DSP Algorithm and Low Complexity Implementation Overview

Wonyong Sung

2 x 1.5 hrs

School of Electrical Engineering
Seoul National University

DSP Algorithms and Applications

- **Filtering**
  - Linear: FIR and recursive (IIR)
  - Nonlinear and time-varying: adaptive filter
  - Usage: noise elimination, frequency compensation, sample rate conversion
- **Transformation**
  - FFT: frequency analysis, indirect convolution
  - DCT: real arithmetic, image/video processing
  - Usage: spectrum analysis, OFDM(Orthogonal Frequency Division Multiplexing)
- **Communication blocks**
  - NCO: digital oscillator,
  - PLL: phase locked loop
  - ADC/DAC
  - QPSK, QAM, OFDM, CDMA
  - ECC: CRC, Hamming, BCH, RS, LDPC, convolutional coding

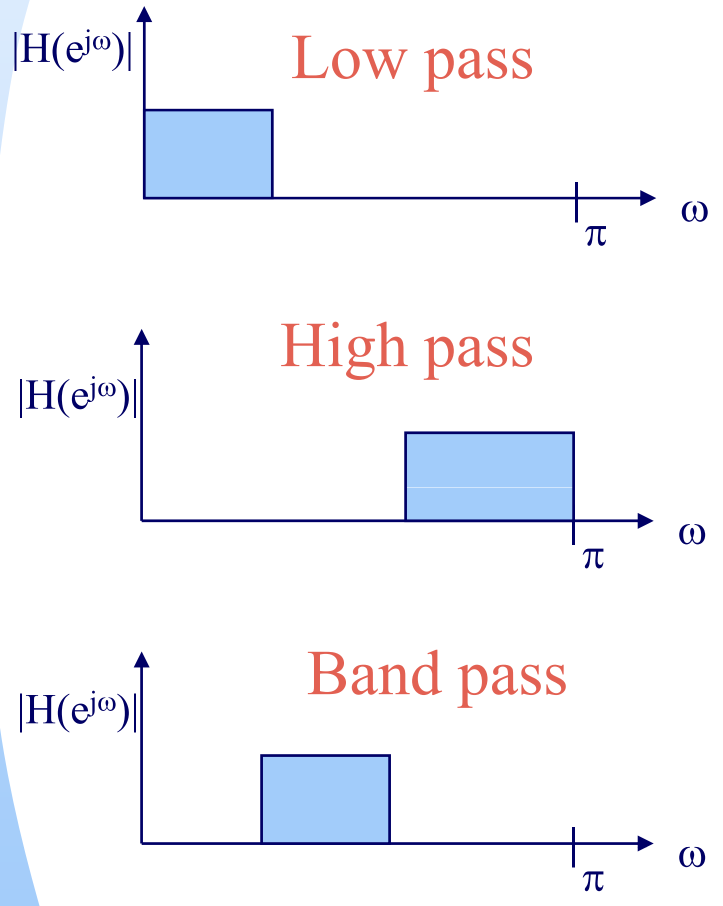
Algorithm considerations for system design

- **Performance in terms of signal processing**
  - For example, RLS is better than LMS in terms of signal robustness (adaptation speed)…
  - FIR is better in terms of phase linearity when compared with recursive filtering
    - For image processing, only FIR filtering is adequate.
- **Number of arithmetic ops.(multiplications)**
  - FIR filtering demands a lot of multiplications
    - But not always, for narrowband filtering, it needs a smaller one.
- **Algorithm complexity, parallel & regularity**
  - Seems more important in these days as there are abundant of arithmetic elements in a chip.
  - Parallel structure is good for HW based design.
- **If you start with a poor algorithm, there is not much way to recover the disadvantages!**
  - You need to consider both performance and implementation characteristics.

Digital Filters

- **Types of Digital Filters:**
  - low-pass, band-pass, high-pass, notch-filter, allpass, etc.
- **FIR and IIR Digital Filters**
- **Multiplierless filters**
- **Filters for sampling rate conversion**
- **Structures of Digital Filters**
  - Direct, cascade, parallel forms
  - State-space realizations
  - Orthogonal digital filter
- **Quantization Errors, Stability, accuracy**

Types of Digital Filters



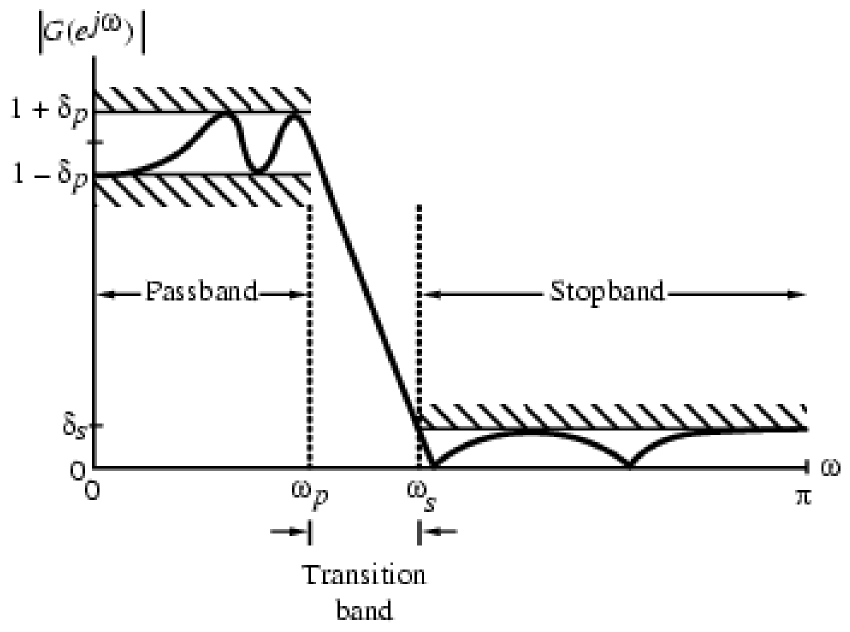
- **Usages:**
  - Low pass: anti-aliasing, smoothing, noise reduction
  - High pass: DC removal, baseline wander reduction
  - Band pass: noise reduction
- **Design:**
  - Choose FIR or IIR filter coefficients to approximate desired frequency response.
  - Usually the designed filter coefficients are not unique! Leaving large design space to be explored. Passband, stopband ripples.

Filter design and implementation

- Filter design: determining the transfer function (H(z)) from the given frequency domain specification. The location of poles and zeroes are determined.
- Filter implementation: determining the filter structure (direct form, 2nd order cascade form, …) , pole-zero pairing if needed, word-length and memory structure for reducing the hardware cost, machine cycles, or power consumption.

Digital filter specifications

- **For example the magnitude response $|G(e^{j\omega})|$ of a digital lowpass filter may be given as indicated below**



* Transition bandwidth is important for filter order determination.

Digital filter specifications

- 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

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

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

Digital filter specifications

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

$$|G(e^{j\omega})| \cong 1 \pm \delta_p$$

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

- In the stopband $\omega_s \le \omega \le \pi$ we require that with a deviation δ_s

$$|G(e^{j\omega})| \cong 0$$

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

Digital filter specifications

Filter specification parameters

- ω_p - passband edge frequency
- ω_s - stopband edge frequency
- δ_p - peak ripple value **in the passband**
- δ_s - peak ripple value **in the stopband**

Digital filter specifications

- **Practical specifications are often given in terms of** loss function (in dB)
- $$\mathrm{G}(\omega) = -20\log_{10}\left|G(e^{j\omega})\right|$$
- Peak passband ripple

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

- Minimum stopband attenuation

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

FIR, IIR digital filters

- **Let {h[n]}: impulse response**

  {x(n)}: input, {y(n)}: output
- **Finite impulse response (FIR) filter:**

$$y(n) = \sum_{j=0}^{J-1} h(j)x(n-j)$$

- **Has only zeroes (no poles).**
- **Usually, implemented as a feed-forward type.**
- **Infinite impulse response (IIR) filter**

$$y(n) = \sum_{i=1}^{P} a(i)y(n-i) + \sum_{k=0}^{Q} b(k)x(n-k)$$

- **Both poles and zeroes. The length of impulse (unitpulse)response may be infinite!**
- **Recursive formula will impact on computation methods (feedback).**
- **Stability concerns:**
  - The magnitude of y(n) may become infinity even all x(n) are finite!
  - coefficient values,
  - quantization error

FIR filters

- **Direct form structure, which has a form of convolution, is usually used. Cascade or parallel forms are a little bit complex in terms of structure. The quantization effects of direct form FIR filters are still tolerable, in most cases.**
- **Symmetric coefficients FIR**
  - Linear phase: critical for image processing
  - Halve the # of multiplications
- **Filter design**
  - Windowing
  - CAD – Parks McClellan method
- **The needed order is usually high.**
- **Good for interpolation, decimation filtering**

Linear phase filter - symmetric FIR

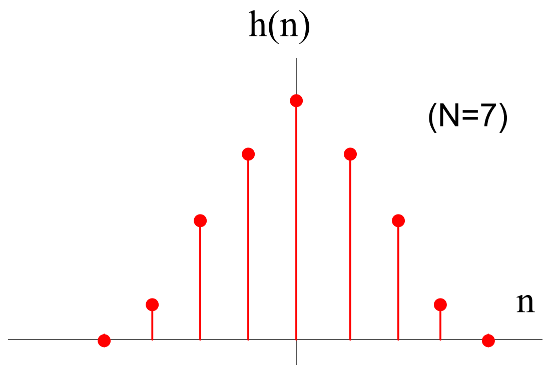
- **h(n) = h(-n)**
- **Evaluate the frequency response (assuming that N is odd) and h(n) is real-valued**

$$H(z) = \sum_{n=n_1}^{n_2} h(n) \quad z^{-n} = \sum_{n=-\frac{N-1}{2}}^{\frac{N-1}{2}} h(n) \quad z^{-n}$$

if $h(n) = h(-n)$ we get

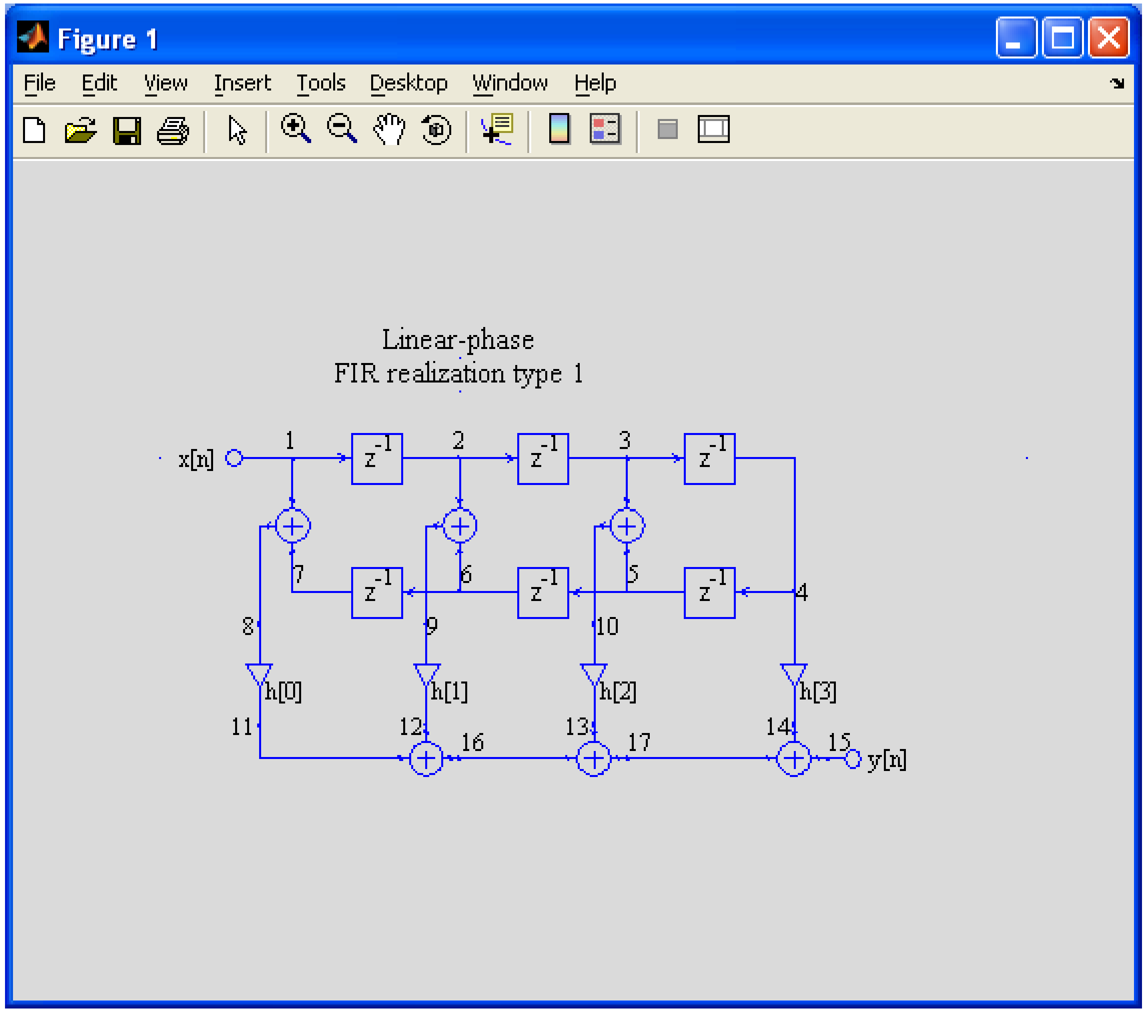
$$H(e^{j2\pi\Omega}) = h(0) + \sum_{n=1}^{\frac{N-1}{2}} h(n) \quad (e^{-j2\pi n\Omega} - e^{+j2\pi n\Omega})$$

$$H(e^{j2\pi\Omega}) = h(0) + 2 \sum_{n=1}^{\frac{N-1}{2}} h(n) \quad \cos[2\pi n\Omega]$$



The **frequency response is real**: phase shift is 0 or 180 degrees

Linear-phase type 1



Needed number of coefficients

❖ **For equiripple LP FIR filters:**

$$\boxed{N_e = \frac{2}{3}\log\left[\frac{1}{10\, D_{pass}\, D_{stop}}\right]\frac{F_s}{F_{stop} - F_{pass}}}$$

✓Independent of BW (F_{pass})!
✓Weak (logarithmic) dependence on the
Pass band ripple level and the Stop band attenuation
✓**Linear dependence on the transition band!**

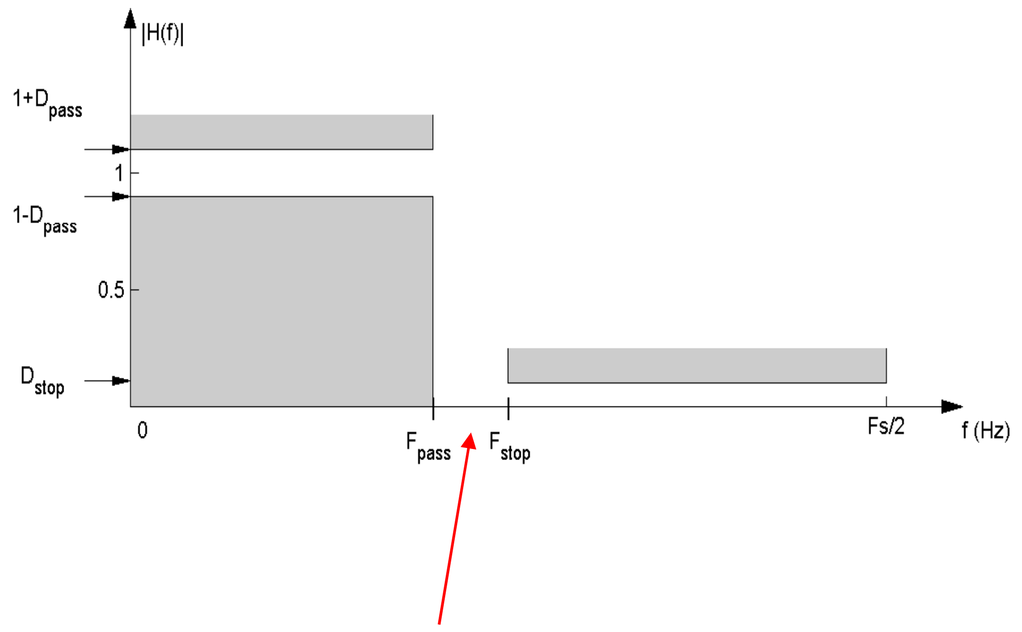
- Our example: Ne = 29 (compared to 31)
- Problem: Very narrow filters -> **Decimating**

Example

- **Develop an AM-SSB modulator using Simulink. There are a few methods for generating SSB signal. Here you try to use a bandpass filter. The filter specification that I suggest is a bandpass filter having the following specification. This filter is designed assuming that the message signal has the frequency range of 0.2KHz ~ 3.8KHz. Carrier freq = 12KHz, Sampling freq = 48KHz**
  - f1: stopband edge: 11.8KHz
  - f2: passband edge: 12.2KHz
  - f3: passband edge: 15.8KHz
  - f4: stopband edge: 16.2KHz
    - Passband ripple: -0.5dB~+0.5dB, stopband attenuation: 40dB

FIR filter design

- **Signal Processing Toolkit of MATLAB**
- **Define the** frequency response template**.**
- **Case of LPF:**
  - ✓ Pass band End F_{pass}
  - ✓ Pass band Ripple D_{pass}
  - ✓ Stop band Start F_{stop}
  - ✓ Stop band Attenuation D_{stop}



F_{stop}-F_{pass} = **Transition band**

Design of Equiripple Linear-Phase FIR Filters

- The linear-phase FIR filter obtained by minimizing the peak absolute value of

$$\varepsilon = \max_{\omega \in R} |\mathrm{E}(\omega)|$$

is usually called the ***equiripple FIR filter***

- After ε is minimized, the weighted error function $\mathrm{E}(\omega)$ exhibits an equiripple behavior in the frequency range *R*

Design of Equiripple Linear-Phase FIR Filters

- The general form of frequency response of a causal linear-phase FIR filter of length 2*M*+1:
$$H(e^{j\omega}) = e^{-jM\omega} e^{j\beta} \breve{H}(\omega)$$
where the amplitude response $\breve{H}(\omega)$ is a real function of ω
- Weighted error function is given by
$$\mathcal{E}(\omega) = W(\omega)[\breve{H}(\omega) - D(\omega)]$$
where $D(\omega)$ is the desired amplitude response and $W(\omega)$ is a positive weighting function

Design of Equiripple Linear-Phase FIR Filters

- For filter design,

$$D(\omega) = \begin{cases} 1, & \text{in the passband} \\ 0, & \text{in the stopband} \end{cases}$$

- $\breve{H}(\omega)$ is required to satisfy the above desired response with a ripple of $\pm\delta_p$ in the passband and a ripple of δ_s in the stopband

