Structures For The Realization of Discrete-Time Systems

Let us consider the implementation of linear constant-coefficient difference equations:

$$y(n) = -\sum_{k=1}^{N} a_k y(n-k) + \sum_{k=0}^{M} b_k x(n-k)$$

These can be characterized by the rational system function:

$$H(z) = \frac{\sum_{k=0}^{M} b_k z^{-k}}{1 + \sum_{k=1}^{N} a_k z^{-k}}$$

The difference equation can be viewed as a computational procedure (an algorithm) for determining the output sequence.

We can draw a block diagram consisting of delay elements, multipliers, and adders — this is called a realization.

We can have many realizations that implement the same difference equation. Important figures of merit for a realization are:

- **Computational complexity** — the number and mixture of arithmetic operations, and how well these are matched to the hardware. Performance measures are:
  - — Number of instructions per second (MIPS)
  - — Number of floating-point operations (MFLOPS)
  - — Number of operations per second (MOPS)
  - — Number of times real-time (xRT) (the number of seconds of CPU time required to process one second of the input signal)
- **Memory requirements** — the number of bytes or words of memory required to store the system parameters, inputs, outputs, and intermediate computations, and the SPEED of the memory.
- **Finite word-length effects** — how much quantization noise is generated in processing the data? Overall system SNR? The number of bits and the sizes of the accumulators and multipliers required at each stage of the filter to maintain sufficient precision.
- Other issues such as parallel processing, pipelines, number of long-word instructions, vectorizability, etc.

(Section 7.1 in the book is an excellent introduction - please read carefully.)

Structures for FIR Systems: Direct Form Implementations

An FIR system is described by

$$y(n) = \sum_{k=0}^{M-1} b_k x(n-k)$$

or, by the system function,

$$H(z) = \sum_{k=0}^{M-1} b_k z^{-k}$$

The impulse response is given by

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

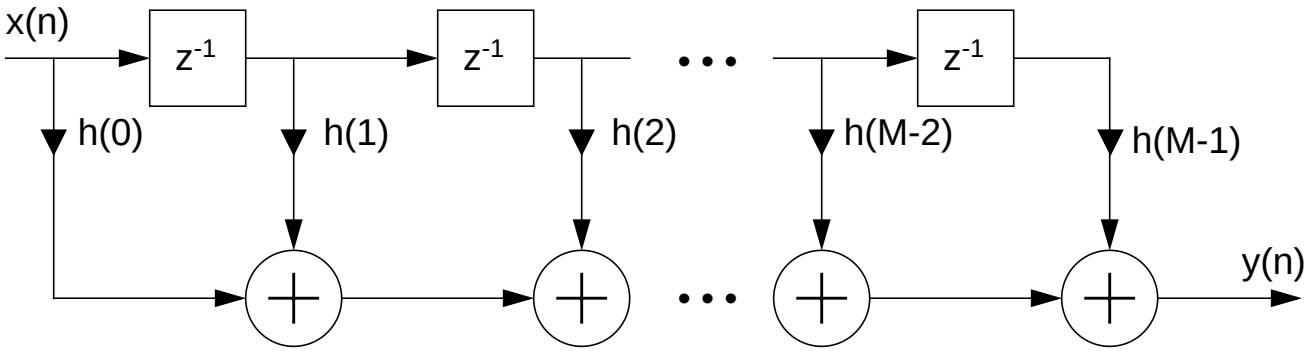
The output is computed as:

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

Observe that:

- Requires M-1 memory locations to store M-1 previous values of input
- Requires M multiplications and M-1 additions per input sample
- Requires either a circular buffer or shift registers
- Often called a tapped-delay line or transversal filter

