Realization structures for FIR filters

The FIR filter is characterized by the transfer function, $H(z)$, given by

$$H(z) = \sum_{n=0}^{N-1} h(n)\, z^{-n}$$

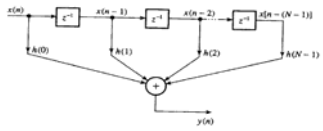
Realization structures are essentially block (or flow) diagram representations of the different theoretically equivalent ways the transfer function can be arranged. In most cases, they consist of an interconnection of multipliers, adders/summers and delay elements. There are many FIR realization structures, but only those that are in common use are presented here.

Transversal structure

The transversal (or tapped delay) structure is depicted in Figure 7.28. The input, $x(n)$, and output, $y(n)$, of the filter for this structure are related simply by

$$y(n) = \sum_{m=0}^{N-1} h(m)x(n-m) \qquad (7.39)$$

In the figure, the symbol z^{-1} represents a delay of one sample or unit of time. Thus $x(n-1)$ is $x(n)$ delayed by one sample. In digital implementations, the boxes labelled



z^{-1} could represent shift registers or more commonly memory locations in a RAM. The transversal filter structure is the most popular FIR structure.

The output sample, $y(n)$, is a weighted sum of the present input, $x(n)$, and $N-1$ previous samples of the input, that is $x(n-1)$ to $x(n-N)$. For the transversal structure, the computation of each output sample, $y(n)$, requires

- $N-1$ memory locations to store the $N-1$ input samples,
- N memory locations to store the N coefficients,
- N multiplications, and
- $N-1$ additions.

Linear phase structure

A variation of the transversal structure is the linear phase structure which takes advantage of the symmetry in the impulse response coefficients for linear phase FIR filters to reduce the computational complexity of the filter implementation.

In a linear phase filter, the coefficients are symmetrical, that is $h(n) = \pm h(N-n-1)$. Thus the filter equation can be re-written to take account of this symmetry with a consequent reduction in both the number of multiplications and additions. For type 1 and 2 linear phase filters, the transfer function can be written as

$$H(z) = \sum_{n=0}^{(N-1)/2-1} h(n)[z^{-n} + z^{-(N-1-n)}] + h\left(\frac{N-1}{2}\right) z^{-(N-1)/2} \quad N \text{ odd} \qquad (7.40a)$$

$$H(z) = \sum_{n=0}^{N/2-1} h(n)[z^{-n} + z^{-(N-1-n)}] \quad N \text{ even} \qquad (7.40b)$$

The corresponding difference equations are given by

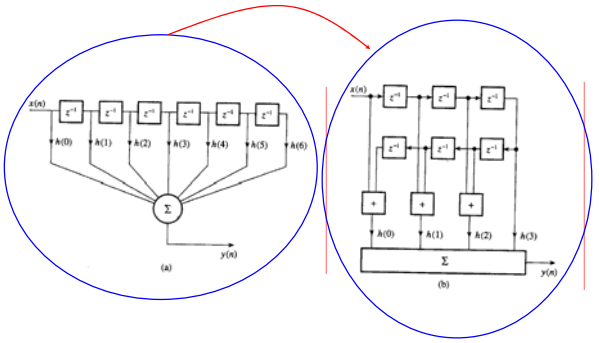
$$y(n) = \sum_{k=0}^{(N-1)/2-1} h(k)\{x(n-k) + x[n-(N-1-k)]\} + h[(N-1)/2]x[n-(N-1)/2] \qquad (7.41a)$$

$$y(n) = \sum_{k=0}^{(N-1)/2-1} h(k)\{x(n-k) + x[n-(N-1-k)]\} \qquad (7.41b)$$

A comparison of Equations 7.39 and 7.41 shows that the linear phase structure is computationally more efficient, requiring approximately half the number of multiplications and additions. However, in most DSP processors Equation 7.39 leads to a more efficient implementation, because the computational advantage in Equation 7.41 is lost in the more complex indexing of data implied.

A linear phase FIR filter has seven coefficients which are listed below. Draw the realization diagrams for the filter using (a) direct (transversal) and (b) linear phase structures. Compare their computational complexities.

$h(0) = h(6) = -0.032$
$h(1) = h(5) = 0.038$
$h(2) = h(4) = 0.048$
$h(3) = -0.048$



Other structures

Fast convolution

The fast convolution method involves performing the convolution operation of Equation 7.39 in the frequency domain. As was discussed in Chapter 5, convolution in the time domain is equivalent to multiplication in the frequency domain. In simple terms, filtering here is performed by first computing the DFTs of $x(n)$ and $h(n)$ (suitably zero padded), multiplying these together and then obtaining their inverse. The concept is depicted in Figure 7.30. In practice, techniques known as overlap–add and overlap–save are used in real-time filtering. These are discussed in Chapter 5.

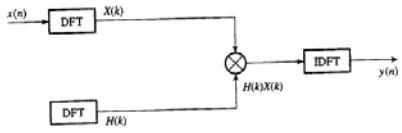


Figure 7.30 An illustration of fast convolution.

Frequency sampling structure

In the frequency sampling structure, the filters are characterized by the samples of the desired frequency response, $H(k)$, instead of its impulse response coefficients. This case has already been discussed in detail. For narrowband filters, most of the frequency samples will be zero, and so the resulting frequency sampling filter will require a smaller number of coefficients and hence multiplications and additions than an equivalent transversal structure. A typical realization diagram is given in Figure 7.22.

Transpose and cascade structures

The transpose structure is similar to the direct structure, except that the partial sums feed into succeeding stages. This method is more susceptible to roundoff noise than the direct method. In the cascade realization, the transfer function, $H(z)$, is expressed as the product of second-order and first-order sections. The transpose and cascade structures are seldom used for FIR filters in current DSP implementations.

Finite wordlength effect in FIR digital filters

In practice, FIR digital filters are often implemented using DSP processors (for example the Texas Instruments TMS320C50), algorithmic-specific DSP chips designed for FIR filtering (such as the INMOS A100) or, where high speed is desired, building blocks of multipliers, memory elements, adders and controllers (for example Plessey's PDSP1600 family). In these cases, the number of bits used to represent the input data to the filter and the filter coefficients and in performing arithmetic operations must be small for efficiency and to limit the cost of the digital filter. The problems caused by using a finite number of bits are referred to as finite wordlength effects, and in general lead to a lowering of the performance of the filter.

In this section, we will discuss the effects of finite wordlength on the performance of FIR digital filters and suggest ways of minimizing these effects. The discussion will centre on the direct form FIR structure as it is the most attractive FIR structure in

modern signal processing, and rounding will be used, being the simplest and most widely used method of quantization.

There are four ways in which finite wordlength affects the performance of FIR digital filters.

(1) *ADC noise* This is the familiar ADC quantization noise which results when the filter input is derived from analog signals. ADC noise limits the signal-to-noise ratio (SNR) obtainable. The effects can be reduced by using additional bits, consistent with inherent signal noise (see Chapter 13), and/or by using multirate techniques to enhance the signal to noise ratio (see Chapter 9).

(2) *Coefficient quantization errors* These result from representing filter coefficients with a limited number of bits. This has the adverse effect of modifying the desired frequency response. In the stopband of a filter, for example, it limits the maximum attenuation possible, thus allowing additional signal transmission. A straightforward solution is to use enough bits to represent filter coefficients. However, optimization techniques allow efficient selection of coefficients to minimize coefficient wordlength.

(3) *Roundoff errors from quantizing results of arithmetic operations* These can occur, for example, by discarding the lower-order bits before storing the results of a multiplication. This is normally forced on us by the wordlength of the processor used. This error reduces the SNR and may be reduced by rounding after double-length summing of products. The extent of the errors introduced depends on the type of arithmetic used and the filter structure.

(4) *Arithmetic overflow* This occurs when partial sums or filter output exceeds the permissible wordlength of the system. Essentially, when an overflow occurs, the output sample will be wrong (normally the sign changes). A way to reduce or avoid an overflow is to scale the filter coefficients by dividing each coefficient by a factor such that the filter output sample never exceeds the permissible wordlength. This is clearly at the cost of reduced signal to noise ratio.

FIR digital filters implementation techniques

The difference equation for an FIR digital filter is given by

$$y(n) = \sum_{k=0}^{N-1} h(k)x(n-k) \qquad (7.48)$$

The coefficients $h(k)$ will have been obtained at the approximation stage, a suitable structure chosen, and an analysis carried out to verify that the number of bits to be used to represent variables and in carrying out arithmetic operations is adequate. The final stage is to implement the filter, and the key issue here is essentially to produce software code and/or hardware realization of the chosen filter structure. The discussions here will be based on the transversal structure which is characterized by Equation 7.48 as it is the most popular.

