Chapter 9
REALIZATION
9.1 Introduction
9.2.1 Direct Realization
9.2.2 Direct Canonic Realization

Victoria, BC, Canada
Email: aantoniou@ieee.org

July 10, 2018

- Previous presentations considered the basics of signal analysis and the characterization and analysis of discrete-time systems in general.

Introduction

- Previous presentations considered the basics of signal analysis and the characterization and analysis of discrete-time systems in general.
- The remaining presentations deal with the design of discrete-time systems that can be used to reshape the spectral characteristics of discrete-time signals, namely, digital filters.

Introduction

- Previous presentations considered the basics of signal analysis and the characterization and analysis of discrete-time systems in general.
- The remaining presentations deal with the design of discrete-time systems that can be used to reshape the spectral characteristics of discrete-time signals, namely, digital filters.
- The design of digital filters encompasses all the activities that need to be undertaken from the point where a need for a specific type of digital filter is identified to the point where a prototype is constructed, tested, and approved.

- In general the design of digital filters involves the following steps:

- In general the design of digital filters involves the following steps:
  - Approximation

- In general the design of digital filters involves the following steps:
  - Approximation
  - Realization

- In general the design of digital filters involves the following steps:
  - Approximation
  - Realization
  - Study of arithmetic errors

- In general the design of digital filters involves the following steps:
  - Approximation
  - Realization
  - Study of arithmetic errors
  - Implementation

Approximation Step

- The *approximation step* is the process of generating a transfer function that would satisfy the desired specifications which may concern the amplitude or phase response or even the time-domain response of the filter.

Approximation Step

- The *approximation step* is the process of generating a transfer function that would satisfy the desired specifications which may concern the amplitude or phase response or even the time-domain response of the filter.

- The available methods for the solution of the approximation problem can be classified as *direct* or *indirect*.

Approximation Step

- The *approximation step* is the process of generating a transfer function that would satisfy the desired specifications which may concern the amplitude or phase response or even the time-domain response of the filter.

- The available methods for the solution of the approximation problem can be classified as *direct* or *indirect*.

- In direct methods, the problem is solved directly in the z domain.

Approximation Step

- The *approximation step* is the process of generating a transfer function that would satisfy the desired specifications which may concern the amplitude or phase response or even the time-domain response of the filter.

- The available methods for the solution of the approximation problem can be classified as *direct* or *indirect*.

- In direct methods, the problem is solved directly in the z domain.

- In indirect methods, a continuous-time transfer function is first obtained and then converted into a corresponding discrete-time transfer function.

Approximation Step

- The *approximation step* is the process of generating a transfer function that would satisfy the desired specifications which may concern the amplitude or phase response or even the time-domain response of the filter.
- The available methods for the solution of the approximation problem can be classified as *direct* or *indirect*.
- In direct methods, the problem is solved directly in the z domain.
- In indirect methods, a continuous-time transfer function is first obtained and then converted into a corresponding discrete-time transfer function.
- Nonrecursive filters are always designed through direct methods whereas recursive filters can be designed either through direct or indirect methods.

- Approximation methods can also be classified as *closed-form* or *iterative*.

- Approximation methods can also be classified as *closed-form* or *iterative*.
- In closed-form methods, the problem is solved through a small number of design steps using a set of closed-form formulas.

- Approximation methods can also be classified as *closed-form* or *iterative*.
- In closed-form methods, the problem is solved through a small number of design steps using a set of closed-form formulas.
- In iterative methods, an initial solution is assumed and through the application of optimization methods a series of progressively improved solutions are obtained until some design criterion is satisfied.

- In general, the designer is interested in approximation methods that

Approximation Step *Cont'd*

- In general, the designer is interested in approximation methods that
  - are simple,

- In general, the designer is interested in approximation methods that
  - are simple,
  - are reliable,

- In general, the designer is interested in approximation methods that
  - are simple,
  - are reliable,
  - yield precise designs,

Approximation Step *Cont'd*

- In general, the designer is interested in approximation methods that
  - are simple,
  - are reliable,
  - yield precise designs,
  - require minimal computation effort, and so on.

- The *realization* or *synthesis* of a digital filter is the process of generating a digital-filter network or structure from the transfer function or some other characterization of the filter.

Realization Step

- The *realization* or *synthesis* of a digital filter is the process of generating a digital-filter network or structure from the transfer function or some other characterization of the filter.
- The network obtained is said to be the *realization* of the transfer function.

- The *realization* or *synthesis* of a digital filter is the process of generating a digital-filter network or structure from the transfer function or some other characterization of the filter.
- The network obtained is said to be the *realization* of the transfer function.
- As for approximation methods, realization methods can be classified as *direct* or *indirect*.

- The *realization* or *synthesis* of a digital filter is the process of generating a digital-filter network or structure from the transfer function or some other characterization of the filter.
- The network obtained is said to be the *realization* of the transfer function.
- As for approximation methods, realization methods can be classified as *direct* or *indirect*.
- In direct methods the filter structure is obtained directly from a given discrete-time transfer function whereas in indirect realizations it is obtained indirectly from an equivalent continuous-time transfer function.

- Many realization methods have been proposed in the past that lead to digital-filter structures of varying complexity and properties.

- Many realization methods have been proposed in the past that lead to digital-filter structures of varying complexity and properties.
- The designer is usually interested in realizations that

- Many realization methods have been proposed in the past that lead to digital-filter structures of varying complexity and properties.
- The designer is usually interested in realizations that
  - are easy to implement in very-large-scale integrated (VLSI) circuit form,

- Many realization methods have been proposed in the past that lead to digital-filter structures of varying complexity and properties.
- The designer is usually interested in realizations that
  - are easy to implement in very-large-scale integrated (VLSI) circuit form,
  - require the minimum number of unit delays, adders, and multipliers,

- Many realization methods have been proposed in the past that lead to digital-filter structures of varying complexity and properties.
- The designer is usually interested in realizations that
  - are easy to implement in very-large-scale integrated (VLSI) circuit form,
  - require the minimum number of unit delays, adders, and multipliers,
  - are not seriously affected by the use of finite-precision arithmetic in the implementation, and so on.

- Designs of all types from that of a refrigerator to an electrical drill entail *imperfections* of various sorts brought about by modeling inaccuracies, component tolerances, unusual or unexpected nonlinear effects, and so on.

- Designs of all types from that of a refrigerator to an electrical drill entail *imperfections* of various sorts brought about by modeling inaccuracies, component tolerances, unusual or unexpected nonlinear effects, and so on.
- A design will be approved to the extent that design imperfections do not violate the desired specifications.

Study of Arithmetic Errors

- Designs of all types from that of a refrigerator to an electrical drill entail *imperfections* of various sorts brought about by modeling inaccuracies, component tolerances, unusual or unexpected nonlinear effects, and so on.
- A design will be approved to the extent that design imperfections do not violate the desired specifications.
- In digital filters and digital systems in general, most imperfections are caused by *numerical imprecision* of some form and studying the ways in which numerical imprecision will manifest itself needs to be undertaken.

- During the approximation step, the coefficients of the transfer function are determined to a high degree of precision.

- During the approximation step, the coefficients of the transfer function are determined to a high degree of precision.
- In practice, however, digital hardware has finite precision that depends

Study of Arithmetic Errors *Cont'd*

- During the approximation step, the coefficients of the transfer function are determined to a high degree of precision.
- In practice, however, digital hardware has finite precision that depends
  - on the length of registers used to store numbers

- During the approximation step, the coefficients of the transfer function are determined to a high degree of precision.
- In practice, however, digital hardware has finite precision that depends
  - on the length of registers used to store numbers
  - the type of number system used (e.g., signed-magnitude, two's complement)

- During the approximation step, the coefficients of the transfer function are determined to a high degree of precision.
- In practice, however, digital hardware has finite precision that depends
  - on the length of registers used to store numbers
  - the type of number system used (e.g., signed-magnitude, two's complement)
  - the type of arithmetic used (e.g., fixed-point or floating-point), etc.

- In order to accommodate filter coefficients in registers, they must be quantized (e.g., rounded or truncated).

- In order to accommodate filter coefficients in registers, they must be quantized (e.g., rounded or truncated).
- When the transfer function coefficients are quantized, errors are introduced in the amplitude and phase responses of the filter, which are commonly referred to as *quantization errors*.

- Quantization errors can cause the digital filter to violate the required specifications and in extreme cases even to become unstable.

- Quantization errors can cause the digital filter to violate the required specifications and in extreme cases even to become unstable.
- Like the filter coefficients, the signals to be processed as well as the internal signals of a digital filter (e.g., the products generated by multipliers) must also be quantized.

- Quantization errors can cause the digital filter to violate the required specifications and in extreme cases even to become unstable.
- Like the filter coefficients, the signals to be processed as well as the internal signals of a digital filter (e.g., the products generated by multipliers) must also be quantized.
- Errors introduced by the quantization of signals can be treated as *noise sources* and, as a consequence, they can have a dramatic effect on the processed signal.

- Quantization errors can cause the digital filter to violate the required specifications and in extreme cases even to become unstable.
- Like the filter coefficients, the signals to be processed as well as the internal signals of a digital filter (e.g., the products generated by multipliers) must also be quantized.
- Errors introduced by the quantization of signals can be treated as *noise sources* and, as a consequence, they can have a dramatic effect on the processed signal.
- In short, the *effects of arithmetic errors* on the performance of the filter must be investigated and ways must be found to mitigate any problems associated with numerical imprecision.

Implementation

- The *implementation* of a digital filter can assume two forms, namely, *software* or *hardware*.

Implementation

- The *implementation* of a digital filter can assume two forms, namely, *software* or *hardware*.
- A software implementation involves the simulation of the filter network or difference equation on a general-purpose digital computer, workstation, or DSP chip.

Implementation

- The *implementation* of a digital filter can assume two forms, namely, *software* or *hardware*.
- A software implementation involves the simulation of the filter network or difference equation on a general-purpose digital computer, workstation, or DSP chip.
- A hardware implementation involves the conversion of the filter network into a dedicated piece of hardware.

- The choice of implementation is usually critically dependent on the application at hand.

- The choice of implementation is usually critically dependent on the application at hand.
- In *nonreal-time* applications where a record of the data to be processed is available in a database, a software implementation would be entirely satisfactory.

- The choice of implementation is usually critically dependent on the application at hand.
- In *nonreal-time* applications where a record of the data to be processed is available in a database, a software implementation would be entirely satisfactory.
- In *real-time* applications, however, where data must be processed at a very high rate, e.g., in communication systems, a hardware implementation is mandatory.

- The choice of implementation is usually critically dependent on the application at hand.
- In *nonreal-time* applications where a record of the data to be processed is available in a database, a software implementation would be entirely satisfactory.
- In *real-time* applications, however, where data must be processed at a very high rate, e.g., in communication systems, a hardware implementation is mandatory.
- Often the best engineering solution might be partially in terms of software and partially in terms of hardware since software and hardware are highly exchangeable nowadays.

- The natural order of the four basic design steps is as stated in the preceding discussion, namely, approximation, realization, study of imperfections, and implementation.

- The natural order of the four basic design steps is as stated in the preceding discussion, namely, approximation, realization, study of imperfections, and implementation.
- However, realization is the easiest to learn and for this reason it will be treated first.

- The natural order of the four basic design steps is as stated in the preceding discussion, namely, approximation, realization, study of imperfections, and implementation.
- However, realization is the easiest to learn and for this reason it will be treated first.
- The approximation step, study of errors, and other design considerations will be discussed in later presentations.

Implementation *Cont'd*

- The natural order of the four basic design steps is as stated in the preceding discussion, namely, approximation, realization, study of imperfections, and implementation.
- However, realization is the easiest to learn and for this reason it will be treated first.
- The approximation step, study of errors, and other design considerations will be discussed in later presentations.
- See end of Chap. 9 for a discussion on the implementation of digital filters.

Realization

- A great collection of direct and indirect realization methods have evolved over the past 40 years.

Realization

- A great collection of direct and indirect realization methods have evolved over the past 40 years.
- In direct methods, the transfer function is put in some form that enables the identification of an interconnection of elemental digital-filter subnetworks.

- A great collection of direct and indirect realization methods have evolved over the past 40 years.
- In direct methods, the transfer function is put in some form that enables the identification of an interconnection of elemental digital-filter subnetworks.
- Some of these methods are as follows:
  - Direct
  - Direct canonic
  - State-space
  - Lattice
  - Parallel
  - Cascade

- An arbitrary causal *nonrecursive* filter can be represented by the equation

$$Y(z) = N(z)X(z) = \left[\sum_{i=0}^{N} a_i z^{-i}\right] X(z)$$

Direct Realization

- An arbitrary causal *nonrecursive* filter can be represented by the equation

$$Y(z) = N(z)X(z) = \left[\sum_{i=0}^{N} a_i z^{-i}\right] X(z)$$

- We can write

$$Y(z) = \left[a_0 + z^{-1}\sum_{i=1}^{N} a_i z^{-i+1}\right] X(z)$$

or $$Y(z) = [a_0 + z^{-1}N_1(z)]X(z)$$

where $$N_1(z) = \sum_{i=1}^{N} a_i z^{-i+1}$$

...

$$Y(z) = N(z)X(z) = [a_0 + z^{-1}N_1(z)]X(z)$$

- Therefore, transfer function $N(z)$ can be realized by using
  - a multiplier with a constant a_0
  - a unit delay, and
  - a network with a transfer function $N_1(z)$

. . .

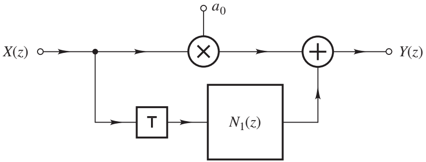
$$Y(z) = N(z)X(z) = [a_0 + z^{-1}N_1(z)]X(z)$$

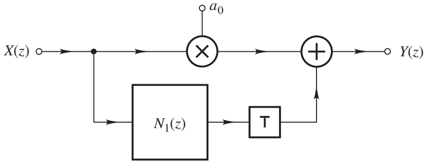
- Therefore, transfer function $N(z)$ can be realized by using
  - a multiplier with a constant a_0
  - a unit delay, and
  - a network with a transfer function $N_1(z)$
- The unit delay can be connected in cascade with the network for $N_1(z)$ to form $z^{-1}N_1(z)$ and the multiplier can be connected in parallel with the network for $z^{-1}N_1(z)$ to form $a_0 + z^{-1}N_1(z)$.

...

$$Y(z) = [a_0 + z^{-1}N_1(z)]X(z)$$

- There are two possible realizations as follows:





...

$$N_1(z) = \sum_{i=1}^{N} a_i z^{-i+1}$$

- Proceeding as before, we can now write

$$N_1(z) = a_1 + z^{-1} \sum_{i=2}^{N} a_i z^{-i+2} = a_1 + z^{-1} N_2(z)$$

where

$$N_2(z) = \sum_{i=2}^{N} a_i z^{-i+2}$$

...

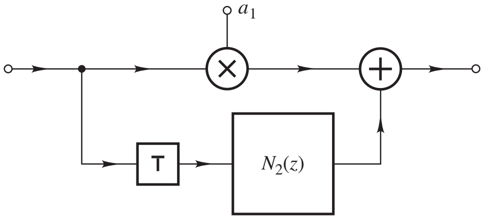
$$N_1(z) = a_1 + z^{-1} N_2(z)$$

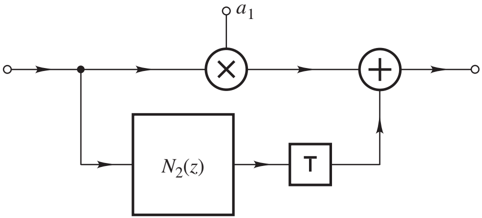
- Therefore, transfer function $N_1(z)$ can be realized by using
  - a multiplier with a constant a_1
  - a unit delay, and
  - a network with a transfer function $N_2(z)$

...

$$N_1(z) = a_1 + z^{-1}N_2(z)$$

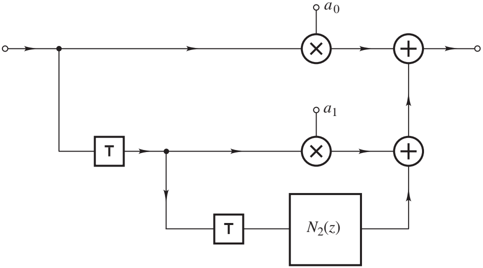
- Therefore, transfer function $N_1(z)$ can be realized by using
  - a multiplier with a constant a_1
  - a unit delay, and
  - a network with a transfer function $N_2(z)$
- The unit delay can be connected in cascade with the network for $N_2(z)$ to form $z^{-1}N_2(z)$ and the multiplier can be connected in parallel with the network for $z^{-1}N_2(z)$ to form $a_0 + z^{-1}N_2(z)$.

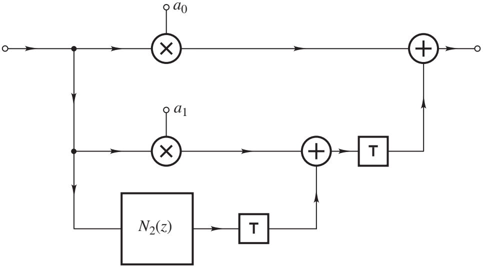

a_1
T
$N_2(z)$


a_1
$N_2(z)$
T

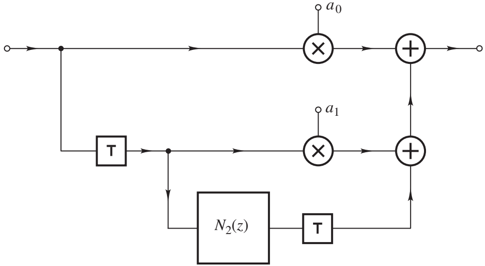
- Since there are two possible realizations for $N_1(z)$ it follows that there are four possible realizations for $N(z)$ as illustrated in the next two slides.

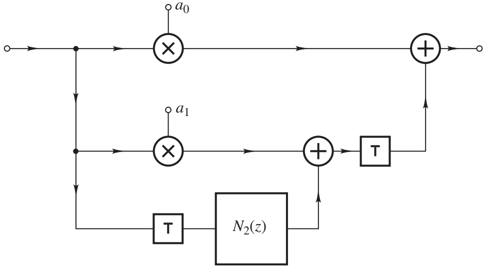
Direct Realization *Cont'd*





Direct Realization *Cont'd*



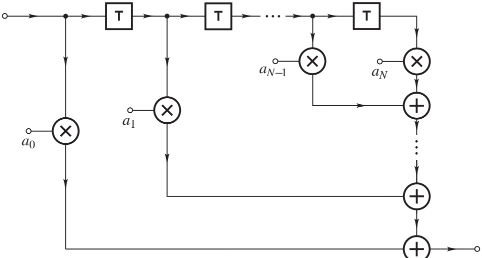


- Repeating the same procedure N times will reduce the realization of $N_N(z)$ to a single multiplier with a constant a_N.

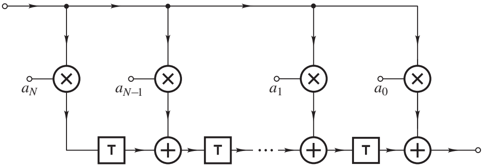
- Repeating the same procedure N times will reduce the realization of $N_N(z)$ to a single multiplier with a constant a_N.
- Since there are N iterations in the procedure, and each iteration multiplies the number of distinct realizations by 2, a total of 2^N realizations are possible for $N(z)$.

- Repeating the same procedure N times will reduce the realization of $N_N(z)$ to a single multiplier with a constant a_N.
- Since there are N iterations in the procedure, and each iteration multiplies the number of distinct realizations by 2, a total of 2^N realizations are possible for $N(z)$.
- Three of the possible realizations are shown in the next two slides.

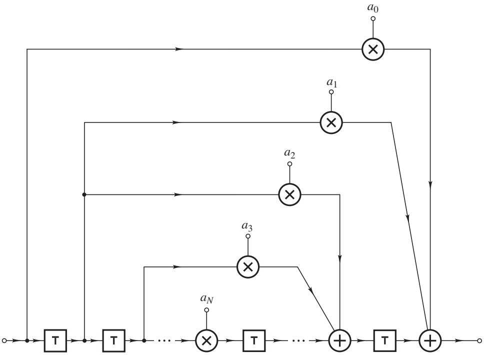
Direct Realization *Cont'd*



(*a*)

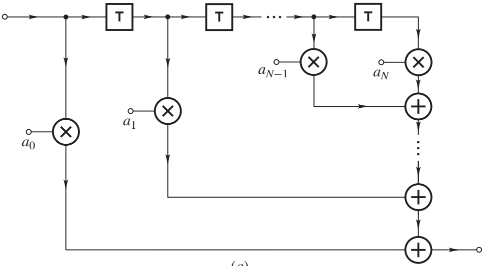


(*b*)

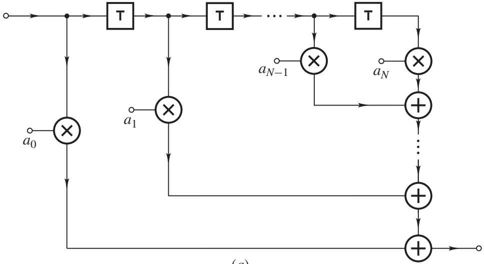


(c)

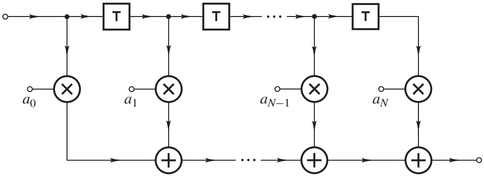
- Another interesting realization can be obtained from the first of the previous structures by noting that the outputs of multipliers a_0, a_1, . . . , a_N can be added in the reverse order without changing the output.



(a)



(*a*)



(*d*)

Direct Realization *Cont'd*

- The derived new structure is seen to be highly regular and it is, therefore, attractive for VLSI implementation (see Sec. 8.3).

- A causal *recursive* digital filter characterized by an Nth-order transfer function can be represented by the equation

$$\frac{Y(z)}{X(z)} = H(z) = \frac{N(z)}{D(z)} = \frac{N(z)}{1 + D'(z)}$$

where

$$N(z) = \sum_{i=0}^{N} a_i z^{-i} \quad \text{and} \quad D'(z) = \sum_{i=1}^{N} b_i z^{-i}$$

- A causal *recursive* digital filter characterized by an Nth-order transfer function can be represented by the equation

$$\frac{Y(z)}{X(z)} = H(z) = \frac{N(z)}{D(z)} = \frac{N(z)}{1 + D'(z)}$$

where

$$N(z) = \sum_{i=0}^{N} a_i z^{-i} \quad \text{and} \quad D'(z) = \sum_{i=1}^{N} b_i z^{-i}$$

- We can write

$$Y(z) = N(z)X(z) - D'(z)Y(z)$$

or $$Y(z) = U_1(z) + U_2(z)$$

where $$U_1(z) = N(z)X(z) \quad \text{and} \quad U_2(z) = -D'(z)Y(z)$$

Direct Realization *Cont'd*

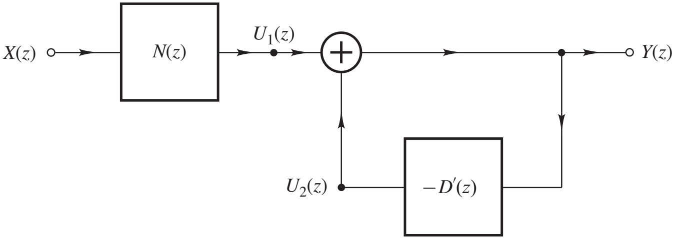
...

$$Y(z) = N(z)X(z) - D'(z)Y(z) \qquad \text{(A)}$$

or $$Y(z) = U_1(z) + U_2(z)$$

where $$U_1(z) = N(z)X(z) \quad \text{and} \quad U_2(z) = -D'(z)Y(z)$$

- Therefore, the recursive filter can be obtained by realizing $N(z)$ and $-D'(z)$ as nonrecursive filters and then connecting the resulting two nonrecursive filters as shown in the block diagram:



- Each of the two nonrecursive filters can be realized using the procedure outlined before.

- Each of the two nonrecursive filters can be realized using the procedure outlined before.
- Since

$$N(z) = \sum_{i=0}^{N} a_i z^{-i} \quad \text{and} \quad -D'(z) = -\sum_{i=1}^{N} b_i z^{-i}$$

the only difference between the two transfer functions is that

- Each of the two nonrecursive filters can be realized using the procedure outlined before.
- Since

$$N(z) = \sum_{i=0}^{N} a_i z^{-i} \quad \text{and} \quad -D'(z) = -\sum_{i=1}^{N} b_i z^{-i}$$

the only difference between the two transfer functions is that

  – the coefficient for z^0 is zero in $-D'(z)$ and

- Each of the two nonrecursive filters can be realized using the procedure outlined before.
- Since

$$N(z) = \sum_{i=0}^{N} a_i z^{-i} \quad \text{and} \quad -D'(z) = -\sum_{i=1}^{N} b_i z^{-i}$$

the only difference between the two transfer functions is that

  - the coefficient for z^0 is zero in $-D'(z)$ and
  - its coefficients are the negatives of the coefficients of the denominator of the transfer function.

Example

Realize the transfer function

$$H(z) = \frac{a_0 + a_1 z^{-1} + a_2 z^{-2}}{1 + b_1 z^{-1} + b_2 z^{-2}}$$

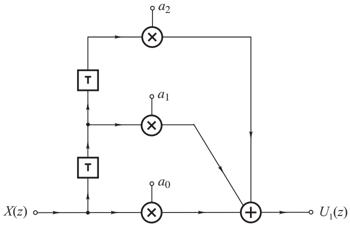
using the direct method.

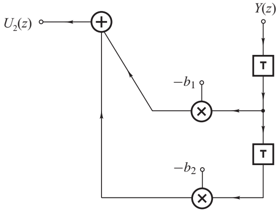
Solution The recursive structure can be obtained by realizing the nonrecursive transfer functions

$$N(z) = a_0 + a_1 z^{-1} + a_2 z^{-2} \quad \text{and} \quad -D'(z) = -b_1 z^{-1} - b_2 z^{-2}$$

Example *Cont'd*

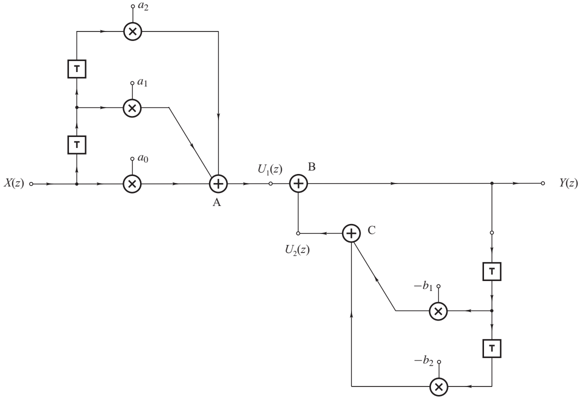
A pair of possible realizations for $N(z)$ and $-D'(z)$ are as shown:



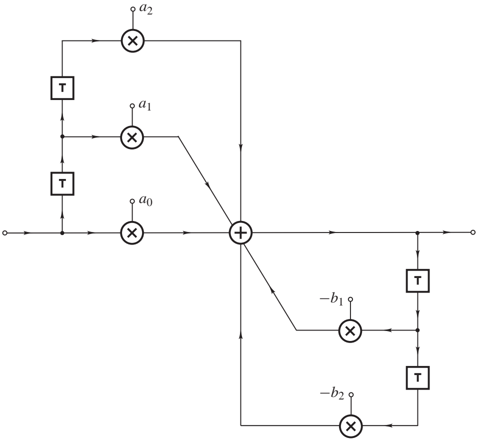


Example *Cont'd*

Connecting the realizations for $N(z)$ and $-D'(z)$ according to the block diagram, we get:



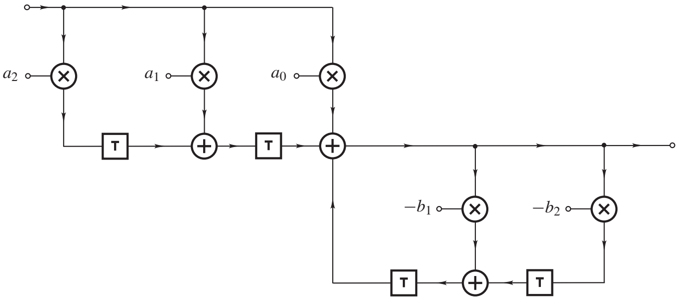
Now if we combine adders A, B, and C into a 5-input adder, we get the simplified structure shown:



(*a*)

Example *Cont'd*

Another possible direct realization that can be obtained using the same method is as follows:



(*b*)

Direct Canonic Realization

- The minimum number of unit delays required to realize an Nth-order transfer function is N.

- The minimum number of unit delays required to realize an Nth-order transfer function is N.
- If the number of unit delays in an Nth-order digital-filter structure is N, then the structure is said to be *canonic*.

Direct Canonic Realization

- The minimum number of unit delays required to realize an Nth-order transfer function is N.
- If the number of unit delays in an Nth-order digital-filter structure is N, then the structure is said to be *canonic*.
- The structures that can be obtained with the direct realization require $2N$ unit delays.

Direct Canonic Realization

- The minimum number of unit delays required to realize an Nth-order transfer function is N.
- If the number of unit delays in an Nth-order digital-filter structure is N, then the structure is said to be *canonic*.
- The structures that can be obtained with the direct realization require $2N$ unit delays.
- However, one of the many possibilities can be rendered canonic through a simple technique, as will be shown.

- The equation

$$\frac{Y(z)}{X(z)} = H(z) = \frac{N(z)}{D(z)} = \frac{N(z)}{1 + D'(z)}$$

can be expressed

$$Y(z) = \frac{N(z)X(z)}{1 + D'(z)}$$

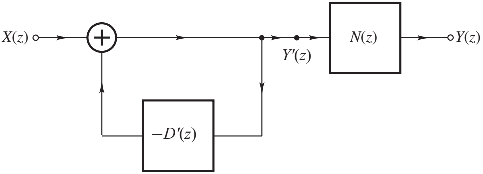
or

$$Y(z) = N(z)Y'(z) \quad \text{where} \quad Y'(z) = \frac{X(z)}{1 + D'(z)}$$

...

$$Y(z) = N(z)Y'(z) \quad \text{where} \quad Y'(z) = \frac{X(z)}{1 + D'(z)}$$

- These equations represent the system shown in the figure.



(a)

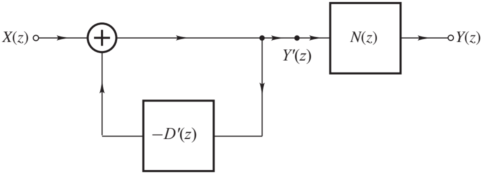
...

$$Y(z) = N(z)Y'(z) \quad \text{where} \quad Y'(z) = \frac{X(z)}{1 + D'(z)}$$

- These equations represent the system shown in the figure.
- Therefore, a digital-filter structure can be obtained by realizing the transfer functions

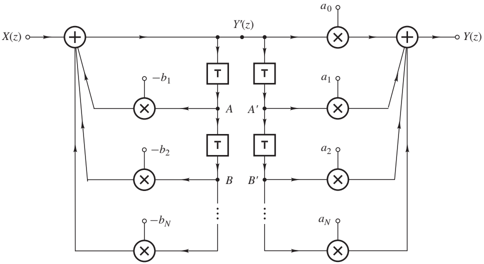
$$N(z) \quad \text{and} \quad \frac{1}{1 + D'(z)}$$

and then connecting the two realizations as shown:



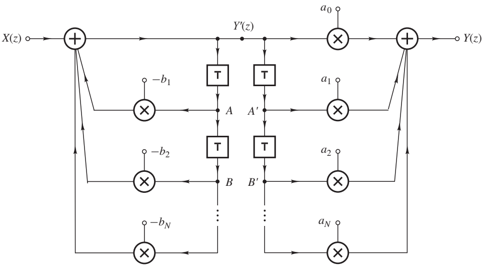
(a)

- Using the direct realization method, the configuration shown can be obtained.



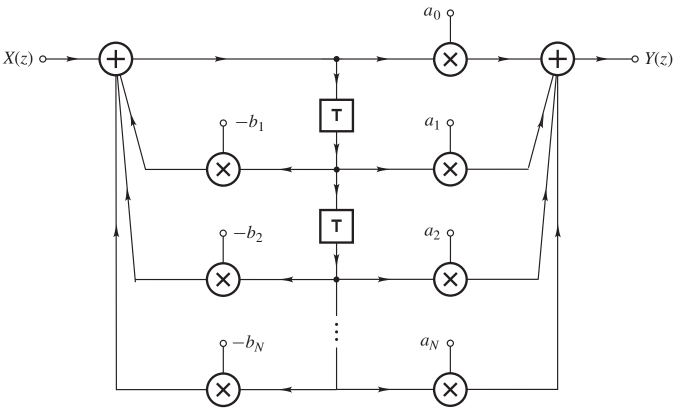
(b)

- Using the direct realization method, the configuration shown can be obtained.
- We note that the signals at nodes A', B' , . . . are exactly the same as the signals at nodes A, B, . . . and, therefore, nodes A, B, . . . can be connected to nodes A', B' , . . . and one set of unit delays can be eliminated.



(*b*)

Direct Canonic Realization *Cont'd*



This slide concludes the presentation.
Thank you for your attention.