Design of Equiripple Linear-Phase FIR Filters

❖ **Thus, weighting function can be chosen either as**

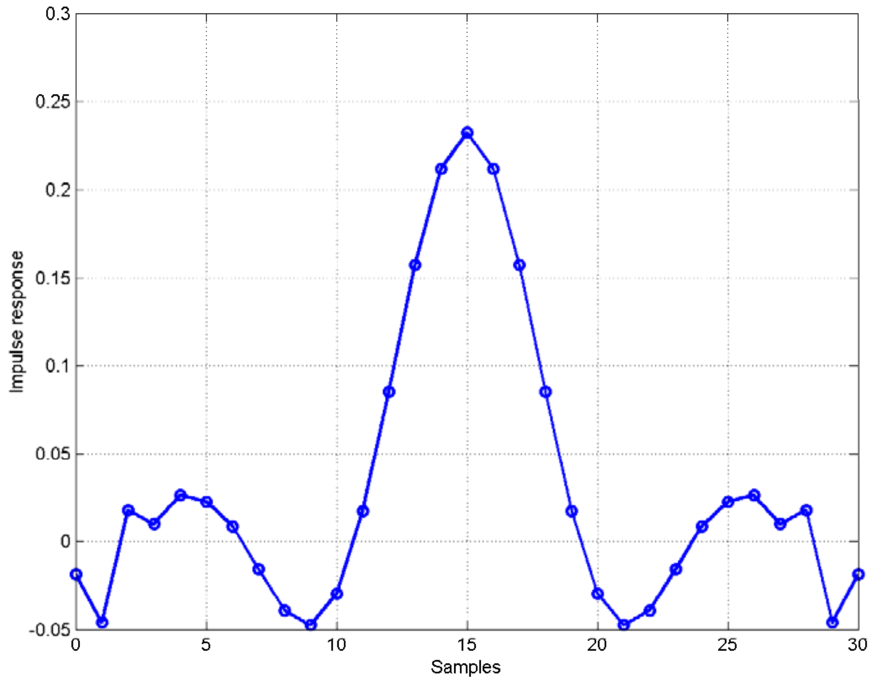
$$W(\omega) = \begin{cases} 1, & \text{in the passband} \\ \delta_p/\delta_s, & \text{in the stopband} \end{cases}$$

or

$$W(\omega) = \begin{cases} \delta_s/\delta_p, & \text{in the passband} \\ 1, & \text{in the stopband} \end{cases}$$

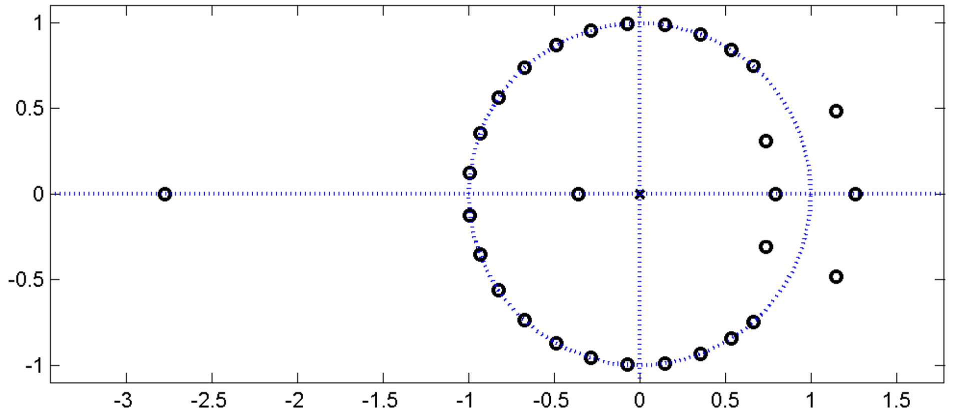
Example 1: Equiripple LP FIR (1)

- **LPF example:**
  - ✓ Pass band End F_{pass} = 0.1
  - ✓ Pass band Ripple D_{pass}=0.05 (5%)
  - ✓ Stop band Start F_{stop}= 0.13
  - ✓ Stop band Attenuation D_{stop} = 0.1 (1/10)
- **Minimum of 31 coefficients needed to achieved required specifications**
- **Even-symmetric impulse response**

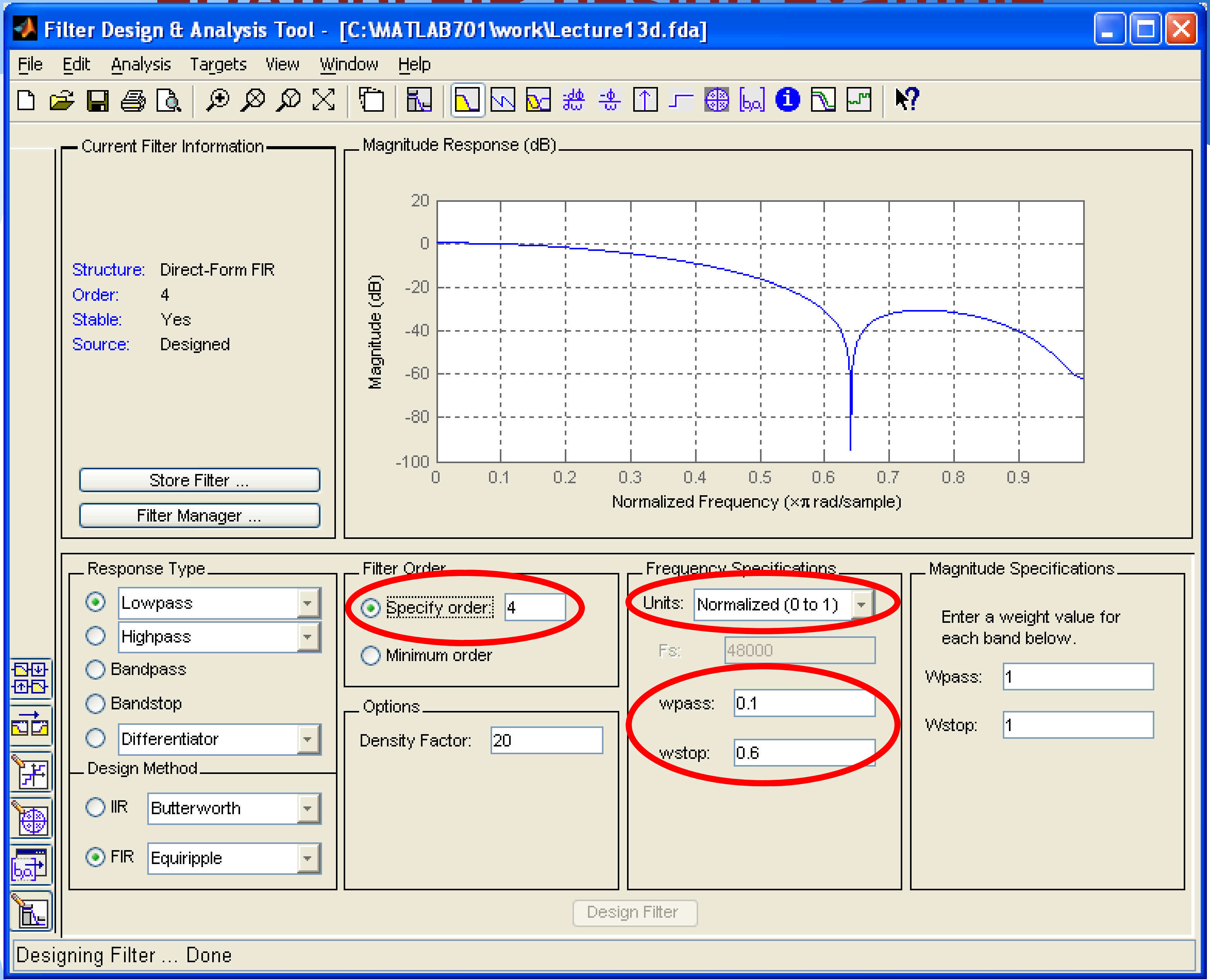


Example 1: Equiripple LP FIR (3)

- **Pole-Zero plot in the z-plane**
- **FIR -> pole at zero (causal)**
- **Location of zeros:**
  - ✓ On the unit circle in the Stop band
  - ✓ Far from unit circle in Pass band -> ripples



FDAtool FIR design example



Filter Design & Analysis Tool - [C:\MATLAB701\work\Lecture13d.fda]

Structure: Direct-Form FIR
Order: 4
Stable: Yes
Source: Designed

Response Type: [✓ Lowpass] Highpass Bandpass Bandstop Differentiator

Design Method: IIR Butterworth [✓ FIR] Equiripple

Filter Order: [✓ Specify order: 4] Minimum order

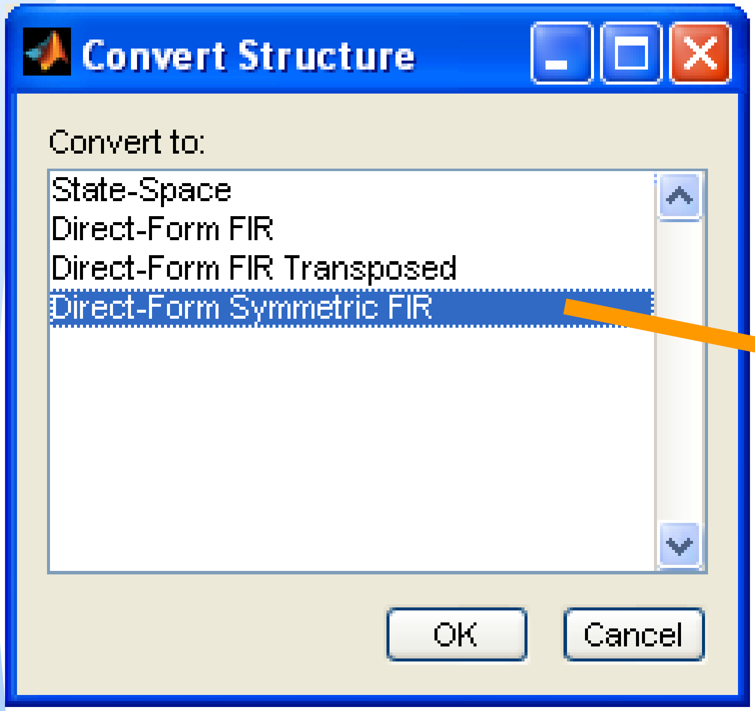
Options: Density Factor: 20

Frequency Specifications:
Units: Normalized (0 to 1)
Fs: 48000
wpass: 0.1
wstop: 0.6

Magnitude Specifications:
Enter a weight value for each band below.
Wpass: 1
Wstop: 1

Designing Filter ... Done

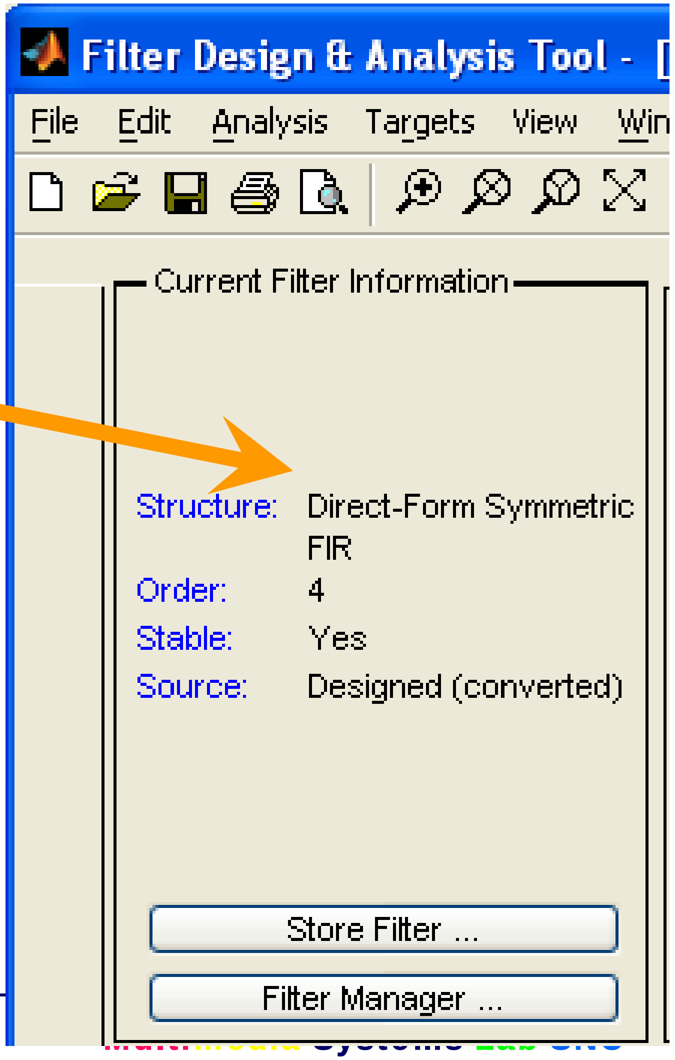
Edit, Convert Structure …



Convert Structure

Convert to:

State-Space
Direct-Form FIR
Direct-Form FIR Transposed
Direct-Form Symmetric FIR



Current Filter Information

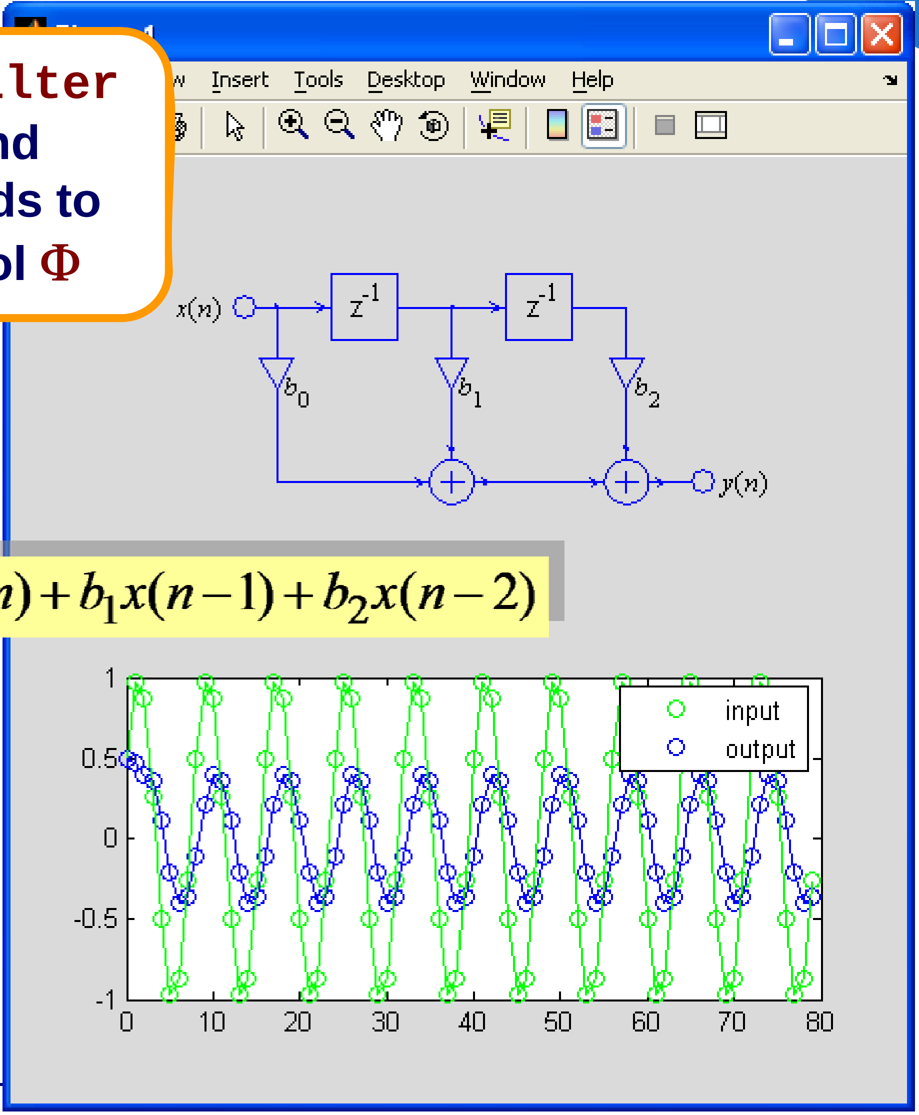
Structure:	Direct-Form Symmetric FIR
Order:	4
Stable:	Yes
Source:	Designed (converted)

MATLAB example 1

```
N = 80; k = 0:(N-1);
b0 = 1;
b1 = -1;
b2 = 1;
B = [b0 b1 b2];
f = 1/8;
x = sin(2*pi*f*k+pi/6);
y = filter(B,1,x);
subplot(2,1,1)
systemFIR(0,0,4,5,10,'b')
subplot(2,1,2)
plot(k,x,'go', k,y,'bo',...
    k,x,'g-', k,y,'b-')
legend('input','output')
```

MATLAB `filter` command corresponds to the symbol Φ

$$y(n) = b_0 x(n) + b_1 x(n-1) + b_2 x(n-2)$$



IIR or recursive filter

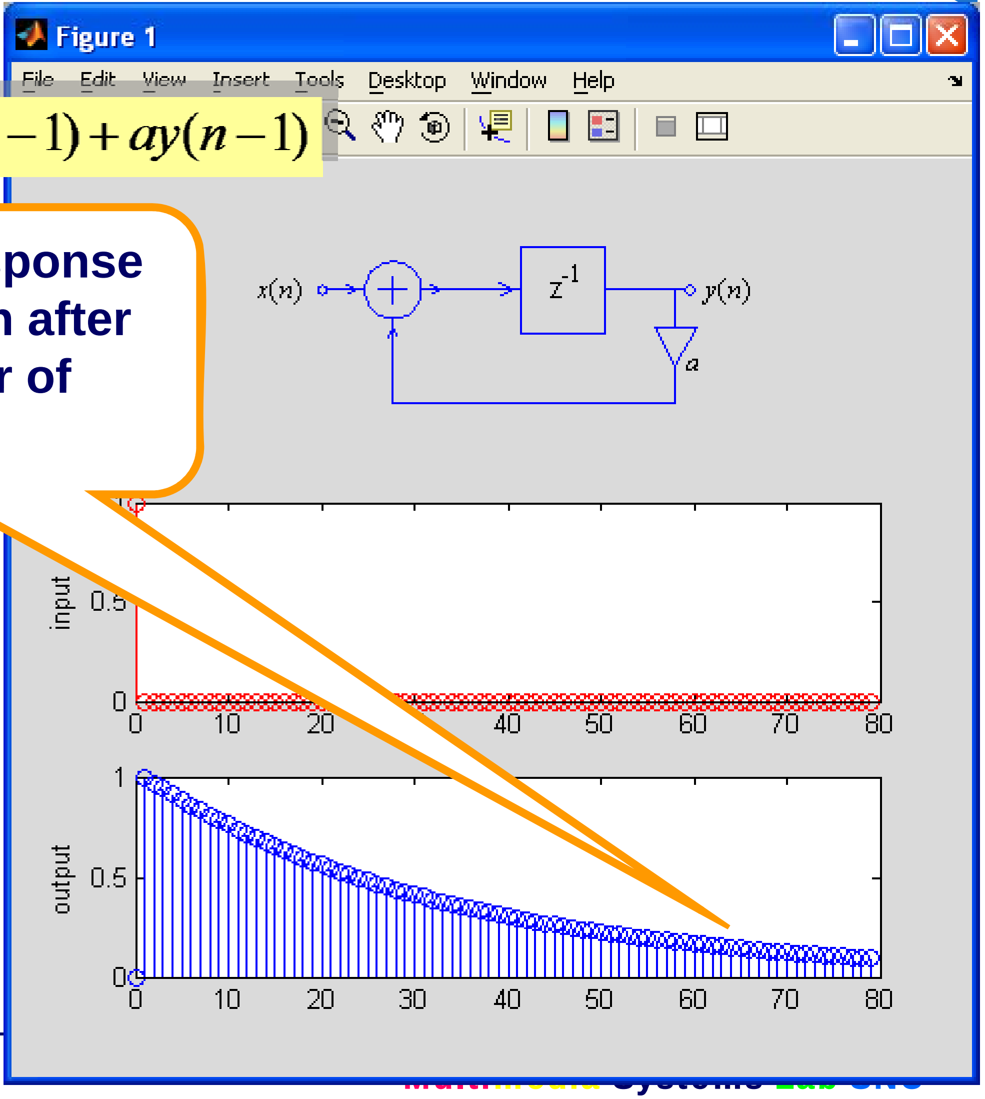
- **Utilize poles and zeros**
  - Poles are for shaping pass bands
  - Zeros are for stop bands
- **The poles should be inside of the unit circle in the z-domain for stability**
  - Finite word-length effects can affect
  - Direct forms are more susceptible.
- **Usually, the order needed is smaller when compared to FIR filters**
- **Direct forms are not good fixed-point implementations**
  - 2nd order cascade, ...

MATLAB example 3

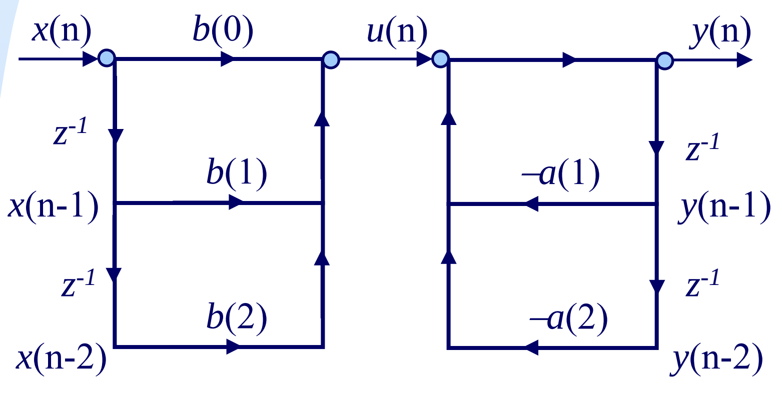
```
N = 80; k = 0:(N-1);
a = 0.97;
B = [0 1];
A = [1 -a];
x = (k==0);
y = filter(B,A,x);
subplot(3,1,1)
draw1stIIR(0,0,4,5,10,'b')
subplot(3,1,2)
stem(k,x,'r')
ylabel('input')
subplot(3,1,3)
stem(k,y,'b')
ylabel('output')
```

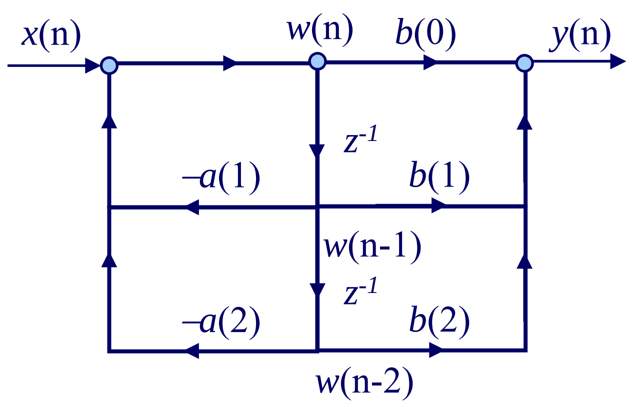
$$y(n) = x(n-1) + ay(n-1)$$

The impulse response does not vanish after finite number of samples



Basic 2nd Order IIR Structures





- **Direct form I realization**
- **5 Multiply 4 Additions per y(n)**
- **4 registers storing x(n-1), x(n-2), y(n-1), y(n-2)**

- **Direct form II realization a.k.a** BiQuad
- **5 multiply, 4 additions per y(n)**
- **2 registers storing w(n-1), w(n-2)**

Different structures determines different execution order, different numerical properties, but retain the same algebraic relation between input and output.

Cascaded and Parallel Structures

If the characteristic polynomial A(z) has real-valued coefficients, then it can be factorized as:

$$A(z)=\left(\prod_{i=1}^{P_1}(1-\alpha_i z^{-1})\right)$$

$$\cdot\left(\prod_{k=1}^{P_2}(1-2r_k\cos\beta_k z^{-1}+r_k^2 z^{-2})\right)$$

$$=\left(\prod_{i=1}^{P_1}A_i(z)\right)\cdot\left(\prod_{k=1}^{P_2}A_k(z)\right)$$

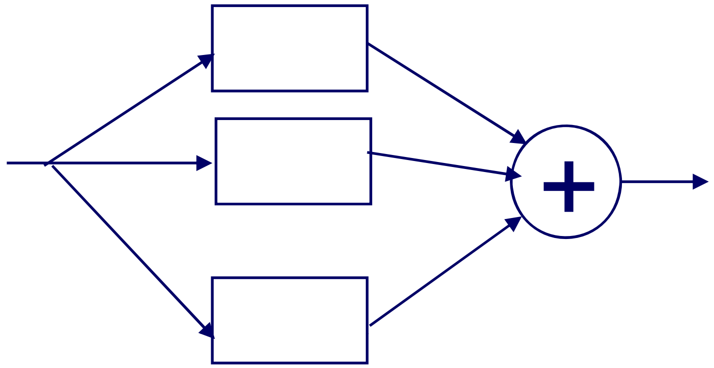
$$P=P_1+P_2$$

Cascaded realization

$$\begin{tikzcd} {} \arrow[r] & \boxed{\dfrac{\gamma_1}{1-\alpha_1 z^{-1}}} \arrow[r] & \boxed{\dfrac{\gamma_{P_1}}{1-\alpha_{P_1} z^{-1}}} \arrow[r] & \boxed{\dfrac{\eta_{P_2}}{1-2r_{P_2}\beta_{P_2}z^{-1}+r_{P_2}^2 z^{-2}}} \arrow[r] & {} \end{tikzcd}$$

Parallel realization

$$H(z)=\sum_{i=1}^{P_1}\frac{\mu_i}{1-\alpha_i z^{-1}}+\sum_{k=1}^{P_2}\frac{\nu_k+\rho_k z^{-1}}{1-2r_k\cos\beta_k z^{-1}+r_k^2 z^{-2}}$$



2nd-order Sections

- Generally implement high-order IIR filters as a sum or cascade of 2nd-order filters to reduce the sensitivity of the overall response to the precision of each coefficient
- High-order polynomial is factored to find poles and zeros, and each 2nd-order filter (biquad) implements two zeros and two poles (real or complex conjugate)
- Poles and zeros are grouped in pairs—usually try to group in such a way that minimizes peak gain of each section
- If cascaded, 2nd-order sections are usually ordered from lowest gain to highest gain to minimize overflow likelihood
- Scaling may be needed for overall response or for each section individually (computation vs. quality)

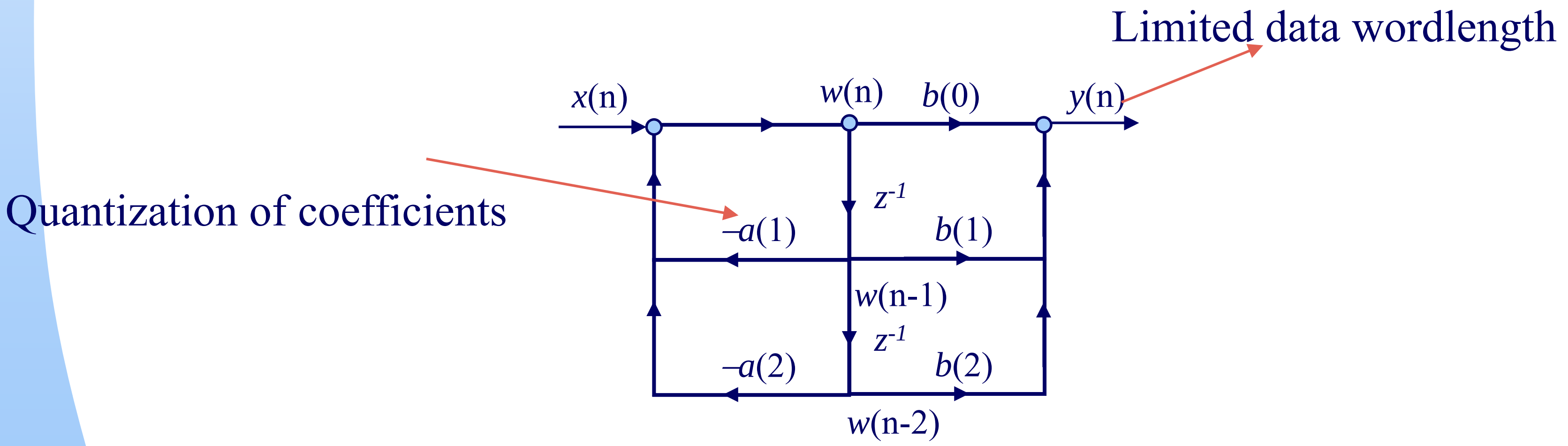
$$H(z) = \frac{b_0 + b_1 z^{-1} + b_2 z^{-2}}{1 + a_1 z^{-1} + a_2 z^{-2}}$$

Pole-zero pairing

- **When pairing poles and zeros, it is better to combine nearby poles and zeroes for a same 2nd order structure.**
- **Purpose: to reduce the dynamic range of internal signal**
  - It is not good to magnify (attenuate) a signal and then attenuate (magnify) it much in terms of quantization noise sensitivity.
  - Criteria: ***Maximal dynamic range***: a ratio of (a) the maximum magnitude response computed over the whole frequency range to (b) the minimum magnitude response in the passband

Implementation Issues

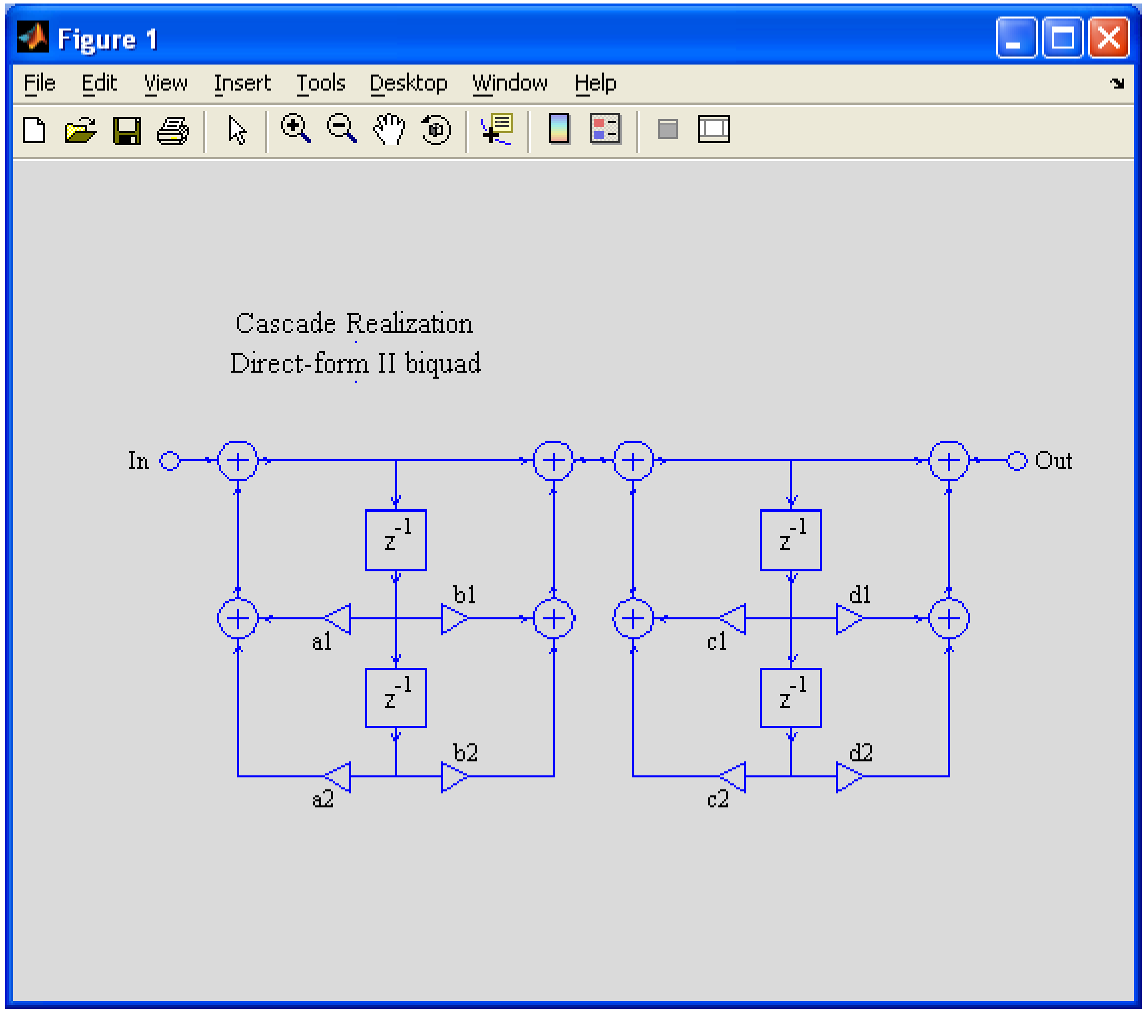
- **Coefficient quantization**
  - Due to limited coefficients word-length.
  - This affects the frequency response.
  - Especially, the pass-band or stop-band ripple can be increasing (make a frequency distortion.)
- **Roundoff error (least significant bits lost)**
  - Due to limited data word-length
  - This introduces random-like quantization noise. Make a noisier filter.



Engineering an IIR filter

- Verify filter performance with quantized coefficients
- Assess roundoff error by modeling the truncation as a *noise source* in the filter structure. Determine the transfer function from each roundoff noise source to the output.
- Calculate overflow situations by looking for peaks in frequency response from input to each accumulation point in the filter.

Cascade direct form II



Comparison of the complexity of different IIR filters

Structure	Number of multiplications	Number of additions and subtractions	Total number of operations	Required bitwidth
Direct form	16	16	40	21
Cascade	13	16	34	12
Parallel	18	16	39	11
Continued fraction	18	16	35	23
Ladder	17	32	50	14
Wave digital	11	30	47	12

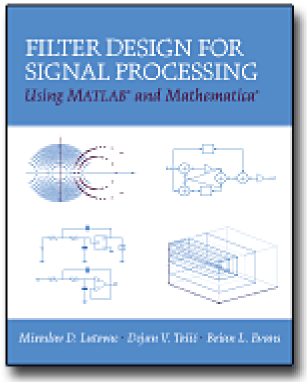
From: Miodrag Bolic

Further reading

M. D. Lutovac, D. V. Tošić, B. L. Evans

Filter Design for Signal Processing Using MATLAB and Mathematica

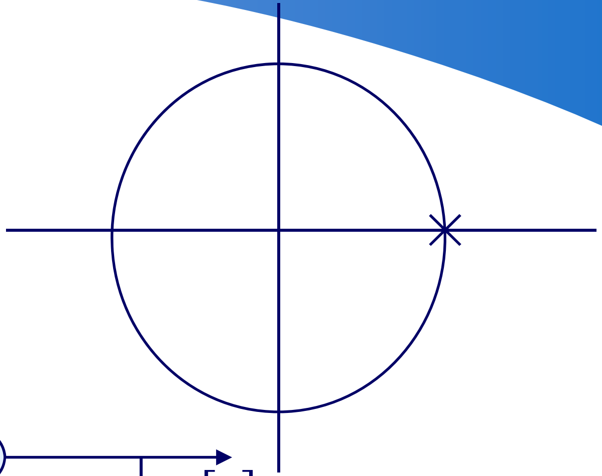
Prentice Hall
Upper Saddle River, New Jersey
ISBN 0-201-36130-2, (c) 2001

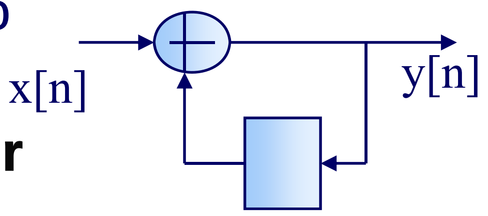

FILTER DESIGN FOR
SIGNAL PROCESSING
Using MATLAB and Mathematica
Miroslav D. Lutovac · Dejan V. Tošić · Brian L. Evans

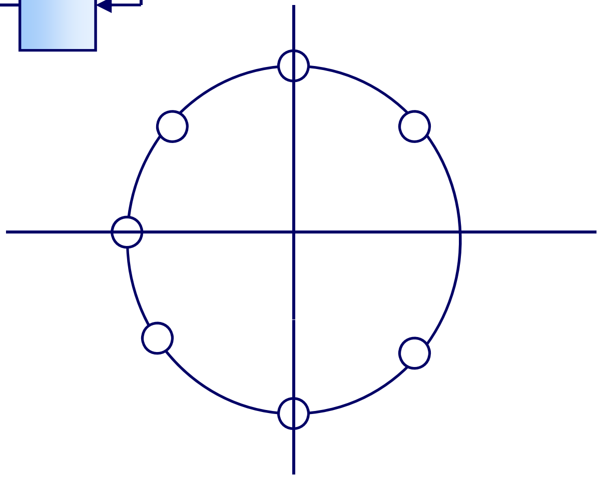
http://kondor.etf.bg.ac.yu/~lutovac/

Multiplierless filters

- **Integrator**
  $H(z) = 1/1-z^{-1}$
  - Integrator has a pole at z=1, so it is a kind of lowpass filter with no multiplier. Should be careful for overflow due to DC accumulation.
- **Moving average (MA) filter**
  $H(z) = (1-z^{-N})/(1-z^{-1})$
  - MA filter has many zeroes around the unit circle except for z=1. It is a lowpass filter. It can be modified to a bandpass or highpass filter.
  - $= 1+z^{-1}+z^{-2}+\ldots+z^{-(N-1)}$
- **Good for FPGA's with DSP blocks**

