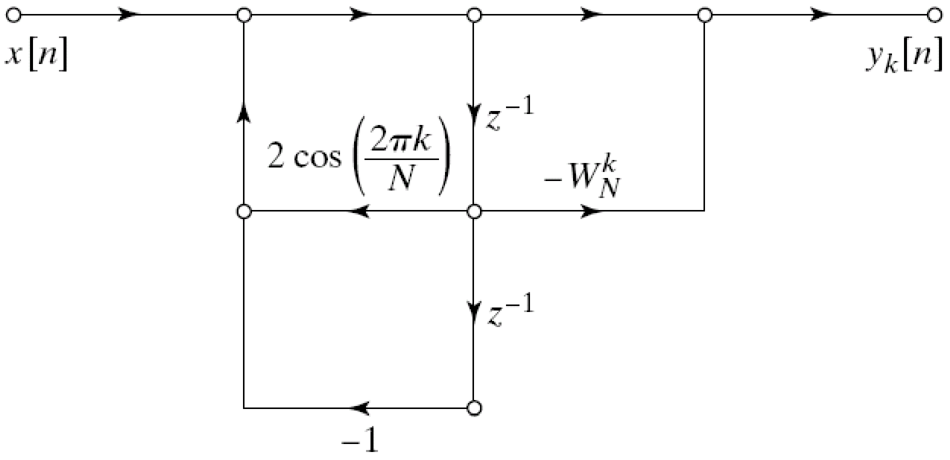


$$H_k(z) = \frac{1}{1 - e^{j\frac{2\pi}{N}k} z^{-1}}$$

- 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

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

- Complexity for one DFT coefficient
  - Poles: 2N real multiplications and 4N real additions
  - Zeros: Need to be implement 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

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

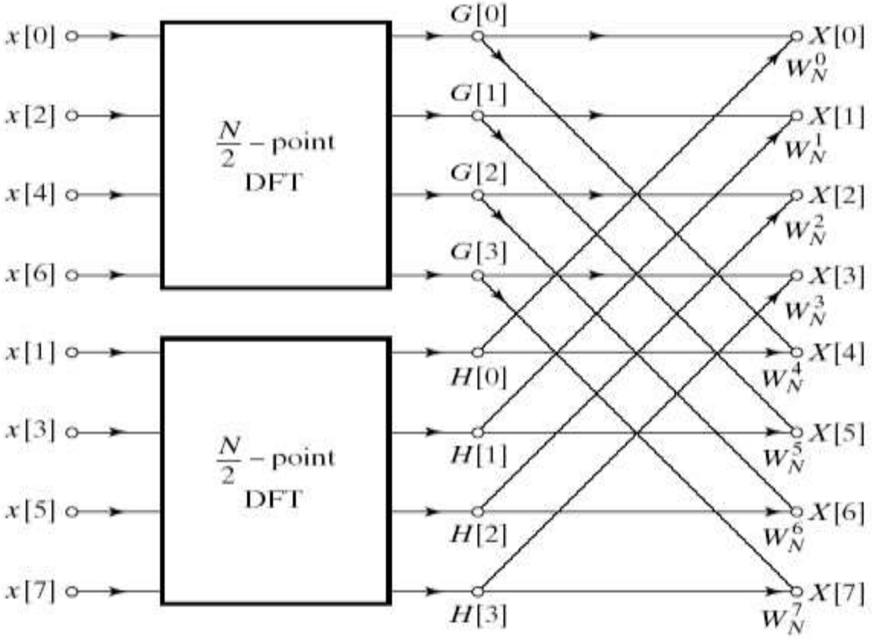
- Substitute variables n=2r for n even and n=2r+1 for odd

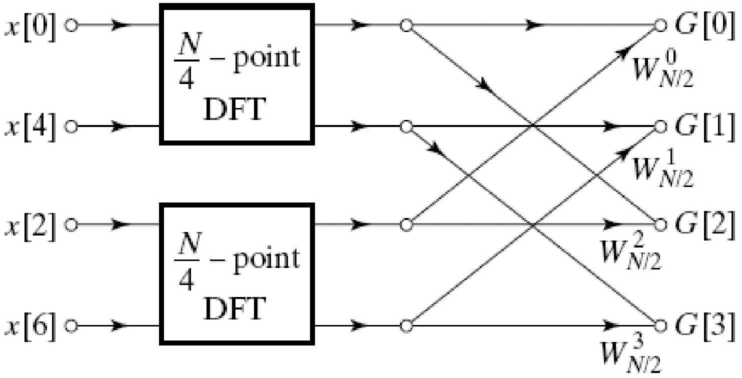
$$\begin{aligned} X[k] &= \sum_{r=0}^{N/2-1} x[2r]W_N^{2rk} + \sum_{r=0}^{N/2-1} x[2r+1]W_N^{(2r+1)k} \\ &= \sum_{r=0}^{N/2-1} x[2r]W_{N/2}^{rk} + W_N^k \sum_{r=0}^{N/2-1} x[2r+1]W_{N/2}^{rk} \\ &= G[k] + W_N^k H[k] \end{aligned}$$

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

Decimation In Time

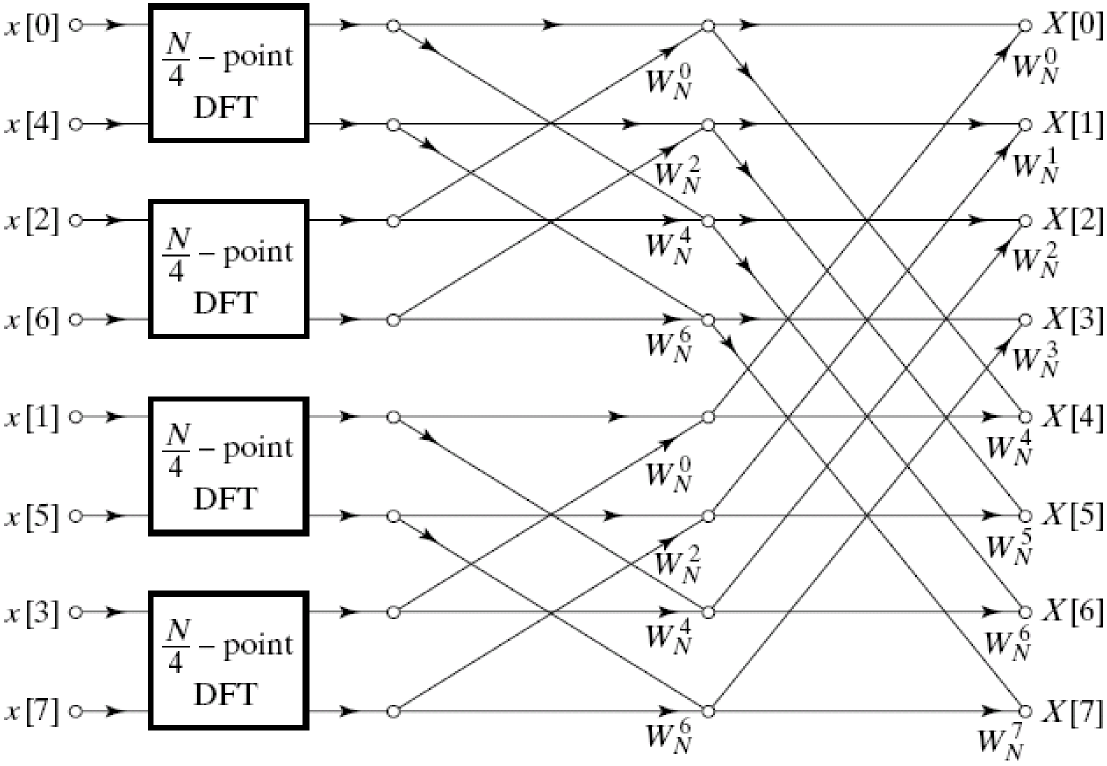
- 8-point DFT example using decimation-in-time
- Two N/2-point DFTs
  - $2(N/2)^2$ complex multiplications
  - $2(N/2)^2$ complex additions
- Combining the DFT outputs
  - N complex multiplications
  - N complex additions
- Total complexity
  - $N^2/2+N$ complex multiplications
  - $N^2/2+N$ complex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



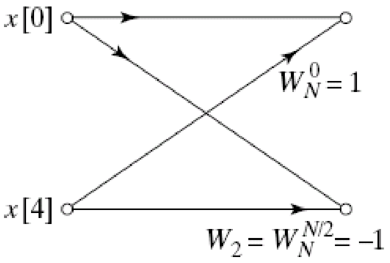


Decimation In Time Cont’d

- After two steps of decimation in time

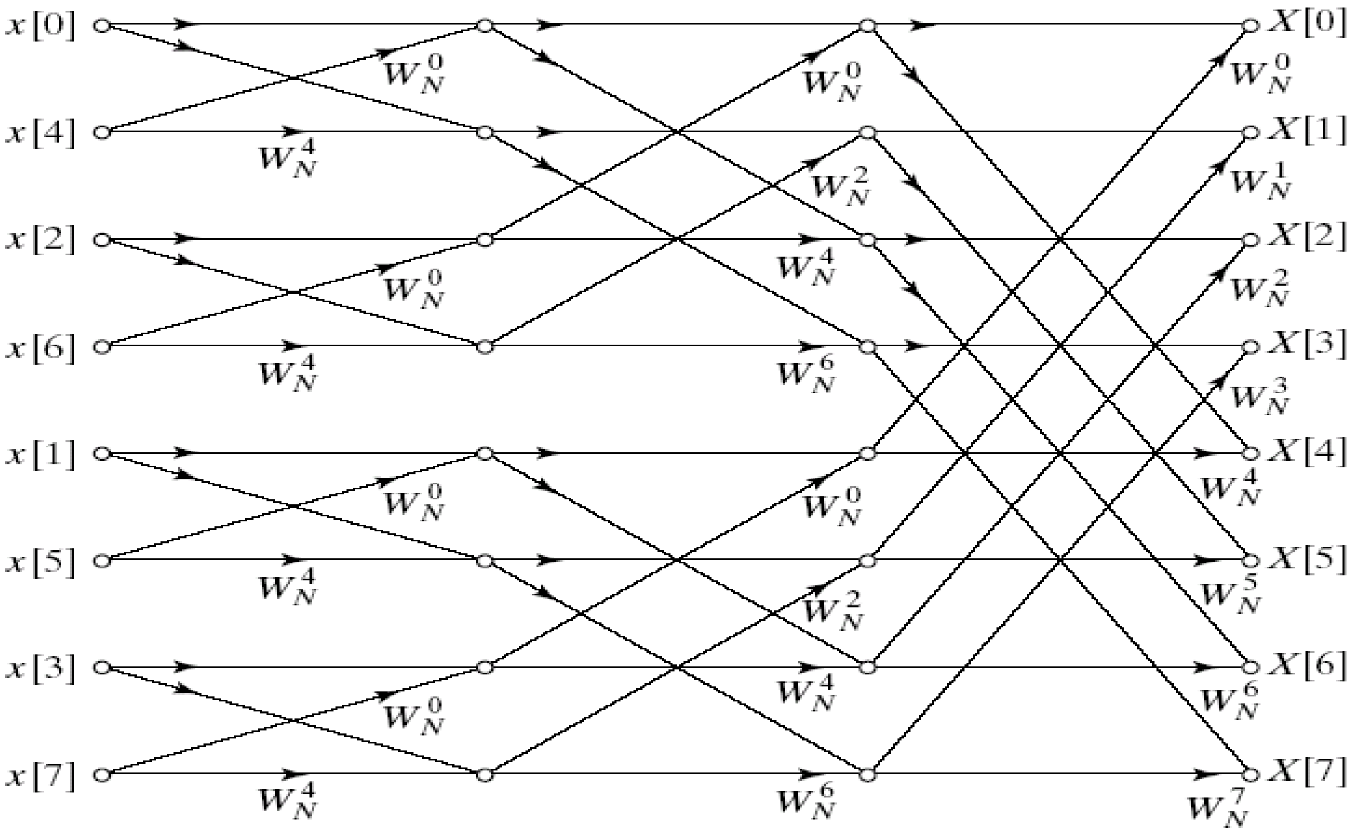


- Repeat until we’re left with two-point DFT’s



Decimation-In-Time FFT Algorithm

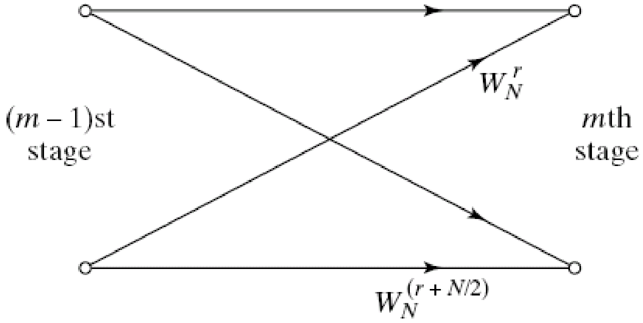
- Final flow graph for 8-point decimation in time



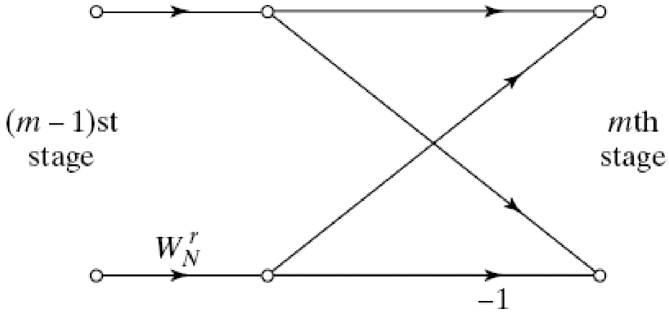
- Complexity:
  - $N\log_2 N$ complex multiplications and additions

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

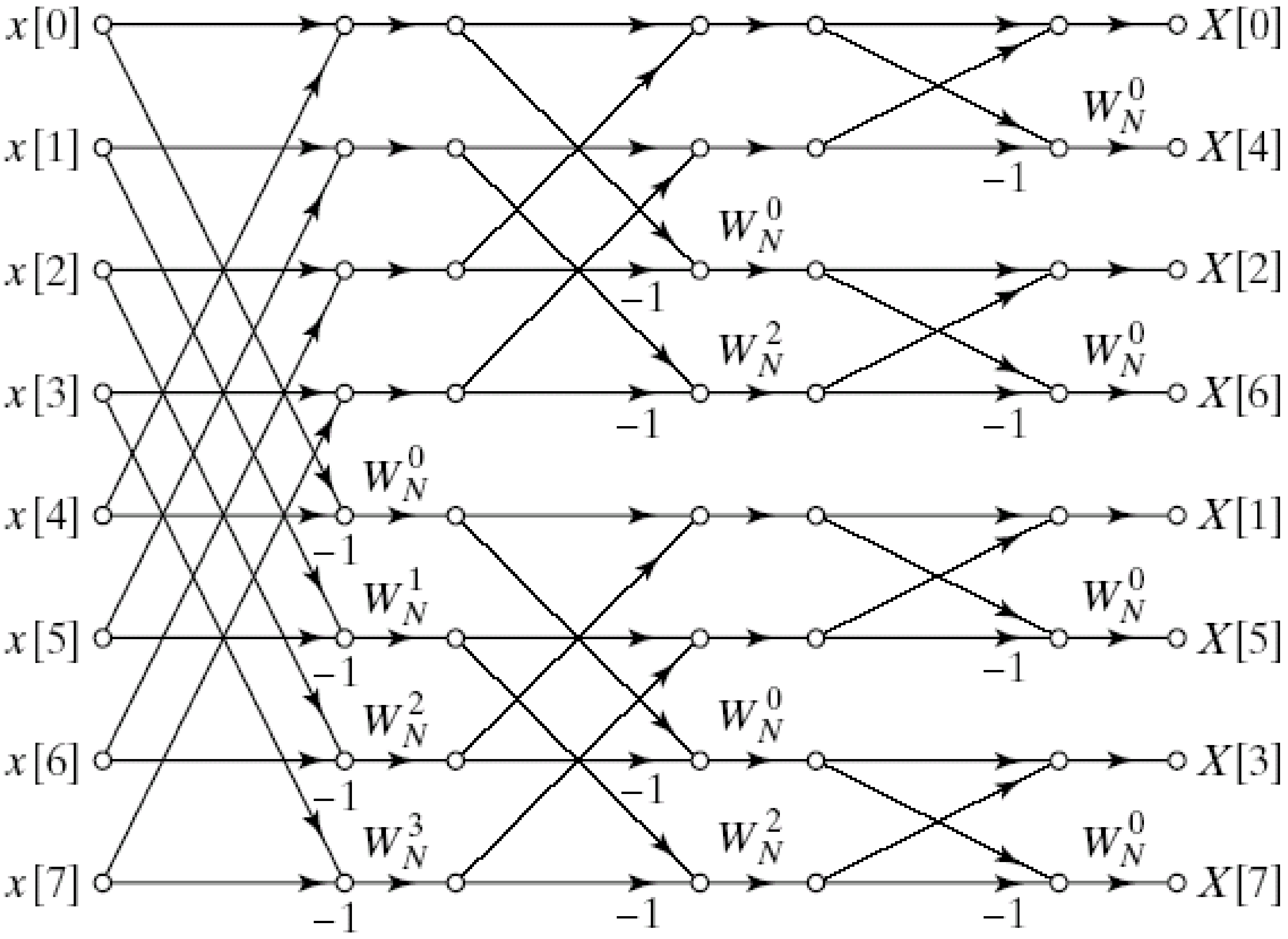
$$X[2r] = \sum_{n=0}^{N/2-1} x[n] W_N^{n2r} + \sum_{n=0}^{N/2-1} x[n + N/2] W_N^{(n+N/2)2r} = \sum_{n=0}^{N/2-1} \left(x[n] + x[n + N/2]\right) W_{N/2}^{nr}$$

- Similarly for odd-numbered frequencies

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

Decimation-In-Frequency FFT Algorithm

- Final flow graph for 8-point decimation in frequency



x[0]
x[1]
x[2]
x[3]
x[4]
x[5]
x[6]
x[7]
W_N^0
W_N^1
W_N^2
W_N^3
-1
-1
-1
-1
W_N^0
W_N^2
W_N^0
W_N^2
-1
-1
-1
-1
W_N^0
W_N^0
W_N^0
W_N^0
-1
-1
-1
-1
X[0]
X[4]
X[2]
X[6]
X[1]
X[5]
X[3]
X[7]