

# Digital Filter Structures

- The convolution sum description of an LTI discrete-time system can, in principle, be used to implement the system
- For an IIR finite-dimensional system this approach is not practical as here the impulse response is of infinite length
- However, a direct implementation of the IIR finite-dimensional system is practical

# Digital Filter Structures

- Here the input-output relation involves a finite sum of products:

$$y[n] = - \sum_{k=1}^N d_k y[n-k] + \sum_{k=0}^M p_k x[n-k]$$

- On the other hand, an FIR system can be implemented using the convolution sum which is a finite sum of products:

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

# Digital Filter Structures

- The actual implementation of an LTI digital filter can be either in software or hardware form, depending on applications
- In either case, the signal variables and the filter coefficients cannot be represented with infinite precision

# Digital Filter Structures

- However, a direct implementation of a digital filter based on either the difference equation or the finite convolution sum may not provide satisfactory performance due to the finite precision arithmetic
- It is thus of practical interest to develop alternate realizations and choose the structure that provides satisfactory performance under finite precision arithmetic

# Digital Filter Structures

- A structural representation using interconnected basic building blocks is the first step in the hardware or software implementation of an LTI digital filter
- The structural representation provides the key relations between some pertinent internal variables with the input and output that in turn provides the key to the implementation

# Block Diagram Representation

- In the time domain, the input-output relations of an LTI digital filter is given by the convolution sum

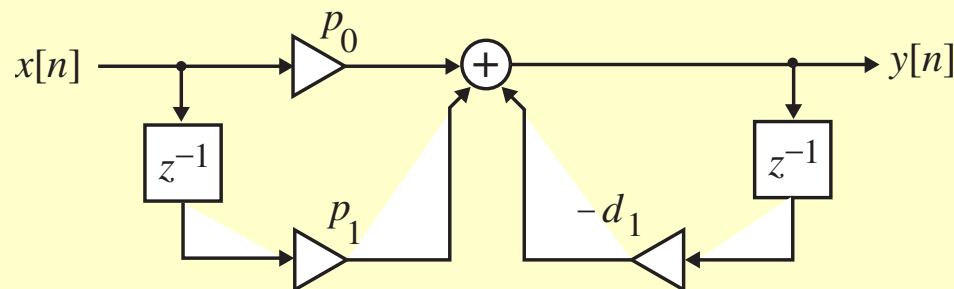
$$y[n] = \sum_{k=-\infty}^{\infty} h[k] x[n-k]$$

or, by the linear constant coefficient difference equation

$$y[n] = - \sum_{k=1}^N d_k y[n-k] + \sum_{k=0}^M p_k x[n-k]$$

# Block Diagram Representation

- For the implementation of an LTI digital filter, the input-output relationship must be described by a valid computational algorithm
- To illustrate what we mean by a computational algorithm, consider the causal first-order LTI digital filter shown below



# Block Diagram Representation

- The filter is described by the difference equation

$$y[n] = -d_1 y[n-1] + p_0 x[n] + p_1 x[n-1]$$

- Using the above equation we can compute  $y[n]$  for  $n \geq 0$  knowing the initial condition  $y[-1]$  and the input  $x[n]$  for  $n \geq -1$ :



# Block Diagram Representation

$$y[0] = -d_1 y[-1] + p_0 x[0] + p_1 x[-1]$$

$$y[1] = -d_1 y[0] + p_0 x[1] + p_1 x[0]$$

$$y[2] = -d_1 y[1] + p_0 x[2] + p_1 x[1]$$

$$\vdots$$

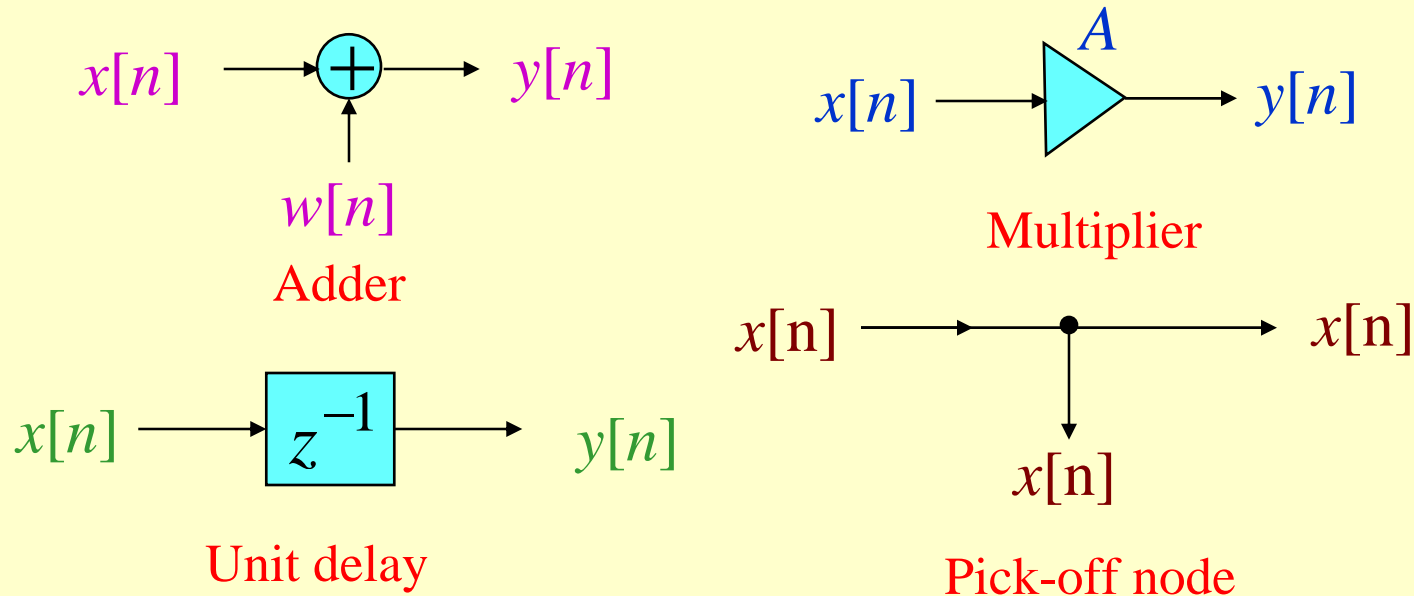
- We can continue this calculation for any value of the time index  $n$  we desire

# Block Diagram Representation

- Each step of the calculation requires a knowledge of the previously calculated value of the output sample (delayed value of the output), the present value of the input sample, and the previous value of the input sample (delayed value of the input)
- As a result, the first-order difference equation can be interpreted as a valid computational algorithm

# Basic Building Blocks

- The computational algorithm of an LTI digital filter can be conveniently represented in block diagram form using the basic building blocks shown below



# Basic Building Blocks

## Advantages of block diagram representation

- (1) Easy to write down the computational algorithm by inspection
- (2) Easy to analyze the block diagram to determine the explicit relation between the output and input

# Basic Building Blocks

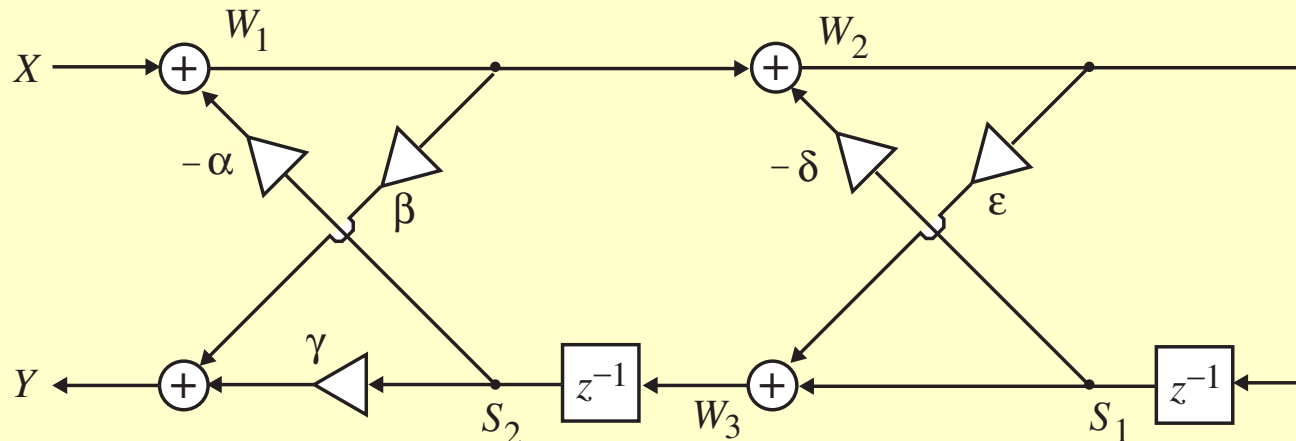
- (3) Easy to manipulate a block diagram to derive other “equivalent” block diagrams yielding different computational algorithms
- (4) Easy to determine the hardware requirements
- (5) Easier to develop block diagram representations from the transfer function directly