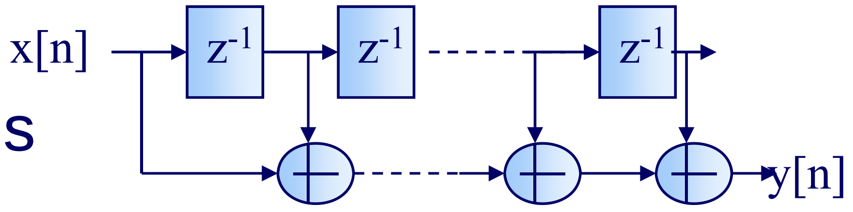


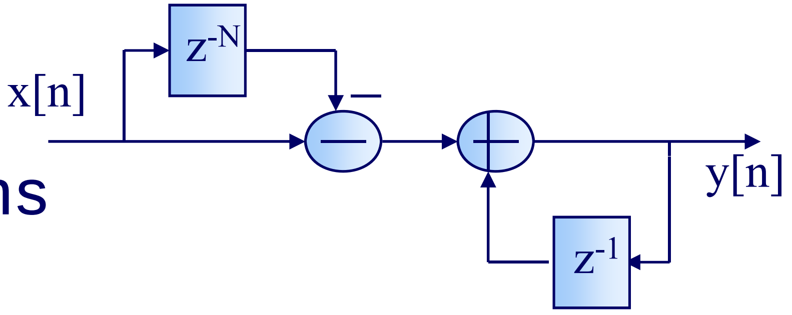




Two moving averager implementations

- **Feed-forward type:**
  - Higher number of arithmetic operations
  - Better stability
- **Feed-back type:**
  - Lower number of arithmetic operations
  - Should be careful against overflows





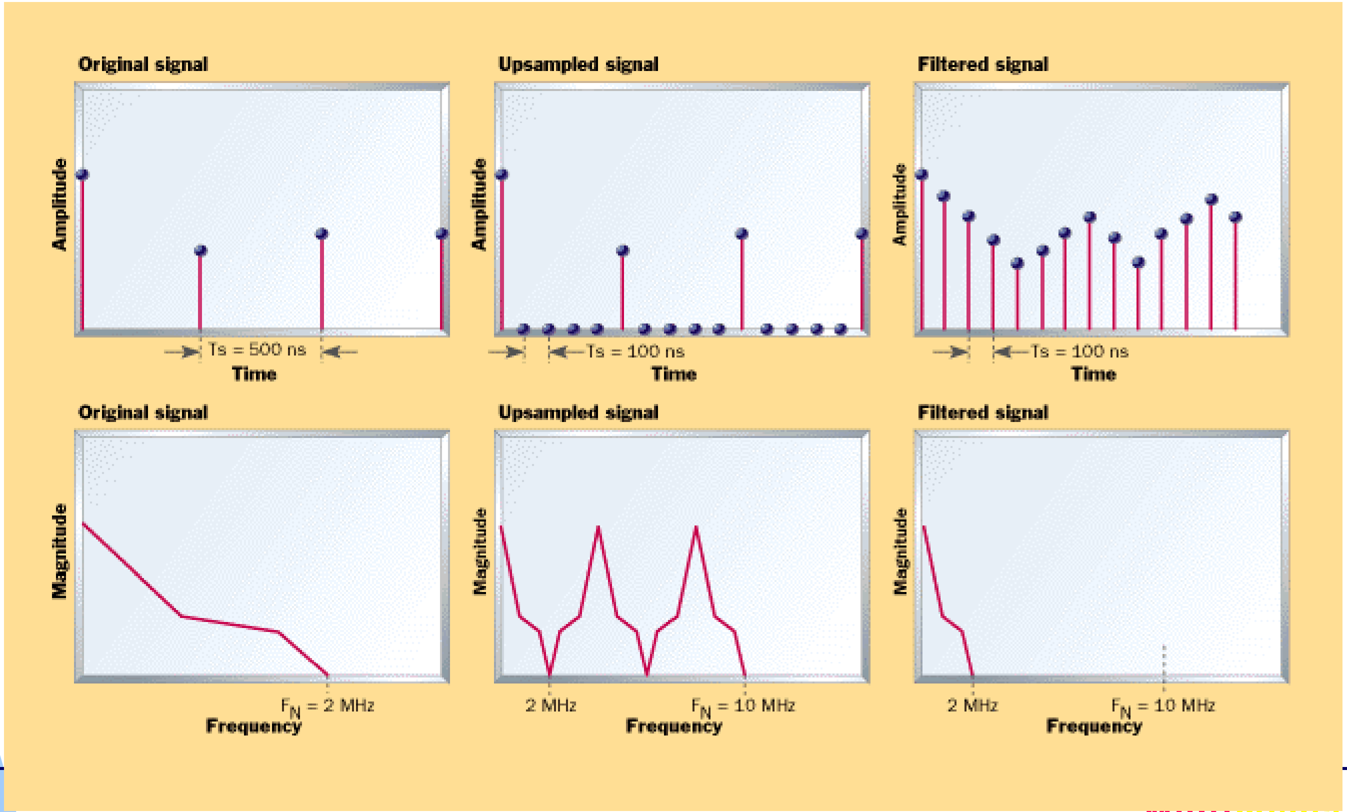
- **Both structures have the same frequency response with infinite precision arithmetic**

Key points

- **Different types and structures of digital filters may achieve the same or similar goals, but with different implementation costs, numerical properties**
- **Design space that can be explored**
  - Filter types: FIR, IIR, adaptive filters
  - Specifications/performance goals
  - Filter structures
    - Direct, parallel, cascade …
  - Numerical properties (floating point/fixed point, etc.)

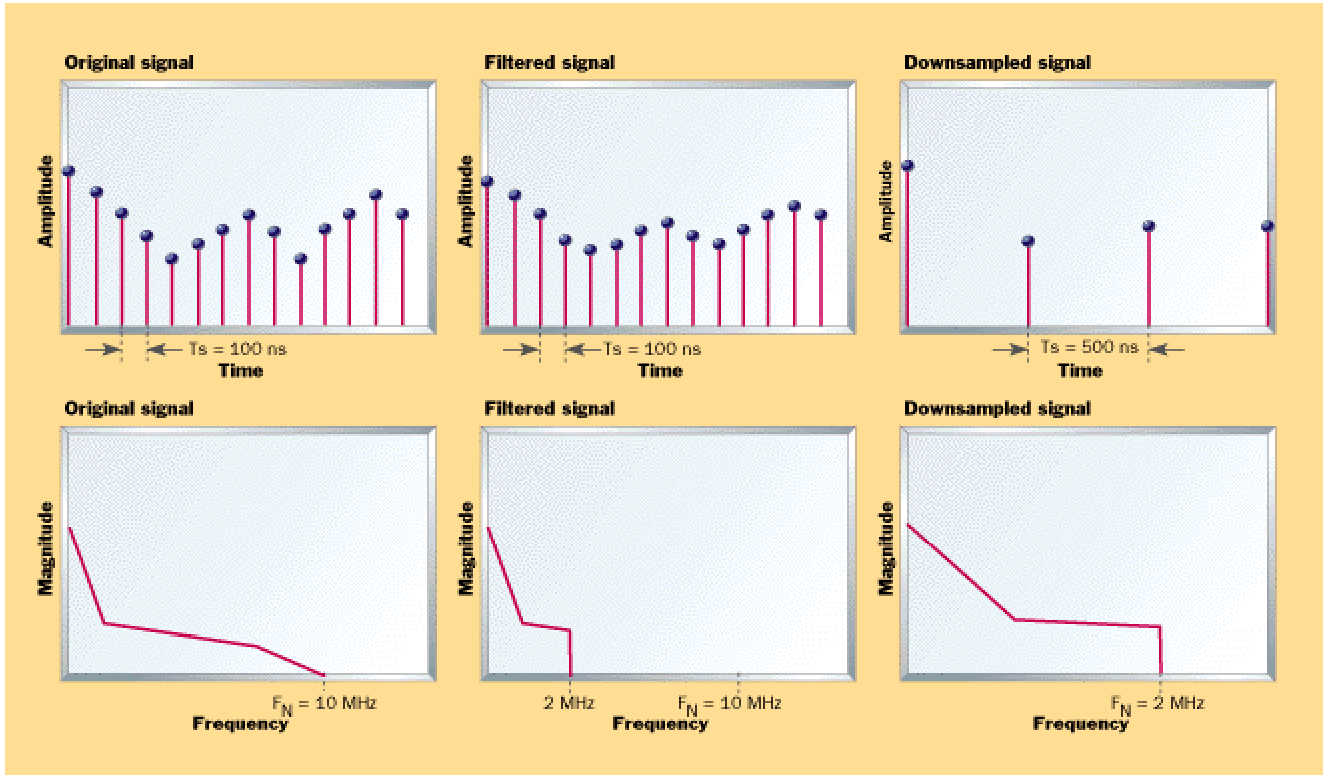
Interpolation (Sample rate increase)

- Insert N-1 zeroes between the samples, then conduct lowpass filtering (pi/N).
- Among N tap inputs, only one is non-zero (no need to compute N-1 mult.)



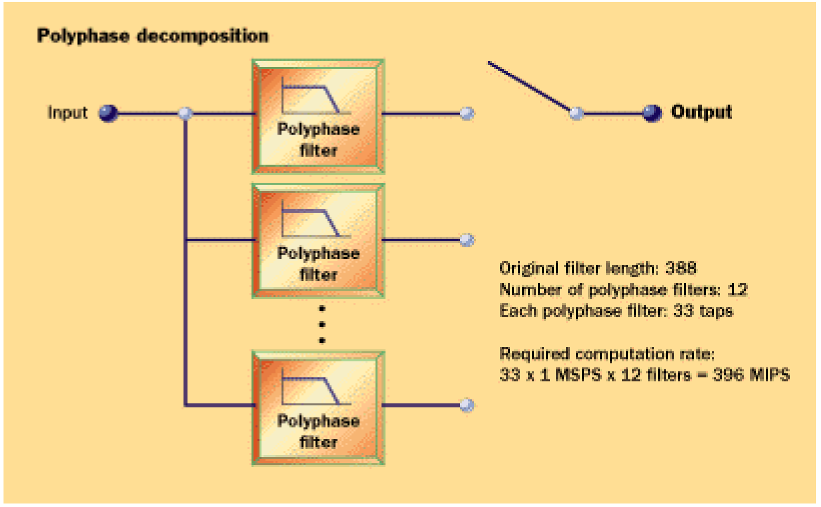
Decimation (Sample rate reduction)

- Conduct lowpass filtering (pi / N).
- Among N output samples, select only one. So, we do not need to compute N-1 output samples.
  - (This is not possible for recursive filters because of feedback!)



FIR filter implementation for decimation, interpolation

- For interpolation: (N-1) among N input values at the tap are zero! No need to conduct multiplications
- For decimation: We only need to compute 1 output sample at every N output.
- It is interpreted as polyphase filter with the tap length of about M/N.

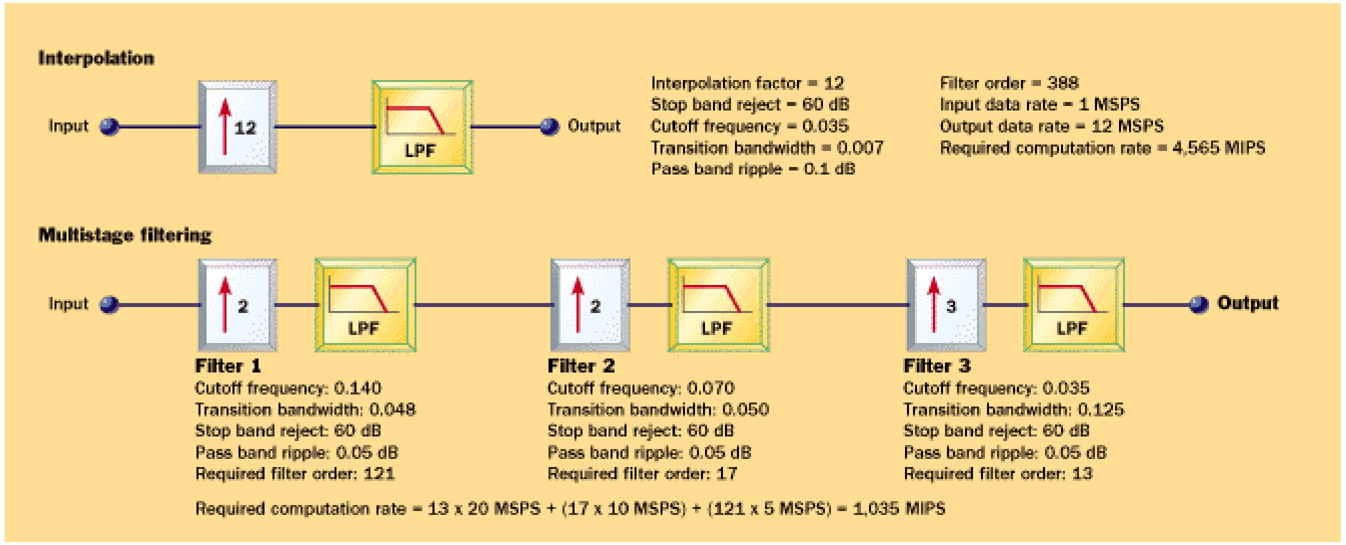

Polyphase decomposition
Input
Polyphase filter
Polyphase filter
Polyphase filter
Output
Original filter length: 388
Number of polyphase filters: 12
Each polyphase filter: 33 taps
Required computation rate:
33 x 1 MSPS x 12 filters = 396 MIPS

Here's an example of a 12-tap FIR filter that implements interpolation by a factor of four. The coefficients are h0–h11, and three data samples, x0–x2 (with the newest, x2, on the left) have made their way into the filter's delay line:

h0	h1	h2	h3	h4	h5	h6	h7	h8	h9	h10	h11	Result
x2	0	0	0	x1	0	0	0	x0	0	0	0	x2·h0+x1·h4+x0·h8
0	x2	0	0	0	x1	0	0	0	x0	0	0	x2·h1+x1·h5+x0·h9
0	0	x2	0	0	0	x1	0	0	0	x0	0	x2·h2+x1·h6+x0·h10
0	0	0	x2	0	0	0	x1	0	0	0	x0	x2·h3+x1·h7+x0·h11

FIR filtering for narrowband signal

- ❖ Narrow band filtering (large order needed): It is advantageous to conduct decimation followed by post-filtering (operates at low sampling rate) -> the total number of mult's is reduced.
- ❖ Application to IF filtering for digital radio



Interpolation factor = 12
Stop band reject = 60 dB
Cutoff frequency = 0.035
Transition bandwidth = 0.007
Pass band ripple = 0.1 dB

Filter order = 388
Input data rate = 1 MSPS
Output data rate = 12 MSPS
Required computation rate = 4,565 MIPS

Filter 1
Cutoff frequency: 0.140
Transition bandwidth: 0.048
Stop band reject: 60 dB
Pass band ripple: 0.05 dB
Required filter order: 121

Filter 2
Cutoff frequency: 0.070
Transition bandwidth: 0.050
Stop band reject: 60 dB
Pass band ripple: 0.05 dB
Required filter order: 17

Filter 3
Cutoff frequency: 0.035
Transition bandwidth: 0.125
Stop band reject: 60 dB
Pass band ripple: 0.05 dB
Required filter order: 13

Required computation rate = 13 x 20 MSPS + (17 x 10 MSPS) + (121 x 5 MSPS) = 1,035 MIPS

Example with SSB generation

- What if the carrier frequency for AM in HW#1 is 2,048KHz. Assume that the sampling frequency for the carrier is 4*2,048KHz.
- Design the bandpass filter for removing the lower sideband.
- Assume that the message signal has the sampling frequency of 8KHz. Design the interpolation filter for interpolating the message signal.
- Consider a heterodyne modulator, where the 1st stage modulation frequency is 12KHz. The generated SSB is, then, modulated to 2,048KHz.

Discussion

- FIR filter can be implemented using direct form or fast convolution methods like FFT.
- IIR filters are often factored into products (cascade realization) or sum (parallel realization) of 1st order or 2nd order sections due to numerical concerns.
- There are numerous possible realization structures of the same digital filters.
- Digital filter coefficients are not always exact. It is possible to realize a digital filter with the same desired properties but different filter structures and coefficients to exploit favorable implementation alternatives. A state space model is a good example.

Quantization error, stability, overflow

- **An IIR filter is BIBO stable if all the poles of its transfer function H(z) lie within the unit circle in z-plane.**
- **A stable IIR digital filter may become unstable when its coefficients are subject to severe quantization due to finite length of registers.**
- **Hence an IIR filter that is stable when designed with a 32 bit machine may become unstable when implemented with an 8-bit micro-controller!**

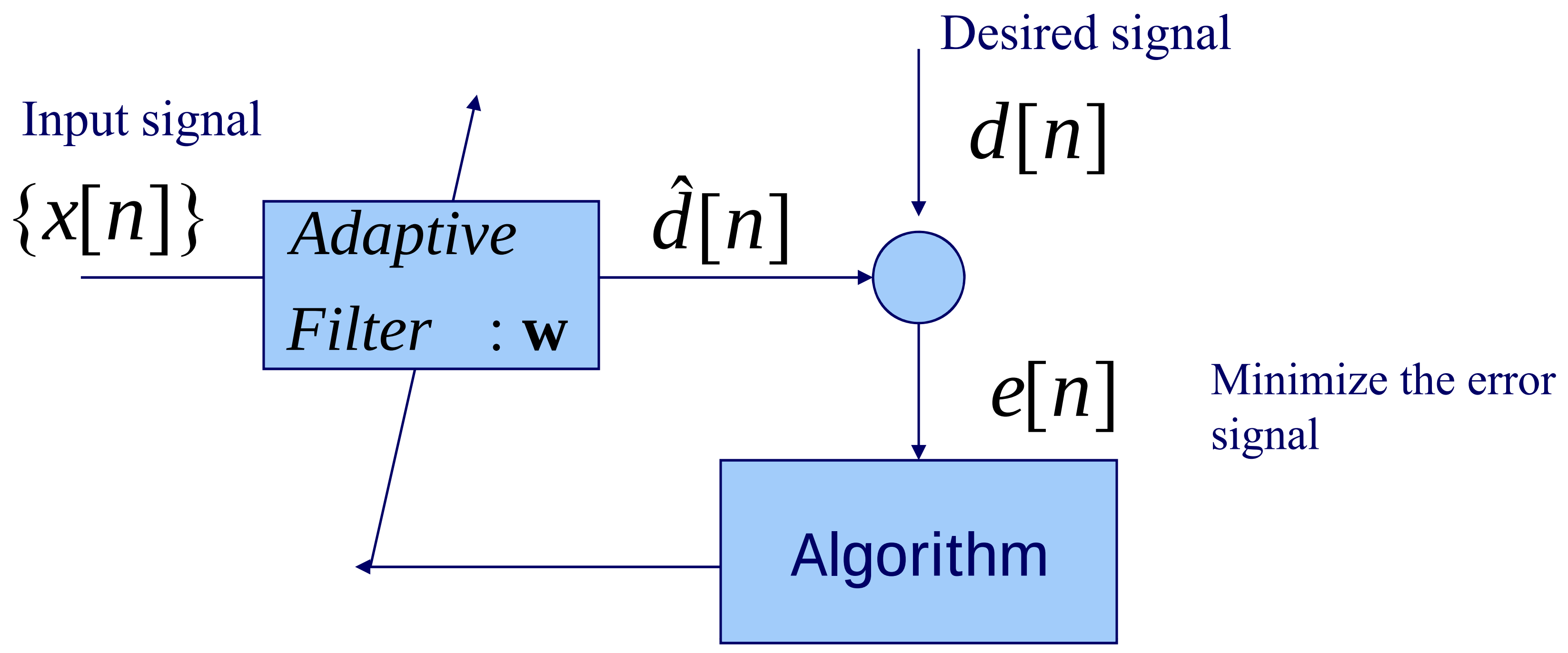
- **Overflow**
  - Dynamic range of intermediate results must be bounded.
  - Otherwise, overflow check must be used and that is very costly.
  - Saturation arithmetic may reduce the error caused by overflow.