As examination of the equation will show, the computation of $y(n)$ involves only multiplications, additions/subtractions, and delays. Thus, to implement a filter, we need the following basic components:

- memory (RAM) to store the present and past input samples, $x(n)$ and $x(n-k)$;
- memory (RAM or ROM) for storing the filter coefficients, the $h(k)$;
- a multiplier (software or hardware);
- adders or arithmetic logic unit (ALU).

These components together with a means of controlling them constitute the digital filter. If the source of the input data is analog, then we need an ADC as well. Similarly, if the output destination is analog we need a DAC. Thus the structure of a real-time filter has the form depicted in Figure 7.34.

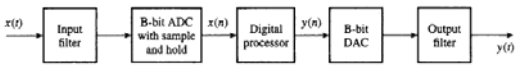
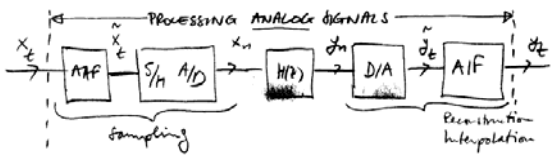


Figure 7.34 A simplified block diagram of a real-time digital filter with analog input/output signals.

Filtering continuous-time signals using digital filters



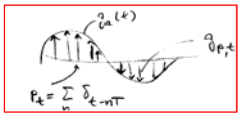
sampling interval $T \Rightarrow F_T = \frac{1}{T}$

$g_n = g_a(nT) \quad \forall n$

$G_a(j\Omega) = \int g_a(t) e^{-j\Omega t} dt$

$G(e^{j\omega}) = \sum_n g_n e^{-j\omega n}$ (4)

relate ?!



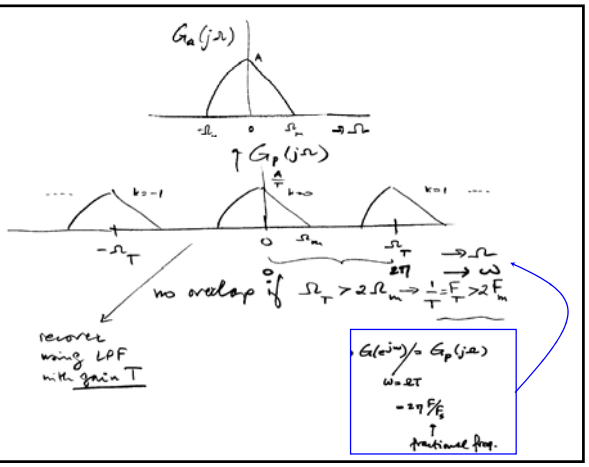
$$G_p(j\Omega) = \frac{1}{T}\sum_k G_a\left(j(\Omega - k\Omega_T)\right)$$

$\propto$ sum of shifted & scaled replicas of $G_a(j\Omega)$

$k=0$: baseband

$-\Omega_T/2 \le \Omega \le \Omega_T/2$ Nyquist band

Nyquist freq.



reconstruction

$$G_a(j\Omega) = \begin{cases} T \cdot G_p(j\Omega) & -\Omega_T/2 < \Omega < \Omega_T/2 \\ 0 & \text{o.w.} \end{cases}$$

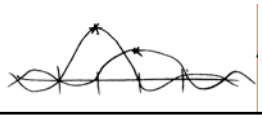
$$g_{a,t} = \frac{1}{2\pi}\int_{-\Omega_T/2}^{\Omega_T/2} T\left[\sum_n g_n e^{-j\Omega nT}\right] e^{j\Omega t}\, d\Omega$$

$$= \frac{T}{2\pi}\sum_n g_n \int_{-\Omega_T/2}^{\Omega_T/2} e^{j\Omega(t-nT)}\, d\Omega$$

$$\frac{e^{j\Omega(t-nT)}}{j(t-nT)}\Big|_{-\Omega_T/2}^{\Omega_T/2}$$

$$= \sum_n g_n \cdot \frac{2T}{2\pi}\frac{\sin(\Omega_T(t-nT)/2)}{(t-nT)} \cdot \frac{2}{\Omega_T}\frac{\sin \Omega_T(t-nT)/2}{(t-nT)}$$

$$= \sum_n g_n \frac{\sin(\Omega_T(t-nT)/2)}{\Omega_T(t-nT)/2}$$



Anti-Aliasing Filter (AAF) Design

AAA: anti aliasing attenuation