# Analysis of Block Diagrams

- Carried out by writing down the expressions for the output signals of each adder as a sum of its input signals, and developing a set of equations relating the filter input and output signals in terms of all internal signals
- Eliminating the unwanted internal variables then results in the expression for the output signal as a function of the input signal and the filter parameters that are the multiplier coefficients

# Analysis of Block Diagrams

- Example - Analyze the cascaded lattice structure shown below where the  $z$ -dependence of signal variables are not shown for brevity



# Analysis of Block Diagrams

- The output signals of the four adders are given by

$$W_1 = X - \alpha S_2$$

$$W_2 = W_1 - \delta S_1$$

$$W_3 = S_1 + \varepsilon W_2$$

$$Y = \beta W_1 + \gamma S_2$$

- From the figure we observe

$$S_2 = z^{-1}W_3$$

$$S_1 = z^{-1}W_2$$



# Analysis of Block Diagrams

- Substituting the last two relations in the first four equations we get

$$W_1 = X - \alpha z^{-1}W_3$$

$$W_2 = W_1 - \delta z^{-1}W_2$$

$$W_3 = z^{-1}W_2 + \varepsilon W_2$$

$$Y = \beta W_1 + \gamma z^{-1}W_3$$

- From the second equation we get  $W_2 = W_1 / (1 + \delta z^{-1})$  and from the third equation we get  $W_3 = (\varepsilon + z^{-1})W_2$

# Analysis of Block Diagrams

- Combining the last two equations we get

$$W_3 = \frac{\varepsilon + z^{-1}}{1 + \delta z^{-1}} W_1$$

- Substituting the above equation in

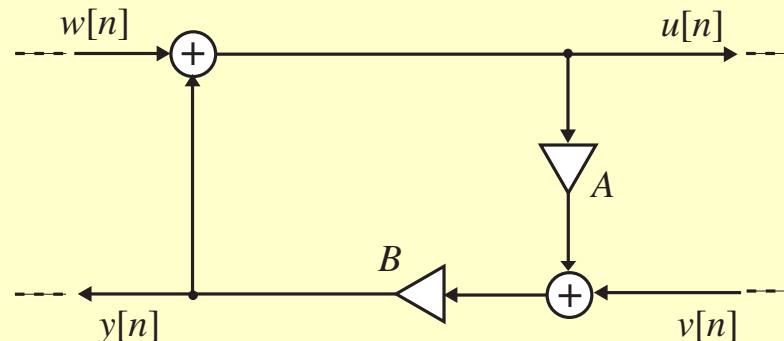
$$W_1 = X - \alpha z^{-1} W_3, \quad Y = \beta W_1 + \gamma z^{-1} W_3$$

we finally arrive at

$$H(z) = \frac{Y}{X} = \frac{\beta + (\beta\delta + \gamma\varepsilon)z^{-1} + \gamma z^{-2}}{1 + (\delta + \alpha\varepsilon)z^{-1} + \alpha z^{-2}}$$

# The Delay-Free Loop Problem

- For physical realizability of the digital filter structure, it is necessary that the block diagram representation contains no delay-free loops
- To illustrate the delay-free loop problem consider the structure below



# The Delay-Free Loop Problem


- Analysis of this structure yields

$$u[n] = w[n] + y[n]$$

$$y[n] = B(v[n] + Au[n])$$

which when combined results in

$$y[n] = B(v[n] + A(w[n] + y[n]))$$

-  The determination of the current value of  $y[n]$  requires the knowledge of the same value

# The Delay-Free Loop Problem

- However, this is physically impossible to achieve due to the finite time required to carry out all arithmetic operations on a digital machine
- Method exists to detect the presence of delay-free loops in an arbitrary structure, along with methods to locate and remove these loops without the overall input-output relation