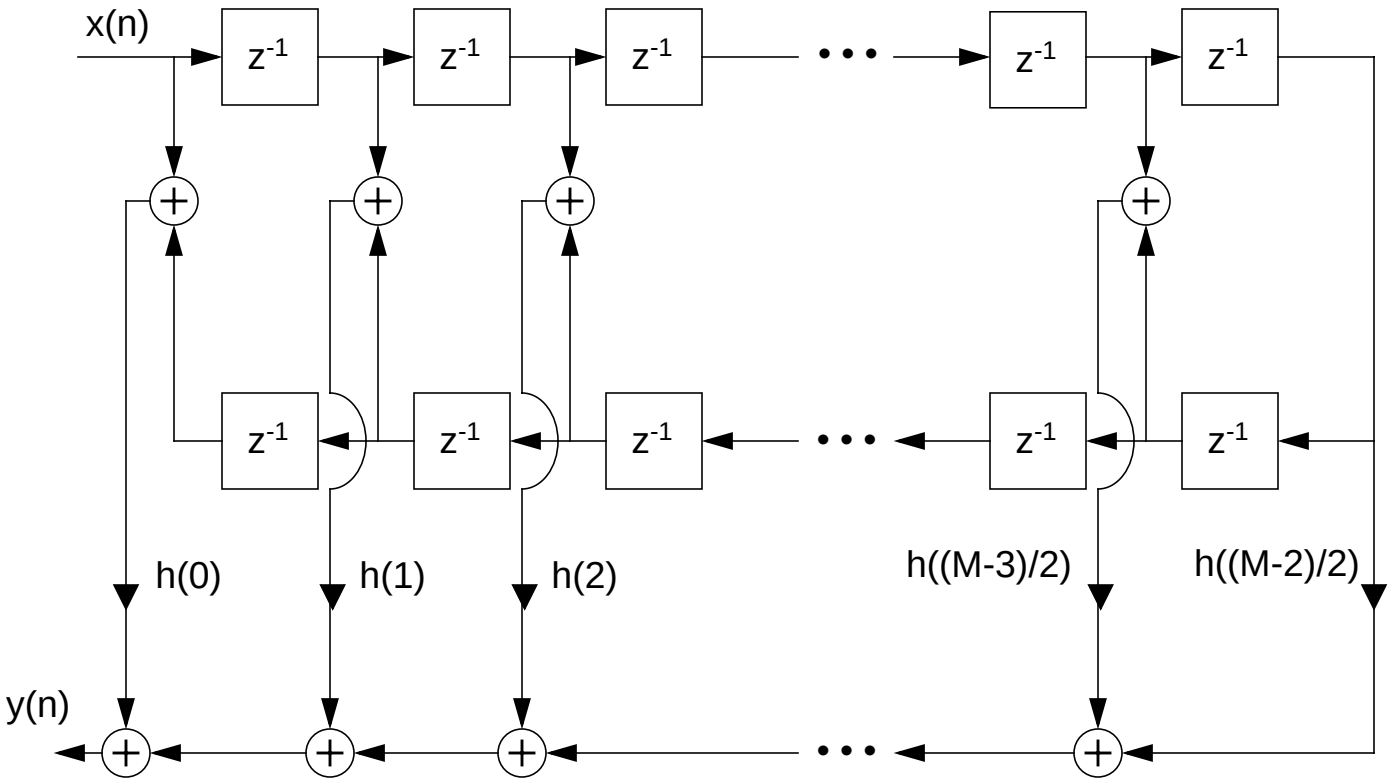
A direct form implementation is shown below. What would code for this filter look like? Can you generate code automatically from the graph?



When the filter is linear phase (symmetric or antisymmetric),

$$h(n) = \pm h(M-1-n)$$

and M/2 (M even) or (M-1)/2 (M odd) multiplications are required. How?



An efficient direct form implementation of a linear-phase FIR filter (M odd).

Cascade Forms

We can factor $H(z)$ into second-order systems:

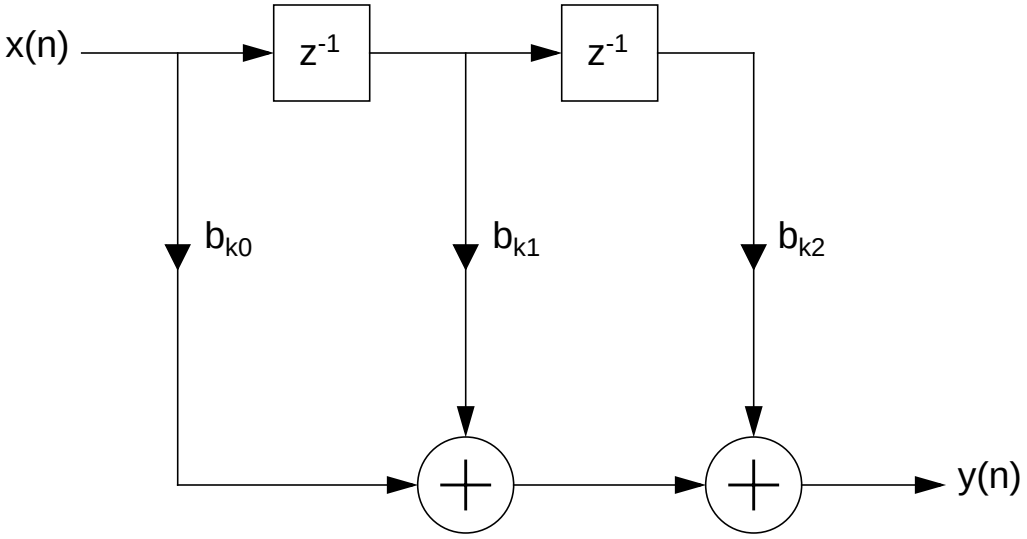
$$H(z) = \prod_{k=1}^{K} H_k(z)$$

where

$$H_k(z) = b_{k0} + b_{k1}z^{-1} + b_{k2}z^{-2}$$

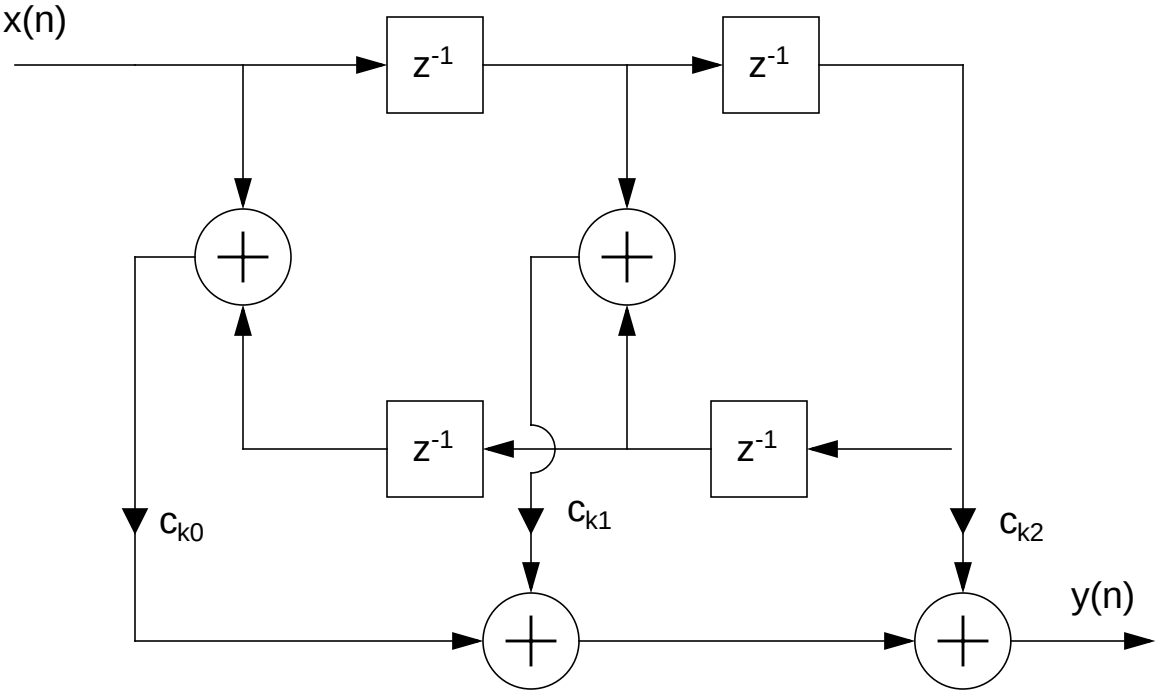
K is the integer part of (M+1)/2.

A simple cascade realization:



In the case of linear-phase filters, since the zeros of $H(z)$ exhibit a symmetry, we can form efficient fourth-order sections:

$$H(z) = c_{k0} + c_{k1}z^{-1} + c_{k2}z^{-2} + c_{k1}z^{-3} + c_{k0}z^{-4}$$



Frequency Sampling Structures

- A frequency sampling structure is an alternative structure for an FIR filter in which the parameters that characterize the filter are the values of the desired frequency response.

$$H(k+\alpha) = \sum_{n=0}^{M-1} h(n) e^{-j2\pi(k+\alpha)n/M}, \qquad k = 0, 1, 2, \ldots, M-1$$

where $\alpha = 0 \; or \; \frac{1}{2}$ (depending on which set of samples are desired).

We can express the impulse response in terms of the frequency samples:

$$h(n) = \frac{1}{M} \sum_{n=0}^{M-1} H(k+\alpha) e^{-j2\pi(k+\alpha)n/M}$$

This can be implemented in a pole/zero parallel realization: