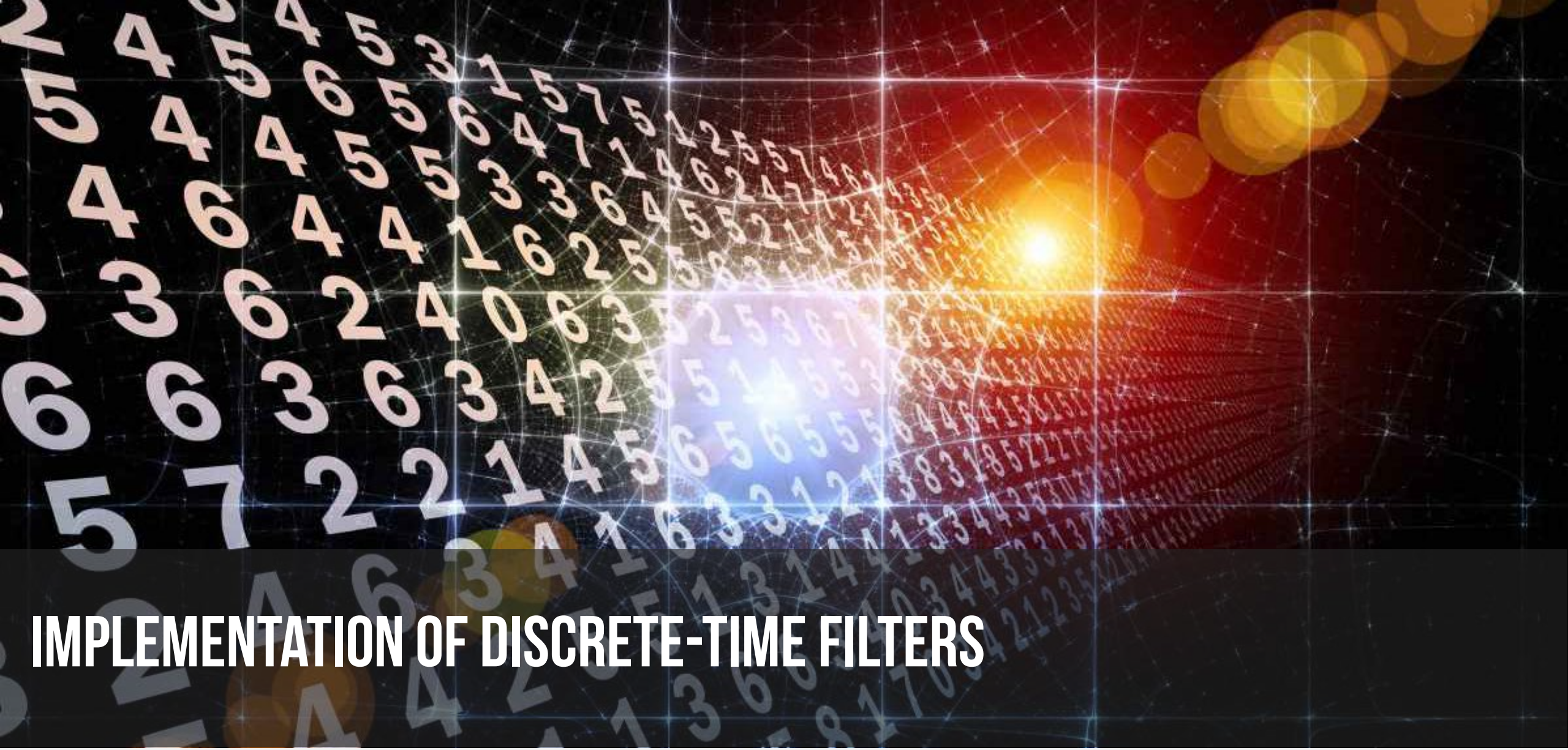
PENGOLAHAN SINYAL DIGITAL

Adhi Harmoko Saputro

UNIVERSITAS INDONESIA
Veritas, Probitas, Iustitia



IMPLEMENTATION OF DISCRETE-TIME FILTERS

Adhi Harmoko Saputro

UNIVERSITAS INDONESIA
Veritas, Probitas, Iustitia

INTRODUCTION

- To process signals, we have to design and implement systems called *filters* (or spectrum analysers in some contexts)

- The filter design issue is influenced by such factors as the type of the filter (i.e., IIR or FIR) or the form of its implementation (structures).

- IIR filters as designed and used in DSP, can be modeled by rational system functions or, equivalently, by difference equations.

INTRODUCTION

- We begin to consider problems associated with quantization effects when finite-precision arithmetic is used in the implementation.
- Digital hardware contains processing elements that use finite-precision arithmetic. When filters are implemented either in hardware or in software, filter coefficients as well as filter operations are subjected to the effects of these finite-precision operations

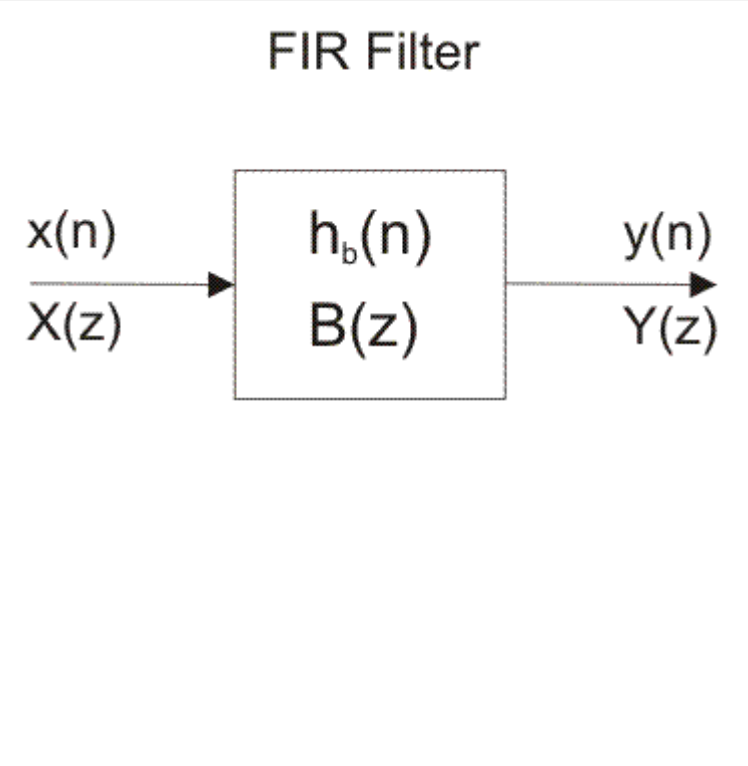
INFINITE IMPULSE RESPONSE (IIR)

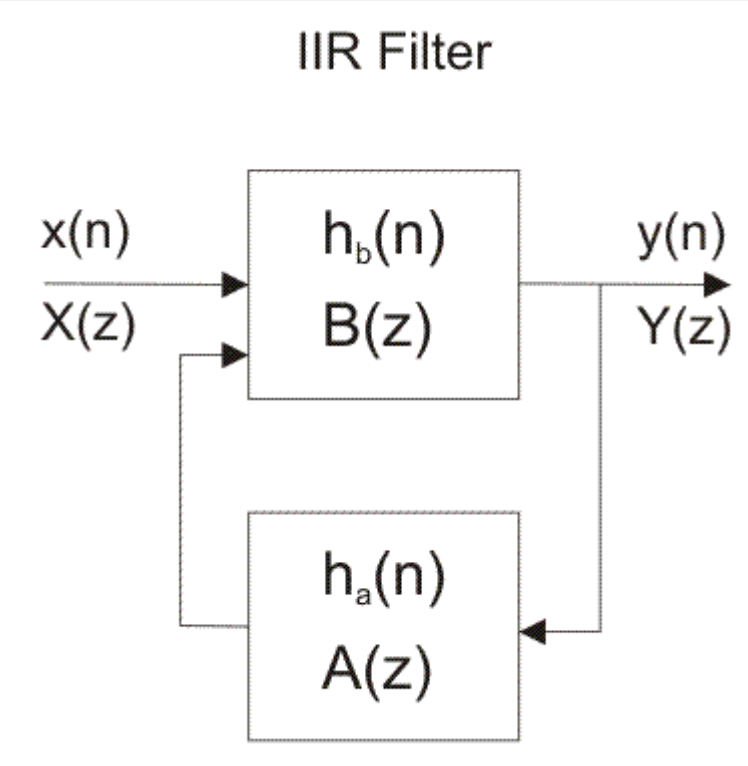
- A property applying to many linear time-invariant systems.
  - Electronic and digital filters

- Having an impulse response which does not become exactly zero past a certain point, but continues indefinitely.

- Contrast to a finite impulse response (FIR)
  - The impulse response h(n) does become exactly zero at times t > T for some finite T, thus being of finite duration.

INFINITE IMPULSE RESPONSE (IIR)





INFINITE IMPULSE RESPONSE (IIR)

- **Filter type:** low pass (LP), band pass (BP) or high pass (HP)
- **Analog prototype:** Butterworth or Chebyshev
- **Filter order:** maximum 16; must be *even* for a BP filter
- **Filter passband:** based on sampling rate of 8000 samples/s (maximum signal frequency 4000 Hz)
- **Passband ripple:** maximum tolerable variation in filter gain within the passband, expressed in decibels (Chebyshev filter only)

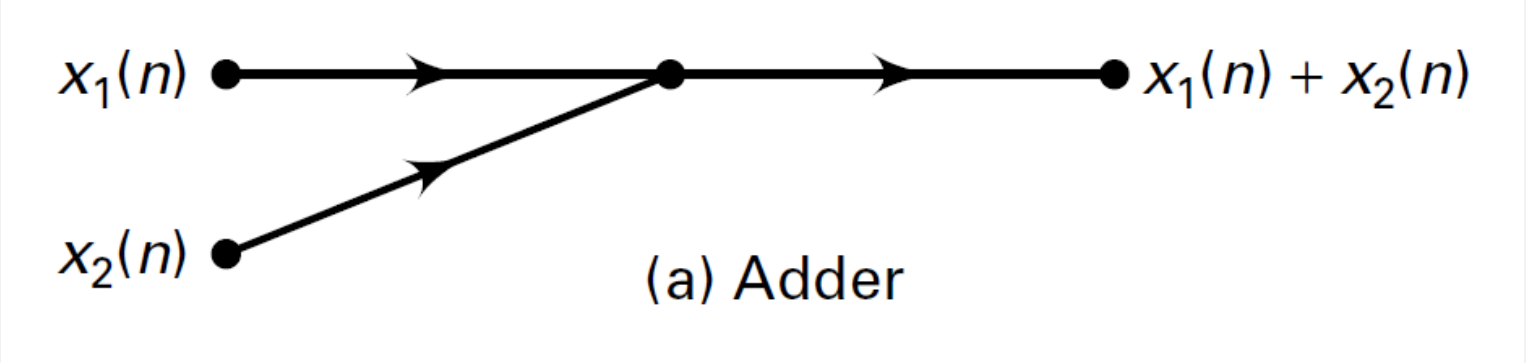
IIR FILTER STRUCTURES

Adhi Harmoko Saputro

UNIVERSITAS INDONESIA
Veritas, Probitas, Iustitia

BASIC ELEMENTS

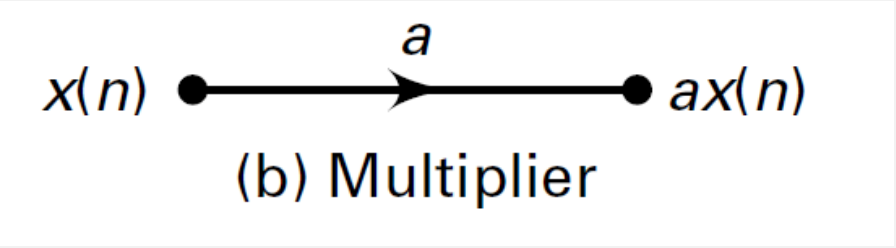
- **Adder:**
  - This element has two inputs and one output
  - Note that the addition of three or more signals is implemented by successive two-input adders.



(a) Adder

BASIC ELEMENTS

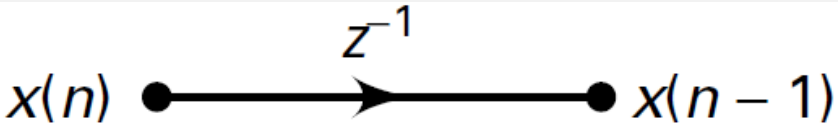
- **Multiplier (gain):**
  - This is a single-input, single-output element
  - Note that the multiplication by 1 is understood and hence not explicitly shown



(b) Multiplier

BASIC ELEMENTS

- **Delay element (shifter or memory):**
  - This element delays the signal passing through it by one sample
  - It is implemented by using a shift register



(c) Delay element

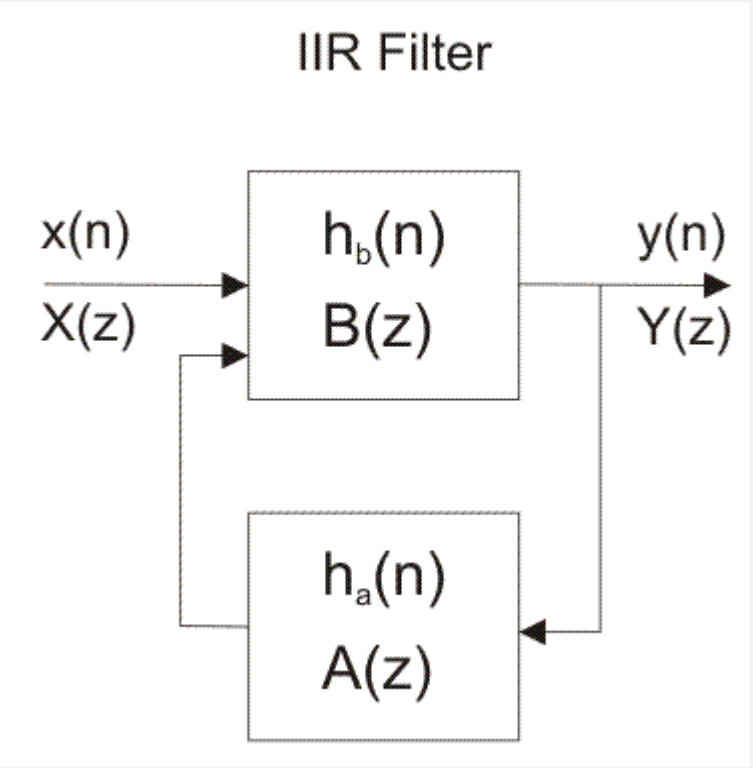
BASIC ELEMENTS

Adhi Harmoko Saputro

IIR FILTER STRUCTURES

- The system function of an IIR filter is given by

$$H(z)=\frac{B(z)}{A(z)}=\frac{\sum_{n=0}^{M} b_n z^{-n}}{\sum_{n=0}^{N} a_n z^{-n}}=\frac{b_o+b_1 z^{-1}+\ldots+b_M z^{-M}}{1+a_1 z^{-1}+\ldots+a_N z^{-N}};\quad a_o=1$$



  - b_n and a_n are the coefficients of the filter
  - The order of such an IIR filter is called N if $a_N \neq 0$.