Adaptive filter and adaptive signal processing

- **Changing the filter coefficients for adapting to the environment or signal change.**
- **The filter coefficients are changing.**
- **Fixed-approach:**
  - If the channel were fixed then a possible solution could be based on the *Wiener filter* approach
  - We need to know in such case the correlation matrix of the transmitted signal and the cross correlation vector between the input and desired response.
- **When the filter is operating in an unknown environment these required quantities need to be found from the accumulated data.**
  - Continuous adaptation is needed.
  - LMS and RLS, ...

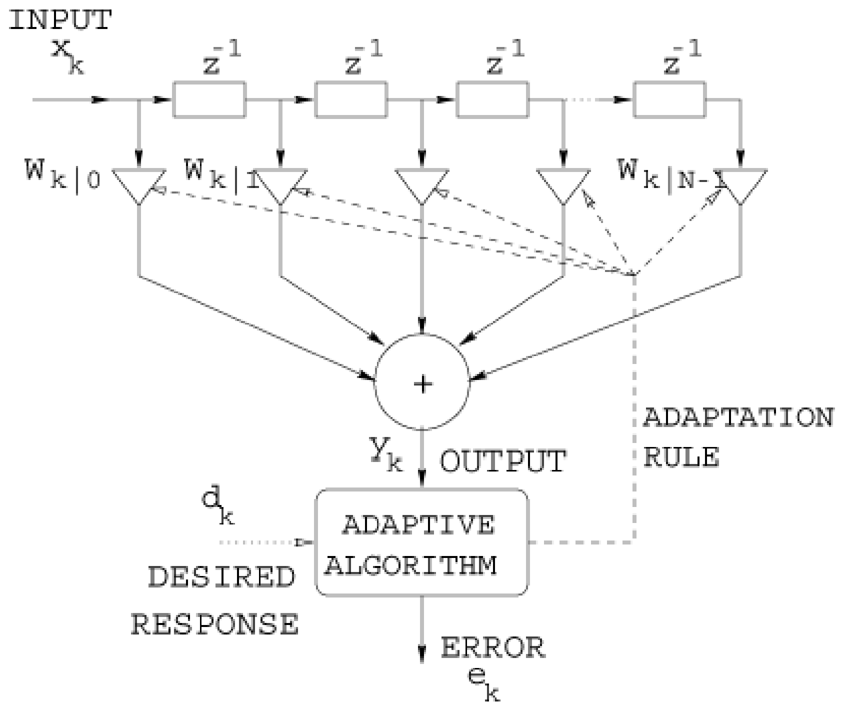
Adaptive signal processing

❖ **A possible framework is:**

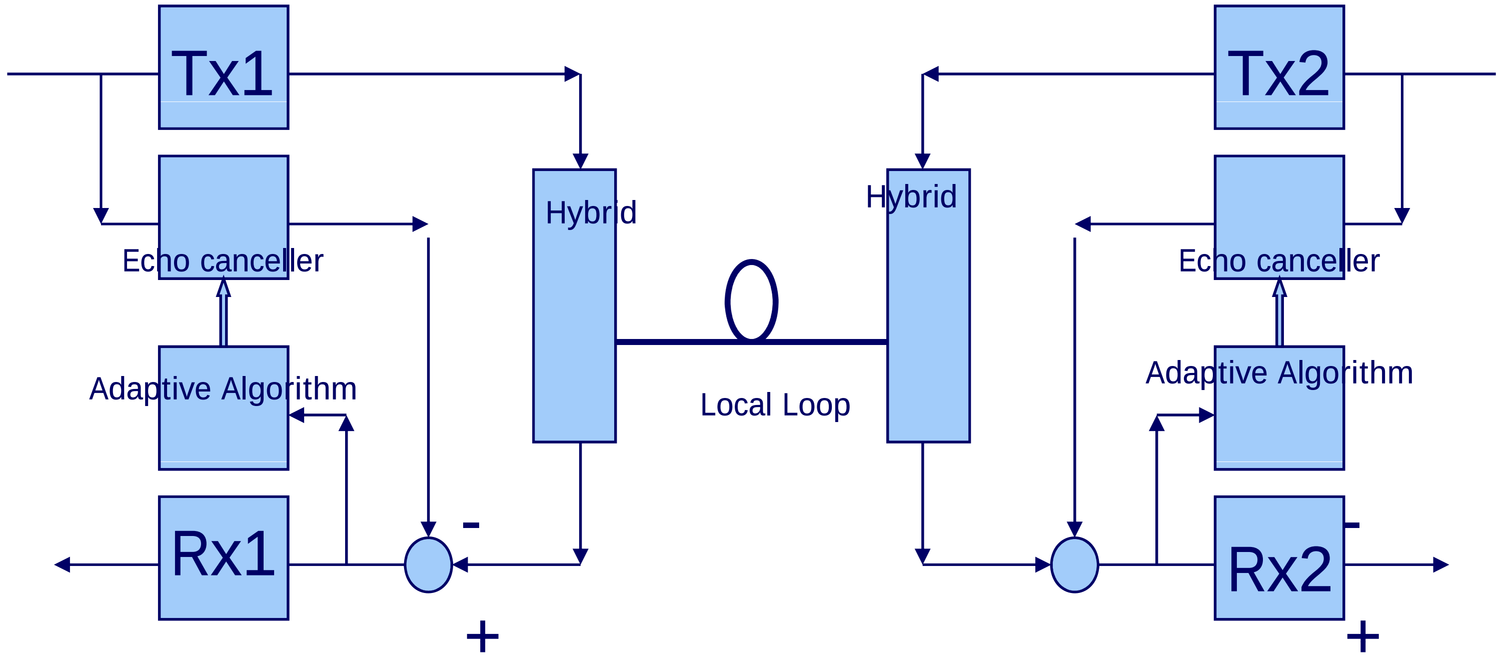


Applications of adaptive filters

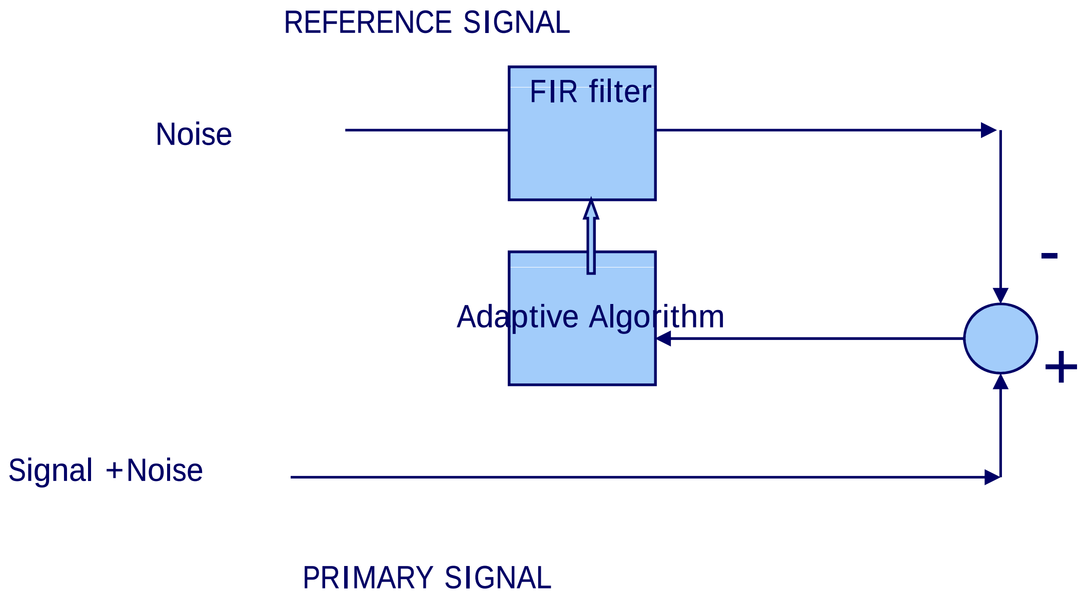
- **Applications are many**
  - Digital Communications
  - Channel Equalisation
  - Adaptive noise cancellation
  - Adaptive echo cancellation
  - System identification
  - Smart antenna systems
  - Blind system equalisation
  - And many, many others



Echo cancellers in local loops

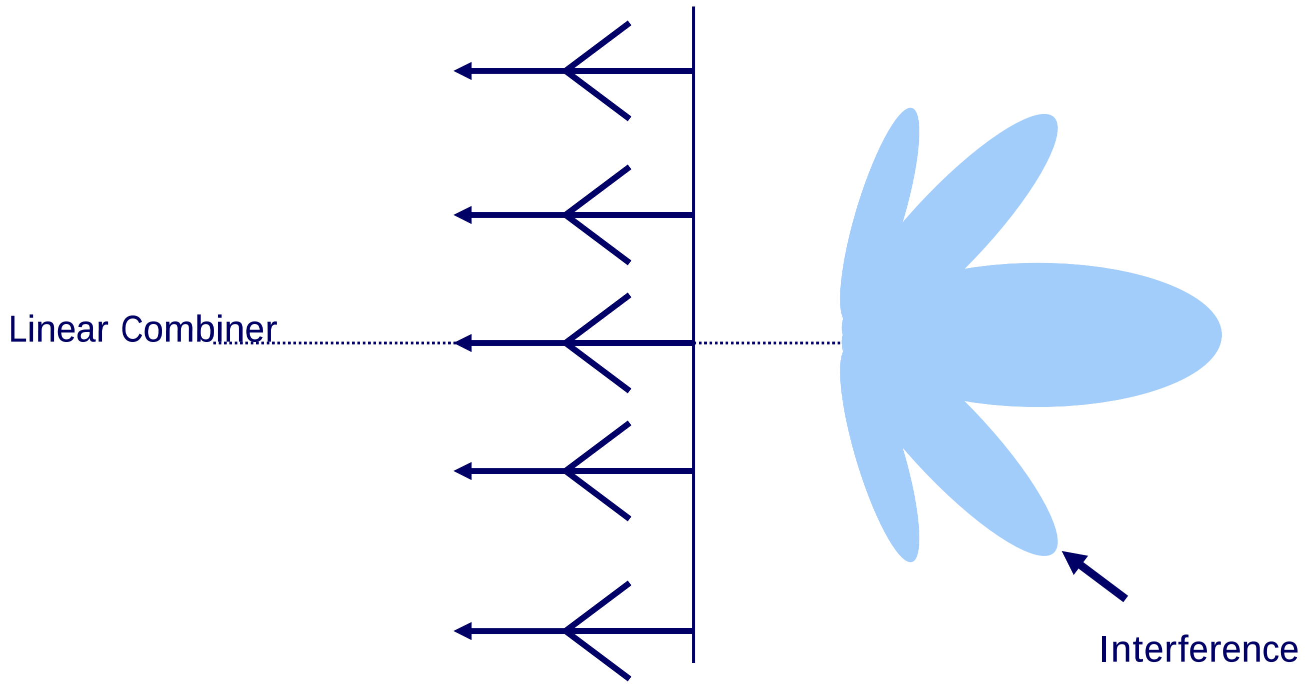


Adaptive noise canceller



Adaptive signal processing

❖ **Adaptive Arrays**



Adaptive signal processing

- Basic principles:
- 1) Form an objective function (performance criterion)
- 2) Find gradient of objective function with respect to FIR filter weights
- 3) There are several different approaches that can be used at this point
- 3) Form a differential/difference equation from the gradient.

Adaptive signal processing

- Let the desired signal be $d[n]$
- The input signal $x[n]$
- The output $y[n]$
- Now form the vectors

- So that

$$\mathbf{x}[n] = [x[n] \quad x[n-1] \quad . \quad x[n-m+1]]^T$$

$$\mathbf{h} = [h[0] \quad h[1] \quad . \quad h[m+1]]^T$$

$$y[n] = \mathbf{x}[n]^T \mathbf{h}$$

- Objective function

$$J(\mathbf{w}) = E\{[d[n] - y[n]]^2\}$$

$$J(\mathbf{w}) = \sigma_d^2 - \mathbf{p}^T\mathbf{h} - \mathbf{h}^T\mathbf{p} + \mathbf{h}^T\mathbf{R}\mathbf{h}$$

$$\mathbf{p} = E\{\mathbf{x}[n]d[n]\} \quad \mathbf{R} = E\{\mathbf{x}[n]\mathbf{x}[n]^T\}$$

Adaptive signal processing

- We wish to minimise this function at the instant ***n***
- Using ***Steepest Descent*** we write
- But

$$\mathbf{h}[n+1] = \mathbf{h}[n] - \frac{1}{2}\mu \frac{\partial J(\mathbf{h}[n])}{\partial \mathbf{h}[n]}$$

$$\frac{\partial J(\mathbf{h})}{\partial \mathbf{h}} = -2\mathbf{p} + 2\mathbf{R}\mathbf{h}$$

Adaptive signal processing

- So that the *"weights update equation"*

$$\mathbf{h}[n+1] = \mathbf{h}[n] + \mu(\mathbf{p} - \mathbf{R}\mathbf{h}[n])$$

- Since the objective function is quadratic this expression will converge in ***m*** steps
- The equation is not practical
- If we knew $\mathbf{R}$ and $\mathbf{p}$ a priori we could find the required solution (Wiener) as

$$\mathbf{h}_{opt} = \mathbf{R}^{-1}\mathbf{p}$$

Adaptive signal processing

- However these matrices are not known
- Approximate expressions are obtained by ignoring the expectations in the earlier complete forms

- This is very crude. However, because the update equation accumulates such quantities, progressive we expect the crude form to improve

$$\hat{\mathbf{R}}[n] = \mathbf{x}[n]\mathbf{x}[n]^T \qquad \hat{\mathbf{p}}[n] = \mathbf{x}[n]d[n]$$

The LMS algorithm

- Thus we have

$$\mathbf{h}[n+1] = \mathbf{h}[n] + \mu \mathbf{x}[n](d[n] - \mathbf{x}[n]^T \mathbf{h}[n])$$

- Where the error is

$$e[n] = (d[n] - \mathbf{x}[n]^T \mathbf{h}[n]) = (d[n] - y[n])$$

- And hence can write

$$\mathbf{h}[n+1] = \mathbf{h}[n] + \mu \mathbf{x}[n] e[n]$$

- This is sometimes called the stochastic gradient descent

Convergence

The parameter μ is the ***step size***, and it should be selected carefully

- If too small it takes too long to converge, if too large it can lead to instability

$$0 < \mu < \frac{2}{\lambda_{\max}}$$

Convergence

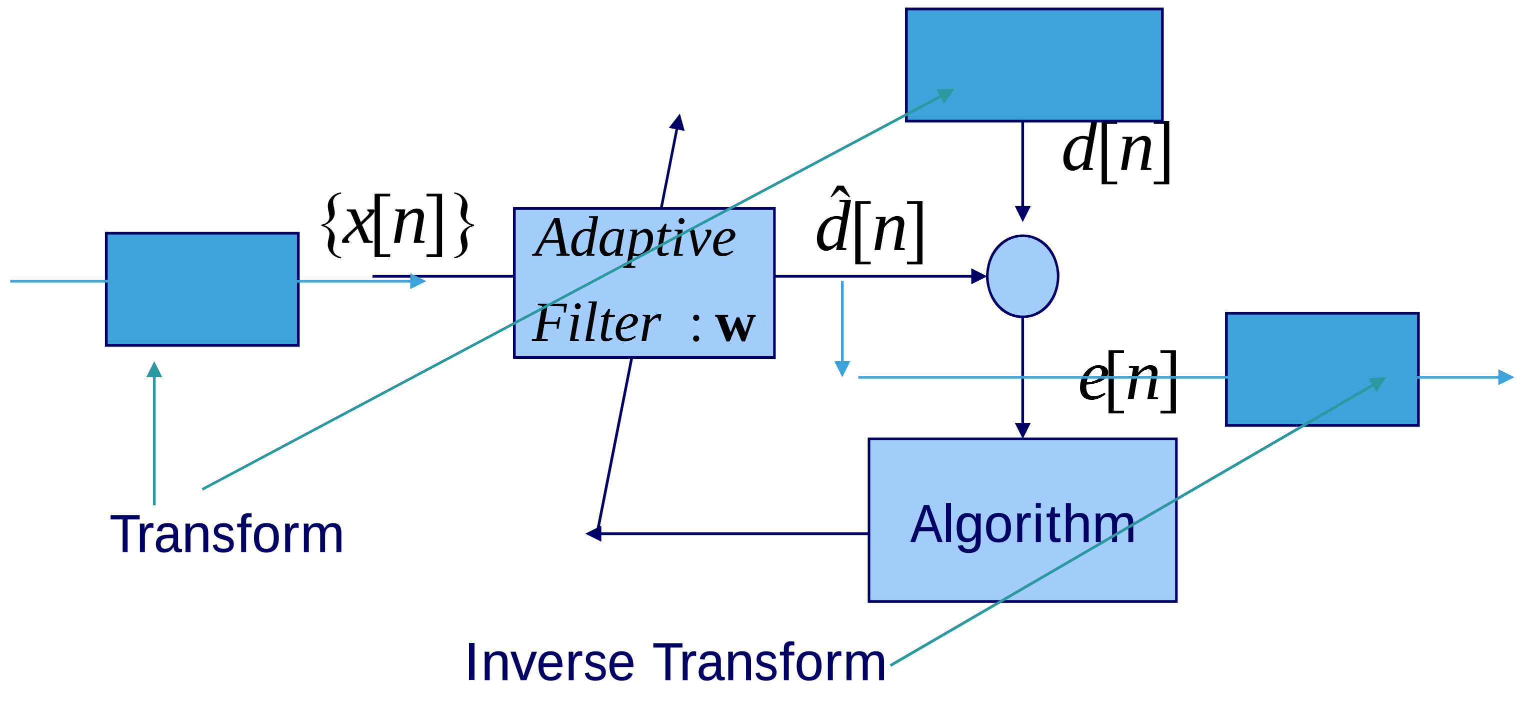
- We require that
- Or

$$\left|1-\mu\lambda_{\max}\right|<1$$

- In practice we take a much smaller value than this

$$0<\mu<\frac{2}{\lambda_{\max}}$$

Transform based LMS



Least squares adaptive

- with

$$\mathbf{R}[n] = \sum_{i=1}^{n} \mathbf{x}[i]\mathbf{x}[i]^T$$

- We have the Least Squares solution

$$\mathbf{p}[n] = \sum_{i=1}^{n} \mathbf{x}[n]d[n]$$

- However, this is computationally very intensive to implement.
- Alternative forms make use of recursive estimates of the matrices involved.

$$\mathbf{h}[n] = \mathbf{R}[n]^{-1}\mathbf{p}[n]$$

Recursive Least Squares

- Firstly we note that

$$\mathbf{p}[n] = \mathbf{p}[n-1] + \mathbf{x}[n]d[n]$$

- We now use the Inversion Lemma (or the Sherman-Morrison formula)
- Let

$$\mathbf{R}[n] = \mathbf{R}[n-1] + \mathbf{x}[n]\mathbf{x}[n]^T$$

Recursive Least Squares (RLS)

- Let

$$\mathbf{P}[n] = \mathbf{R}[n]^{-1}$$

- Then

$$\mathbf{k}[n] = \frac{\mathbf{R}[n-1]^{-1}\mathbf{x}[n]}{1 + \mathbf{x}[n]^T \mathbf{R}[n-1]^{-1}\mathbf{x}[n]}$$

- The quantity is known as the *Kalman gain*

$$\mathbf{P}[n] = \mathbf{R}[n-1] - \mathbf{k}[n]\mathbf{x}^T[n]\mathbf{P}[n-1]$$

$$\mathbf{k}[n]$$

Recursive Least Squares

- Now use $\mathbf{k}[n] = \mathbf{P}[n]\mathbf{x}[n]$ in the computation of the filter weights

$$\mathbf{h}[n] = \mathbf{P}[n]\mathbf{p}[n] = \mathbf{P}[n](\mathbf{p}[n-1] + \mathbf{x}[n]d[n])$$

- From the earlier expression for $\mathbf{P}[n]$ updates we have

$$\mathbf{P}[n]\mathbf{p}[n-1] = \mathbf{P}[n-1]\mathbf{p}[n-1] - \mathbf{k}[n]\mathbf{x}^T[n]\mathbf{P}[n-1]\mathbf{p}[n-1]$$

- And hence
- $$\mathbf{h}[n] = \mathbf{h}[n-1] + \mathbf{k}[n](d[n] - \mathbf{x}[n]^T\mathbf{h}[n-1])$$

Transforms

School of Electrical Engineering
Seoul National University

Linear transformations

- **Signals represented in the <u>frequency domain</u> have different properties that can be exploited to facilitate efficient digital signal processing**
- **A linear transform is a mapping that converts time domain (or spatial domain) digital signal into frequency domain coefficients.**
- **A linear transform operates on the entire sequence of digital signals.**
- **Linearity: Let T[x] be the linear transform of signal sequence x. Then for arbitrary constant *a, b***

$$T[ax_1 + bx_2] = aT[x_1] + bT[x_2]$$

- **Types of linear transforms**
  - DFT: discrete Fourier transform
  - DCT: discrete cosine transform
  - DWT: discrete wavelet transform
  - KLT: Optimal linear transform

Matrix Formulations

1D linear transform

Represent a finite 1D sequence by a column vector:

$$\underline{X} = \begin{bmatrix} x[1] & x[2] & \cdots & x[N] \end{bmatrix}^T$$

The 1D linear transform can be represented as a matrix-vector product:

$$\underline{Y} = \mathbf{T} \cdot \underline{X}$$

where T is a N × N matrix whose elements may be complex-valued.

