Realizations of Digital Filters

1. INTRODUCTION:

In the preceding lessons we learned how to design Linear Time Invariant (LTI) digital filters, both IIR and FIR. The end result of the design was the transfer function H(z) of the filter or, equivalently, the difference equation it represents. We thus far looked at a filter as a black box, whose input-output relationships are well defined, but whose internal structure is ignored. Now it is time to look more closely at possible internal structures of digital filters, and to learn how to build such filters. It is convenient to break the task of building a digital filter into two stages:

1. Construction of a block diagram of the filter. Such a block diagram is called a **realization** of the filter. Realization of a filter at a block-diagram level is essentially a flow graph of the signals in the filter. It includes operations such as delays, additions, and multiplications of signals by constant coefficients. It ignores ordering of operations, accuracy, scaling and the like. A given filter can be realized in infinitely many ways. Different realizations differ in their properties, and some are better than others.
2. **Implementation** of the realization, either in hardware or in software. At this stage we must concern ourselves with problems neglected during the realization stage: order of operations, signal scaling, accuracy of signal values, accuracy of coefficients, and accuracy of arithmetic operations. We must analyze the effect of such imperfections on the performance of the filter. Finally, we must build the filter – either the hardware or the program code (or both, if the filter is a specialized combination of hardware and software).

2. DIRECT REALIZATIONS OF IIR FILTERS:

Let H(z) be a rational, causal, stable transfer function. We assume, for convenience, that the orders of the numerator and denominator polynomials of the filter transfer function are equal (There is no loss of generality in this assumption since it is always possible to extend numerator or denominator polynomials by adding zero-valued coefficients). Thus *H* (*z*) is given by:

$$\frac{Y(z)}{X(z)} = H(z) = \frac{B(z)}{A(z)} = \frac{\sum_{k=0}^{N} b(k) z^{-k}}{1 + \sum_{k=1}^{N} a(k) z^{-k}}$$

2.1 DIRECT FORMS I AND II:

Let us introduce an auxiliary signal u(n), related to the input and output signals X(n) and Y(n) through:

$$u(n) = -a(1)u(n-1) - \ldots - a(N)u(n-N) + x(n)$$

$$y(n) = b(0)u(n) + b(1)u(n-1) + \ldots + b(N)u(n-N)$$

Or, in the Z domain,

$$U(z) = \frac{1}{A(z)} X(z), \quad Y(z) = B(z)U(z)$$

Figure 1 shows a realization of the previous equations using the three types of building block (in the figure we have used N=3 as an example). By passing u(n) through a chain of N delay elements, we get the signals *{u(n-1), u(n-2)… u(n-N)}* Then we can form (n) as a linear combination of the delayed signals, plus the input signal x(n), as is expressed in last equation. We need multipliers for the coefficients *{-a (1), -a (N)}*, and two-input adders to form the sum. The realization uses feedback.

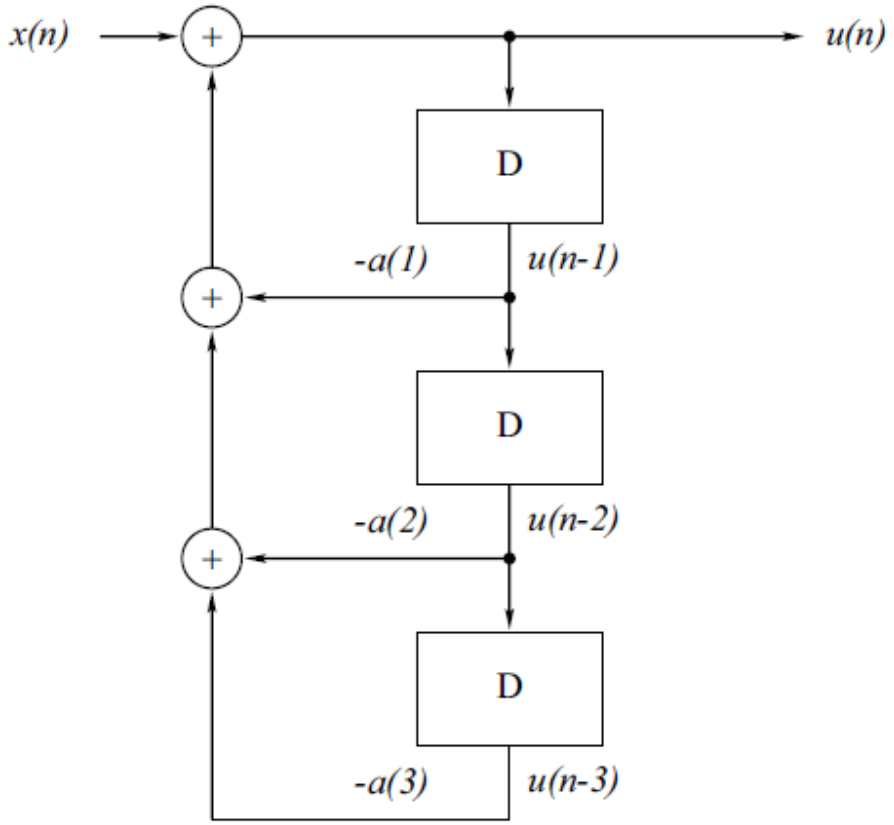

Figure 1: Realization of the auxiliary signal u(n).

We can now use the previous y(n) equation for generating the output signal *y* (*n*) from the auxiliary signal u(n) and its delayed values. We do this by augmenting Figure 1 with N+1 multipliers for the coefficients {b(0),…b(n)} and N adders. This results in the realization (called **direct form II**) shown in Figure 2. Note that it is **not necessary to increase** the number of delay elements and such realization is minimal (**canonical**) realization of IIR system.