IIR FILTER STRUCTURES

- The difference equation representation of an IIR filter is expressed as

$$y(n) = \sum_{m=0}^{M} b_m x(n-m) - \sum_{m=1}^{N} a_m y(n-m)$$

IIR FILTER STRUCTURES

- Three different structures can be used to implement an IIR filter

1. **Direct form:** In this form the difference equation is implemented directly as given. There are two parts to this filter, namely the moving average part and the recursive part (or equivalently, the numerator and denominator parts). Therefore this implementation leads to two versions: direct form I and direct form II structures.

2. **Cascade form:** In this form the system function $H(z)$ is factored into smaller 2nd-order sections, called *biquads*. The system function is then represented as a *product* of these biquads. Each biquad is implemented in a direct form, and the entire system function is implemented as a *cascade* of biquad sections.

IIR FILTER STRUCTURES

- Three different structures can be used to implement an IIR filter

3. **Parallel form:** This is similar to the cascade form, but after factorization, a partial fraction expansion is used to represent $H(z)$ as a *sum* of smaller 2nd-order sections. Each section is again implemented in a direct form, and the entire system function is implemented as a *parallel* network of sections..

DIRECT FORM

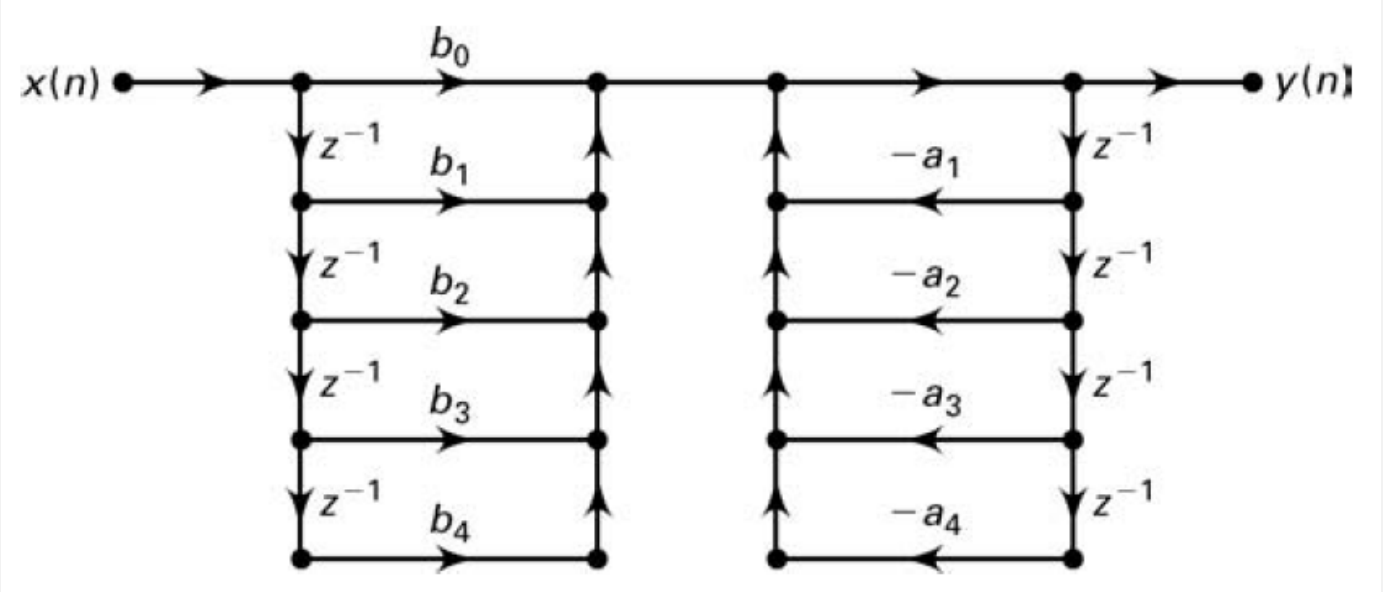
- The difference equation is implemented as given using delays, multipliers, and adders.
- For the purpose of illustration, let $M = N = 4$.
  - Then the difference equation is

$$y(n) = b_o x(n) + b_1 x(n-1) + b_2 x(n-2) + b_3 x(n-3) + b_4 x(n-4)$$
$$- a_1 y(n-1) - a_2 y(n-2) - a_3 y(n-3) - a_4 y(n-4)$$

DIRECT FORM

- The difference equation can be implemented as

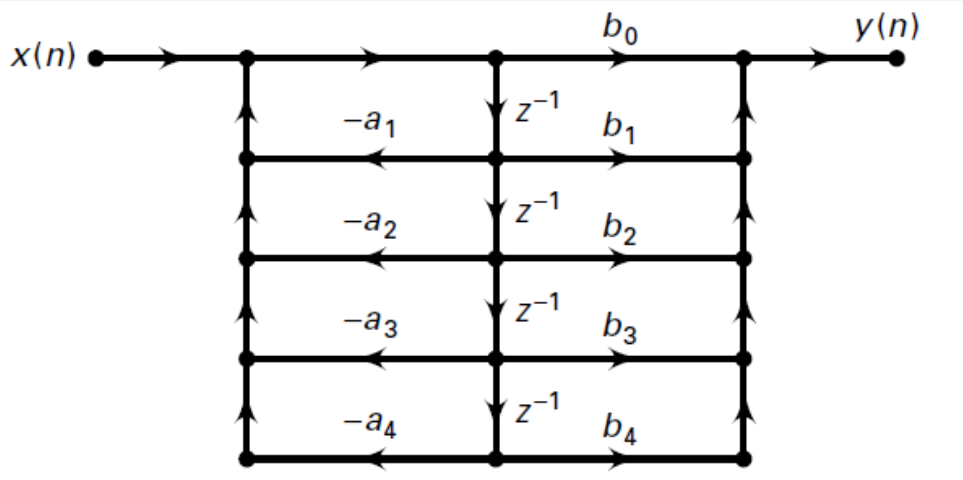
$$y(n) = b_o x(n) + b_1 x(n-1) + b_2 x(n-2) + b_3 x(n-3) + b_4 x(n-4)$$
$$- a_1 y(n-1) - a_2 y(n-2) - a_3 y(n-3) - a_4 y(n-4)$$