2D linear transform

Often consider 2D <u>separable</u> linear transform. That is a 1D linear transform is first applied to each row of the 2D signal, and then a second 1D transform is applied to each column of the transformed signal. It consists of two consecutive matrix-matrix product:

U and V are the transformation matrices

$$\mathbf{Y} = \mathbf{U} \cdot \mathbf{X} \cdot \mathbf{V}$$

Discrete Fourier Transform

The 1D DFT is defined as:

$$Y(k) = \sum_{m=0}^{N-1} X(m) W_N^{mk}$$

for 0 ≤ m ≤ N–1, where the data {X(m); 0 ≤ m ≤ N–1} may be real-valued or complex-valued, and

Requires N^2 complex-valued MAC operations, or $4N^2$ real-valued MAC operations.

Fast Fourier Transform (FFT) can reduce the DFT computation to O(N log N)

$$W_N = \exp(-j2\pi / N)$$

Each complex-valued arithmetic:

$$(a + jb)(c + jd) = (ac - bd) + j(bc + ad)$$

Requires 4 real-valued multiplications and two additions.

Half if one operand is a real number.

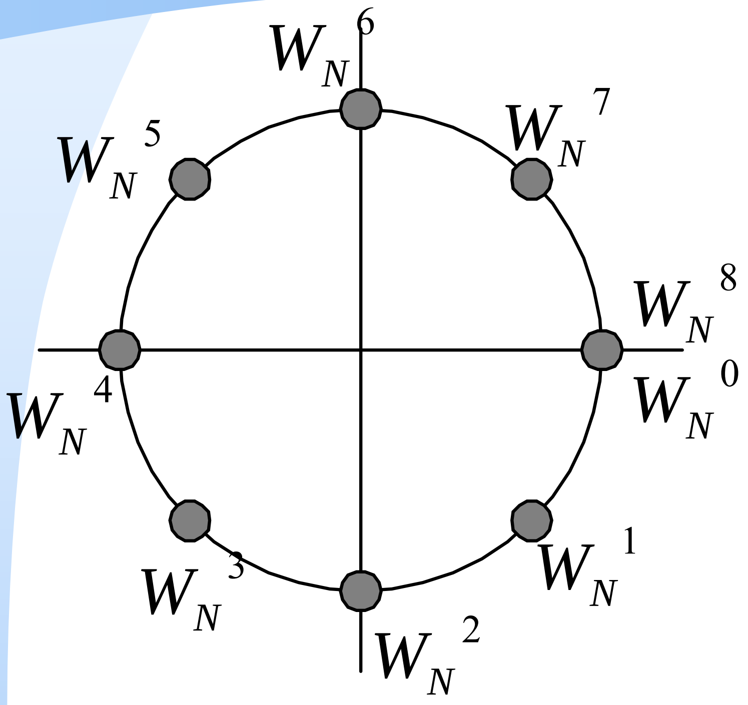
Special arithmetic algorithms such as CORDIC can be used to implement complex-valued multiplication effectively.

$$\begin{cases} \text{DFT}: \; X[k] = \sum_{n=0}^{N-1} x[n] \cdot e^{-jk\frac{2\pi}{N}n}, \quad k = 0,1,\cdots,N-1 \\ \text{IDFT}: x[n] = \frac{1}{N} \cdot \sum_{k=0}^{N-1} X[k] \cdot e^{jk\frac{2\pi}{N}n}, \quad n = 0,1,\cdots,N-1 \end{cases}$$

- Use Notation $\underline{W_N = e^{-j\frac{2\pi}{N}}}$ then,

$$\begin{cases} \text{DFT}: \; X[k] = \sum_{n=0}^{N-1} x[n] \cdot W_N^{kn}, \quad k = 0,1,\cdots,N-1 \\ \text{IDFT}: x[n] = \frac{1}{N} \cdot \sum_{k=0}^{N-1} X[k] \cdot W_N^{-kn}, \quad n = 0,1,\cdots,N-1 \end{cases}$$

- What's the meaning of W_N^{kn} ????



the kn$_{th}$ point among the N points on the unit circle counted from the origin clockwise in circular manner.

(NOTE) $W_N^{N}=1, \quad W_N^{N/2}=W_2^{1}=-1,$

$$W_N^{N/P}=W_P^{1} \ \left(=e^{-j\frac{2\pi}{N}\cdot\frac{N}{P}}=e^{-j\frac{2\pi}{P}}\right)$$

2. Decimation-in-Time Factorization

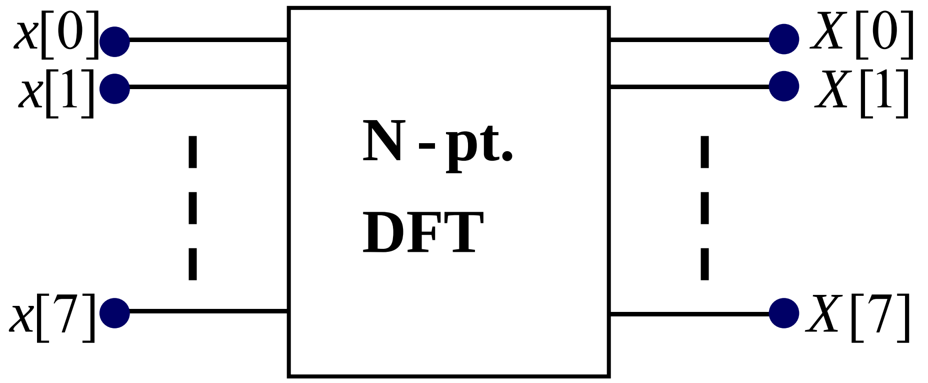
$$X[k] = \underbrace{\sum_{n=0}^{N-1} x[n] \cdot W_N^{kn}}_{N-pt.DFT}, \quad k = 0,1,\cdots,N-1 \quad (N = 2^{\upsilon})$$

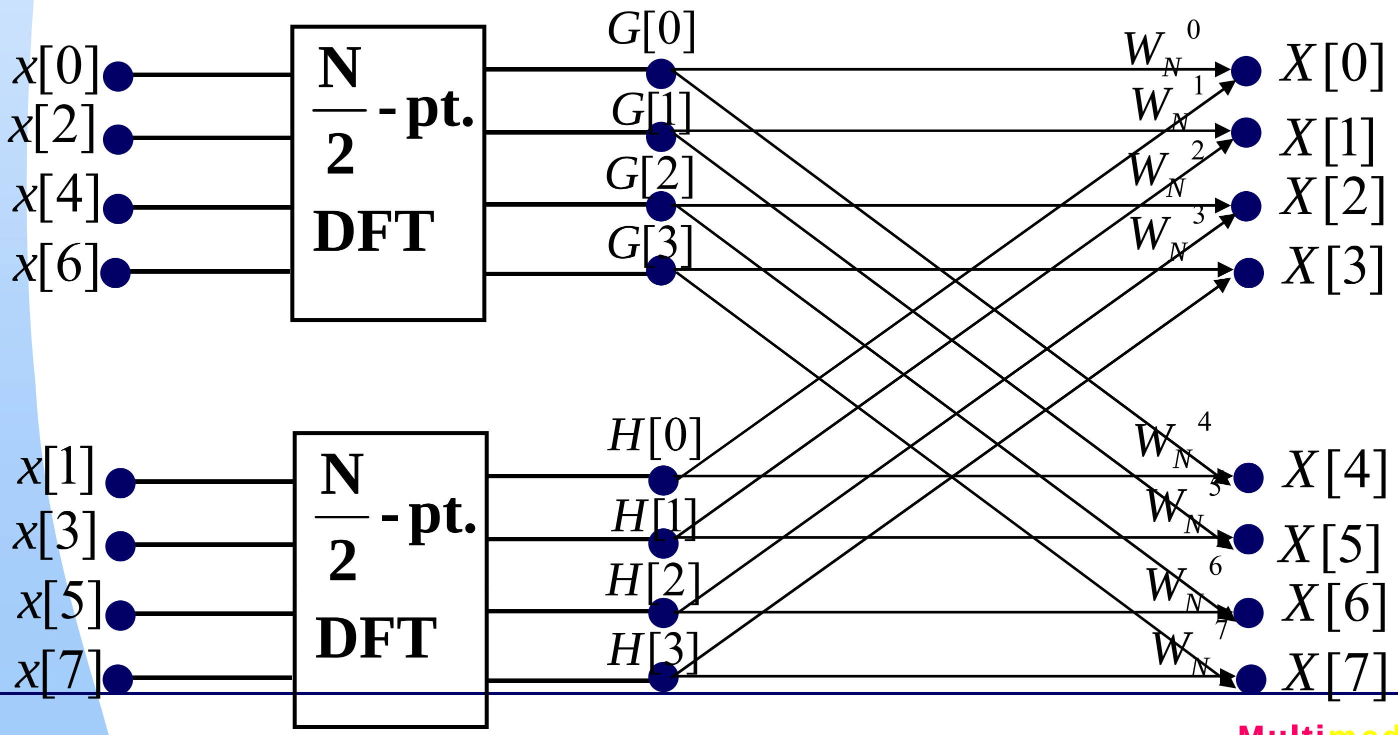
$$= \sum_{n,\,even} + \sum_{n,\,odd}$$

$$= \sum_{r=0}^{N/2-1} x[2r] W_N^{2rk} + \sum_{r=0}^{N/2-1} x[2r+1] W_N^{(2r+1)k}$$

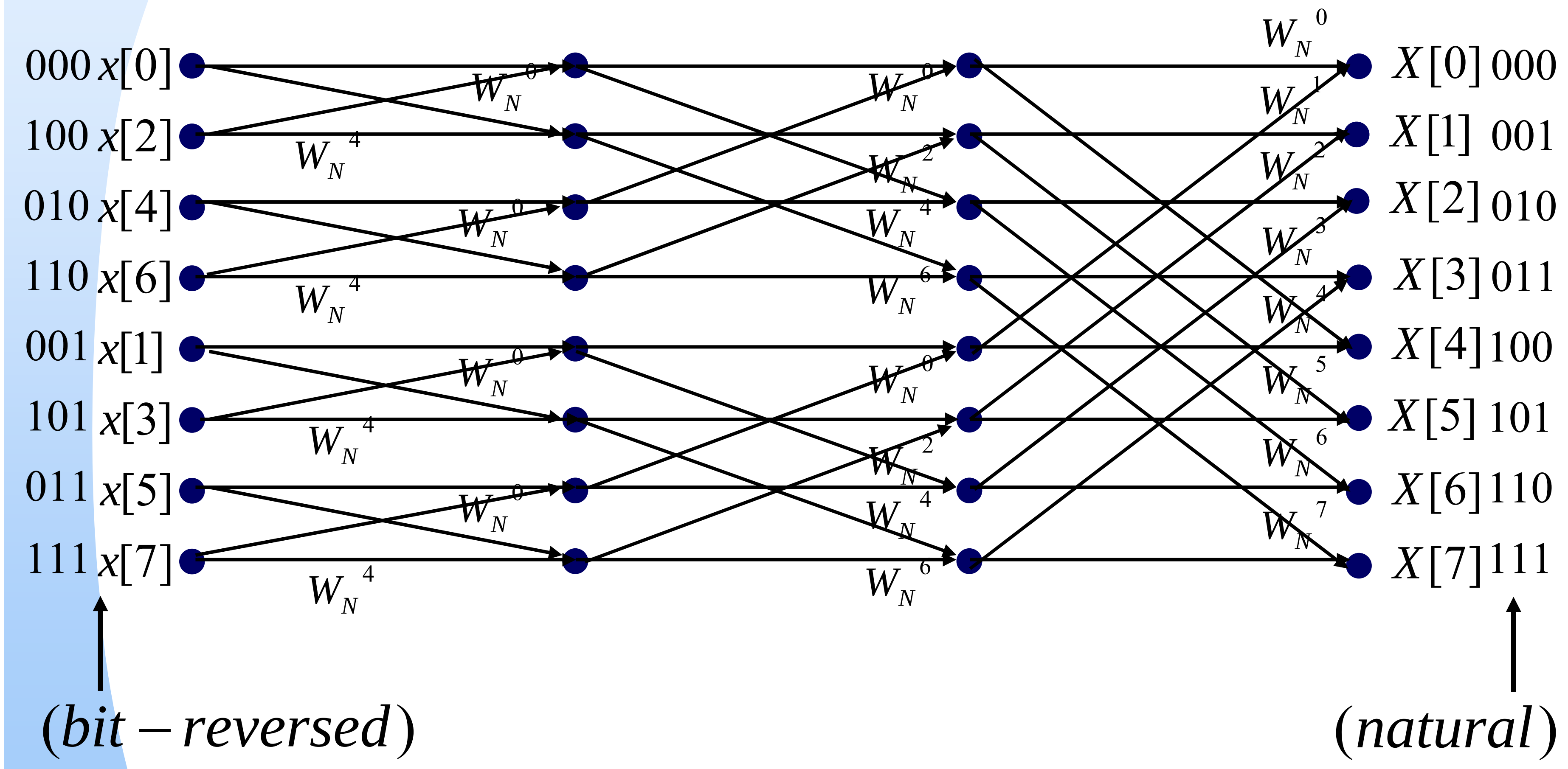
$$= \underbrace{\sum_{r=0}^{N/2-1} x[2r] W_{N/2}^{rk}}_{N/2-pt.DFT,G(k)} + W_N^{k} \cdot \underbrace{\sum_{r=0}^{N/2-1} x[2r+1] W_{N/2}^{rk}}_{N/2-pt.DFT,H(k)}$$

$(eg)\ N = 8$


$x[0]$
$x[1]$
$x[7]$
N - pt.
DFT
$X[0]$
$X[1]$
$X[7]$


$x[0]$
$x[2]$
$x[4]$
$x[6]$
$\frac{N}{2}$ - pt.
DFT
$G[0]$
$G[1]$
$G[2]$
$G[3]$
$x[1]$
$x[3]$
$x[5]$
$x[7]$
$\frac{N}{2}$ - pt.
DFT
$H[0]$
$H[1]$
$H[2]$
$H[3]$
W_N^0
W_N^1
W_N^2
W_N^3
W_N^4
W_N^5
W_N^6
W_N^7
$X[0]$
$X[1]$
$X[2]$
$X[3]$
$X[4]$
$X[5]$
$X[6]$
$X[7]$

Final flow graph



Fast Fourier Transform

Decimation in time formulation

```
Function Y = fft(N,x)
If N==1, Y = x;
Else
    xeven=[x(0)x(2)… x(N-2)];
    xodd=[x(1) x(3) … x(N-1)];
    Yeven=fft(N/2,xeven);
    Yodd=fft(N/2,xodd);
    For k=0:N-1,
         Y(k)=Yeven(k mod N/2)
         + W^k*Yodd(k mode N/2);
    end
end
```

- **If L(N) = # of ops for Npt FFT, and N = 2^m, then**

  L(N) = N + 2*L(N/2)

  = N + 2*[N/2 + 2*L(N/2^2)]

  = 2N + 2^2*L(N/2^2)

  = … = mN + 2^m L(N/2^m)

  = (m+1)N ~ O(N $\log_2$N)

- **Basic computation unit: twilde factor (each operation)**
  - 4 multiply, 4 addition

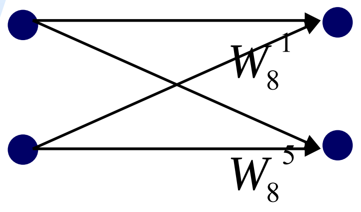
$$\begin{bmatrix} p_r \\ p_i \end{bmatrix} = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix} \begin{bmatrix} q_r \\ q_i \end{bmatrix} + \begin{bmatrix} s_r \\ s_i \end{bmatrix}$$

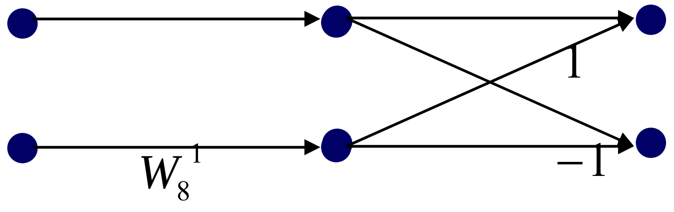
$$\theta = 2\pi k / N$$

• Remarks

- cascaded processing : # stages = $\upsilon = \log_2 N$

- butterflies : # /stage = N/2





- # computations (complex)

$$\begin{cases} \text{multiplications : 2/butterfly} \rightarrow \text{1/butterfly} \\ \text{additons : 2/butterfly} \rightarrow \text{2/butterfly} \end{cases}$$

total # of computations (complex)

$$\begin{cases} \text{multiplications : } 1 \cdot \text{N/2} \cdot \upsilon = \frac{N}{2} \log_2 N \\ \text{additons : } 2 \cdot \text{N/2} \cdot \upsilon = N \log_2 N \end{cases}$$

- in-phase compuations

- input data ordering : bit-reversed.

3. Decimation-in-frequency Factorization

(Sande- Tuckey) $\longrightarrow$ **FFT**

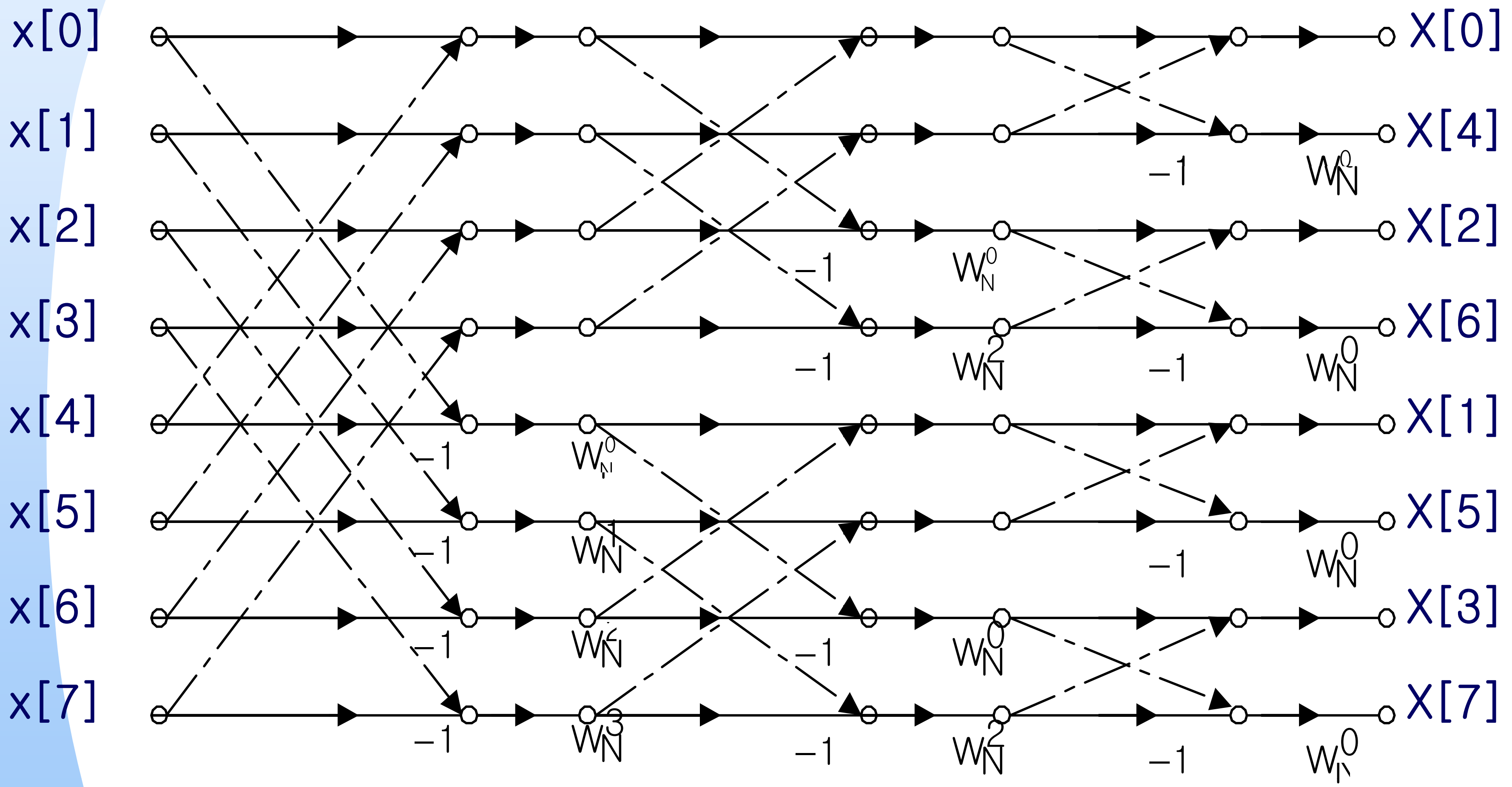
$$X[k] = \sum_{n=0}^{N-1} x[n] W_N^{nk}, \qquad k = 0,1,\ldots, N-1, \qquad N = 2^{\nu}$$

$$= \sum_{n=0}^{\frac{N}{2}-1} x[n] W_N^{nk} + \sum_{n=\frac{N}{2}}^{N-1} x[n] W_N^{nk}$$

$$= \sum_{n=0}^{\frac{N}{2}-1} x[n] W_N^{nk} + W_N^{\frac{kN}{2}} \sum_{n=0}^{\frac{N}{2}-1} x[n+\frac{N}{2}] W_N^{nk}$$

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

Final flow graph



-Remarks

- # Stages : $\nu = \log_2 N$
- # butterflies : $\dfrac{N}{2}$
- # computations : $\dfrac{N}{2}\log_2 N \,(complex\ )$
- inplace computations
- output data ordering : bit-reversed

-Question

The flow graph for D-I-F is obtained by reversing.

The direction of the flow graph for D-I-T. Why?

-Omit Sections 9.5-9.7

4. Applications of FFT

(1) Spectrum Analysis

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

  is the spectrum of x[n] , n=0,1,…,N-1

- Inverse transform can be done through the same mechanism

  i) Take the complex conjugate of X[k]

  ii) Pass it through the FFT process,

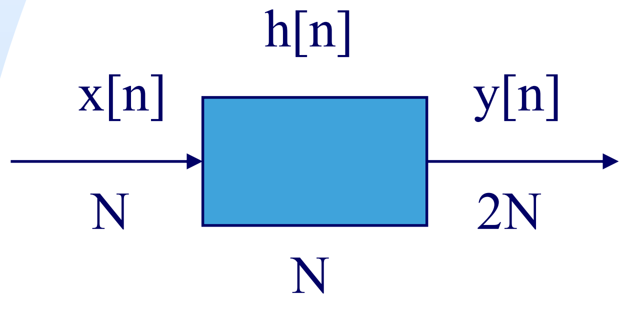
  But with one shift right(/2) operation at each stage

  iii) Finally, take the complex conjugate of the result

$$x[n] = \frac{1}{N} \sum_{k=0}^{N-1} X[k] W_N^{-nk} = [\frac{1}{2^{\nu}} \sum_{k=0}^{N-1} X^*[k] W_N^{nk}]^*$$

- Operation reduction : $N^2 \rightarrow \frac{N}{2}\log_2 N$

(2) Convolution (Filtering)



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

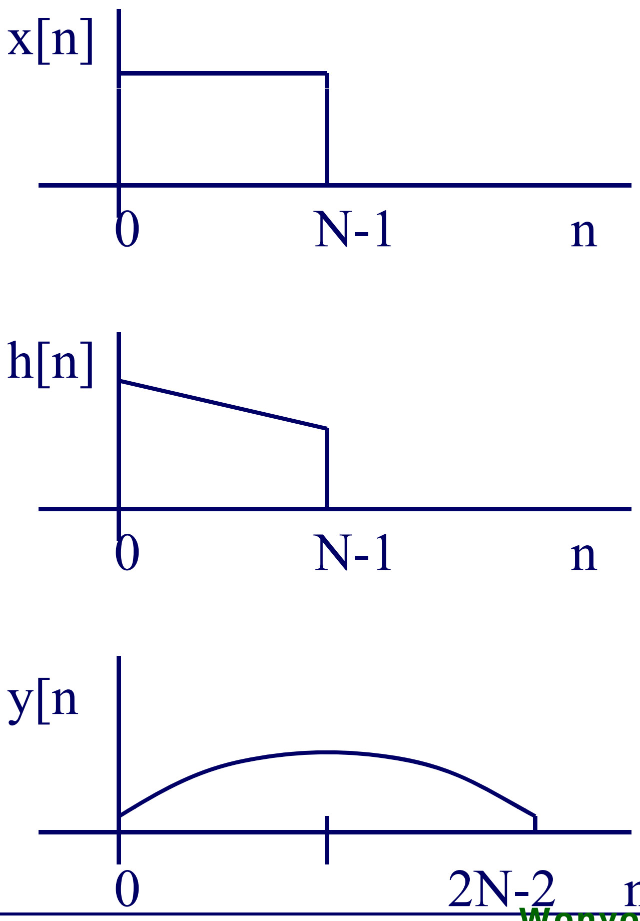
$$= \sum_{k=0}^{N-1} x[k]h[n-k],$$

$$n = 0,1,\ldots,2N-2$$

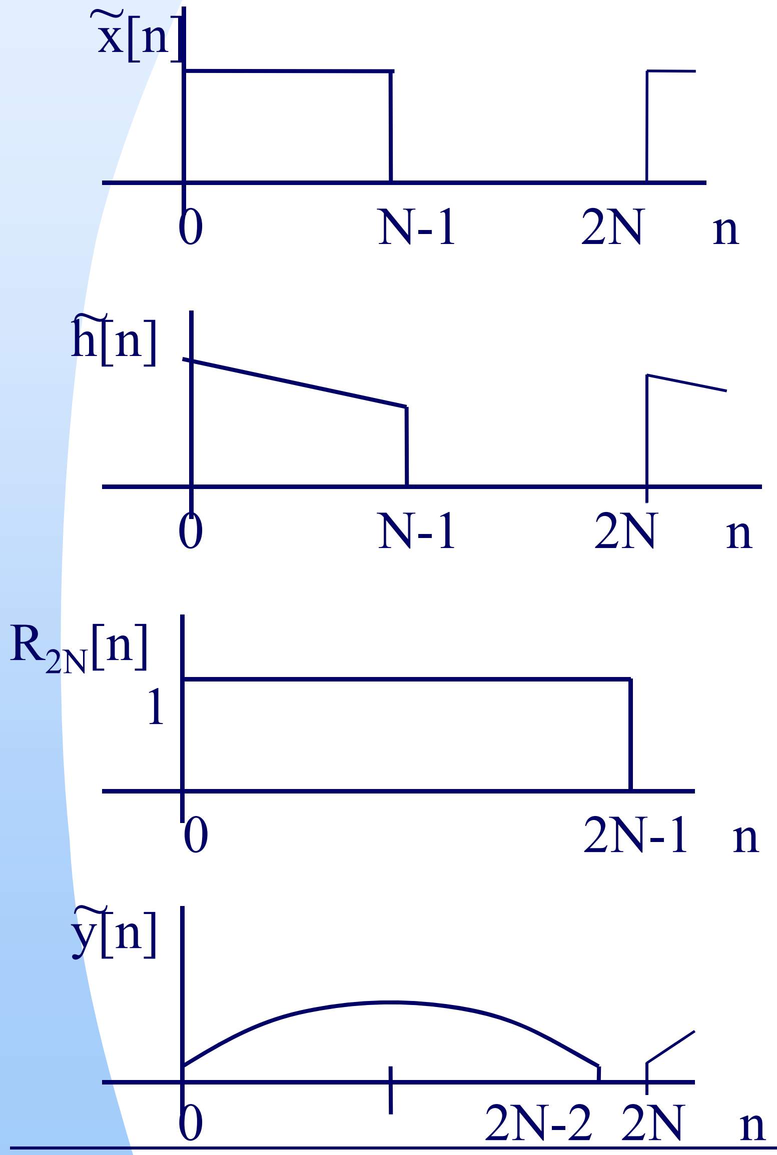
#computation(multi)?

1+2+…+N+N-1+…+1+0

=N^2



-Utilize FFT of 2N-point



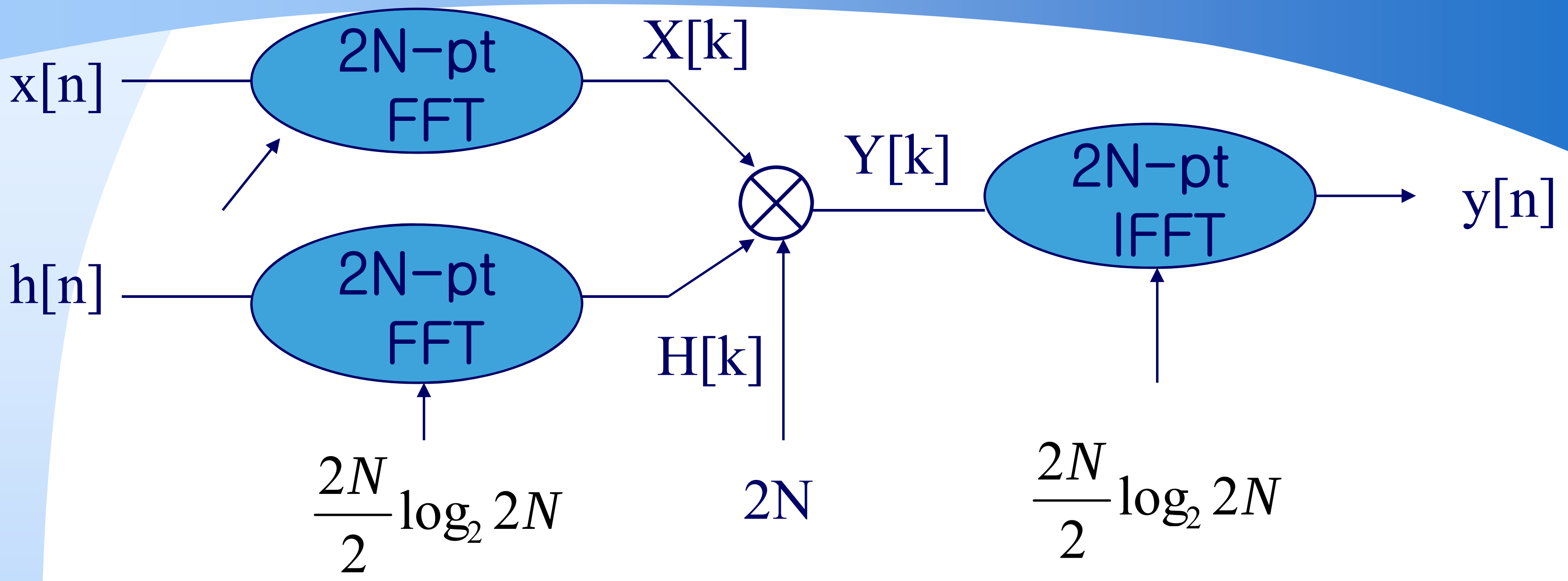
$$y[n] = \tilde{y}[n] * R_{2N}[n]$$

$$= [\sum_{k=0}^{2N-1} \tilde{x}[k]\tilde{h}[n-k]] \cdot R_{2N}[\mathrm{n}],$$

$$n = 0,1,\ldots, 2N-1$$

$$Y[k] = X[k] \cdot H[k],$$

2N-pt DFTs



# operation (multi)

$$3\cdot\frac{2N}{2}\log_2 2N + 2N = 3N\log_2 N + 5N$$

- operation reduction : $\mathrm{N}^2 = 3N\log_2 N + 5N$

$\mathrm{N} = 1024 = 2^{10}$: $1{,}000{,}000 \rightarrow 35{,}000$

(3) Correlation /Power Spectrum

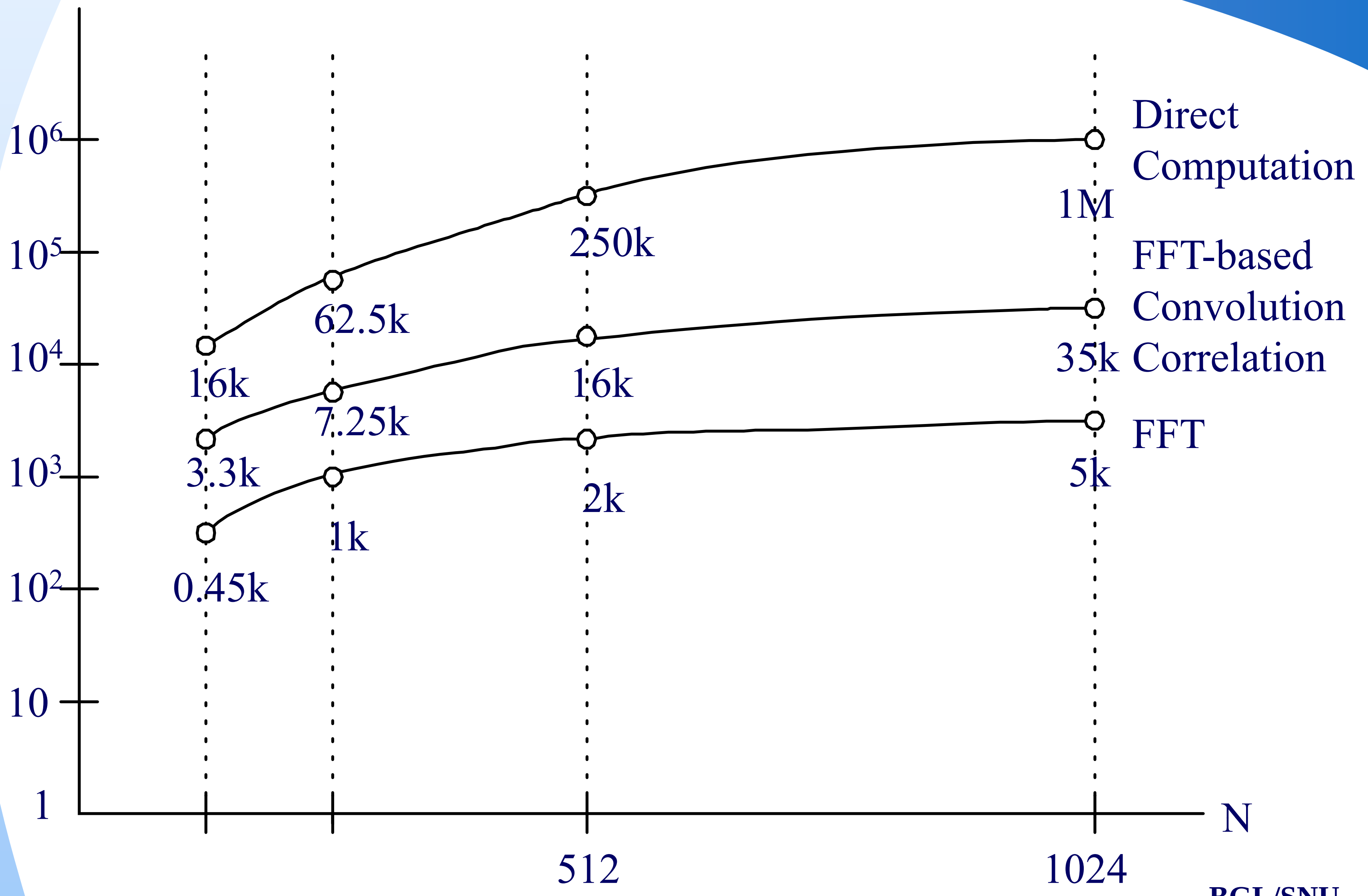
$$z[n] = x[n] * h^*[-n]$$
$$= \sum_{k=0}^{N-1} x[k]h^*[-n+k]$$
$$= [\sum_{k=0}^{N-1} \tilde{x}[k]\tilde{h}^*[-n+k]] \cdot R_{2N}[\mathrm{n}]$$
$$n = 0,1,2,\ldots, 2N-1$$

