Lecture 18

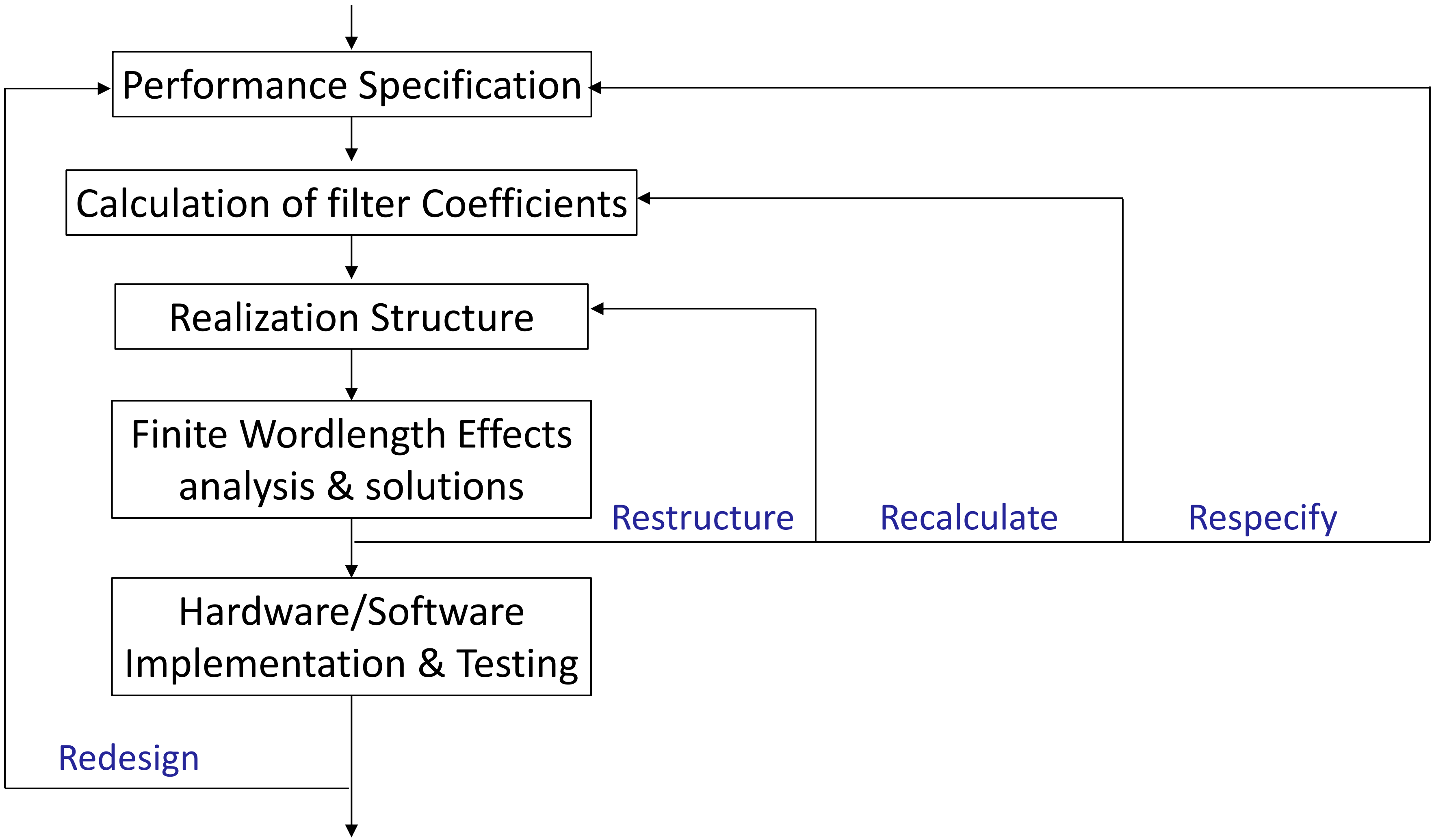
Filter Realisation Structures

Implementing digital filters in software / hardware

Dr. Dimitris Agrafiotis, Dept. Of Electrical & Electronic Engineering

Digital Filter Design Procedure

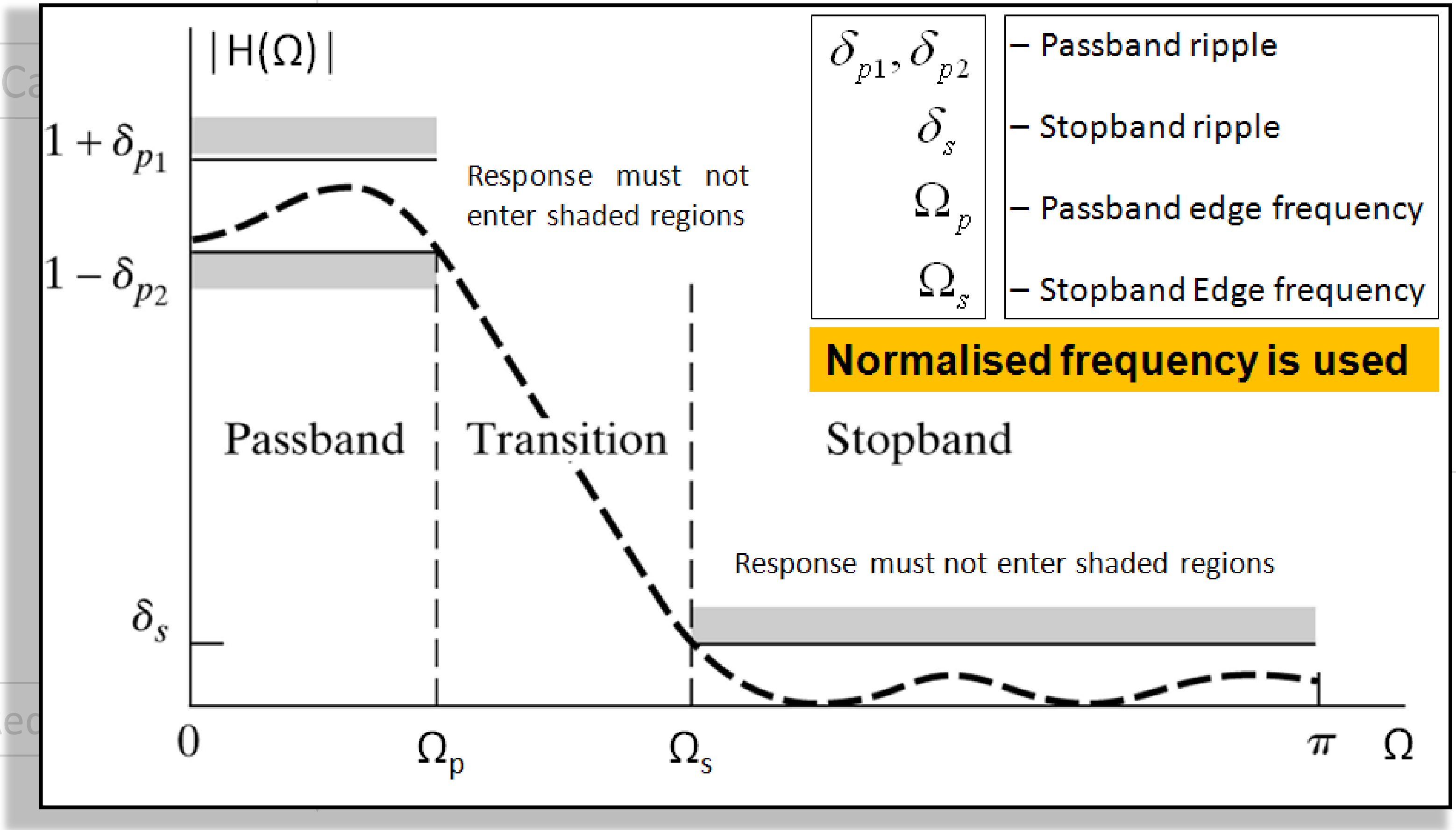
Design Stages for Digital Filters



Digital Filter Design Procedure

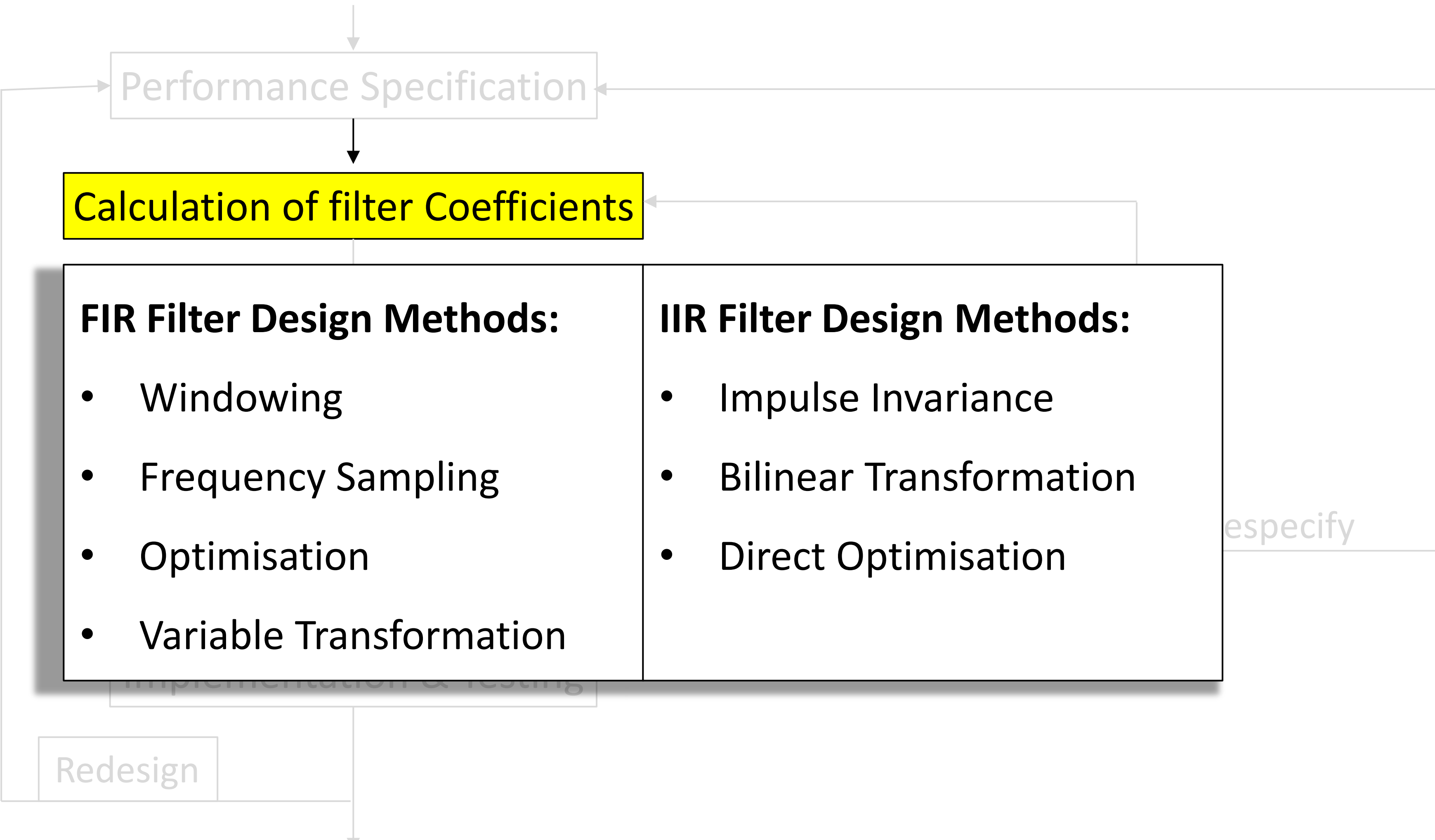
Design Stages for Digital Filters

Performance Specification



Digital Filter Design Procedure

Design Stages for Digital Filters



FIR Filter Design Methods:

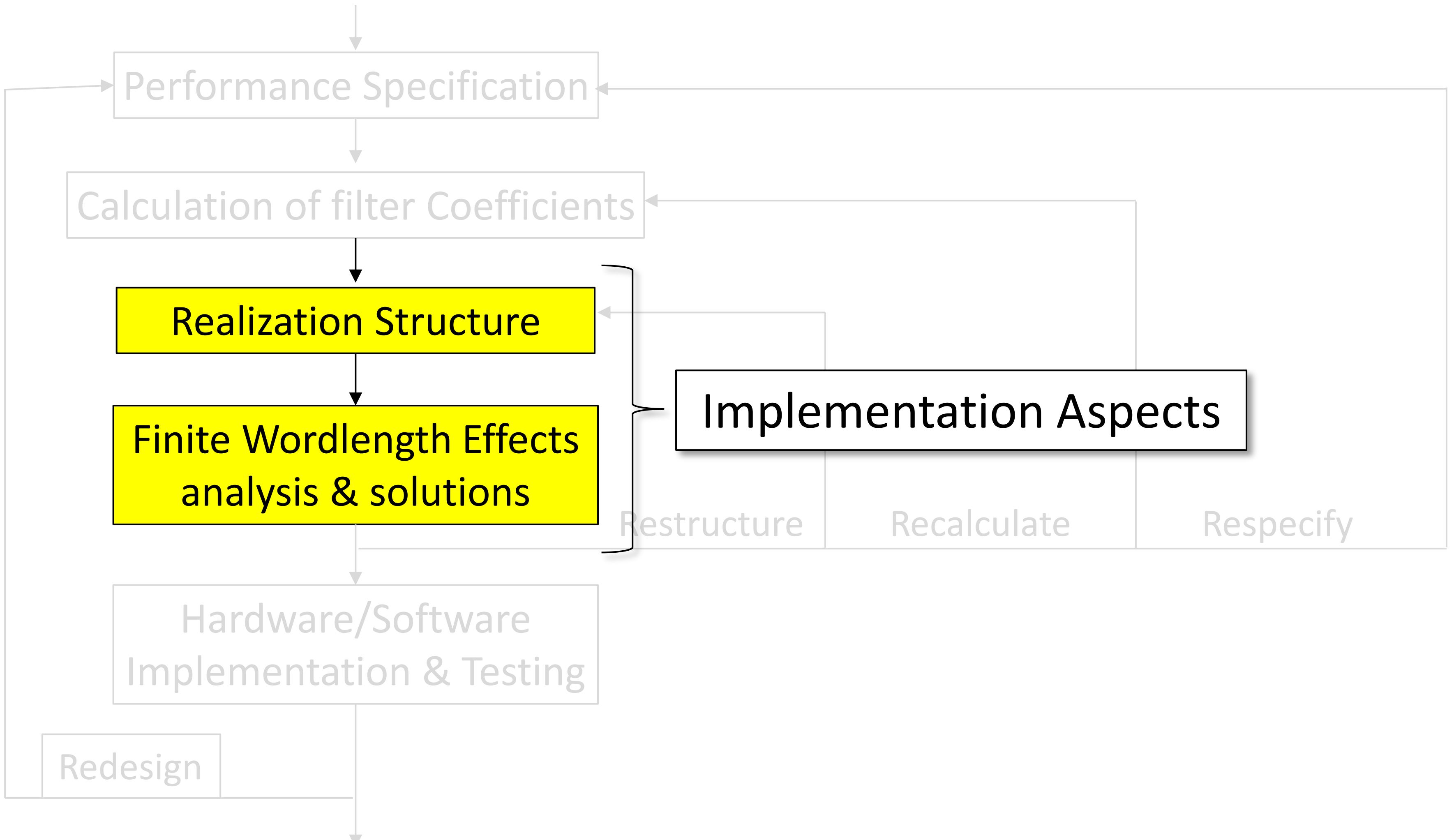
- Windowing
- Frequency Sampling
- Optimisation
- Variable Transformation

IIR Filter Design Methods:

- Impulse Invariance
- Bilinear Transformation
- Direct Optimisation

Digital Filter Design Procedure

Design Stages for Digital Filters



Linear Constant Coefficient Difference Equations

Realisation structures from difference equations

LTI System function		Time Domain : Difference equation
$H(z) = \sum_{k=0}^{N} b_k z^{-k} \Big/ \sum_{k=0}^{N} a_k z^{-k}$	$\Leftrightarrow$	$y[n] = \sum_{k=0}^{N} b_k x[n-k] - \sum_{k=1}^{N} a_k y[n-k]$

1st order system example: $H(z) = \dfrac{b_0 + b_1 z^{-1}}{1 - a_1 z^{-1}} \Leftrightarrow y[n] - a_1 y[n-1] = b_0 x[n] + b_1 x[n-1]$ (1)

Rewrite (1): $$y[n] = a_1 y[n-1] + b_0 x[n] + b_1 x[n-1]$$

Recursive computation of the output ***y*** at any time ***n*** based on the previous output ***y[n-1]*** *,* the current input ***x[n]*** and the previous input sample ***x[n-1]***

Linear Constant Coefficient Difference Equations

Realisation structures from difference equations

LTI System function		Time Domain : Difference equation
$H(z) = \sum_{k=0}^{N} b_k z^{-k} \Big/ \sum_{k=0}^{N} a_k z^{-k}$	$\Leftrightarrow$	$y[n] = \sum_{k=0}^{N} b_k x[n-k] - \sum_{k=1}^{N} a_k y[n-k]$

Realisation structures

- Methods of implementing difference equations in hardware / software
- Described by means of block diagrams or flow graphs
- Software implementation
  - Block diagram serves as the basis for a program that implements the filter
- Hardware implementation
  - Block diagram serves as a basis for determining the hardware architecture
- Multiple equivalent structures result from algebraic/block diagram manipulations

Realisation Structures from Difference Equations

Why multiple realisation structures?

1. **Computational complexity**
   - Number of multiplications, additions, divisions
   - Numbers of memory fetches, numerical comparisons
2. **Memory requirements**
   - Number of memory locations for storing past inputs, outputs and intermediate values
3. **Finite-word length effects**
   - Rounding / truncation of output / intermediate results due to finite precision available (e.g. 16bits or 32bits)
   - Different structures exhibit different behaviour under finite-precision arithmetic

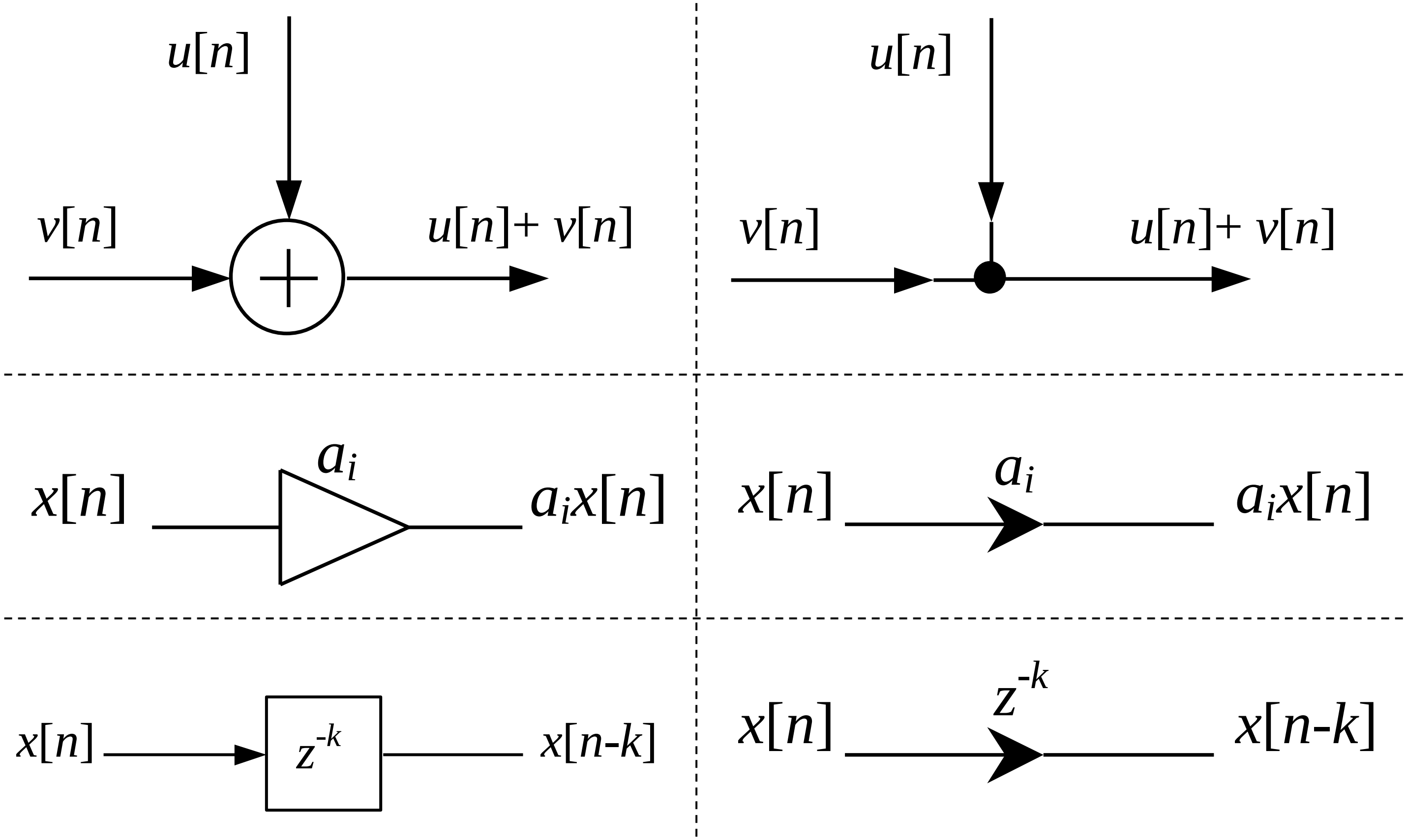
Block Diagram Representation of Difference Equations

Building blocks

Addition

Multiplication

Delay



Block Diagram Notation | Flow Graph Notation

Block Diagram Representation of Difference Equations

Example 1: 2nd order difference equation

2nd order filter: $H(z) = \dfrac{b_0}{1 - a_1 z^{-1} - a_2 z^{-2}} \Leftrightarrow y[n] = a_1 y[n-1] + a_2 y[n-2] + b_0 x[n]$

Computational algorithm:

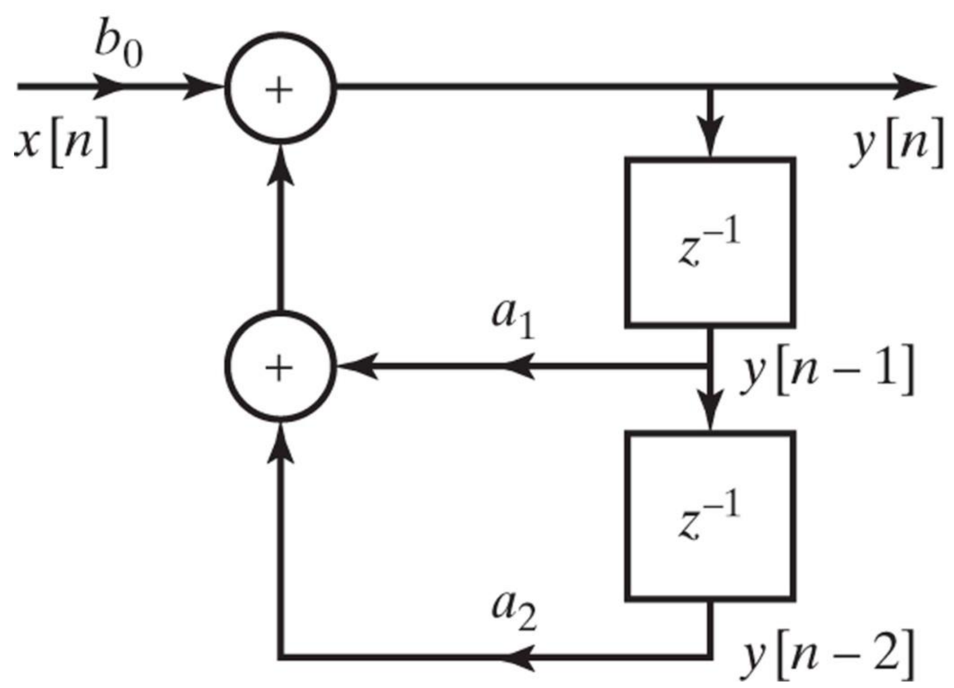
1. Form the products $a_1 y[n-1]$ and $a_2 y[n-2]$
2. Add them
3. Add the result to the product $b_0 x[n]$

Memory requirements

- Storage required for delayed variables y[n-1], y[n-2]
- Storage for coefficients a_1 and a_2 and b_0

Computational complexity

- 2 additions
- 3 multiplications
- 2 memory locations



Block Diagram Representation of Difference Equations

General case: Higher order difference equations

Higher order filter: $H(z)=\sum_{k=0}^{M} b_k z^{-k} \Big/ 1-\sum_{k=1}^{N} a_k z^{-k} \Leftrightarrow y[n]=\sum_{k=1}^{N} a_k y[n-k]+\sum_{k=0}^{M} b_k x[n-k]$

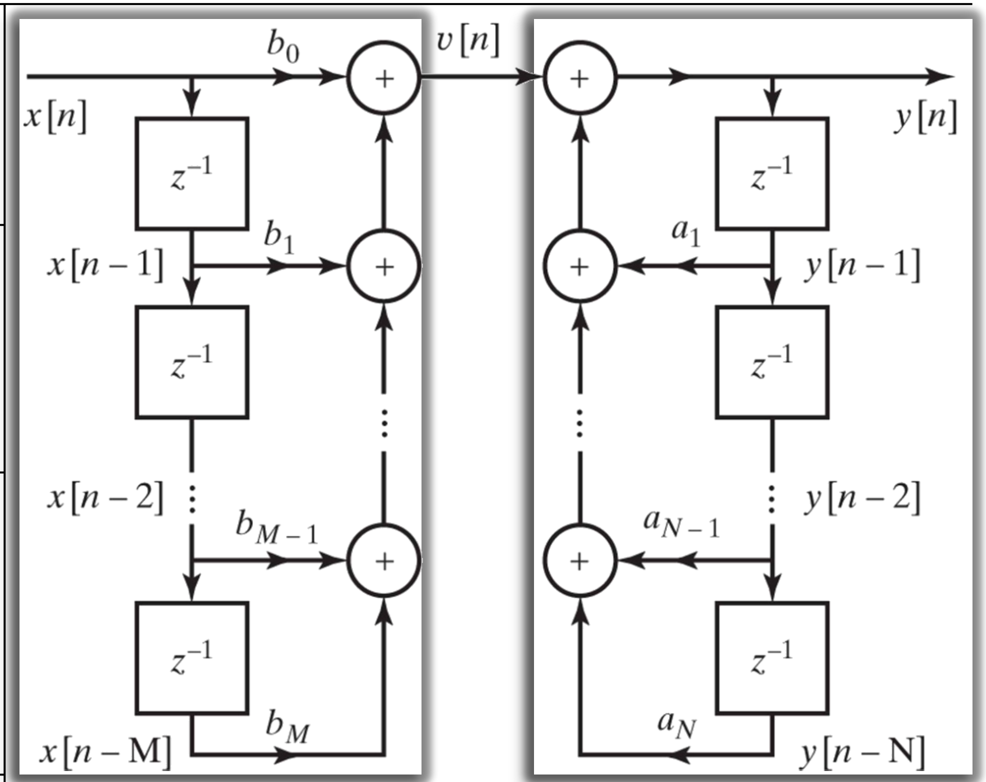
Difference equation 1

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

Difference equation 2

$$y[n]=\sum_{k=1}^{N} a_k y[n-k]+v[n]$$

Block diagram represents a pair of difference equations



$v[n]$ $y(n)$

Filter Realisation Structures

Block Diagram Representation of Difference Equations

General case: Higher order difference equations

Higher order filter: $H(z) = \sum_{k=0}^{M} b_k z^{-k} \Big/ 1 - \sum_{k=1}^{N} a_k z^{-k} \Leftrightarrow y[n] = \sum_{k=1}^{N} a_k y[n-k] + \sum_{k=0}^{M} b_k x[n-k]$

System Function 1

$$V(z) = H_1(z)X(z) = \left(\sum_{k=0}^{M} b_k z^{-k} \right) X(z)$$

System Function 2

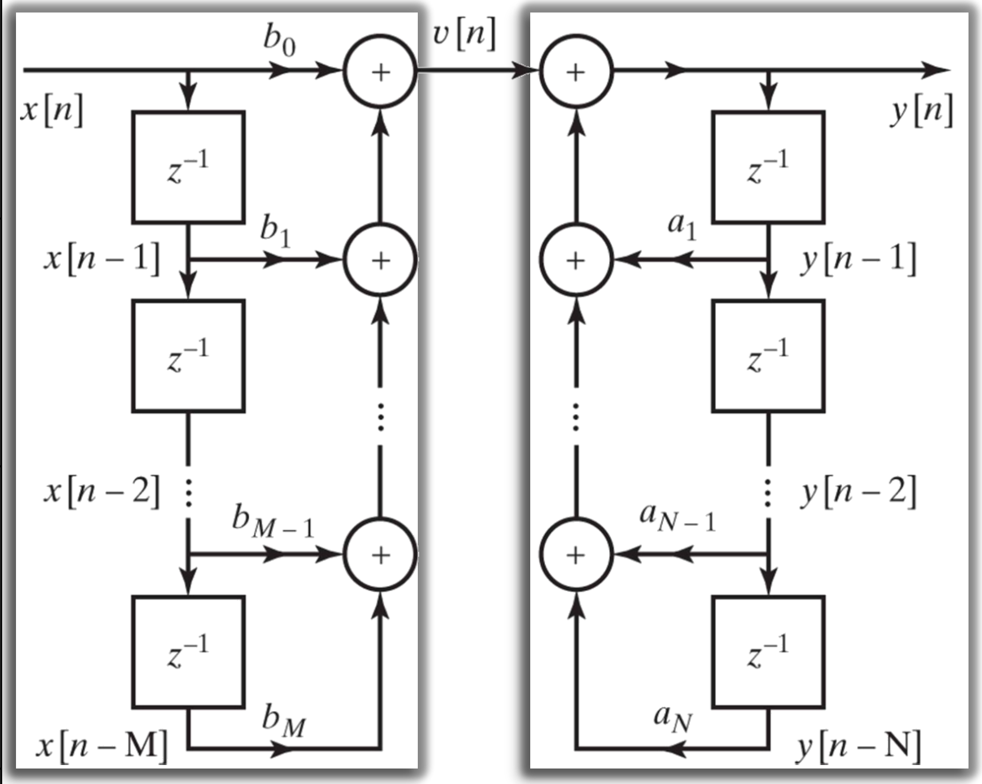
$$Y(z) = H_2(z)V(z) = \left(\frac{1}{1 - \sum_{k=1}^{N} a_k z^{-k}} \right) V(z)$$

System is a cascade of two LTI systems

$$H(z) = H_2(z)H_1(z)$$

$$H_1(z) = \sum_{k=0}^{M} b_k z^{-k}, \quad H_2(z) = 1/(1 - \sum_{k=1}^{N} a_k z^{-k})$$

$$Y(z) = H(z)X(z) = H_2(z)\ H_1(z)X(z)$$



$V(z) = H_1(z)X(z)$ $\qquad$ $Y(z) = H_2(z)V(z)$

Block Diagram Representation of Difference Equations

General case: Higher order difference equations

Higher order filter: $H(z)=\sum_{k=0}^{M} b_k z^{-k} \Big/ 1-\sum_{k=1}^{N} a_k z^{-k} \Leftrightarrow y[n]=\sum_{k=1}^{N} a_k y[n-k]+\sum_{k=0}^{M} b_k x[n-k]$

System Function 1

$$W(z)=H_2(z)X(z)=\left(\frac{1}{1-\sum_{k=1}^{N} a_k z^{-k}}\right)X(z)$$

System Function 2

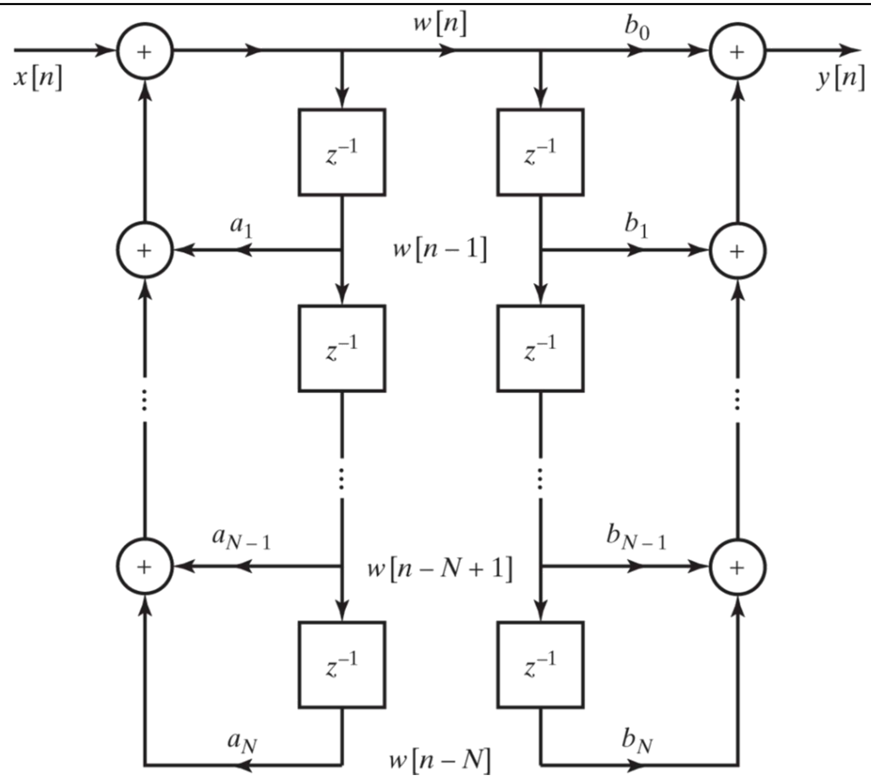
$$Y(z)=H_1(z)W(z)=\left(\sum_{k=0}^{M} b_k z^{-k}\right)W(z)$$

System is a cascade of two LTI systems

$$H(z)=H_1(z)H_2(z)$$

$$H_1(z)=\sum_{k=0}^{M} b_k z^{-k}, \quad H_2(z)=1/(1-\sum_{k=1}^{N} a_k z^{-k})$$

$$Y(z)=H(z)X(z)=H_1(z)\ H_2(z)X(z)$$



$$W(z)=H_2(z)X(z)$$

$$Y(z)=H_1(z)W(z)$$

Filter Realisation Structures

Block Diagram Representation of Difference Equations

General case: Higher order difference equations

Higher order filter: $H(z) = \sum_{k=0}^{M} b_k z^{-k} \Big/ 1 - \sum_{k=1}^{N} a_k z^{-k} \Leftrightarrow y[n] = \sum_{k=1}^{N} a_k y[n-k] + \sum_{k=0}^{M} b_k x[n-k]$

Difference equation 1

$$w[n] = \sum_{k=1}^{N} a_k w[n-k] + x[n]$$

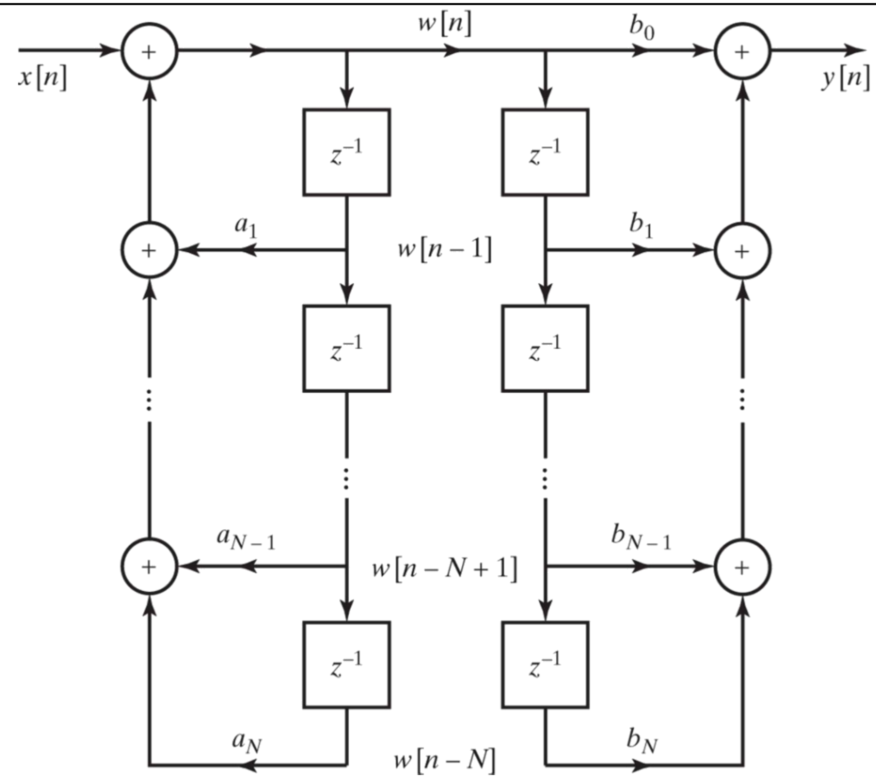
Difference equation 2

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

System is a cascade of two LTI systems

$$H(z) = H_1(z) H_2(z)$$

$$H_1(z) = \sum_{k=0}^{M} b_k z^{-k}, \quad H_2(z) = 1/(1 - \sum_{k=1}^{N} a_k z^{-k})$$



$$Y(z) = H(z) X(z) = H_1(z) \; H_2(z) X(z)$$

$$W(z) = H_2(z) X(z)$$

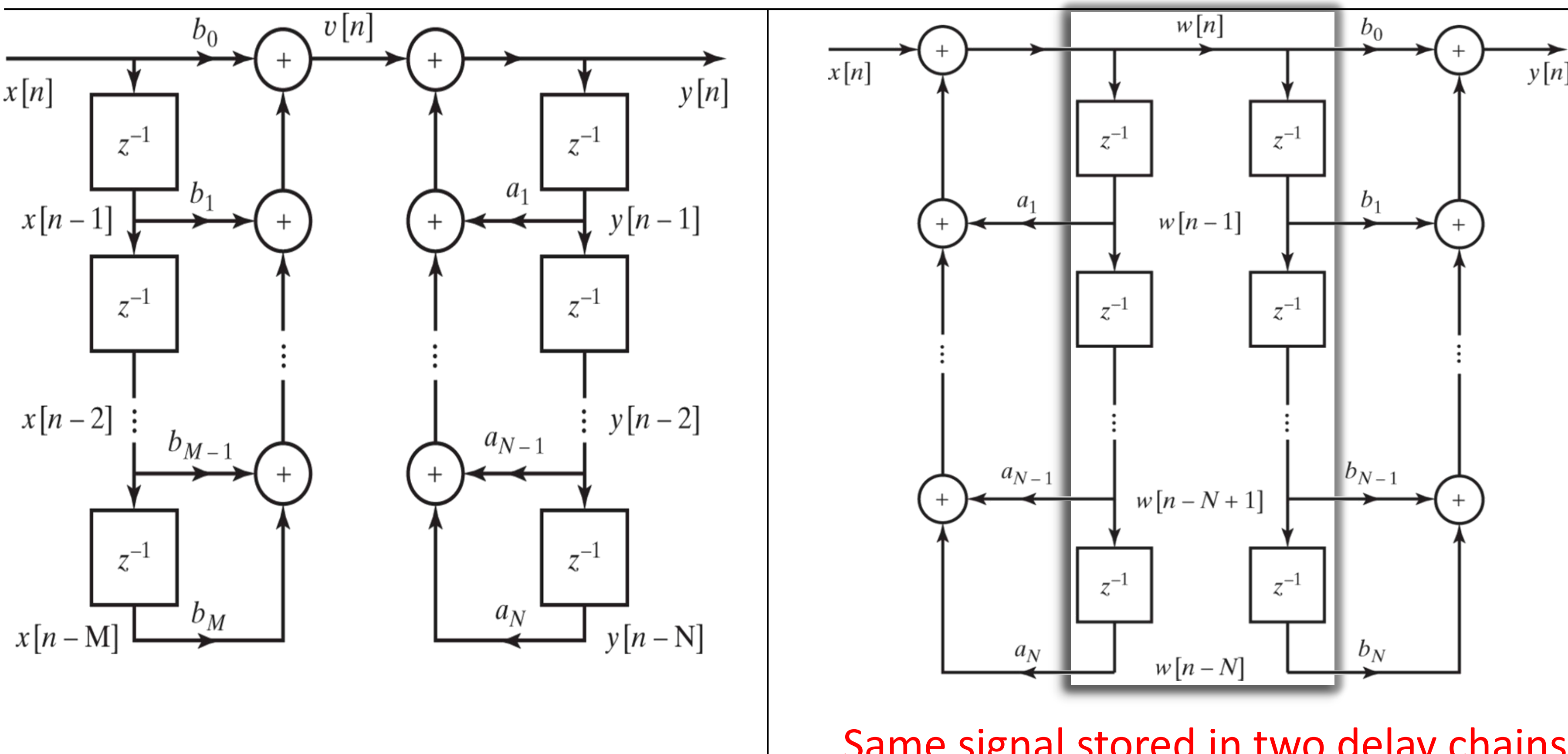
$$Y(z) = H_1(z) W(z)$$

Filter Realisation Structures

Block Diagram Representation of Difference Equations

General case: Higher order difference equations

Higher order filter: $H(z)=\sum_{k=0}^{M} b_k z^{-k} \Big/ 1-\sum_{k=1}^{N} a_k z^{-k} \Leftrightarrow y[n]=\sum_{k=1}^{N} a_k y[n-k]+\sum_{k=0}^{M} b_k x[n-k]$



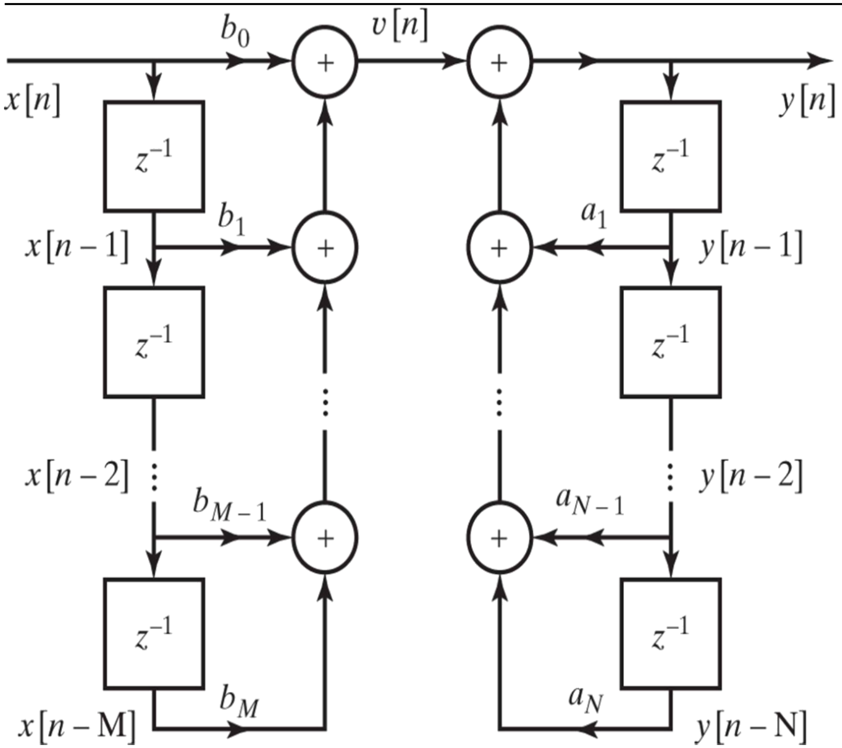
Same signal stored in two delay chains

Filter Realisation Structures

IIR Direct Forms

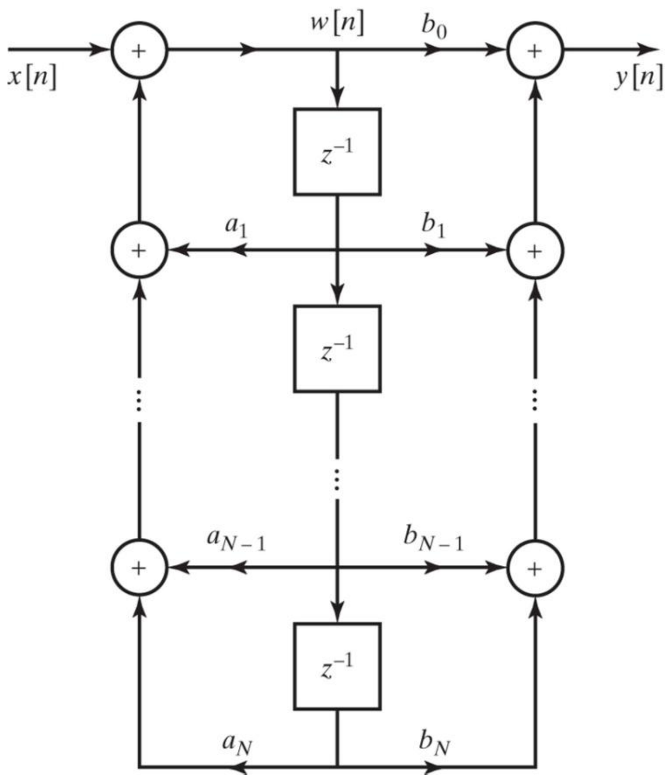
Direct Form I & Direct Form II

Higher order filter: $H(z) = \sum_{k=0}^{M} b_k z^{-k} \Big/ 1 - \sum_{k=1}^{N} a_k z^{-k} \Leftrightarrow y[n] = \sum_{k=1}^{N} a_k y[n-k] + \sum_{k=0}^{M} b_k x[n-k]$



DIRECT FORM I

Direct realisation of the difference equation



DIRECT FORM II – CANONIC FORM

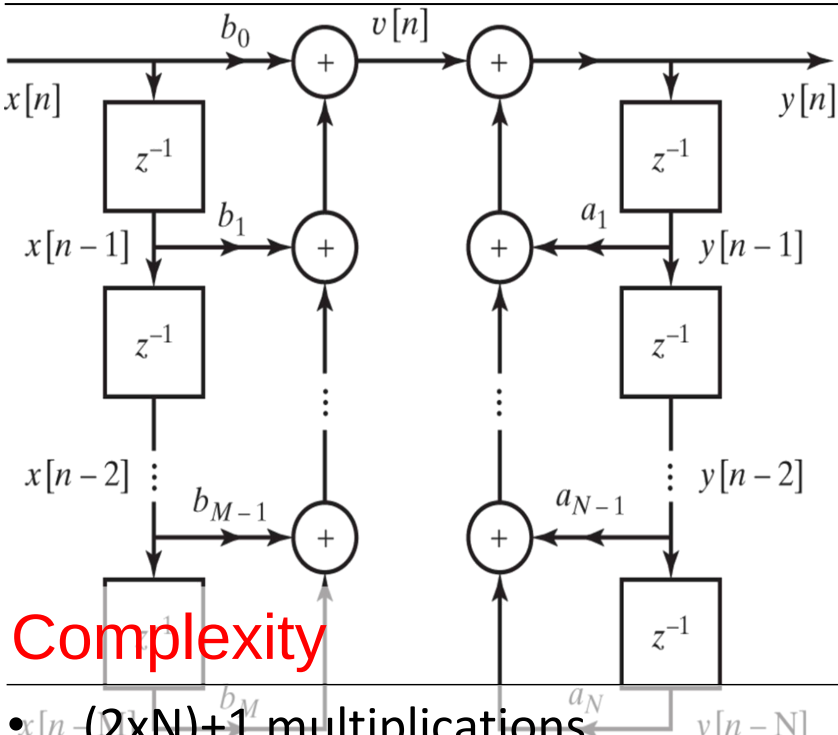
Form with minimum number of delay elements

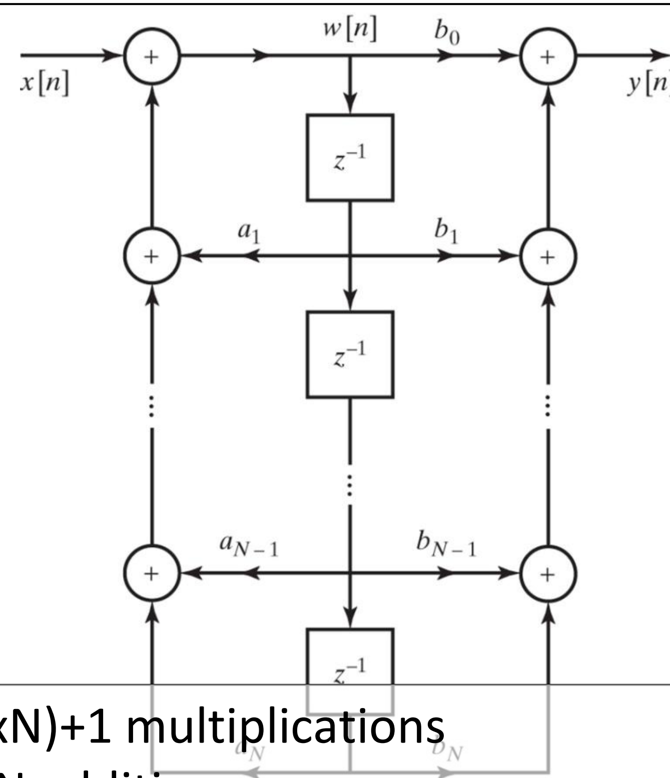
Filter Realisation Structures

IIR Direct Forms

Direct Form I & Direct Form II

Higher order filter: $H(z) = \sum_{k=0}^{M} b_k z^{-k} \Big/ 1 - \sum_{k=1}^{N} a_k z^{-k} \Leftrightarrow y[n] = \sum_{k=1}^{N} a_k y[n-k] + \sum_{k=0}^{M} b_k x[n-k]$





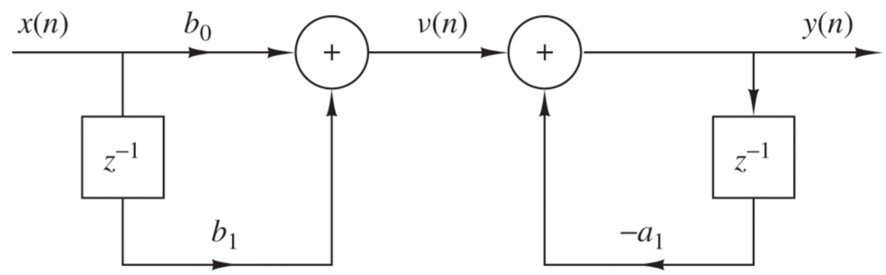
Complexity

- (2xN)+1 multiplications
- 2xN additions
- 2xN memory locations

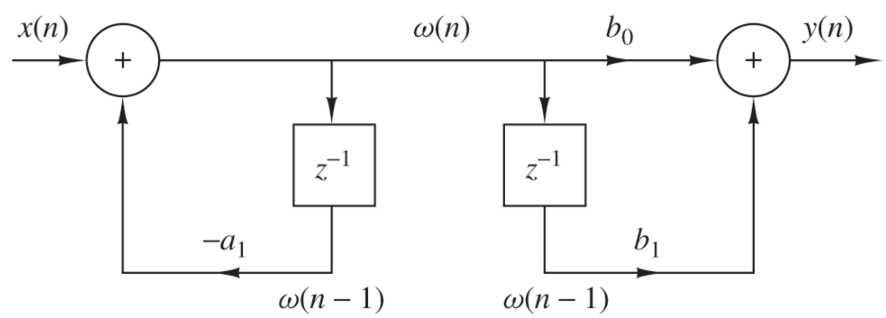
- (2xN)+1 multiplications
- 2xN additions
- **N memory locations**

IIR Direct Forms – From Direct Form I to Direct Form II

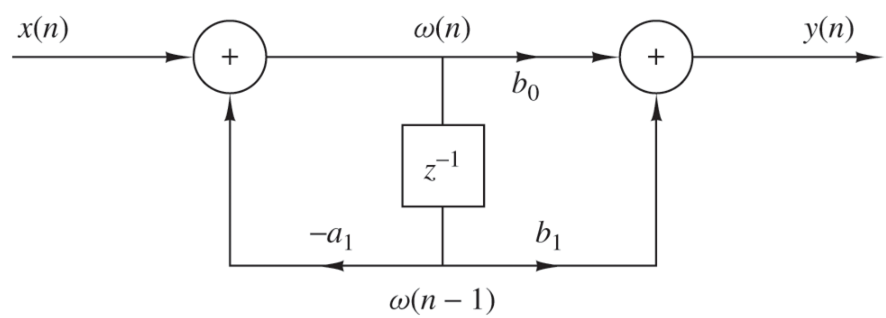
Direct Form I



- Interchange order of cascade
- Common input to delays



Direct Form II



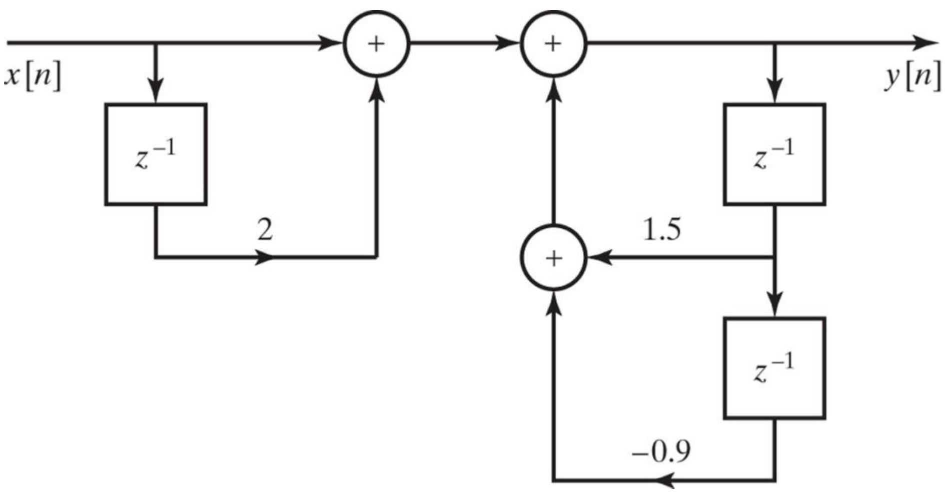
Filter Realisation Structures

IIR Direct Forms

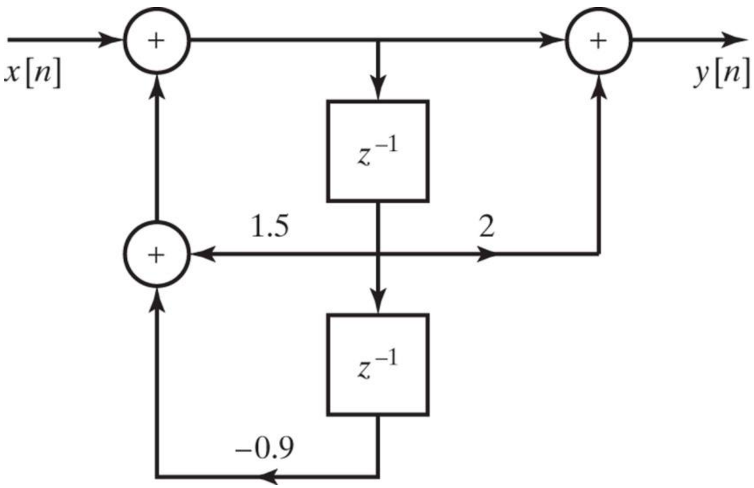
Example 2 : Draw Direct Form I and Direct Form II Block Diagrams

System Function: $H(z) = \dfrac{1+2z^{-1}}{1-1.5z^{-1}+0.9z^{-2}}$

$b_0 = 1,\ b_1 = 2,\ a_1 = +1.5,\ a_2 = -0.9$



DIRECT FORM I



DIRECT FORM II

Coefficients in the feedback branches in the block diagram have opposite signs from the corresponding coefficients a_k in the system function ⇔ Feedback coefficients a_k always have the opposite sign in the difference equation from their sign in the system function

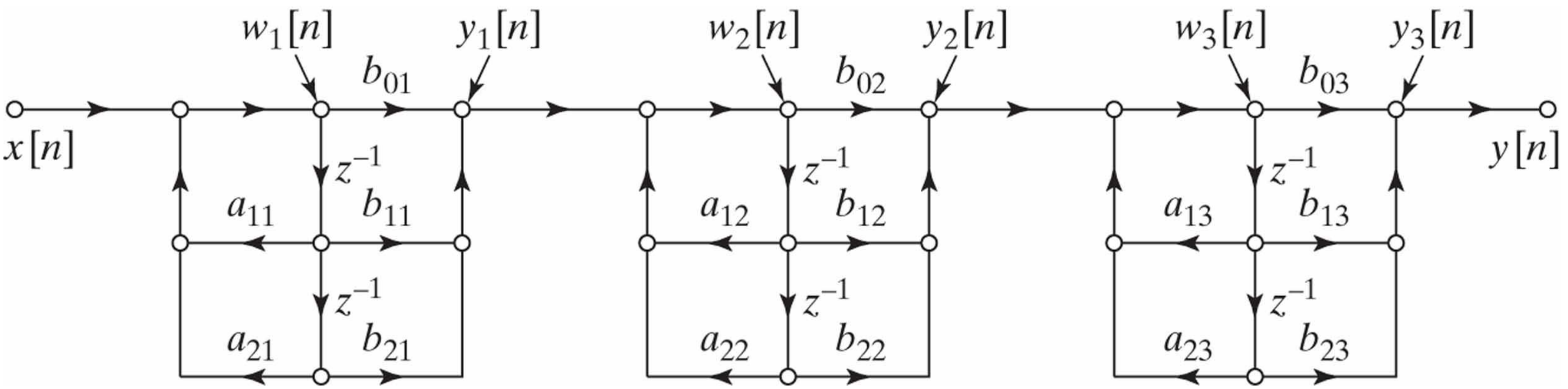
Filter Realisation Structures

IIR Cascade Form

Realisation Structure resulting from Factorization of System Function

System Function: $H(z)=\sum_{k=0}^{M} b_k z^{-k} \Big/ 1-\sum_{k=1}^{N} a_k z^{-k} = \prod_{k=1}^{N_s} H_k(z) = \prod_{k=1}^{N_s} \frac{b_{0k}+b_{1k}z^{-1}+b_{2k}z^{-2}}{1-a_k z^{-1}-a_{2k}z^{-2}}$

- Implement each 2[nd] order systems in one of two Direct Forms
- Multiple equivalent structures can be obtained by different pairings of poles & zeros
- N_s! pairings and N_s! orderings of 2[nd] order systems => $(N_s!)^2$ structures, $N_s=\lfloor (N+1)/2 \rfloor$



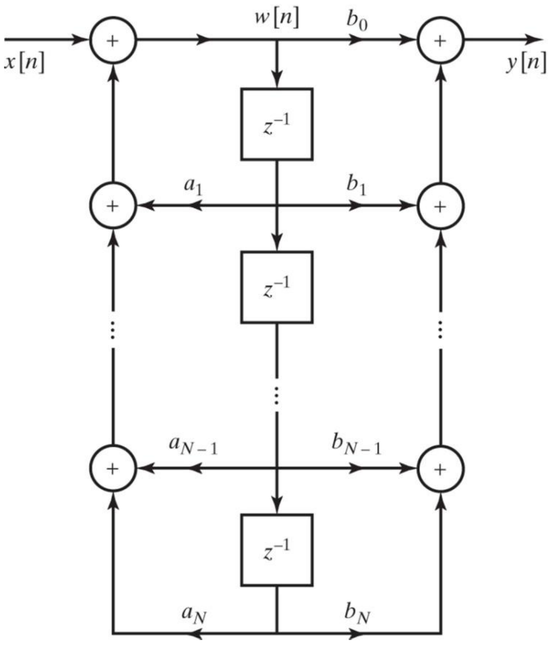
Cascade structure of 6[th] order system with a direct form II realization of each 2[nd] order subsystem

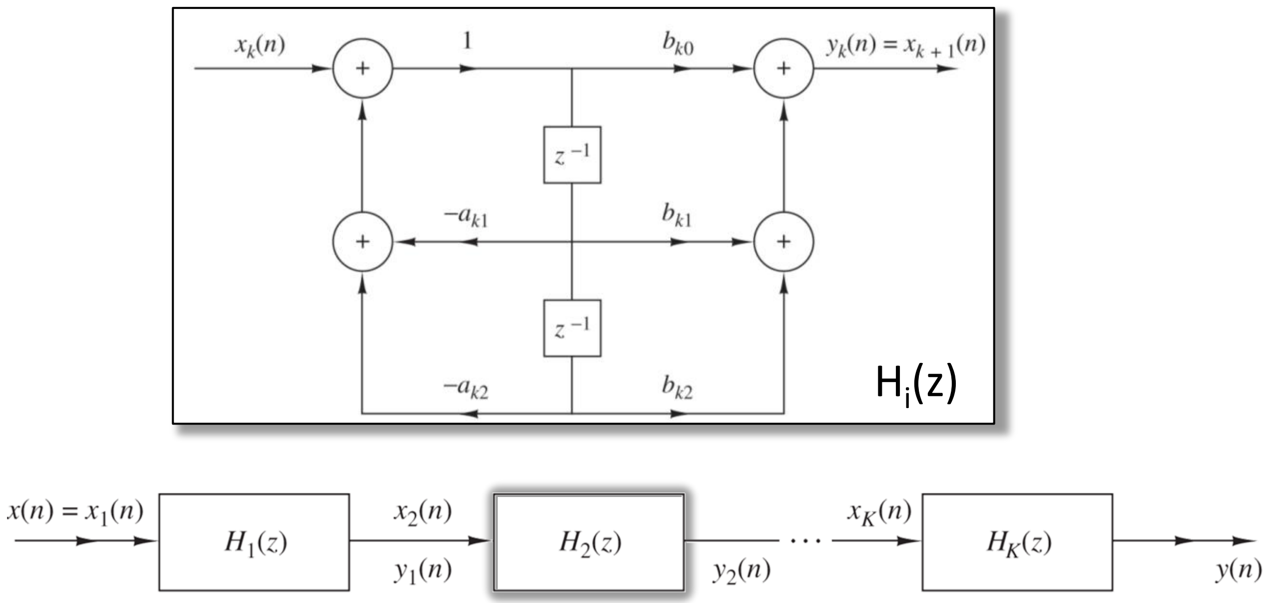
Filter Realisation Structures

IIR Cascade Form

Realisation Structure resulting from Factorization of System Function

System Function: $H(z)=\sum_{k=0}^{M} b_k z^{-k} \Big/ 1-\sum_{k=1}^{N} a_k z^{-k} \;=\prod_{k=1}^{N_s} H_k(z)=\prod_{k=1}^{N_s} \frac{b_{0k}+b_{1k}z^{-1}+b_{2k}z^{-2}}{1-a_k z^{-1}-a_{2k}z^{-2}}$



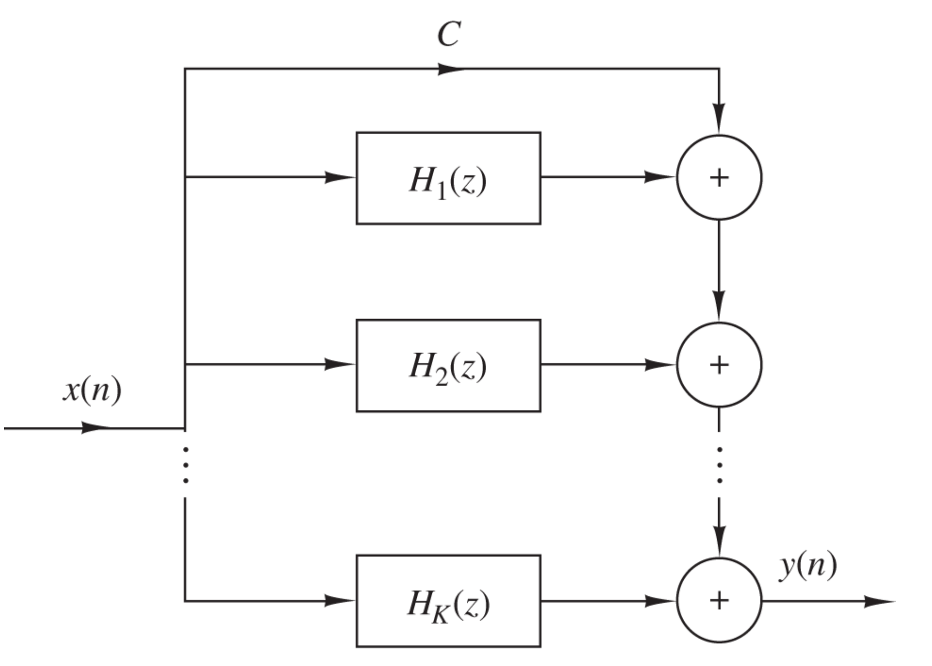


- Different structures -> different behaviour under finite arithmetic
- Overall gain of the system distributed, thus controlling the size of signals at various points

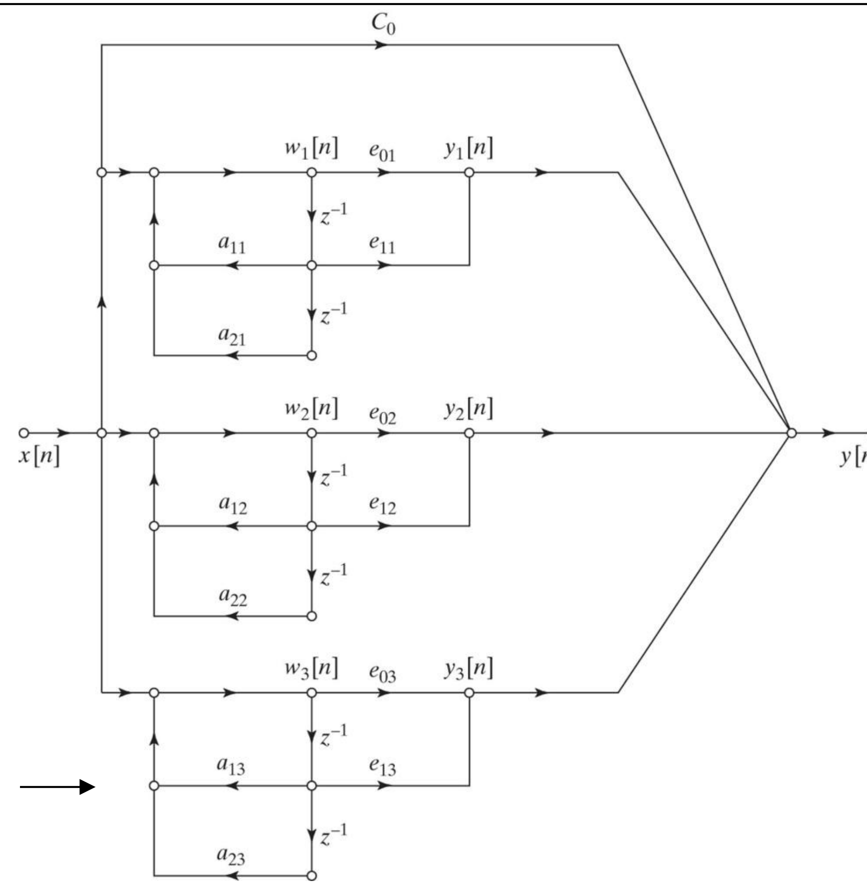
IIR Parallel Form

Realisation resulting from Partial Fraction Expansion of System Function

$$H(z)=\sum_{k=0}^{M} b_k z^{-k} \Big/ 1-\sum_{k=1}^{N} a_k z^{-k} = C+\sum_{i=1}^{L} H_i(z) = C+\sum_{i=1}^{L}\frac{A_i}{1-p_i z^{-1}} = b_0+\sum_{i=1}^{L}\frac{b_{i0}+b_{i1}z^{-1}}{1+a_{i1}z^{-1}+a_{i2}z^{-2}}$$



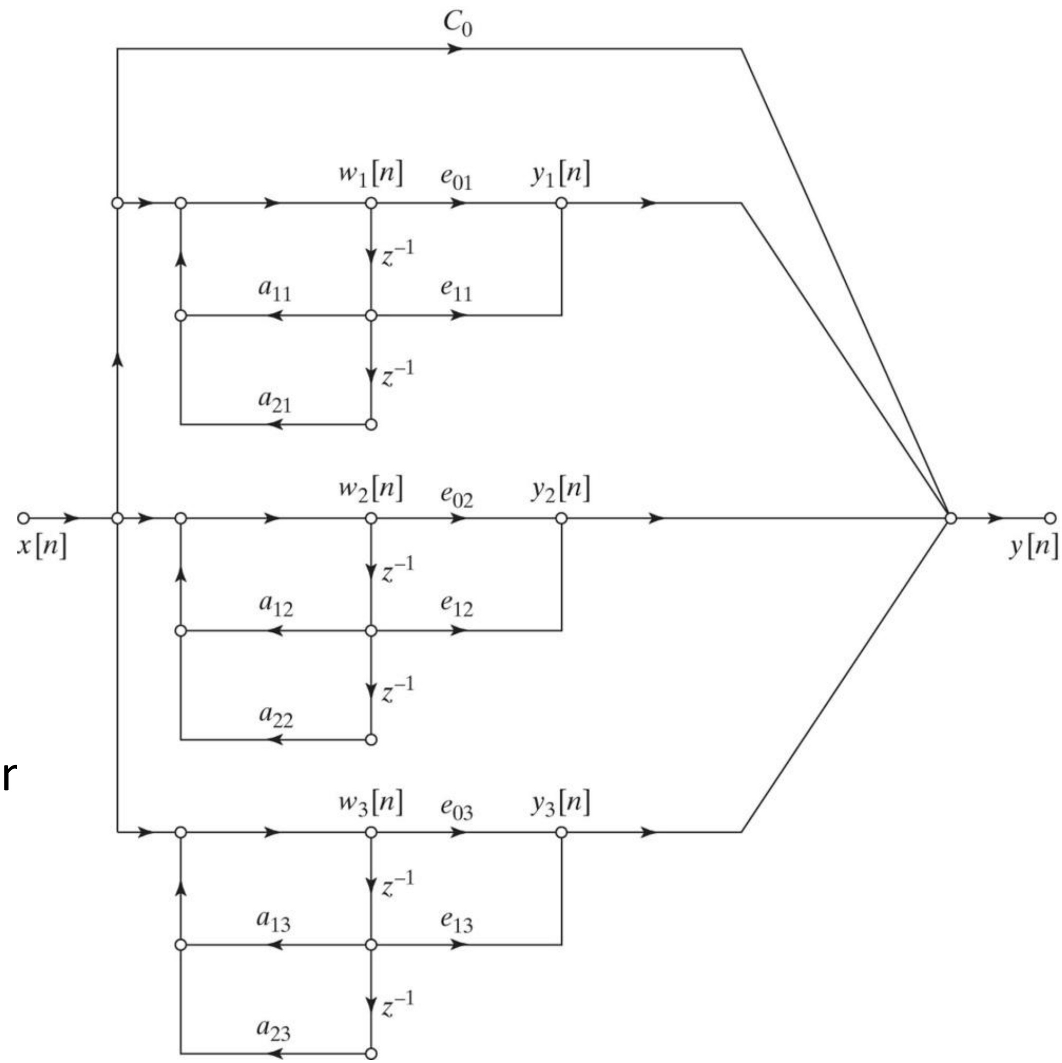
Parallel form structure for 6th-order system →



Filter Realisation Structures

IIR Parallel & Cascade Form

Structure from Partial Fraction Expansion & Factorisation of System Function



6th order

Realisation structure resulting fror

partial fraction expansion

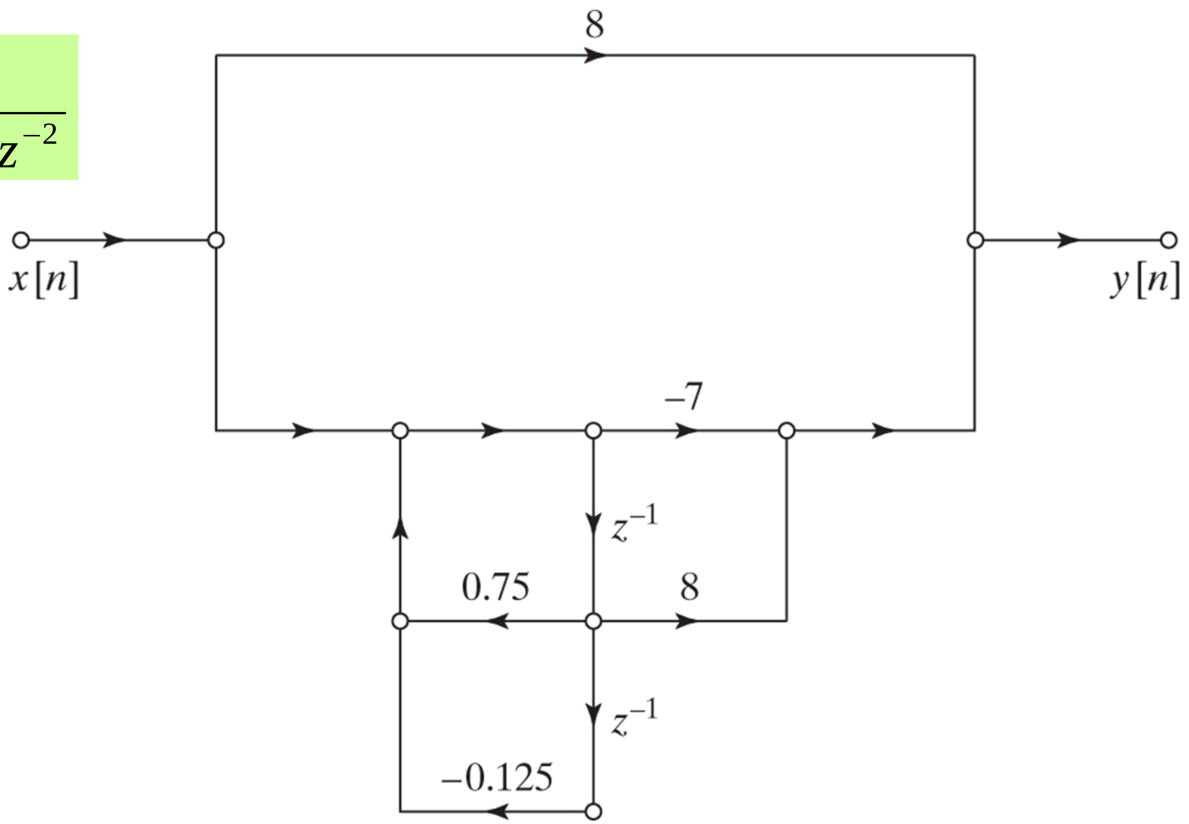
of the system function

IIR Parallel Form

Example 4

$$H(z) = \frac{1 + 2z^{-1} + z^{-2}}{1 - 0.75z^{-1} + 0.125z^{-2}} = C + \sum_{i=1}^{L} H_i(z) = C + \sum_{i=1}^{L} \frac{A_i}{1 - p_i z^{-1}} = b_0 + \sum_{i=1}^{L} \frac{b_{i0} + b_{i1} z^{-1}}{1 + a_{i1} z^{-1} + a_{i2} z^{-2}}$$

$$H(z) = 8 + \frac{-7 + 8z^{-1}}{1 - 0.75z^{-1} + 0.125z^{-2}}$$



Filter Realisation Structures

IIR Parallel Form

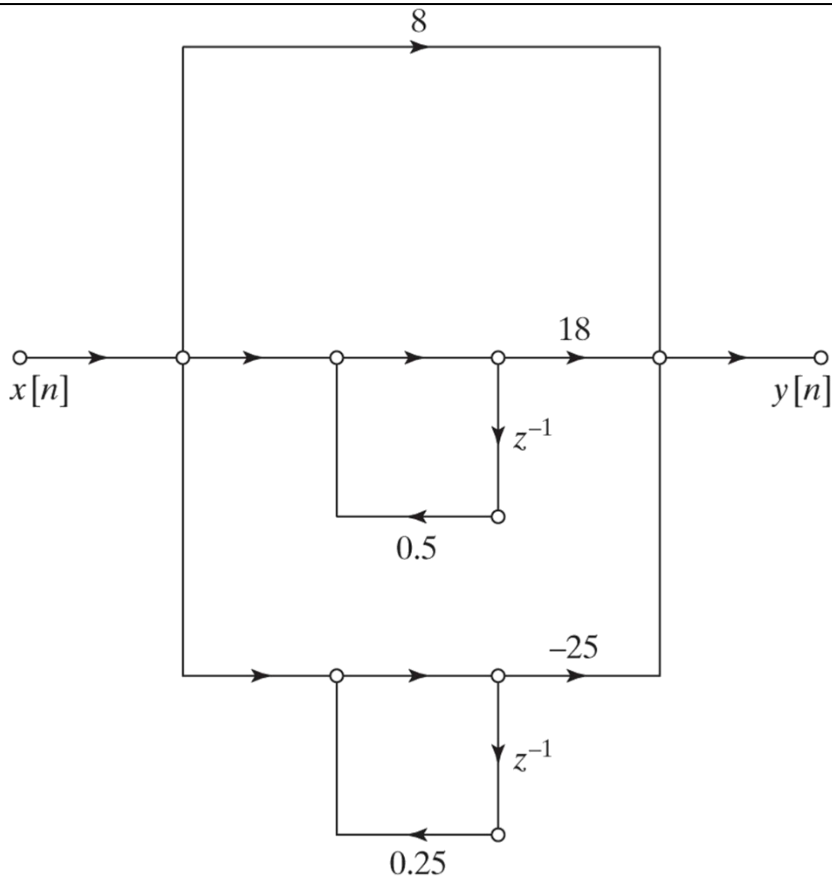
Example 4

$$H(z)=\frac{1+2z^{-1}+z^{-2}}{1-0.75z^{-1}+0.125z^{-2}}=C+\sum_{i=1}^{L}H_i(z)=C+\sum_{i=1}^{L}\frac{A_i}{1-p_iz^{-1}}=b_0+\sum_{i=1}^{L}\frac{b_{i0}+b_{i1}z^{-1}}{1+a_{i1}z^{-1}+a_{i2}z^{-2}}$$

$$H(z)=8+\frac{-7+8z^{-1}}{1-0.75z^{-1}+0.125z^{-2}}$$

Further Expansion

$$H(z)=8+\frac{18}{1-0.5z^{-1}}-\frac{25}{1-0.25z^{-1}}$$



Filter Realisation Structures

IIR Parallel Form

Example 5

The figure below shows a parallel implementation for an IIR filter. A) Evaluate the system function for this filter. B) Draw the corresponding direct form implementation.

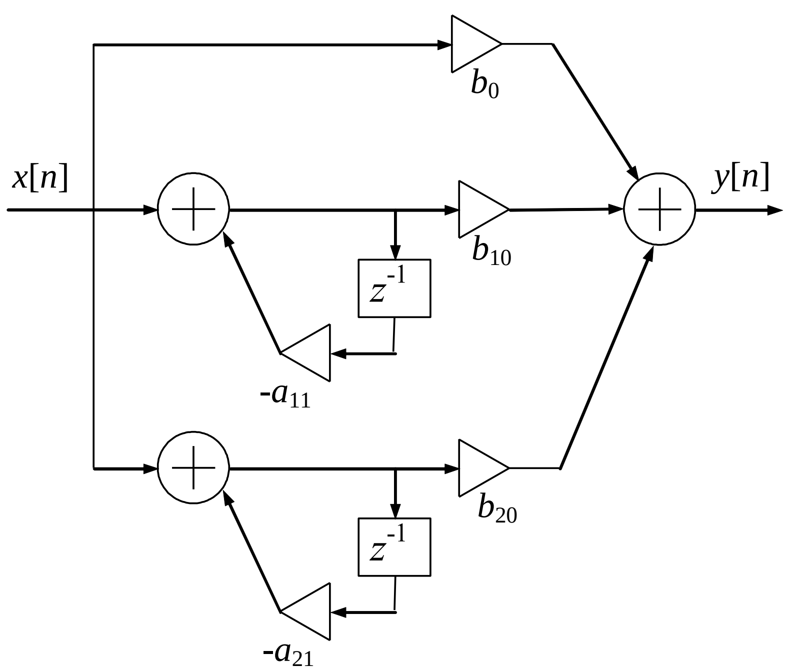
$$H(z) = b_0 + \frac{b_{10}}{1 + a_{11}z^{-1}} + \frac{b_{20}}{1 + a_{21}z^{-1}}$$

$$= \frac{b_0\left(1 + a_{11}z^{-1}\right)\left(1 + a_{21}z^{-1}\right) + b_{10}\left(1 + a_{21}z^{-1}\right) + b_{20}\left(1 + a_{11}z^{-1}\right)}{\left(1 + a_{11}z^{-1}\right)\left(1 + a_{21}z^{-1}\right)}$$

$$H(z) = \frac{\left(b_0 + b_{10} + b_{20}\right) + \left(b_0 a_{11} + b_0 a_{21} + b_{10} a_{21} + b_{20} a_{11}\right)z^{-1} + b_0 a_{11} a_{21} z^{-2}}{1 + \left(a_{11} + a_{21}\right)z^{-1} + a_{11}a_{21}z^{-2}}$$

$$a'_1 = a_{11} + a_{21} \qquad a'_2 = a_{11}.a_{21} \qquad b'_0 = b_0 + b_{10} + b_{20}$$

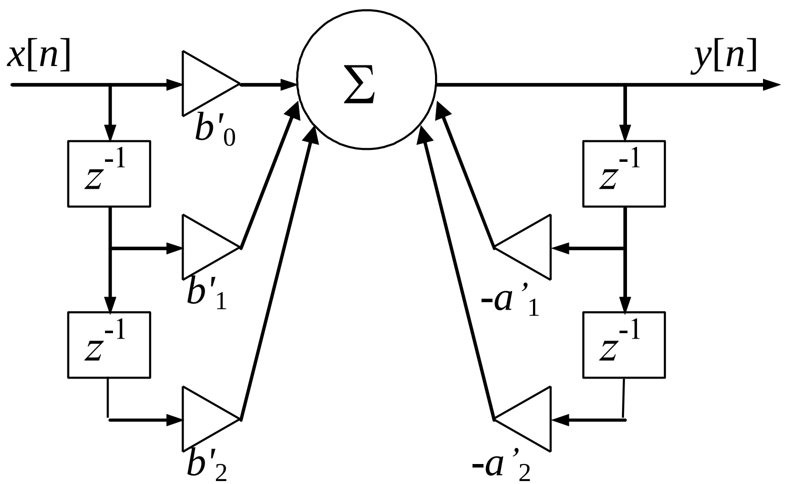
$$b'_1 = b_0.a_{11} + b_0.a_{21} + b_{20}.a_{11} + b_{10}.a_{21} \qquad b'_2 = b_0.a_{11}.a_{21}$$



IIR Parallel Form

Example 5

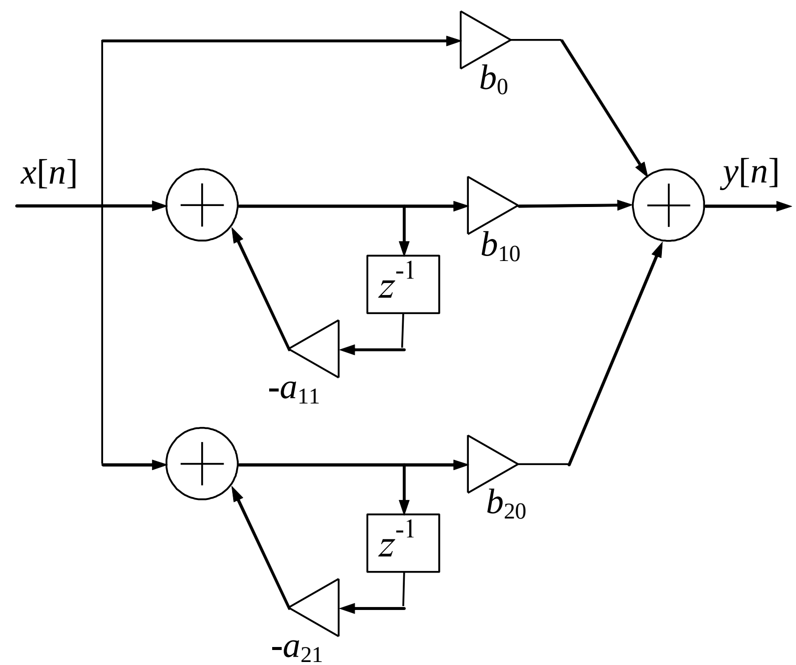
The figure below shows a parallel implementation for an IIR filter. A) Evaluate the transfer function for this filter. B) Draw the corresponding direct form implementation.



$$H(z)=\frac{(b_0+b_{10}+b_{20})+(b_0a_{11}+b_0a_{21}+b_{10}a_{21}+b_{20}a_{11})z^{-1}+b_0a_{11}a_{21}z^{-2}}{1+(a_{11}+a_{21})z^{-1}+a_{11}a_{21}z^{-2}}$$

$$a'_1=a_{11}+a_{21} \quad a'_2=a_{11}.a_{21} \quad b'_0=b_0+b_{10}+b_{20}$$

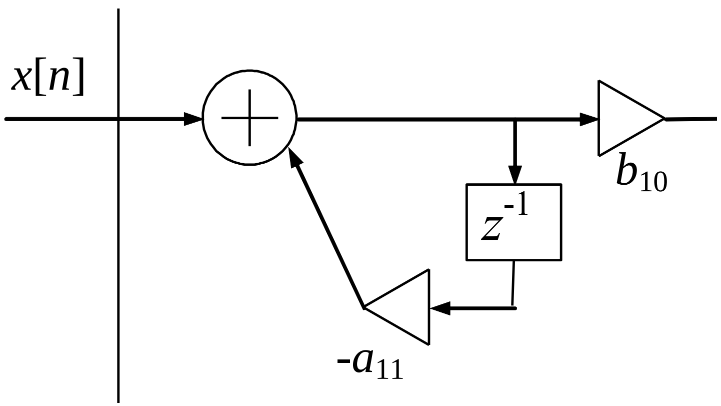
$$b'_1=b_0.a_{11}+b_0.a_{21}+b_{20}.a_{11}+b_{10}.a_{21} \quad b'_2=b_0.a_{11}.a_{21}$$



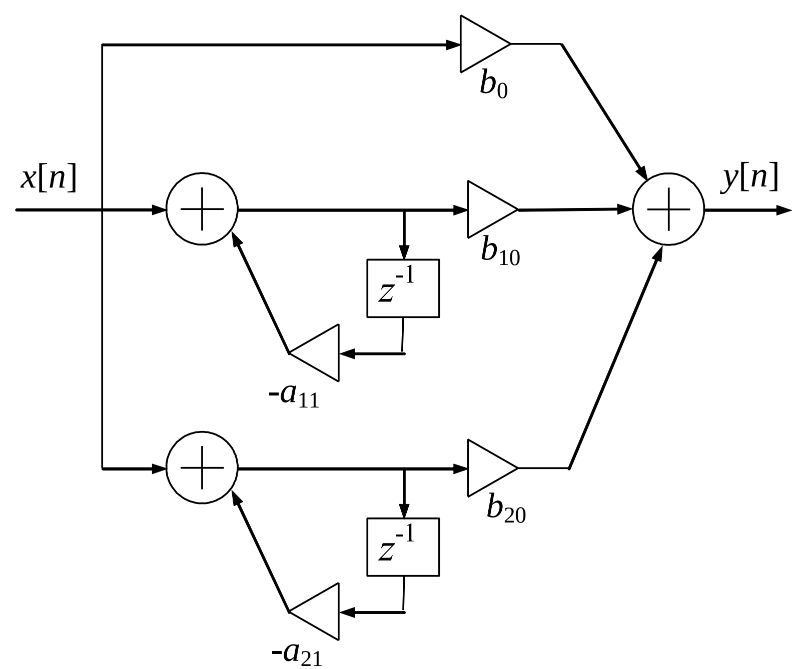
IIR Parallel Form

?

The figure below shows a parallel implementation for an IIR filter. A) Evaluate the transfer function for this filter. B) Draw the corresponding direct form implementation.



Which form is this ?



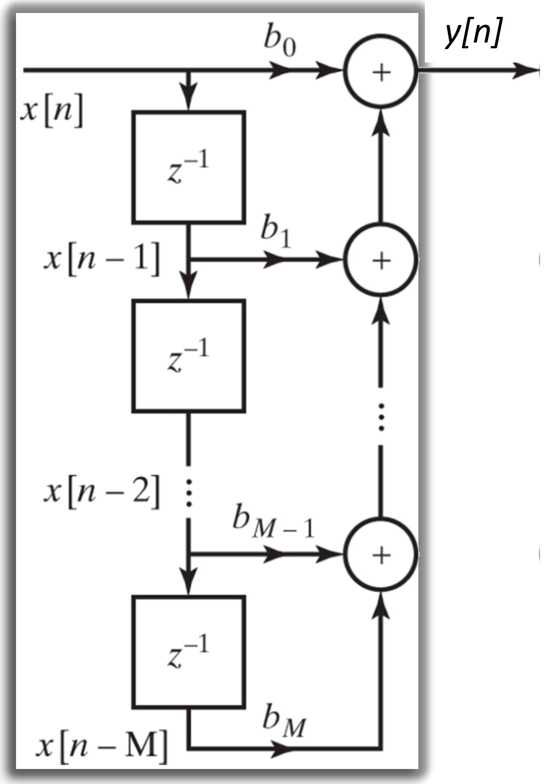
FIR Direct Form Structure

AKA Tapped Delay Line

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

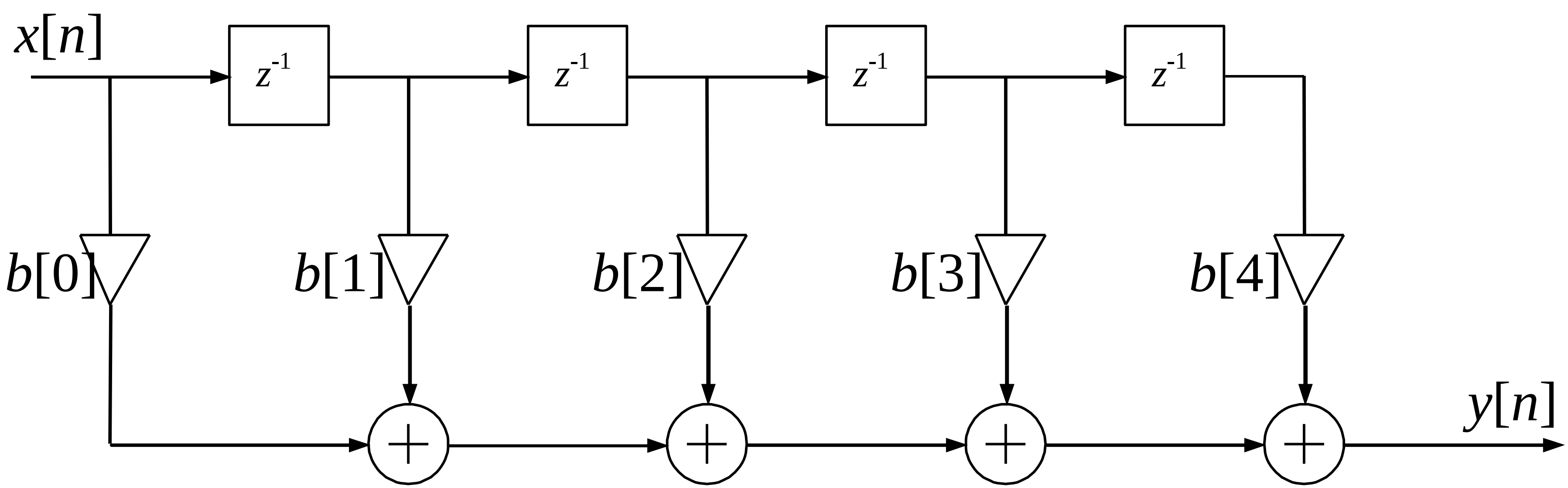
$$\Leftrightarrow Y(z) = \sum_{k=0}^{N} b_k z^{-k} X(z)$$

$$\Leftrightarrow H(z) = \sum_{k=0}^{N} b_k z^{-k}$$



FIR Direct Form Structure

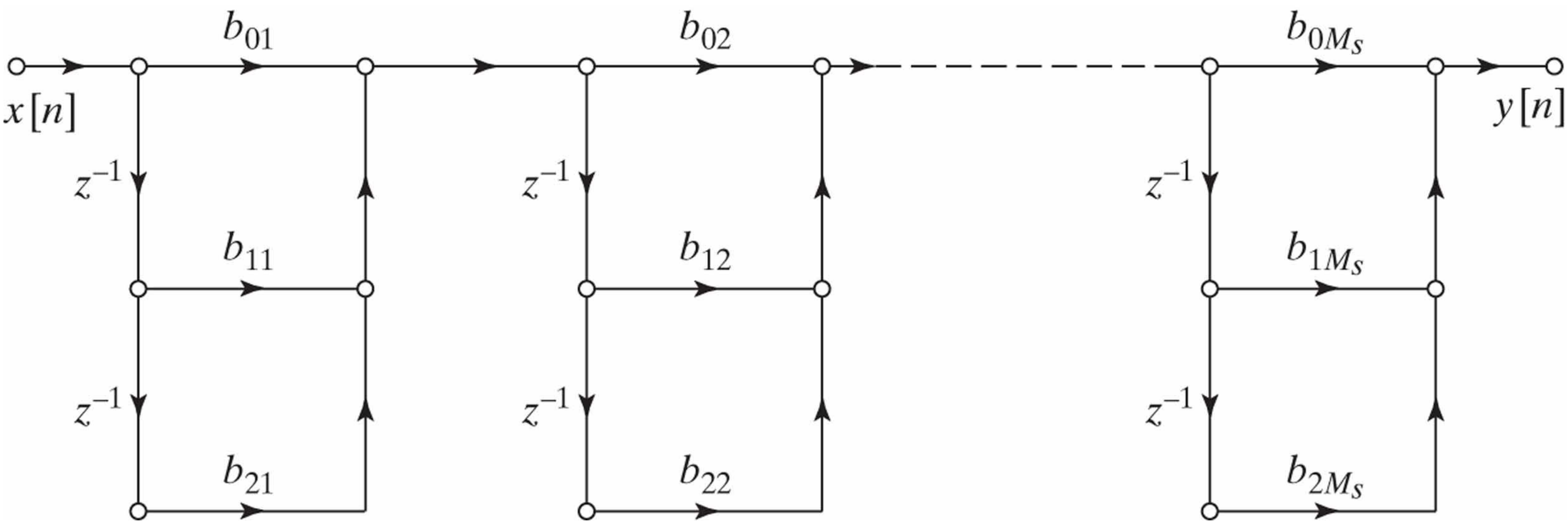
AKA Tapped Delay Line



$$y[n]=\sum_{k=0}^{N} b_k x[n-k]=b[n]*x[n] \;\Leftrightarrow Y(z)=\sum_{k=0}^{N} b_k z^{-k} X(z) \;\Leftrightarrow H(z)=\sum_{k=0}^{N} b_k z^{-k}$$