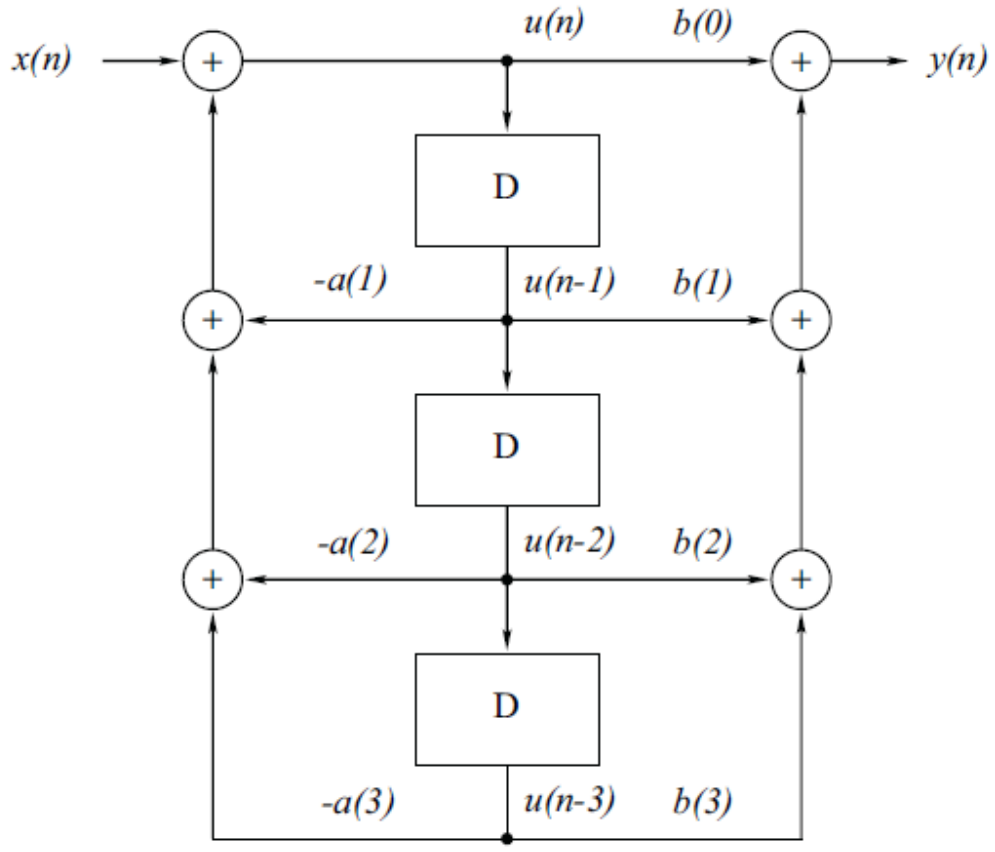

Figure 2: Direct realization (direct form II) of a digital IIR system.

It has the following properties:

1. The number of delay elements N is the maximum of the orders of polynomials A(z) and B(z). Assuming that these polynomials have no common factor, this is the **minimum possible number of delays** in any realization of H(z). The set of values at the outputs of the delay elements is called the **state** of the system.
2. There are 2N+1 multipliers and 2N adders. However, since some coefficients may be zero, there **may be a smaller** number of adders and multipliers.
3. The realization is **recursive**, since it generates present value from past values of these signals. It can be shown that any realization of an IIR system must be recursive if it is required to include only a finite number of delay elements.
4. Assuming that the input signal x(n) is causal. It is necessary **to initialize the state** components before feeding the input signal to the system. The state is usually initialized to zero. However, initialization to nonzero values is sometimes required, depending on the application.

Another direct realization (**direct form I**) of N-th order IIR filter can be constructed using $2N$ delay elements[3], N for the input signal $x(n)$, and N for the output signal $y(n)$. It is based on the auxiliary signal $v(n)$, related to the input and output signals $x(n)$ and $y(n)$ through

$$v(n) = b(0)x(n) + b(1)x(n-1) + \ldots + b(N)x(n-N)$$

$$y(n) = -a(1)y(n-1) - \ldots - a(N)y(n-N) + v(n)$$

Or, in the Z domain,

$$V(z) = B(z)X(z), \quad Y(z) = \frac{1}{A(z)}V(z)$$

And it is shown in Figure 3.

