A survey of theory and design techniques for digital filters

Srinivasan, K. R.; Tavares, s. E.

Report (National Research Council of Canada. Radio and Electrical Engineering Division : ERB), 1970-06

ERB-842

UNCLASSIFIED

NATIONAL RESEARCH COUNCIL OF CANADA

RADIO AND ELECTRICAL ENGINEERING DIVISION

A SURVEY OF THEORY AND DESIGN TECHNIQUES FOR DIGITAL FILTERS

- K. R. SRINIVASAN AND S. E. TAVARES -

OTTAWA

JUNE 1970

ABSTRACT

A survey of design techniques for digital filters is presented in this report. Design techniques employing the z-transform and numerical methods are discussed, and the frequency domain synthesis of digital filters is illustrated. The theory of the discrete Fourier transform and the many possible forms of fast Fourier transform are presented. The realization of digital filters in terms of hardware modules and the impact of LSI technology on them is discussed. Finally, a cost evaluation of digital filters, assuming current LSI technology, is attempted. A comprehensive bibliography is presented.

K.R. Srinivasan was born in Madras, India, on August 8, 1941. He received the B.Sc. degree in Physics from Madras University, India, in 1961 and B.Eng. degree in electrical communication engineering from Indian Institute of Science, Bangalore, India in 1964. He is currently employed in the Electronics division of the National Aeronautical Lab., Bangalore, India. During 1969–70 he was on a Colombo plan fellowship at the National Research Council of Canada. He is interested in digital data processing and data logging systems.

Stafford E. Tavares was born in Kingston, Jamaica, West Indies, on May 11, 1940. He received the B. Eng. degree from McGill University, Montreal, Canada, in 1962, the M.S. degree from the California Institute of Technology, Pasadena, in 1964, and the Ph.D. degree from McGill University in 1968, all in electrical engineering. In 1962 he joined the Radio and Electrical Engineering Division of the National Research Council of Canada. His present areas of interest are coding theory and digital communications.

CONTENTS

		Page
1.00	Introduction	1
2.00	Digital Filter Design by z-transform Method	3
2.01	Difference Equations and z-transforms	3
2.02	Inverse z-transform	5
2.03	Digital Filter Design	6
2.04	Digital Filter Implementation	7
2.05	Recursive and Non-Recursive Filters	10
3.00	Classical Numerical Approach to Digital Filter Design	11
4.00	Frequency Domain Synthesis of Digital Filters	14
4.01	Synthesis of Digital Filters	14
4.02	Bilinear Transformation Technique	14
5.00	Fast Fourier Transform Technique	18
5.01	The Fast-Fourier Transform	18
5.02	The Discrete Fourier Transform	19
5.03	Comparison of Computation Time	20
5.04	Decimation in Time	21
5.05	Decimation in Frequency	27
6.00	Digital Hardware Modules and the Impact of LSI	30
6.01	Hardware Implementation of a Digital Filter	30
6.02	Impact of LSI Technology and Cost Evaluation	35
7.00	Conclusion	38
8.00	Acknowledgment	38
9.00	Bibliography	39

FIGURES

1. Block-schematic of a digital processor
2. Ladder network
3. Digital network of equation 2.34
4. Modified digital network of Fig. 3
5. Another digital network version of Fig. 3
6. The graphical representation of equation 2.37

7. Graphical representation of equation 2.38
8. Filtering operation with five weights
9. Sine-terminated digital filters
10. Ideal low-pass filters
11. Butterworth filter response
12. Digital filter frequency transformation
13. Decomposition of time-series into odd and even-numbered samples
14. Signal flow graph of decimation in time. Eight point DFT reduced to two DFT's
15. Eight-point DFT of Fig. 14 reduced to four two-point DFT's
16. Eight-point DFT of Fig. 13 completely reduced to complex multiplication and addition
17. Rearrangement of the flow graph of Fig. 16 gives naturally ordered time samples
18. Rearrangement of the flow graph of Fig. 16 gives DFT computation without bit reversal
19. Eight-point DFT reduced to 2 four-point DFT's by decimation in frequency
20. Eight-point DFT further reduced to 4 two-point DFT's
21. Complete reduction of the eight-point DFT
22. Rearrangement of Fig. 21 with naturally ordered DFT coefficients
23. Rearrangement of Fig. 21 to give DFT computation without bit-reversal
24. Serial-order parallel digital filter configuration
25. Sixth-order cascaded digital filter configuration
26. Serial-sequential 2-pole filter block schematic
27. Parallel-simultaneous two-pole filter block schematic
28. Cascade realization of digital filter
29. Multiplexing for multiple channels

TABLES

I Comparison of the number of multiplications required using direct and FFT methods

II The present state of the art of MSI/LSI devices

A SURVEY OF THEORY AND DESIGN TECHNIQUES FOR DIGITAL FILTERS

– K.R. Srinivasan* and S.E. Tavares –

1.00 Introduction

Processing of signals by digital techniques as compared to the conventional analog method is gaining wide acceptance in many branches of science and engineering. With the rapid growth in large-scale integration and computer technology, on-line digital information processing is now considered fairly attractive in terms of speed, economy, and accuracy. Linear filtering and spectrum analysis are basic and widely used signal processing operations. The concept of analog filtering is quite familiar and very well established. It varies in complexity from the simple RC circuit used commonly to suppress noise to complex filters such as Tschebyschev and Butterworth types. Recently, active filters making use of low-priced integrated-circuit operational amplifiers have been used very extensively in signal processing operations. At low frequencies, they have proved very competitive with the conventional passive discrete-component filters. This is mainly due to the fact that at low frequencies high value capacitors and inductors are needed and hence the filter becomes bulky and unwieldy.

The signal applied to an analog filter is continuous – that is, it is present all the time – whereas a digital filter requires that the signal be in discrete form; that is, it is the discrete samples of the continuous signal quantized at equal intervals. Analog filtering is based on linear network theory, which in turn is based on the electrical properties of components like resistance, capacitance, and inductance. The performance of the network is defined by a linear differential equation. In contrast, a digital filter is defined in terms of a linear difference equation. In order to realize the difference equation, the input signal should be in the form of discrete samples. These samples are then processed by digital logic which performs the required numerical operations defined by the difference equation. A digital filter may also be considered as a set of numerical weights that are applied to the incoming discrete sampled signal, the values of the weights and the number of them required being dependent on the filtering requirements. The filtering operation consists of multiplying the input signal with the corresponding weights and summing them. As a large number of additions and multiplications are involved in carrying out the filtering operation, in general these operations are performed numerically (digitally) by special or general purpose digital computers. The numerical error encountered in the computation process can be considered as noise superimposed on the signal [27, 40].

---

*National Aeronautical Lab., Bangalore – 17, India

Digital filters find extensive application in such areas as audio/electroacoustics, telephone switching circuits, speech processing, radar and sonar systems, telemetry data processing, and digital control system compensation. Digital filtering techniques appear to be much superior to analog techniques in several areas. As an example, precise equalizers are very difficult to design and realize in the analog form but are much easier in the digital form. Advantages of digital techniques over analog are small size, IC implementation, stability, absence of impedance matching problems (this is due to the inherent isolation in digital circuits), greater flexibility, time-sharing facility, and repeatability.

The basic block-schematic of a digital processor can be conceived as in Fig. 1.

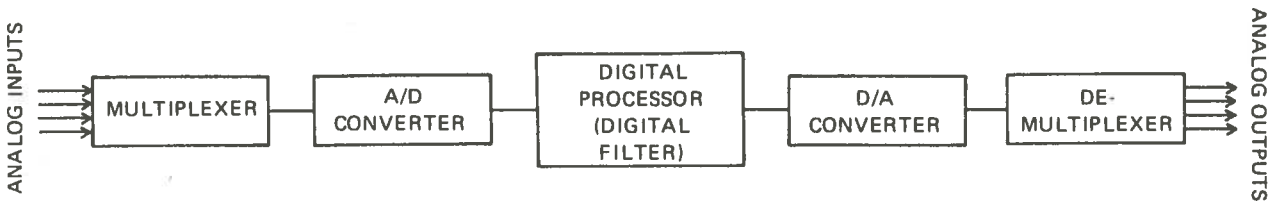


Figure 1 Block-schematic of a digital processor

This consists of an A/D converter which quantizes the analog input and presents the quantized samples to the digital processor – a digital processor which is essentially a digital filter configuration and a D/A converter which converts the filter output back to the analog form. Time-sharing of the processor is achieved via the multiplexer to the input. Numerous variations of the above are possible and in cases where the processing is between two digital terminal systems there is no need of A/D and D/A converters. Due to the high cost of A/D, D/A converters and the digital processor, the design of real-time digital filters was limited in the past. However, with the rapid growth of large scale integration (LSI) technology, the over-all system cost will be lowered, size will be greatly reduced, and reliability increased, and hence digital filtering will be competitive with analog filtering.

Digital filter hardware consists basically of shift registers, adders, and gates, and hence the filter design can be of highly flexible modular form; this approach is elaborated on later in this report. Recently, the approach to digital filters has been in terms of a hardware system, used to realize the particular task. A few companies are already in the process of developing digital filter hardware. The hardware will be fast enough to permit multiplexing for efficient use. For example, as shown in Fig. 1, many-section filters may be implemented by building one section and multiplexing it by commutation. The filtering coefficients may be changed for each section. The conversion rates of presently available A/D and D/A converters limit the useful higher frequency range of digital filters. A/D converters having a conversion rate as high as

200 Mega samples per second are presently available and they tend to be costly, but it is not unreasonable to expect to get higher conversion rates at reasonable prices in the future, thus making digital filters feasible at much higher frequencies.

Recently many techniques have evolved which increase the speed and frequency range of digital filters such as: the discrete Fourier transform, fast Fourier transform, high-speed convolution and correlation. We will examine some of these techniques, including the conventional ones, in this report. At the end of this report an extensive bibliography is presented and it is hoped that it will be useful to those starting in this expanding field.

2.00 Digital Filter Design by z-transform Method

2.01 Difference Equations and z-transforms

Continuous linear dynamic systems are generally analyzed by the use of Laplace and Fourier transforms, and by using general network concepts. Linear discrete systems are analyzed by the use of the z-transform and network concepts. The z-transform is a natural tool for the solution of linear constant-coefficient difference equations, just as the Laplace transform is for the solution of linear constant-coefficient differential equations [4 1].

To understand the difference equation better we can look at it in terms of network concepts.

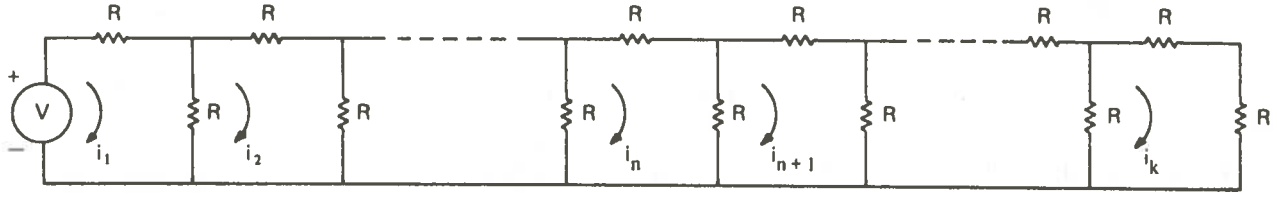


Figure 2 Ladder network

In the ladder network of Fig. 2, the voltage equation for the $(n+1)$th loop can be written as

$$-Ri_n + 3Ri_{n+1} - Ri_{n+2} = 0 \qquad (2.01)$$

or

$$i(n) - 3i(n+1) + i(n+2) = 0 \qquad (2.02)$$

This is a linear, constant-coefficient difference equation of second order. In order to solve this equation we resort to the z-transform, which will be described below.

In sampled-data systems a sampled function is approximated by a train of pulses, each having an area equal to the value of the function at the sampling instant. The sampled function corresponding to $f(t)$ is defined as

$$f^*(t) = f(0)\,\delta(t) + f(T)\,\delta(t - T) + f(2T)\,\delta(t - 2T) + \ldots . \tag{2.03}$$

$$= \sum_{n=0}^{\infty} f(nT)\,\delta(t - nT) \tag{2.04}$$

where T is the sampling period in seconds and $\delta(t)$ is the Dirac delta function. Taking the Laplace transform of the sampled function,

$$F^*(s) \triangleq L[f^*(t)] = \sum_{n=0}^{\infty} f(nT)\,e^{-snT} \tag{2.05}$$

Making a change of variable

$$z = e^{st} \tag{2.06}$$

we have

$$F(z) \triangleq \Big[L[f^*(t)]\Big]_{e^{sT}=z} = \sum_{n=0}^{\infty} f(nT)\,z^{-n} \tag{2.07}$$

where z is interpreted as a complex transform variable. The definition of the z-transform is given by the equation (2.07). Letting $T = 1$, the z-transform can be written as:

$$Z[f(n)] \triangleq F(z) \triangleq \sum_{n=0}^{\infty} f(n)\,z^{-n}. \tag{2.08}$$

$$= f(0) + f(1)z^{-1} + f(2)z^{-2} + \ldots . \tag{2.09}$$

As an illustration, the z-transform of a unit step function given by

$$u(n) \triangleq \begin{cases} 1, & n \geqslant 0 \\ 0, & n < 0 \end{cases} \tag{2.10}$$

is

$$Z[u(n)] = \sum_{n=0}^{\infty} z^{-n} = 1 + z^{-1} + z^{-2} + \ldots = \frac{1}{1 - z^{-1}} \tag{2.11}$$

Similarly, the z-transform of a^n is given by $\dfrac{1}{1 - az^{-1}}$. The z-transform pairs may be obtained from any standard textbook on the z-transform [64].

The z-transform is a linear operation and hence

$$Z[a_1 g_n + a_2 h_n] = a_1 Z[g_n] + a_2 Z[h_n] \tag{2.12}$$

where a_1 and a_2 are constants.

Also,
$$Z[f_{n-k}] = z^{-k} Z[f_n] \tag{2.13}$$

which is the z-transform of a delayed sequence. The z-transform, $F(z)$, of the convolution of two sequences g_n and h_n is equal to the product of their individual z-transforms, that is,

$$F(z) = G(z)H(z) \tag{2.14}$$

where

$$F(z) = Z[f_n],\ G(z) = Z[g_n]\ ,\ H(z) = Z[h_n]\ . \tag{2.15}$$

2.02 Inverse z-transform

As we have seen, the z-transform is given by

$$Z[f(n)] = \sum_{n=0}^{\infty} f(n)z^{-n} = F(z) \tag{2.16}$$

The inverse operation can be written as

$$Z^{-1}[F(z)] = f(n) \tag{2.17}$$

$f(n)$ being obtained from a complex inversion formula

$$f(n) = \frac{1}{2\pi j} \oint F(z)\, z^{n-1} dz \tag{2.18}$$

This formula may be applied to any transform that has a region of convergence.

In the case of a rational function of z^{-1}, we can find an inverse z-transform which does not require the contour integral (2.18). From (2.16) we can write

$$F(z) = f(0) + f(1)z^{-1} + f(2)\, z^{-2} + \ldots . \tag{2.19}$$

In the case of a rational function, the expansion can be made by long division as below. For example,

$$\begin{array}{r|l} & 1 + z^{-1} + z^{-2} + \ldots \\ \hline 1 - z^{-1} & 1 \\ & \underline{1 - z^{-1}} \\ & \quad z^{-1} \\ & \quad \underline{z^{-1} - z^{-2}} \\ & \qquad z^{-2} \\ & \qquad \underline{z^{-2} - z^{-3}} \\ & \qquad\quad \cdot \\ & \qquad\quad \cdot \end{array}$$

Hence

$$Z^{-1}\left[\frac{1}{1 - z^{-1}}\right] = 1 + z^{-1} + z^{-2} + \ldots \tag{2.20}$$

The above computation could be easily implemented on a digital computer. The long division is susceptible to round-off error, but has the advantage that the location of the poles of $F(z)$ need not be found.