# The Delay-Free Loop Problem

- Removal achieved by replacing the portion of the overall structure containing the delay-free loops by an equivalent realization with no delay-free loops
- To illustrate the process, consider the two equations characterizing the structure of Slide No. 21:

$$u[n] = w[n] + y[n]$$

$$y[n] = B(v[n] + Au[n])$$

# The Delay-Free Loop Problem

- Substituting the second equation into the first we arrive at

$$u[n] = w[n] + Bv[n] + ABu[n]$$

- Solving the above we get

$$u[n] = \frac{1}{1 - AB}(w[n] + Bv[n])$$

- From the first equation we have

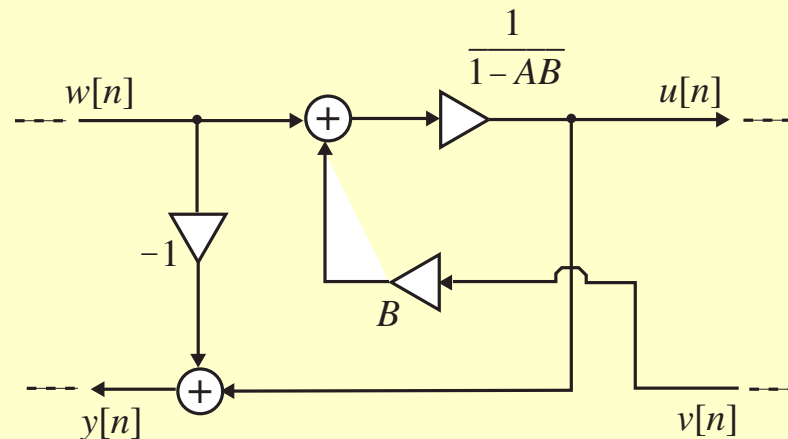
$$y[n] = u[n] - w[n]$$

# The Delay-Free Loop Problem

$$y[n] = u[n] - w[n]$$

$$u[n] = \frac{1}{1 - AB}(w[n] + Bv[n])$$

- A delay-free loop realization based on the above two equations is shown below





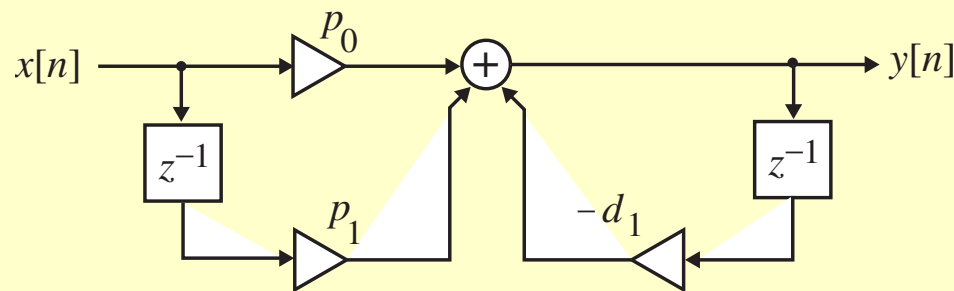
# Canonical and Noncanonical Structures

- A digital filter structure is said to be **canonical** if the number of delays in the block diagram representation is equal to the order of the transfer function
- Otherwise, it is a **noncanonical** structure

# Canonic and Noncanonic Structures

- The structure shown below is noncanonic as it employs two delays to realize a first-order difference equation

$$y[n] = -d_1 y[n-1] + p_0 x[n] + p_1 x[n-1]$$



# Equivalent Structures

- Two digital filter structures are defined to be **equivalent** if they have the same transfer function
- We describe next a number of methods for the generation of equivalent structures
- However, a fairly simple way to generate an equivalent structure from a given realization is via the **transpose operation**