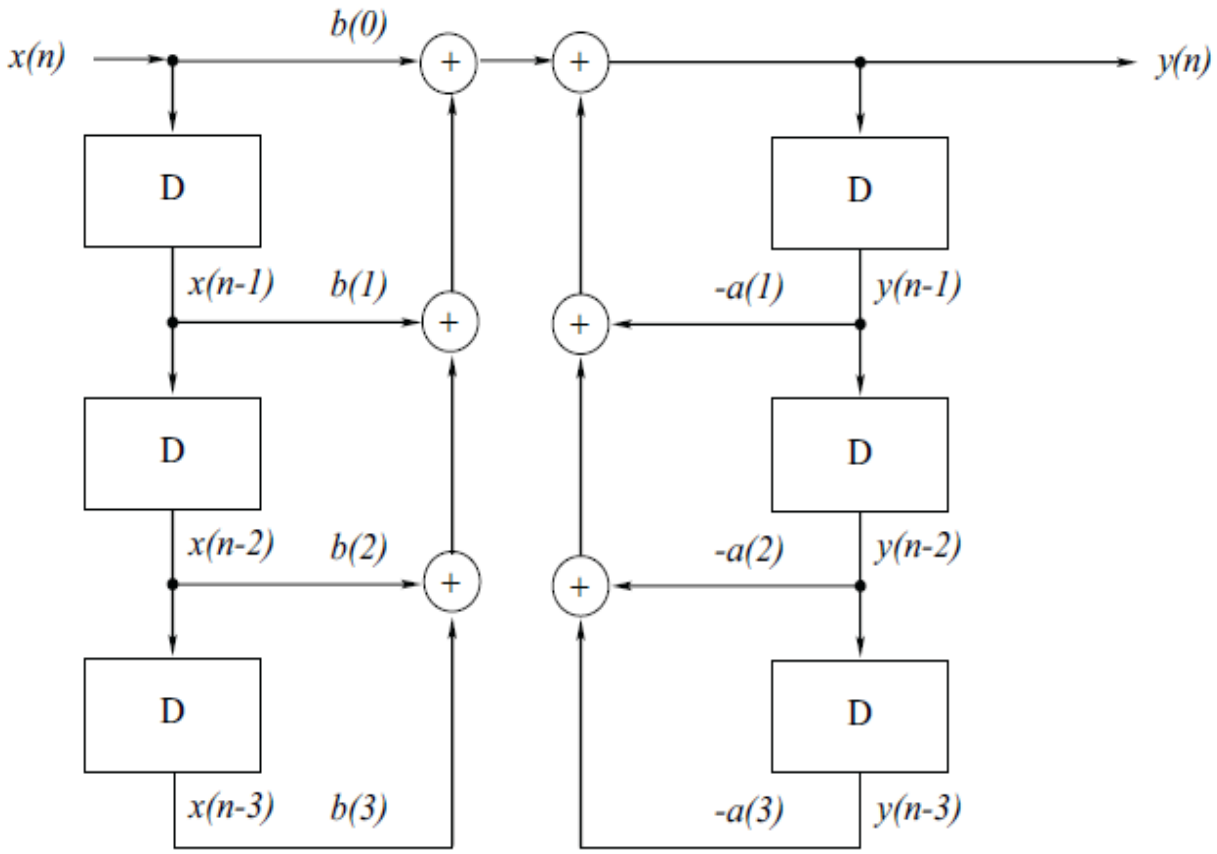

Figure 3: Direct realization (direct form I) of a digital IIR system.

2.2 TRANSPOSED DIRECT FORM II:

We now explore an alternative to the direct form realizations. Figure 4 shows a realization of y(n) in **direct form I** using the three types of building block (in the figure we again use N=3 as an example). The present value of y(n) is built from its own past values and the auxiliary signal v(n). To generate y(n-N), we need N delay elements. These elements can be used for generating all intermediate delays, as shown in the figure. This realization effectively computes y(n) in **direct form I** in the Z domain form.

$$Y(z)=\left(\ldots\left(-a(N)z^{-1}-a(N-1)\right)z^{-1}-\ldots-a(1)\right)z^{-1}Y(z)+V(z)$$

The state of this realization does not consist of pure delays of a single signal, but of linear combinations of different delays. As before, we need N multipliers for the coefficients {-a(1) …-a(N)} and N two-input adders.

We can now use v(n) equation In **direct form I** for generating the auxiliary signal v(n) from the input signal x(n) and its delayed values. We do this by augmenting Figure 4 with N+1 multipliers for the coefficients {b(0), …. b(N)} and N adders. This results in the realization shown in Figure 5. Note that it is **not necessary to increase** the number of delay elements (This is also canonical realization of IIR filter), since the existing elements can include of the necessary delay of the input signal.

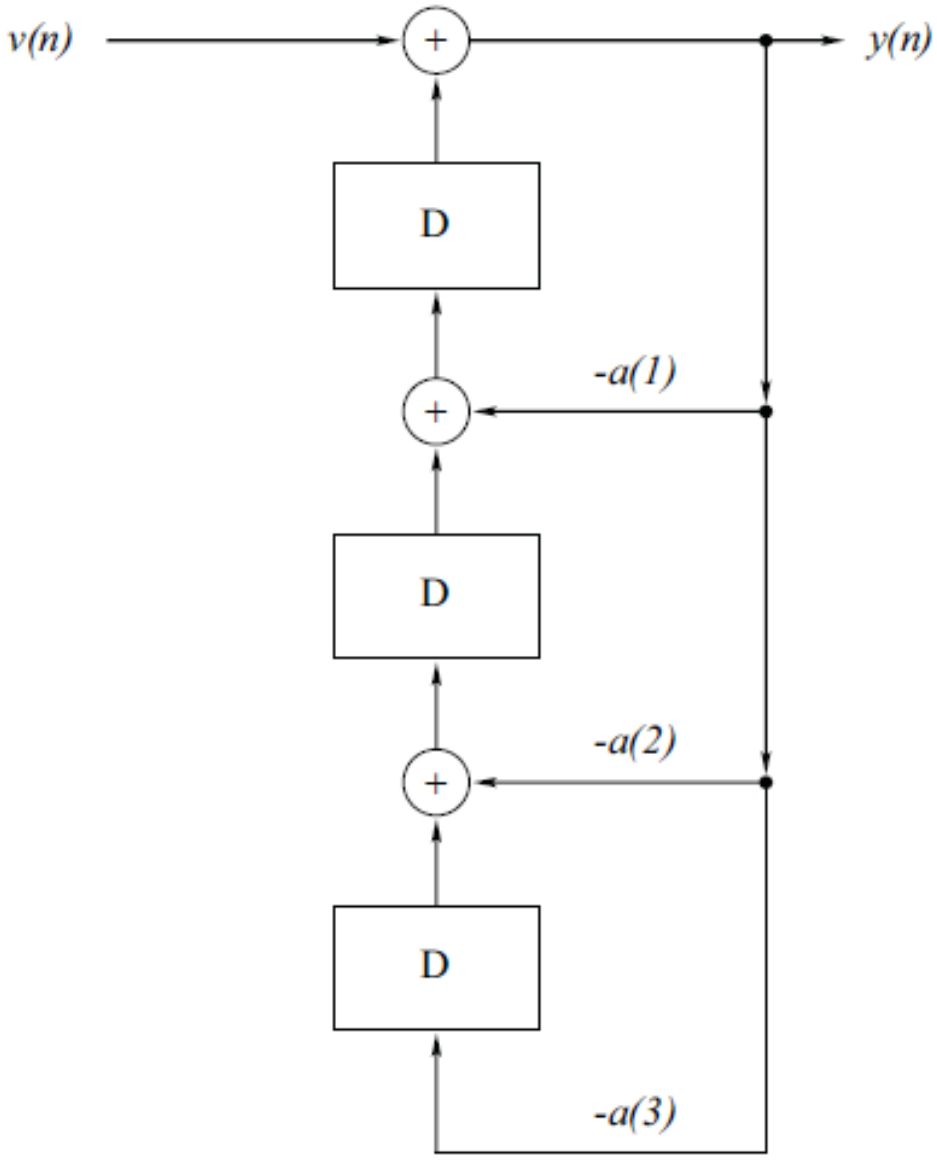

Figure 4: Realization of y(n) from the auxiliary signal v(n).

The realization shown in Figure 5 is known as a transposed direct realization (or **transposed direct form II**), for reasons explained next. The transposed direct form II shares the main properties of the direct form II. In particular, it has the same number of delays, multipliers, and two-input adders (Note that, in Figure 5, there are N-1 three-input adders and 2 two-input adders, which are equivalent to 2N two-input adders. However, the two realizations **have different states** (As long as the state is initialized

to zero, this difference is inconsequential. However, when initialization to a nonzero state is necessary, the two realizations require different computations of the initial state.

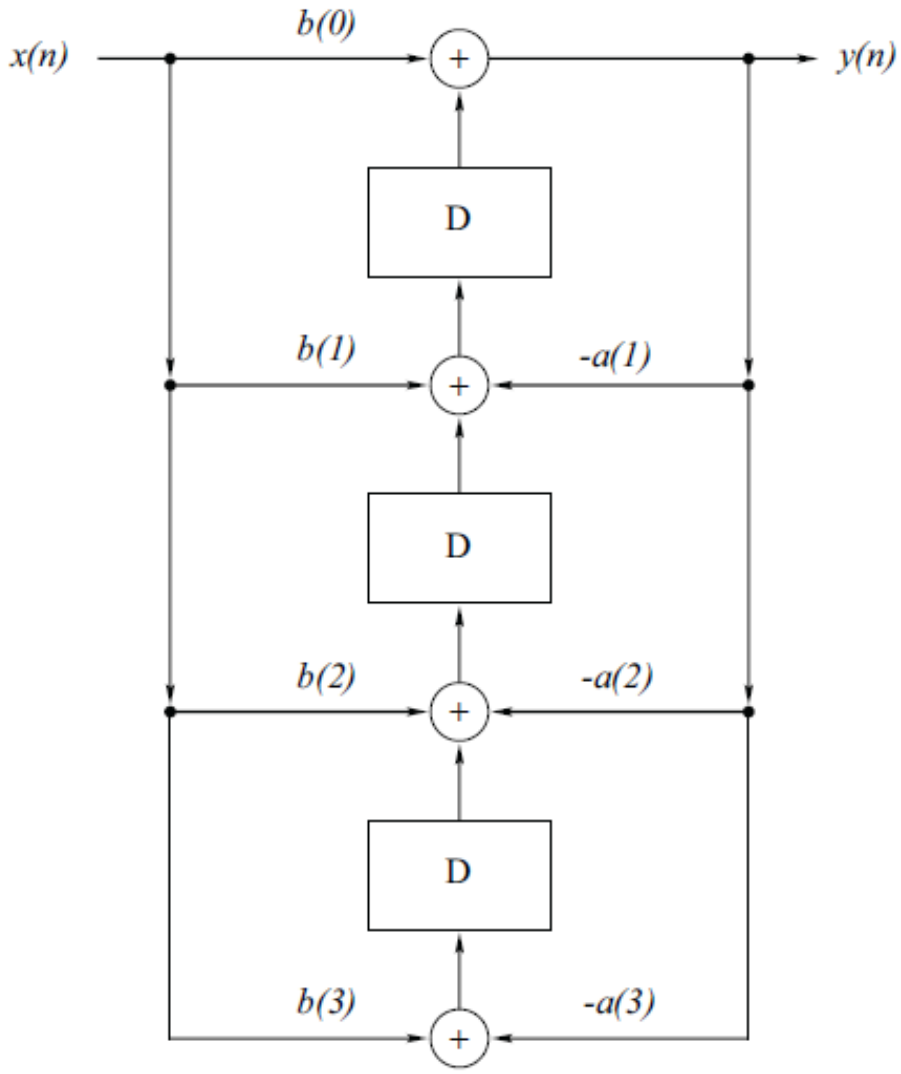

Figure 5: Transposed direct form II realization of a digital IIR filter.

Comparison of Figures 2 and 5 reveals that the later can be obtained from the former by the following sequence of operations:

1. Reversal of the signal flow direction in all lines (i.e., reversal of all arrows).
2. Replacing all adders by contact points, and all contact points by adders.

3. Interchange the input and output.

This sequence of operation is called **transposition** of the realization. A known theorem in the network theory states that transposition of a given realization leaves the transfer function of the realization invariant. Transposition of the transposed realization obviously gives back the original realization. Two such realizations are said to be **dual** to each other.

The procedures **direct** in the codes implements the two direct realizations of an IIR filter. The MATLAB function **filter** performs the same computation more efficiently, since it is coded internally. Therefore, this program is intended for educational purposes, rather than for serious use.

```
function y = direct(typ,b,a,x);
% Synopsis: direct(typ,b,a,x).
% Direct realizations of rational transfer functions.
% Input parameters:
% typ: 1 for direct realization, 2 for transposed
% b, a: numerator and denominator polynomials
% x: input sequence.
% Output:
% y: output sequence.

p = length(a)-1; q = length(b)-1; pq = max(p,q);
a = a(2:p+1); u = zeros(1,pq); % u: the internal state
if (typ == 1),
  for i = 1:length(x),
    unew = x(i)-sum(u(1:p).*a);
    u = [unew,u];
    y(i) = sum(u(1:q+1).*b);
    u = u(1:pq);
  end
elseif (typ == 2),
  for i = 1:length(x),
    y(i) = b(1)*x(i)+u(1);
    u = [u(2:pq),0];
    u(1:q) = u(1:q) + b(2:q+1)*x(i);
    u(1:p) = u(1:p) - a*y(i);
  end
end
```