- This block diagram is called *Direct Form I* structure

DIRECT FORM II STRUCTURE

- The two delay lines are close to each other, connected by a unity gain branch



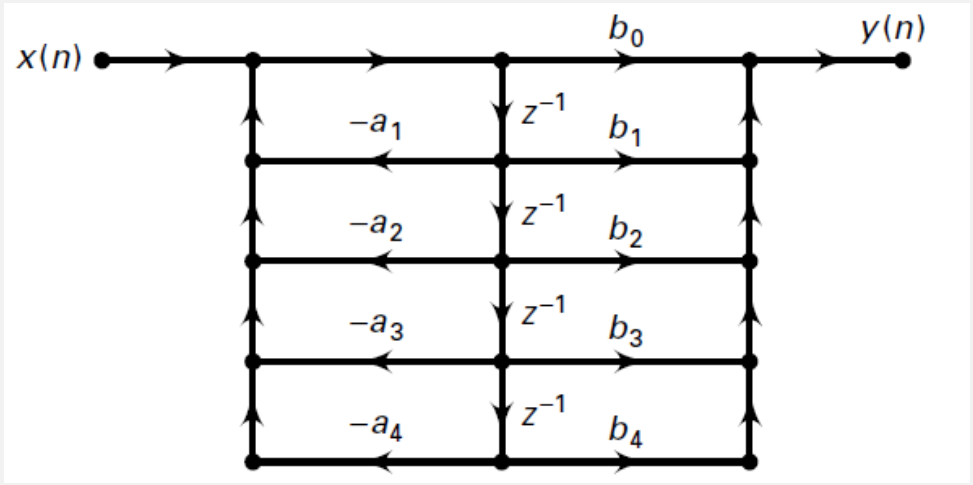
TRANSPOSED STRUCTURE

- An equivalent structure to the direct form can be obtained using a procedure called *transposition*.

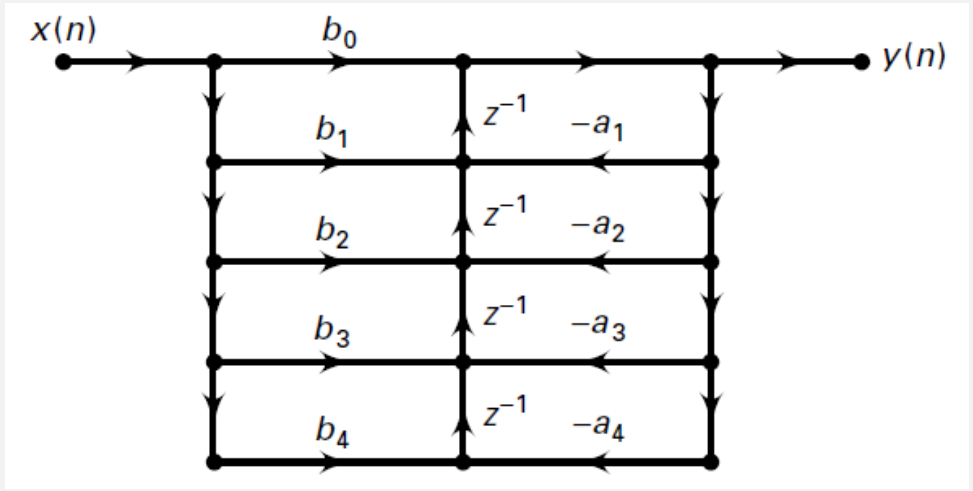
- Three steps are performed:
  1. All path arrow directions are reversed.
  2. All branch nodes are replaced by adder nodes, and all adder nodes are replaced by branch nodes.
  3. The input and output nodes are interchanged.

TRANSPOSED STRUCTURE

- The transposed direct form II structure



direct form II



transposed

MATLAB IMPLEMENTATION

- The direct form structure is described by two row vectors;
  - b containing the $\{b_n\}$ coefficients
  - a containing the $\{a_n\}$ coefficients

- The filter function implements the transposed direct form II structure.

CASCADE FORM

- In this form the system function $H(z)$ is written as a product of 2nd-order sections with real coefficients.
- This is done by factoring the numerator and denominator polynomials into their respective roots and then combining either a complex conjugate root pair or any two real roots into 2nd-order polynomials.

$$H(z) = \frac{b_o + b_1 z^{-1} + \ldots + b_N z^{-N}}{1 + a_1 z^{-1} + \ldots + a_N z^{-N}} = b_o \frac{1 + \frac{b_1}{b_o} z^{-1} + \ldots + \frac{b_N}{b_o} z^{-N}}{1 + a_1 z^{-1} + \ldots + a_N z^{-N}}$$

$$= b_o \prod_{k=1}^{K} \frac{1 + B_{k,1} z^{-1} + B_{k,2} z^{-2}}{1 + A_{k,1} z^{-1} + A_{k,2} z^{-2}}; \quad k = 1, \ldots, K$$

- K is equal to $N/2$, and $B_{k,1}$, $B_{k,2}$, $A_{k,1}$, and $A_{k,2}$ are real numbers representing the coefficients of 2nd-order sections

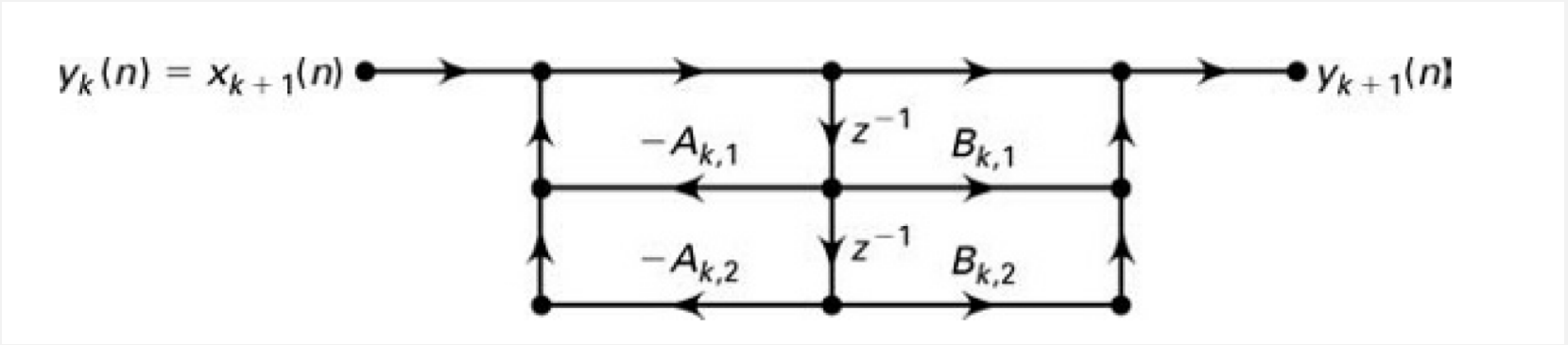
BIQUAD SECTION

- The 2nd-order section

$$H(z)=\frac{Y_{k+1}(z)}{Y_k(z)}=\frac{1+B_{k,1}z^{-1}+B_{k,2}z^{-2}}{1+A_{k,1}z^{-1}+A_{k,2}z^{-2}};\quad k=1,\ldots,K$$

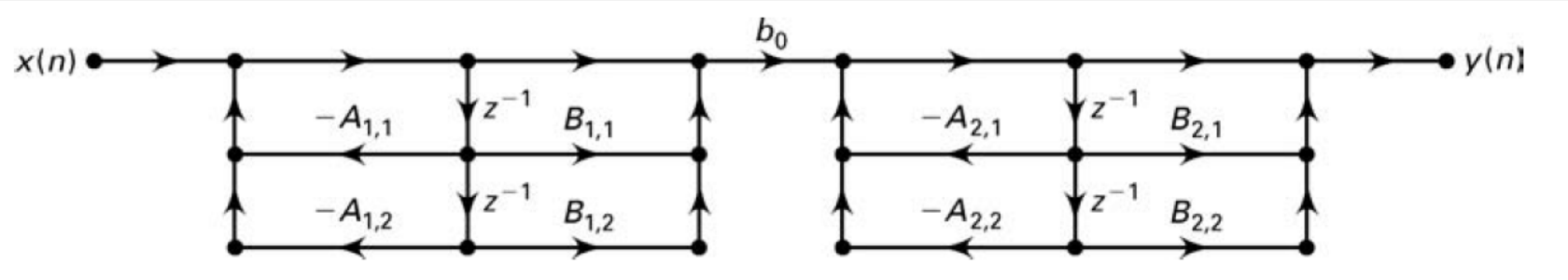
- with

$$Y_1(z)=b_oX(z);\quad Y_{K+1}(z)=Y(z)$$



BIQUAD SECTION STRUCTURE

- a cascade form structure for this 4th-order IIR filter (N = 4)



MATLAB IMPLEMENTATION

```
function [b0,B,A] = dir2cas(b,a);
% DIRECT-form to CASCADE-form conversion (cplxpair version)
% ---------------------------------------------------------
% [b0,B,A] = dir2cas(b,a)
% b0 = gain coefficient
% B = K by 3 matrix of real coefficients containing bk's
% A = K by 3 matrix of real coefficients containing ak's
% b = numerator polynomial coefficients of DIRECT form
% a = denominator polynomial coefficients of DIRECT form
% compute gain coefficient b0
b0 = b(1); b = b/b0; a0 = a(1); a = a/a0; b0 = b0/a0;
%
M = length(b); N = length(a);
if N > M
       b = [b zeros(1,N-M)];
```

MATLAB IMPLEMENTATION

```
elseif M > N
       a = [a zeros(1,M-N)]; N = M;
else
       NM = 0;
end
%
K = floor(N/2); B = zeros(K,3); A = zeros(K,3);
if K*2 == N;
       b = [b 0]; a = [a 0];
end
%
broots = cplxpair(roots(b)); aroots = cplxpair(roots(a));
for i=1:2:2*K
       Brow = broots(i:1:i+1,:); Brow = real(poly(Brow));
       B(fix((i+1)/2),:) = Brow;
       Arow = aroots(i:1:i+1,:); Arow = real(poly(Arow));
       A(fix((i+1)/2),:) = Arow;
end
```

MATLAB IMPLEMENTATION

```
function y = casfiltr(b0,B,A,x);
% CASCADE form realization of IIR and FIR filters
% ----------------------------------------------
% y = casfiltr(b0,B,A,x);
% y = output sequence
% b0 = gain coefficient of CASCADE form
% B = K by 3 matrix of real coefficients containing bk's
% A = K by 3 matrix of real coefficients containing ak's
% x = input sequence
%
[K,L] = size(B);
N = length(x); w = zeros(K+1,N); w(1,:) = x;
for i = 1:1:K
       w(i+1,:) = filter(B(i,:),A(i,:),w(i,:));
end
y = b0*w(K+1,:);
```

MATLAB IMPLEMENTATION

```
function [b,a] = cas2dir(b0,B,A);
% CASCADE-to-DIRECT form conversion
% ---------------------------------
% [b,a] = cas2dir(b0,B,A)
% b = numerator polynomial coefficients of DIRECT form
% a = denominator polynomial coefficients of DIRECT form
% b0 = gain coefficient
% B = K by 3 matrix of real coefficients containing bk's
% A = K by 3 matrix of real coefficients containing ak's
%
[K,L] = size(B);
b = [1]; a = [1];
for i=1:1:K
       b=conv(b,B(i,:)); a=conv(a,A(i,:));
end
b = b*b0;
```

EXAMPLE

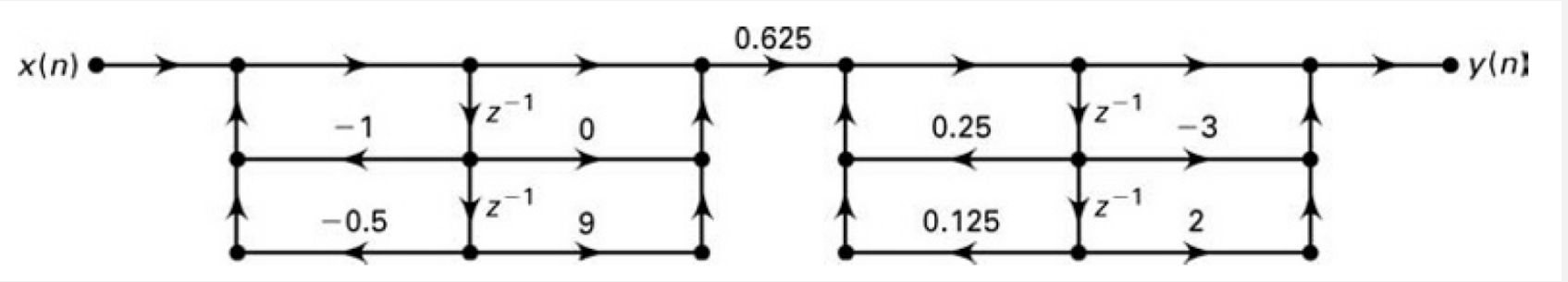
- A filter is described by the following difference equation

$$16y(n) + 12y(n-1) + 2y(n-2) - 4y(n-3) - y(n-4) =$$
$$x(n) - 3x(n-1) + 11x(n-2) - 27x(n-3) + 18x(n-4)$$

- Determine its cascaded form structure.

EXAMPLE

```
>> b=[1 -3 11 -27 18]; a=[16 12 2 -4 -1];
>> [b0,B,A]=dir2cas(b,a)
b0 = 0.0625
B =    1.0000 -0.0000 9.0000
       1.0000 -3.0000 2.0000
A =    1.0000 1.0000 0.5000
       1.0000 -0.2500 -0.1250
```



EXAMPLE

```
>> delta = impseq(0,0,7)
delta = 1 0 0 0 0 0 0 0
>> format long
>> hcas=casfiltr(b0,B,A,delta)
>> hdir=filter(b,a,delta)
```

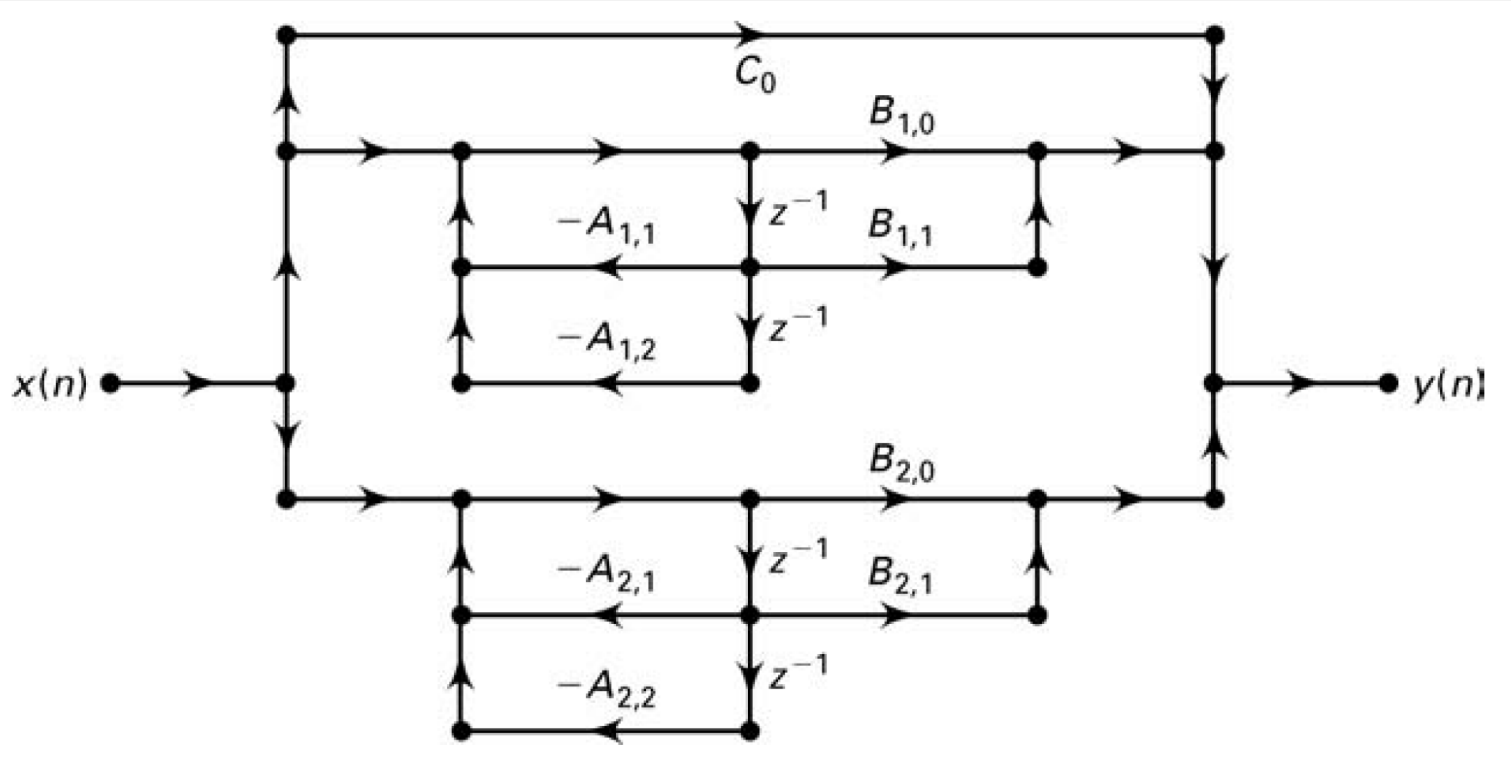
PARALLEL FORM

- The system function $H(z)$ is written as a sum of 2nd-order sections using partial fraction expansion

$$H(z) = \frac{B(z)}{A(z)} = \frac{b_o + b_1 z^{-1} + \ldots + b_M z^{-M}}{1 + a_1 z^{-1} + \ldots + a_N z^{-N}}$$

PARALLEL FORM

- A parallel-form structure for this 4th-order IIR filter.



MATLAB IMPLEMENTATION

- A function dir2par converts the direct-form coefficients $\{b_n\}$ and $\{a_n\}$ into parallel form coefficients $\{B_k,i\}$ and $\{A_k,i\}$.

- The dir2cas function first computes the *z*-domain partial fraction expansion using the residuez function.

- The cplxpair function from MATLAB can be used to arrange pole-and-residue pairs into complex conjugate pole-and-residue pairs followed by real poleand- residue pairs
  - this sorts a complex array into complex conjugate pairs

EXAMPLE

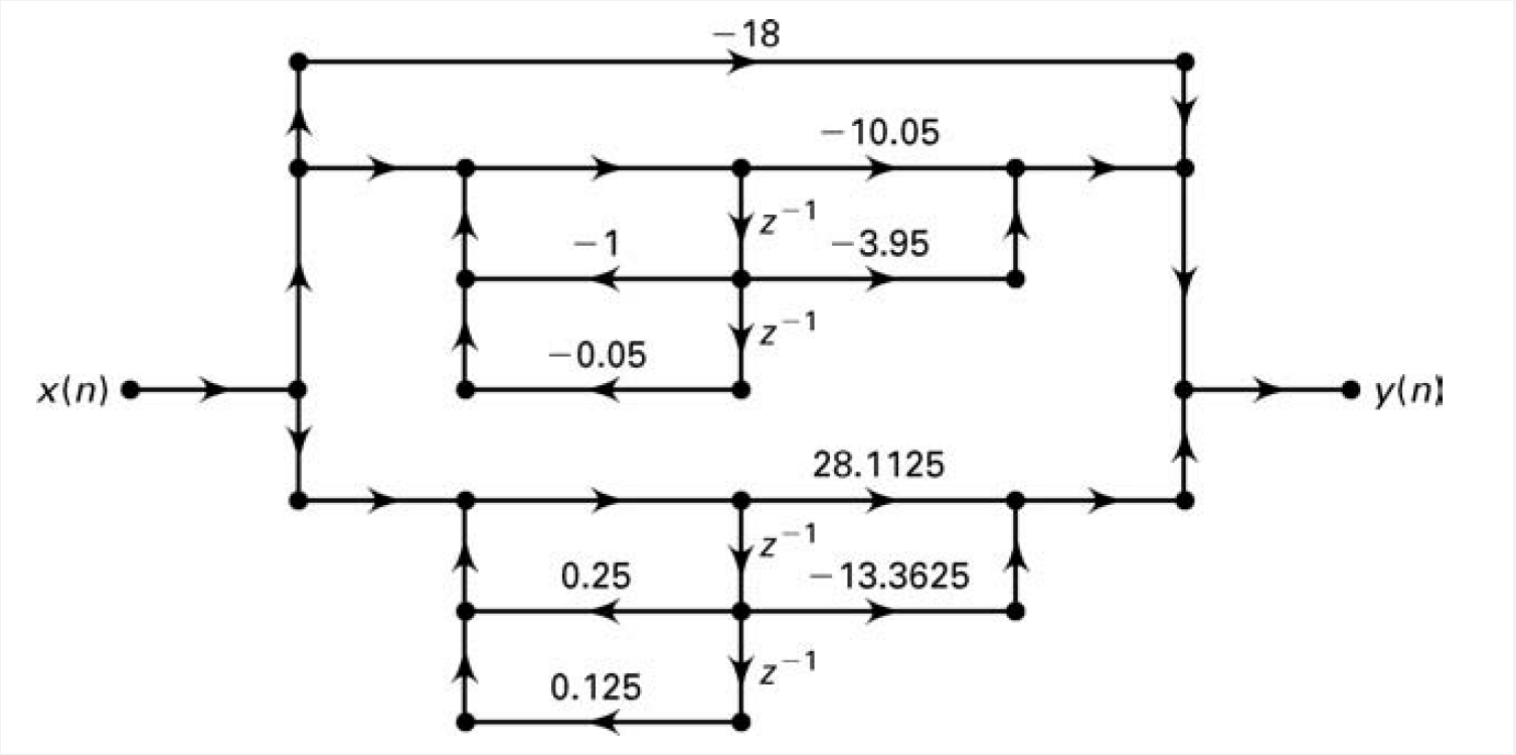
- A filter is described by the following difference equation

$$16y(n) + 12y(n-1) + 2y(n-2) - 4y(n-3) - y(n-4)$$
$$= x(n) - 3x(n-1) + 11x(n-2) - 27x(n-3) + 18x(n-4)$$

- Now determine its parallel form.