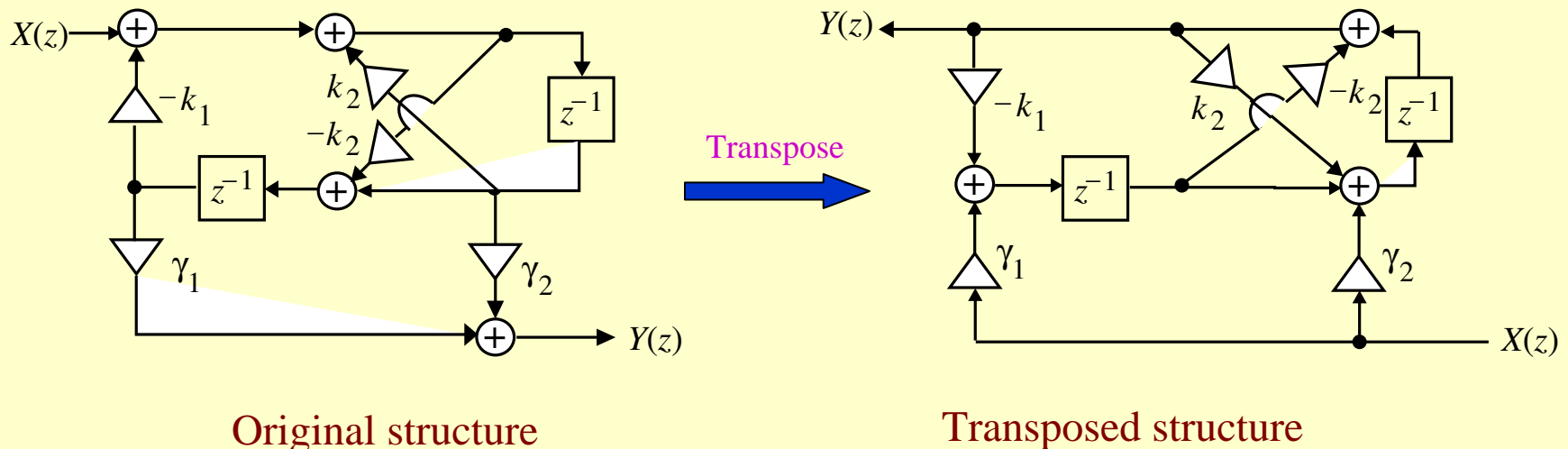
# Equivalent Structures

## Transpose Operation

- (1) Reverse all paths
- (2) Replace pick-off nodes by adders, and vice versa
- (3) Interchange the input and output nodes

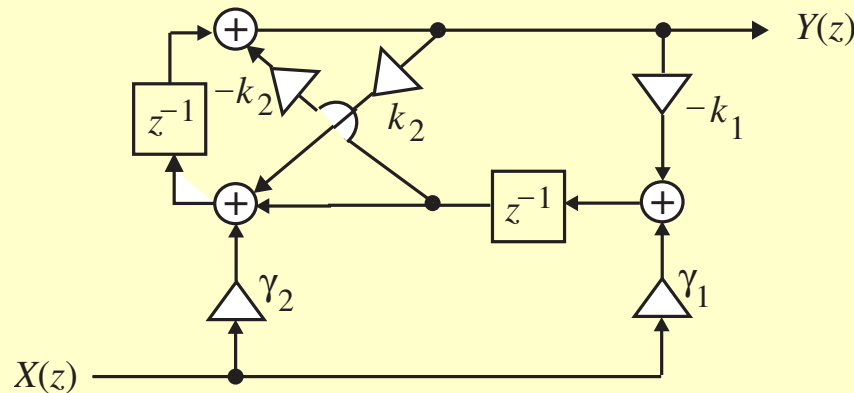
# Equivalent Structures

- Example** – The transpose operation is illustrated below



# Equivalent Structures

- Redrawn transposed structure is shown below



- All other methods for developing equivalent structures are based on a specific algorithm for each structure

# Equivalent Structures

- There are literally an infinite number of equivalent structures realizing the same transfer function
- It is thus impossible to develop all equivalent realizations
- In this course we restrict our attention to a discussion of some commonly used structures

# Equivalent Structures

- Under infinite precision arithmetic any given realization of a digital filter behaves identically to any other equivalent structure
- However, in practice, due to the finite wordlength limitations, a specific realization behaves totally differently from its other equivalent realizations



# Equivalent Structures

- Hence, it is important to choose a structure that has the least quantization effects when implemented using finite precision arithmetic
- One way to arrive at such a structure is to determine a large number of equivalent structures, analyze the finite wordlength effects in each case, and select the one showing the least effects

# Equivalent Structures

- In certain cases, it is possible to develop a structure that by construction has the least quantization effects
- We defer the review of these structures after a discussion of the analysis of quantization effects
- Here, we review some simple realizations that in many applications are quite adequate