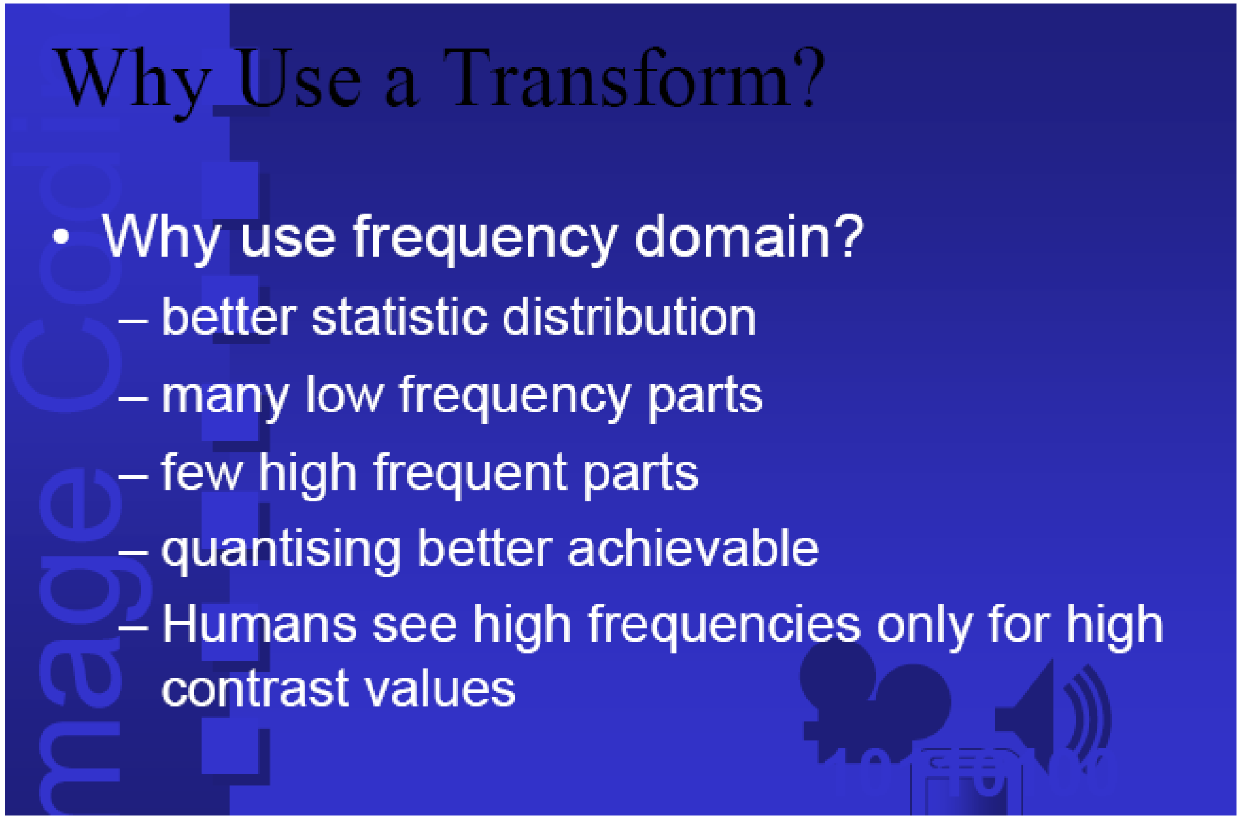
$Z[k] = X[k] \cdot H^*[k],$ 2N-point DFTs

# Operation : $\mathrm{N}^2 \rightarrow 3N \log_2 N + 5N$

- Power spectrum $\mathrm{P[k] = X[k]\, X^*[k]}$

$ Comparison of # computation



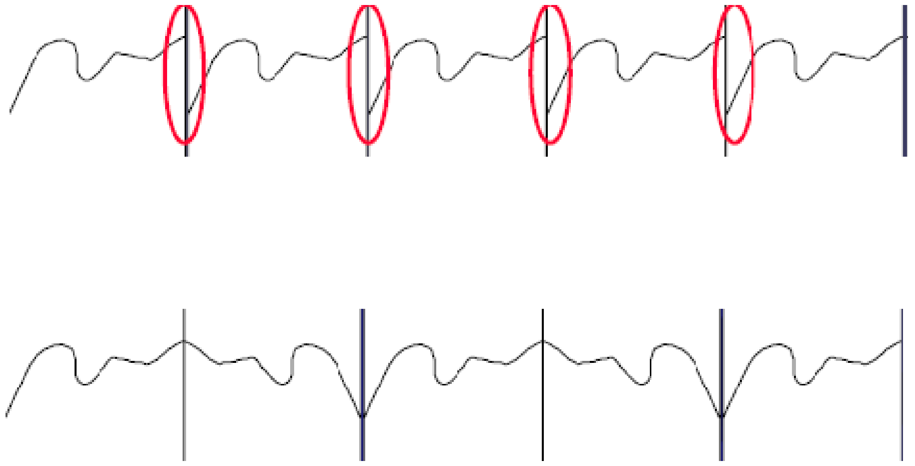


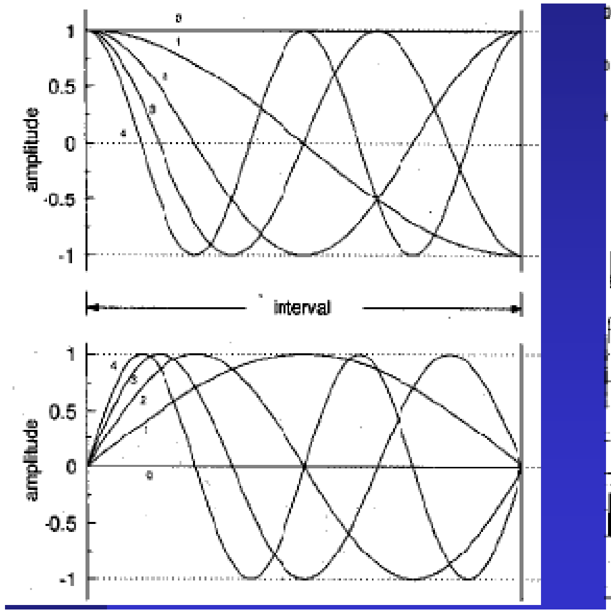
Why Use a Transform?

- Why use frequency domain?
  - better statistic distribution
  - many low frequency parts
  - few high frequent parts
  - quantising better achievable
  - Humans see high frequencies only for high contrast values

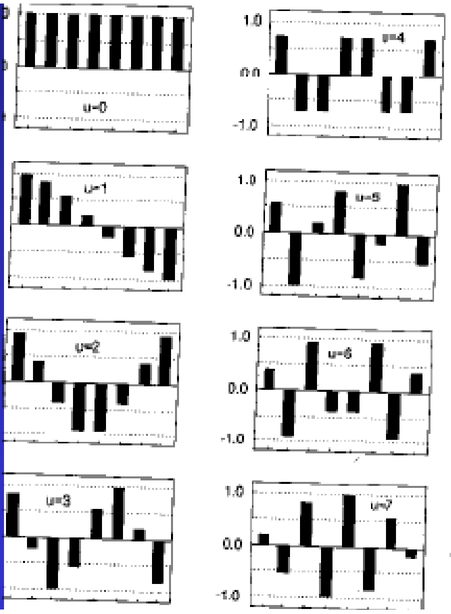
Why not Fourier Transform?

- 8x8 Blocks
- FT: ringing at block edges
- Mirroring produces smooth function
- Sinus coefficients disappear

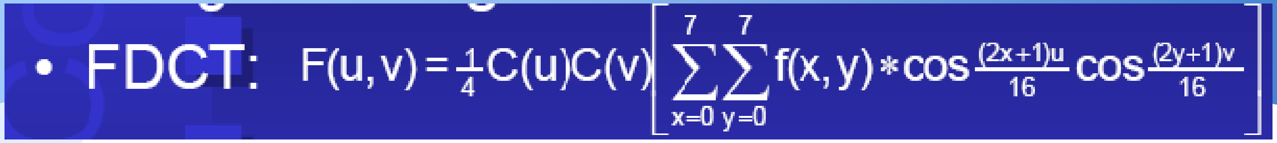




amplitude
1
0.5
0
-0.5
-1
interval



u=0
u=1
u=2
u=3
u=4
u=5
u=6
u=7
1.0
0.0
-1.0



- FDCT: $F(u,v) = \frac{1}{4}C(u)C(v)\left[\sum_{x=0}^{7}\sum_{y=0}^{7} f(x,y) * \cos\frac{(2x+1)u}{16}\cos\frac{(2y+1)v}{16}\right]$

Discrete Cosine/Sine Transform

- **DCT: Fast DCT**

  see note: fastdct.doc

$$F(u) = \alpha(u) \sum_{m=0}^{N-1} x(m) \cos \frac{(2m+1)\pi u}{2N}$$

$$\alpha(u) = \begin{cases} 1 & 1 \le u \le N-1 \\ 1/\sqrt{2} & u = 0 \end{cases};$$

- **DST**

$$G(u) = \sum_{m=0}^{N-1} x(m) \sin \frac{\pi(2m+1)(u+1)}{N+1}$$

- **Both DCT and DST can be expressed as Matrix-vector products.**

- **2D DCT**

$$F(u,v) = \frac{2}{N} \alpha(u)\alpha(v) \sum_{m=0}^{N-1} \sum_{n=0}^{N-1} x(m)x(n) \cos\left\{\frac{(2m+1)\pi u}{2N}\right\} \cos\left\{\frac{(2n+1)\pi n}{2N}\right\}$$

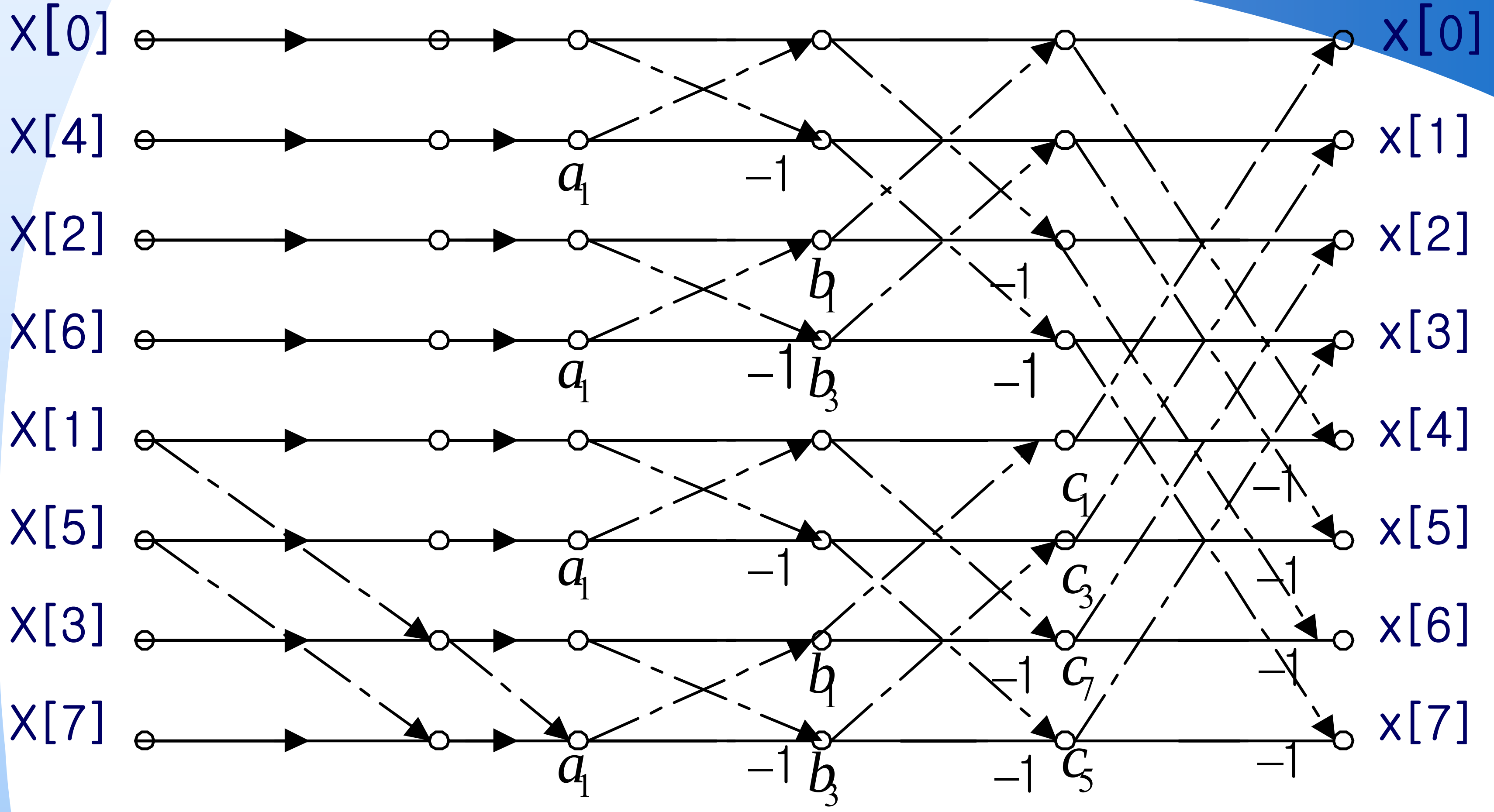
5. Fast Computation of DCT

$$X[\mathrm{k}] = \frac{2}{N} e[k] \sum_{k=0}^{N-1} x[n] \cos \frac{\pi(2n+1)k}{2N} \qquad , k = 0,1,\ldots, N-1$$

$$e[k] = \begin{cases} \dfrac{1}{\sqrt{2}} & , \mathrm{k} = 0 \\ 1 & , \text{otherwise} \end{cases}$$

$$x[\mathrm{n}] = \sum_{k=0}^{N-1} e[k] X[k] \cos \frac{\pi(2n+1)k}{2N} \qquad , n = 0,1,\ldots, N-1$$

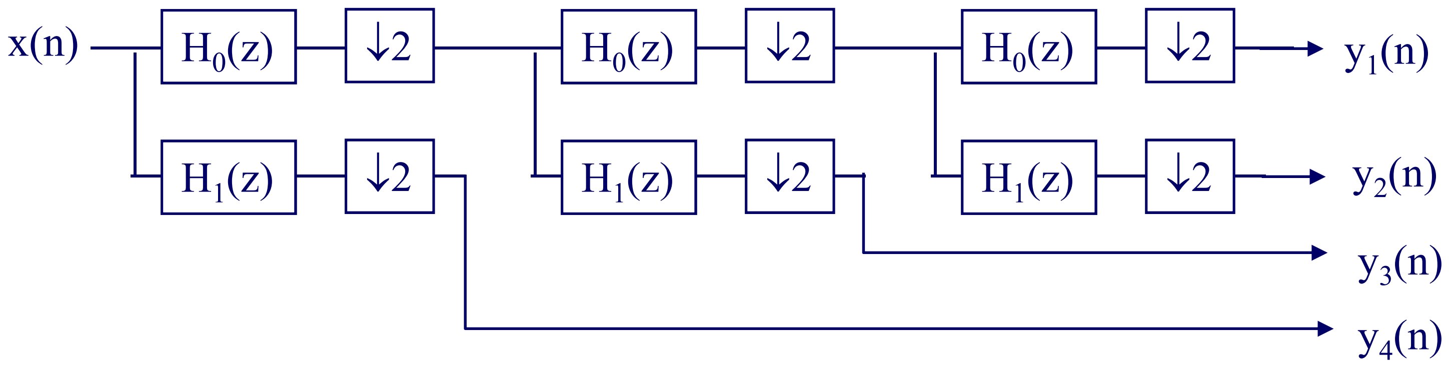
- *Example*: Lee's Algorithm (1984, IEEE Trans , ASSP, Dec)

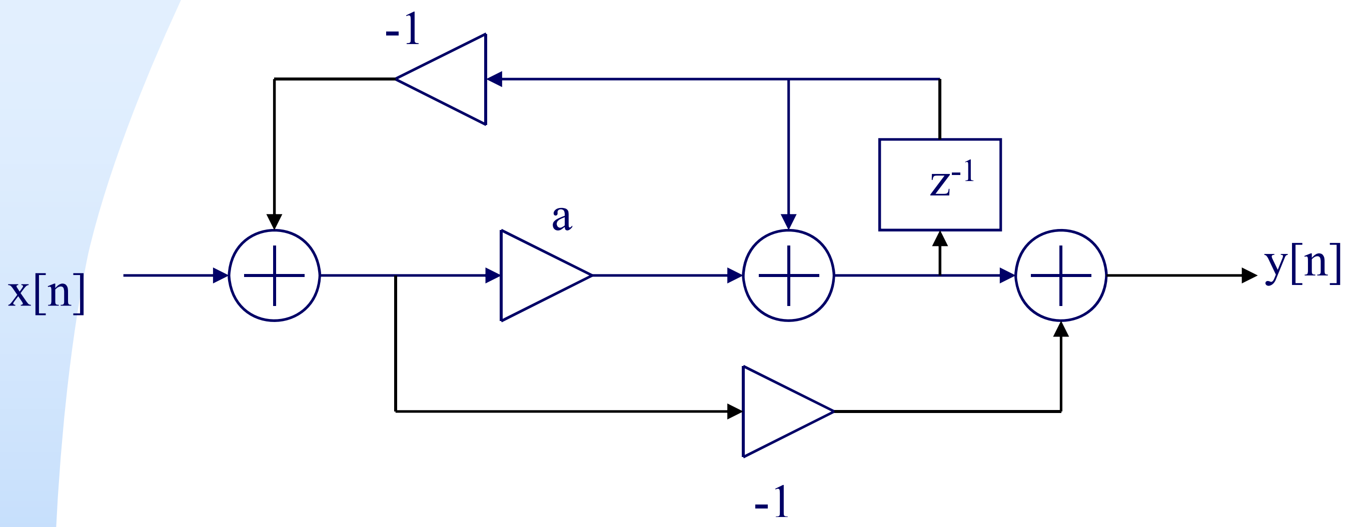


$$a_1 = \frac{1}{2\cos\frac{\pi}{4}} \qquad b_i = \frac{1}{2\cos\frac{i\pi}{8}}, \qquad i = 1,3 \qquad c_i = \frac{1}{2\cos\frac{i\pi}{16}}, i = 1,3,5,7$$

Discrete Wavelet Transform

- $H_0(z)$, $H_1(z)$: low pass and high pass FIR digital filters. Maintain same number of input samples and output samples
- $\downarrow 2$: down-sampling by a factor 2.




-1
a
z^{-1}
x[n]
y[n]
-1