FIR Direct Form Structure

Cascade form from factorisation

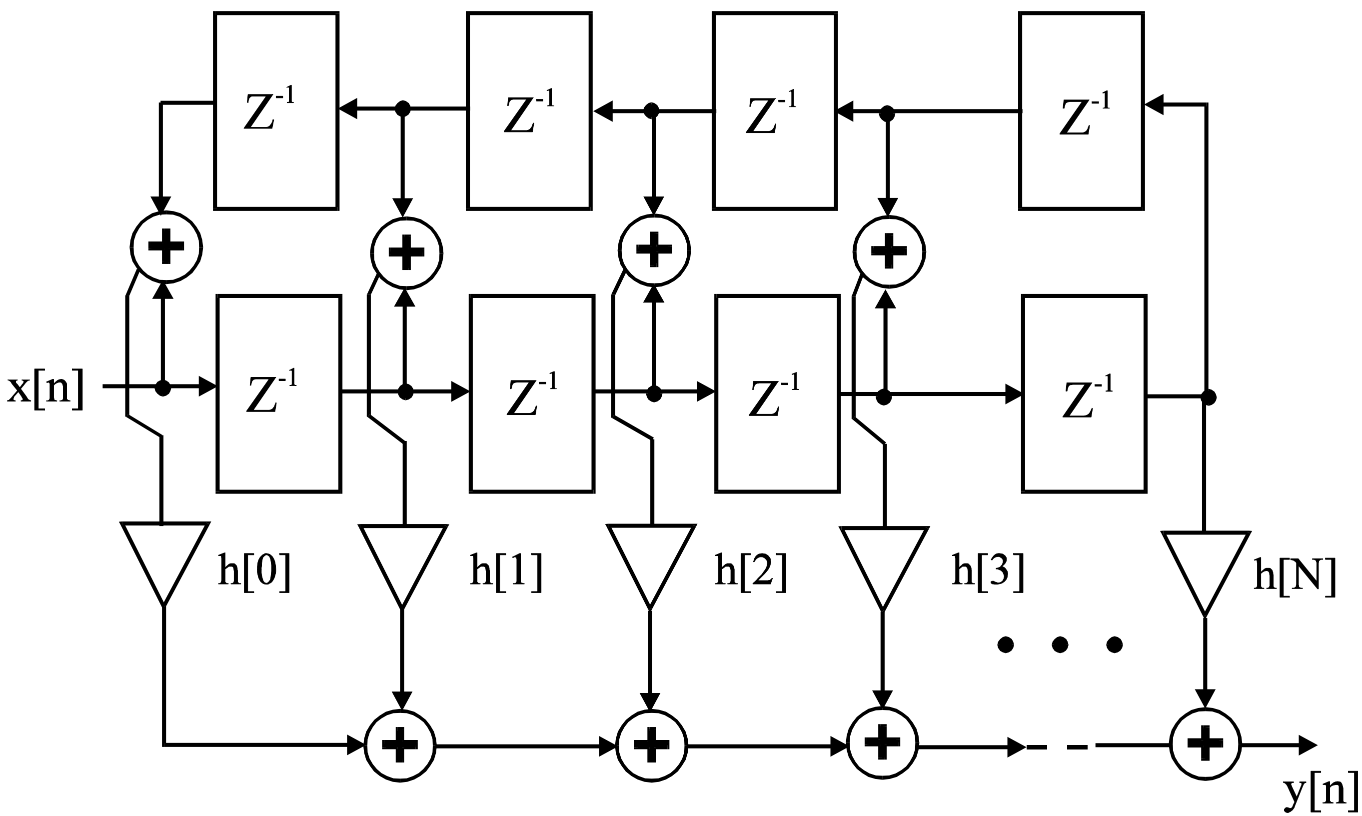


$$y[n]=\sum_{k=0}^{N} b_k x[n-k]=b[n]*x[n] \;\Leftrightarrow Y(z)=\sum_{k=0}^{N} b_k z^{-k} X(z) \;\Leftrightarrow H(z)=\sum_{k=0}^{N} b_k z^{-k}$$

$$H(z)=\sum_{k=0}^{N} b_k z^{-k} = \prod_{k=1}^{N_s}(b_{0_k}+b_{1k}z^{-1}+b_{2k}z^{-2}),\quad N_s=\lfloor (N+1)/2 \rfloor$$

FIR Direct Form Structure

Linear Phase FIR Filters



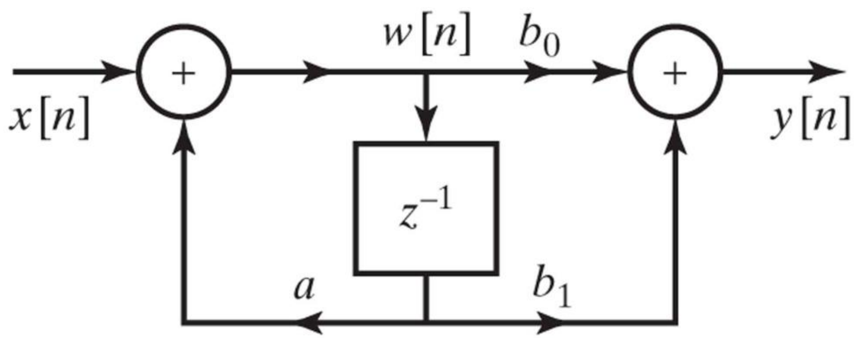
Savings due to symmetry
(number of multipliers halved)

FIR Transposed Form Structure

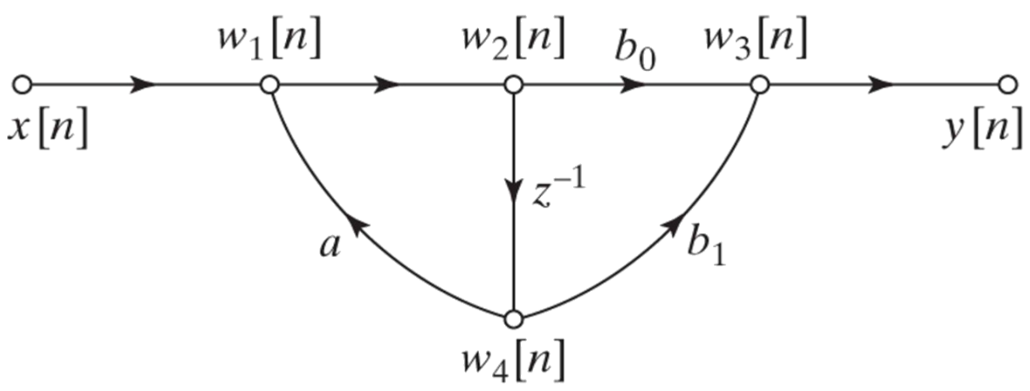
Flow Graph Reversal Theorem

- If we **reverse the directions of all branch transmittances** and **interchange the input and output** in the flow graph the system function remains the same
- The resulting structure is called a **transposed form**

Block Diagram



Flow Graph



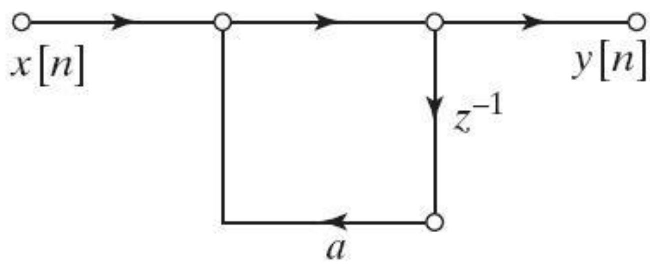
FIR Transposed Form Structure

Flow Graph Reversal Theorem

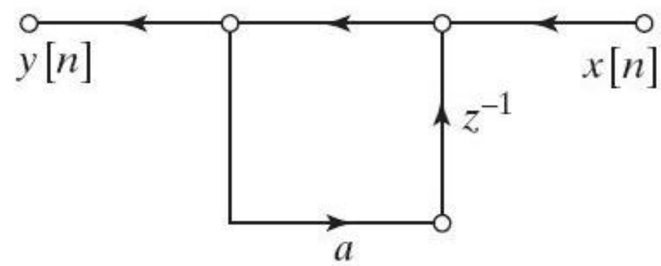
- If we **reverse the directions of all branch transmittances** and **interchange the input and output** in the flow graph the system function remains the same
- The resulting structure is called a **transposed form**

Example

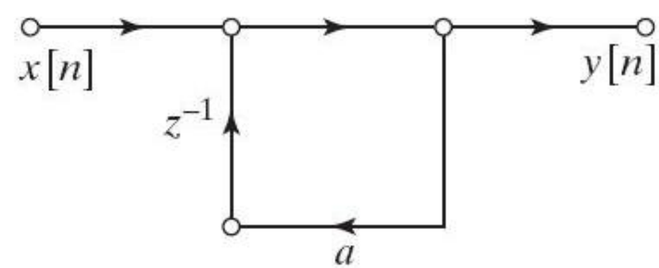
$$H(z) = \frac{1}{1 - az^{-1}}$$



(a)



(b)



(c)

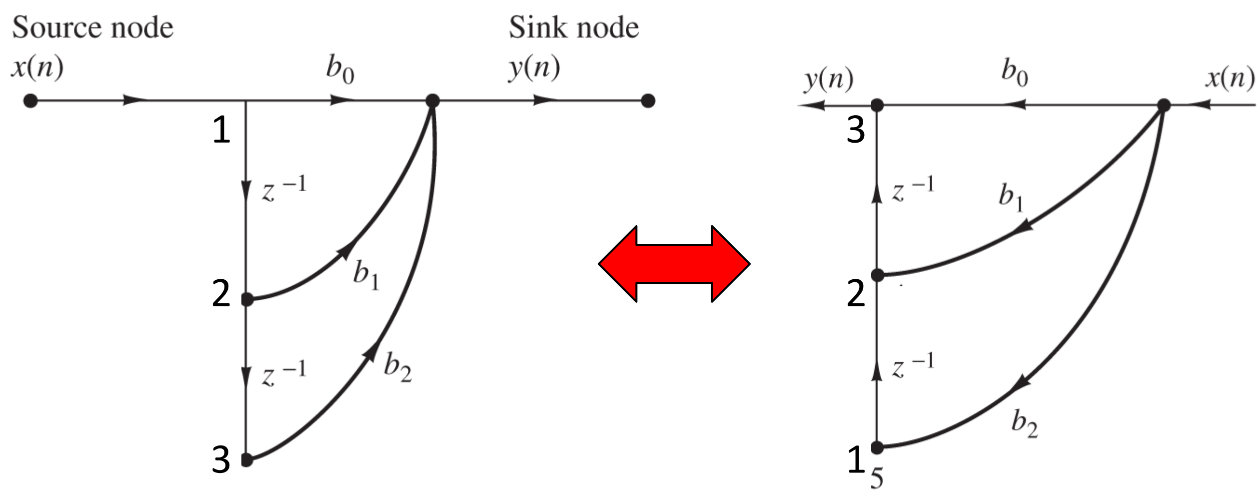
(a) Flow graph of simple 1st-order system.

(b) Transposed form of (a).

(c) Structure of (b) Redrawn with input on left.

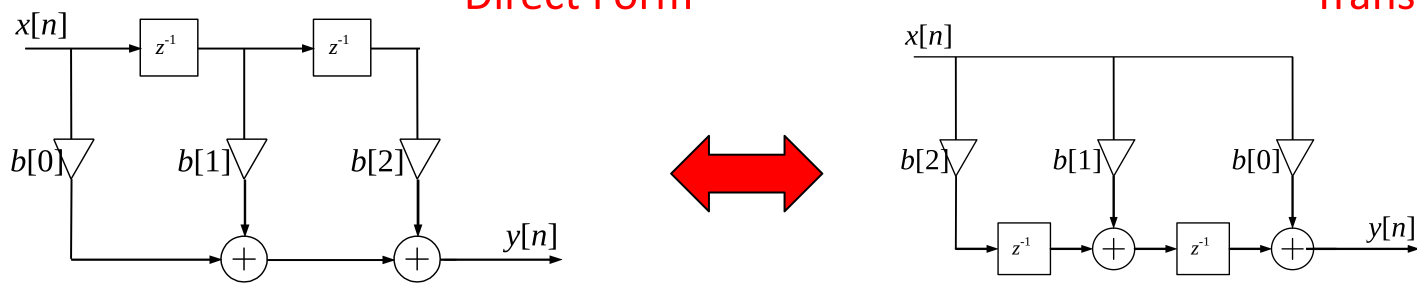
FIR Transposed Form Structure

Flow Graph Reversal Theorem



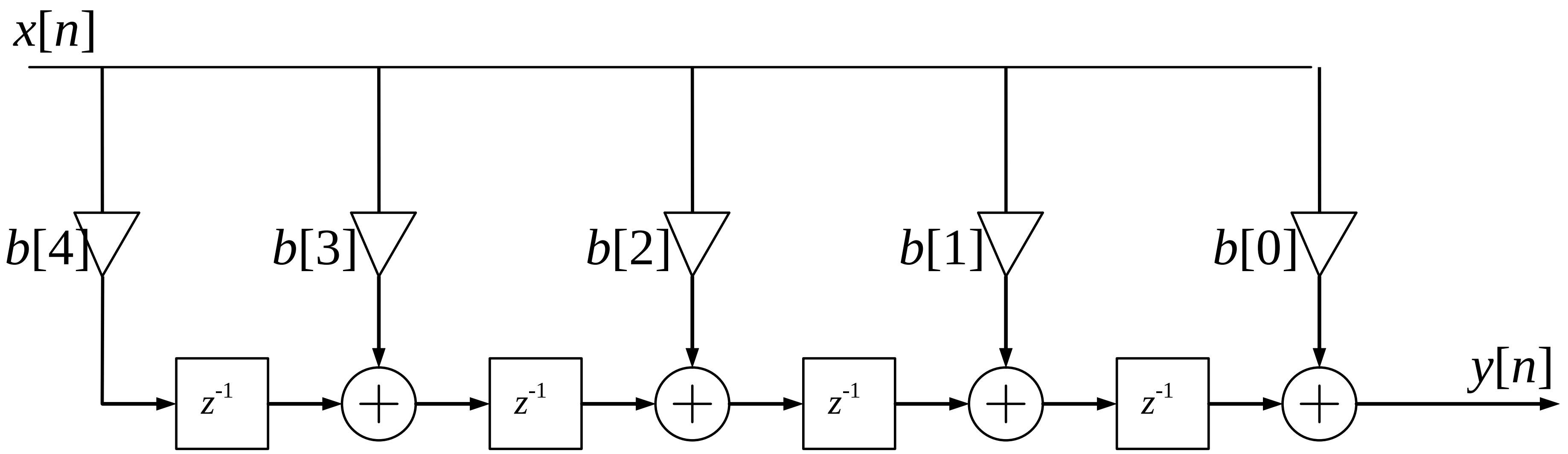
Direct Form

Transposed Form



FIR Transposed Form Structure

Flow Graph Reversal Theorem



Complexity

- L=N+1 multiplications
- N additions
- N memory locations

- Multipliers can be implemented together in one multiplier block and both multiplication & summation can be performed in parallel