EXAMPLE

```
>> b=[1 -3 11 -27 18]; a=[16 12 2 -4 -1];
>> [C,B,A]=dir2par(b,a)
```



EXAMPLE

- To check our parallel structure, let us compute the first 8 samples of the impulse response using both forms

```
>> format long; delta = impseq(0,0,7);
>> hpar=parfiltr(C,B,A,delta)
>> hdir = filter(b,a,delta)
```

FIR FILTER STRUCTURES

Adhi Harmoko Saputro

UNIVERSITAS INDONESIA
Veritas, Probitas, Iustitia

FIR FILTER STRUCTURES

- A finite-duration impulse response filter has a system function of the form

$$H(z) = b_o + b_1 z^{-1} + \ldots + b_{M-1} z^{1-M} = \sum_{n=0}^{M-1} b_n z^{-n}$$

- The impulse response $h(n)$ is

$$h(n) = \begin{cases} b_n, & 0 \le n \le M-1 \\ 0, & \text{else} \end{cases}$$

- The difference equation representation is

$$y(n) = b_o x(n) + b_1 x(n-1) + \ldots + b_{M-1} x(n-M+1)$$

- The order of the filter is $M - 1$, and the *length* of the filter (which is equal to the number of coefficients) is M

FIR FILTER STRUCTURES

- The FIR filter structures are
  - a linear convolution of finite support
  - always stable
  - relatively simple compared to IIR structures
- FIR filters can be designed to have a linear-phase response, which is desirable in some applications.

FIR FILTER STRUCTURES

1. **Direct form:** $y(n) = b_o x(n) + b_1 x(n-1) + \ldots + b_{M-1} x(n-M+1)$

2. **Cascade form:** $H(z) = b_o + b_1 z^{-1} + \ldots + b_{M-1} z^{1-M} = \sum_{n=0}^{M-1} b_n z^{-n}$

this form the system function $H(z)$ is factored into 2nd-order factors, which are then implemented in a cascade connection.

FIR FILTER STRUCTURES

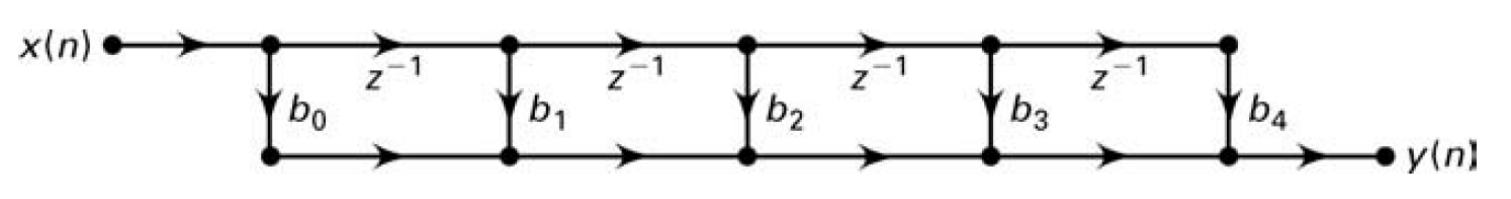
3. **Linear-phase form:**
   its impulse response exhibits certain symmetry conditions
   to reduce multiplications by about half.

4. **Frequency-sampling form:**
   This structure is based on the DFT of the impulse response $h(n)$ and leads to a parallel structure
   suitable for a design technique based on the sampling of frequency response $H(e^{j\omega})$.

DIRECT FORM

- The difference equation is implemented as a tapped delay line since there are no feedback paths.
- Let *M* = 5 (i.e., a 4th-order FIR filter)

$$y(n) = b_o x(n) + b_1 x(n-1) + b_2 x(n-2) + b_3 x(n-3) + b_4 x(n-4)$$



DIRECT FORM

MATLAB IMPLEMENTATION

- The direct form FIR structure is described by the row vector b containing the *{bn}* coefficients.
- The structure is implemented by the filter function, in which the vector a is set to the scalar value 1

CASCADE FORM

- The system function $H(z)$ is converted into products of 2nd-order sections with real coefficients

$$H(z) = b_o + b_1 z^{-1} + \ldots + b_{M-1} z^{-M+1}$$

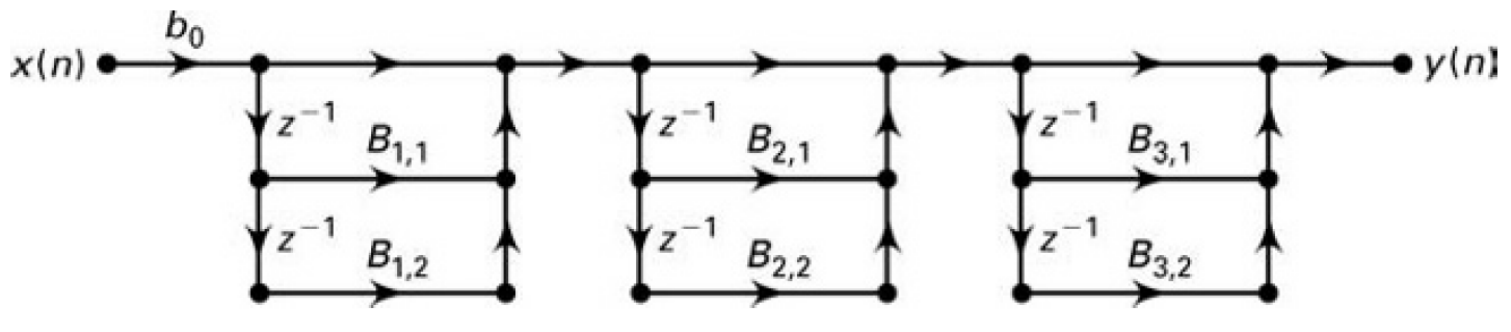
$$= b_o \left(1 + \frac{b_1}{b_o} z^{-1} + \ldots + \frac{b_{M-1}}{b_o} z^{-M+1} \right)$$

$$= b_o \prod_{k=1}^{K} \left(1 + B_{k,1} z^{-1} + B_{k,2} z^{-2} \right)$$

- K is equal to $M/2$, and $B_{k,1}$ and $B_{k,2}$ are real numbers representing the coefficients of 2nd-order sections.

CASCADE FORM

- For $M = 7$ the cascade form is shown as



CASCADE FORM

MATLAB IMPLEMENTATION

- Use our dir2cas function by setting the denominator vector a equal to 1
- Use cas2dir to obtain the direct form from the cascade form.

LINEAR-PHASE FORM

- For frequency-selective filters (e.g., lowpass filters) it is generally desirable to have a phase response that is a linear function of frequency

- where $\beta = 0$ or $\pm\pi/2$ and α is a constant.

$$\angle H\left(e^{j\omega}\right) = \beta - \alpha\omega, \quad -\pi < \omega \leq \pi$$

LINEAR-PHASE FORM

- The linear-phase condition imposes the following symmetry conditions on the impulse response *h*(*n*)
  - a symmetric impulse response

$$h(n) = h(M-1-n); \quad \beta = 0, \alpha = \frac{M-1}{2}, \quad 0 \le n \le M-1$$

  - an antisymmetric impulse response

$$h(n) = -h(M-1-n); \quad \beta = \pm\pi/2, \alpha = \frac{M-1}{2}, \quad 0 \le n \le M-1$$

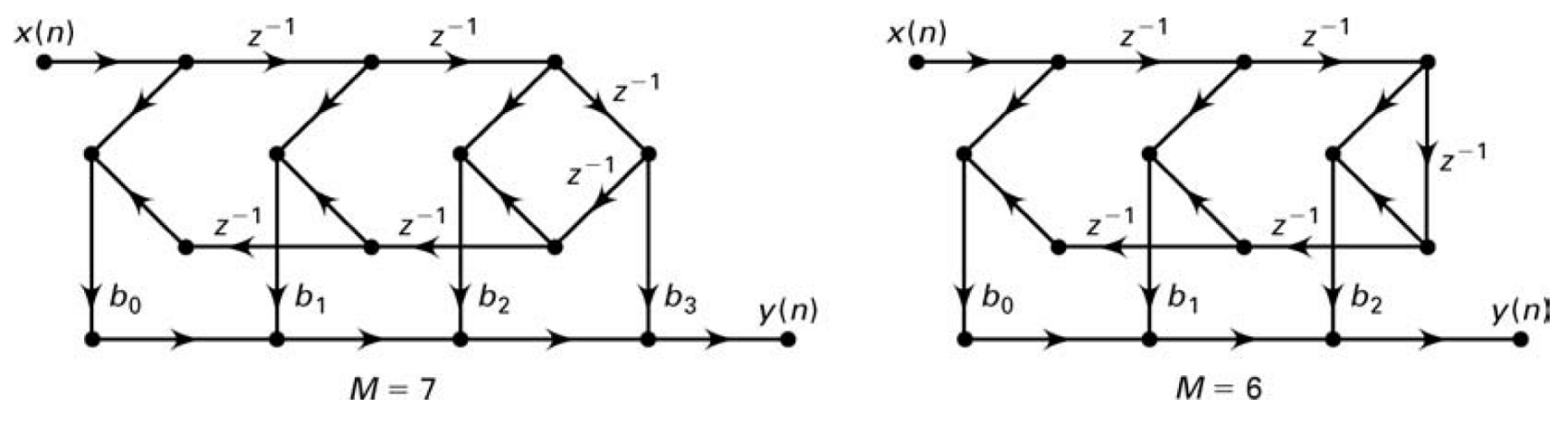
LINEAR-PHASE FORM

- Consider the difference equation with a symmetric impulse response in a symmetric impulse response

$$y(n) = b_o x(n) + b_1 x(n-1) + \ldots + b_1 x(n-M+2) + b_o x(n-M+1)$$
$$= b_o \left[x(n) + x(n-M+1) \right] + b_1 \left[x(n-1) + x(n-M+2) \right] + \ldots$$

LINEAR-PHASE FORM

- The block diagram implementation of the previous difference equation for both odd and even *M*.



LINEAR-PHASE FORM

MATLAB IMPLEMENTATION

- The linear-phase structure is essentially a direct form drawn differently to save on multiplications.
- Hence in a MATLAB representation of the linear-phase structure is equivalent to the direct form.

EXAMPLE

- An FIR filter is given by the system function

$$H(z) = 1 + 16\frac{1}{16}z^{-4} + z^{-8}$$

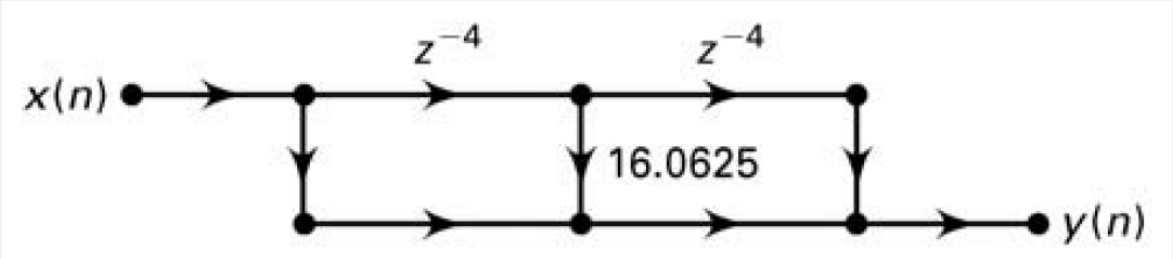
- Determine and draw the direct, linear-phase, and cascade form structures

EXAMPLE

$$H(z) = 1 + 16\frac{1}{16}z^{-4} + z^{-8}$$

- **Direct form:** The difference equation is given by

$$y(n) = x(n) + 16.0625x(n - 4) + x(n - 8)$$

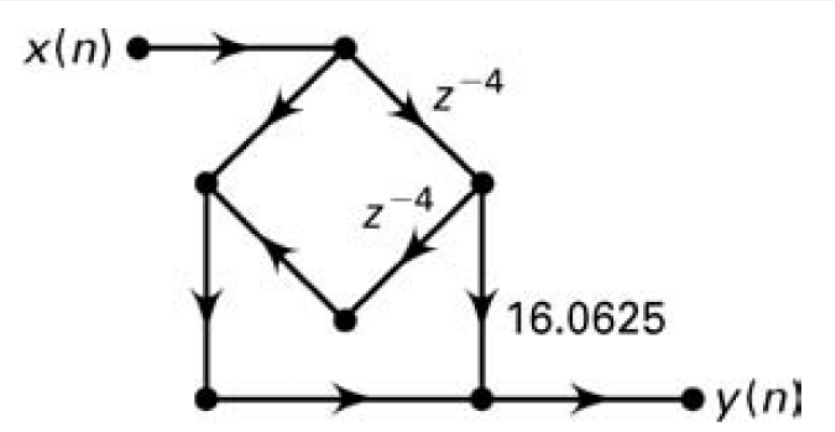


EXAMPLE

$$H(z) = 1 + 16\frac{1}{16}z^{-4} + z^{-8}$$

- **Linear-phase form:** The difference equation can be written in the form

$$y(n) = [x(n) + x(n - 8)] + 16.0625x(n - 4)$$

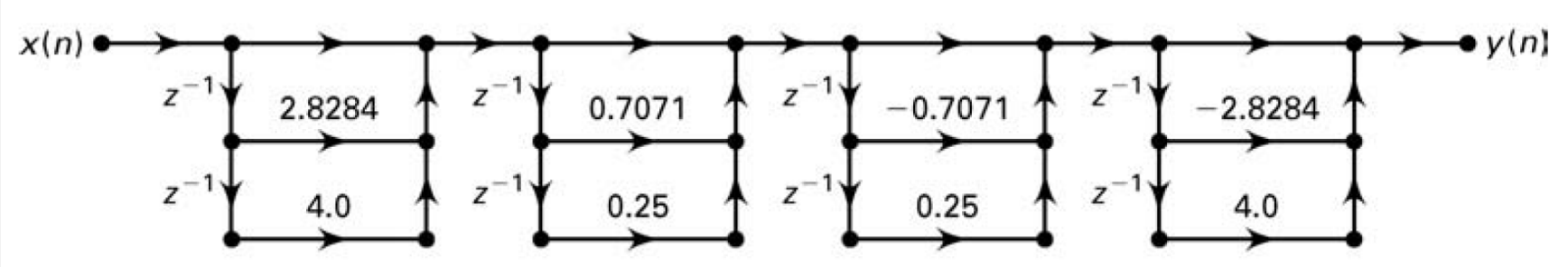


EXAMPLE

$$H(z) = 1 + 16\frac{1}{16}z^{-4} + z^{-8}$$

- **Cascade form:**

>> b=[1,0,0,0,16+1/16,0,0,0,1]; [b0,B,A] = dir2cas(b,1)



TERIMA KASIH

Adhi Harmoko Saputro

UNIVERSITAS INDONESIA
Veritas, Probitas, Iustitia