For a proper rational function of z^{-1}, with N-poles we have,

$$F(z) = \frac{\sum_{i=0}^{N-1} a_i z^{-i}}{1 + \sum_{i=1}^{N} b_i z^{-i}} = z\left[\frac{\sum_{i=0}^{N-1} a_i z^{N-1-i}}{z^N + \sum_{i=1}^{N} b_i z^{N-i}}\right] \tag{2.21}$$

Assuming that the poles p_i are distinct, and expanding the bracketed expression in a partial fraction expansion,

$$F(z) = z\sum_{i=1}^{N} \frac{A_i}{z - p_i} = \sum_{i=1}^{N} \frac{A_i}{1 - p_i z^{-1}} \tag{2.22}$$

$$\therefore f_n = \sum_{i=1}^{N} A_i p_i^n \tag{2.23}$$

where p_i may be complex. This method has the advantage that the value of the signal at time n can be computed without computing any other values.

2.03 Digital Filter Design

Let x_n and y_n represent the input and output, respectively, of a linear discrete system, related by a linear difference equation with constant coefficients of the form

$$y_n + b_1 y_{n-1} + \ldots + b_N y_{n-N} = a_0 x_n + \ldots + a_M x_{n-M} \tag{2.24}$$

Equation (2.24) implies that the output at time n can be computed from the present input and linear combinationof past inputs and outputs. Taking the z-transform of (2.24), term by term, we have

$$Y(z)\left[1 + \sum_{i=1}^{N} b_i z^{-i}\right] = X(z)\sum_{i=0}^{M} a_i z^{-i} \ldots \tag{2.25}$$

$$\therefore Y(z) = X(z)\frac{\sum_{i=0}^{M} a_i z^{-i}}{1 + \sum_{i=1}^{N} b_i z^{-i}} \ldots \tag{2.26}$$

or

$$Y(z) = X(z)\,H(z) \tag{2.27}$$

where
$$H(z) = \frac{\sum_{i=0}^{M} a_i z^{-i}}{1 + \sum_{i=1}^{N} b_i z^{-i}} \quad \ldots \tag{2.28}$$

is the transfer function relating x to y. $H(z)$ in (2.28), which represents a time-invariant linear operation on discrete-time signals, characterizes a digital filter. The response of the system given by $H(z)$ to a discrete-time signal (x_n) is

$$y_n = \sum_{k=0}^{\infty} x_k h_{n-k} \tag{2.29}$$

From equation (2.24), we obtain

$$y_n = a_0 x_n + a_1 x_{n-1} + \ldots + a_M x_{n-M} - b_1 y_{n-1} - \ldots - b_N y_{n-N} \tag{2.30}$$

As mentioned earlier, the output at an instant n depends on the present input and a finite linear combination of the past inputs and outputs.

The frequency response of a digital filter is found by the values of its transfer function on the unit circle in the z-plane ($|z| = 1$). For stability of linear continuous systems, the necessary and sufficient condition is that the roots of $F(s)$ lie in the left half of the s-plane. These conditions have been studied in detail and we have the criteria of Routh-Hurwitz and Leenard-Chipard. A linear discrete system is stable if and only if the roots of $F(z)$ lie inside the unit circle in the z-plane.

2.04 Digital Filter Implementation

As seen earlier, the design of a digital filter rests in choosing the proper transfer function $H(z)$, to realize the particular task. In the case of digital filters, unlike the analog case, the implementation of a difference equation or system of difference equations to realize the particular transfer function $H(z)$ is fairly simple.

To illustrate this, let

$$H(z) = \frac{\sum_{i=0}^{M} a_i z^{-i}}{1 + \sum_{i=1}^{N} b_i z^{-i}} \quad \ldots \tag{2.31}$$

then
$$Y(z) = H(z)\, X(z)$$
$$= \frac{\sum_{i=0}^{M} a_i z^{-i}}{1 + \sum_{i=1}^{N} b_i z^{-i}} \; X(z) \quad \ldots \tag{2.32}$$

$$\therefore \; Y(z) = X(z) \sum_{i=0}^{M} a_i z^{-i} - Y(z) \sum_{i=1}^{N} b_i z^{-i} \; . \tag{2.33}$$

Hence we get the output y_n in the time domain as

$$y_n(t) = \sum_{i=0}^{M} a_i x_{n-i}(t) - \sum_{i=1}^{N} b_i y_{n-i}(t) \tag{2.34}$$

Hence the output y_n can be computed in a digital computer by inserting the $(M + 1)$ most recent samples of the input x_n and the N previous values of the outputs. We can draw the digital network for the above difference equation as in Fig. 3.

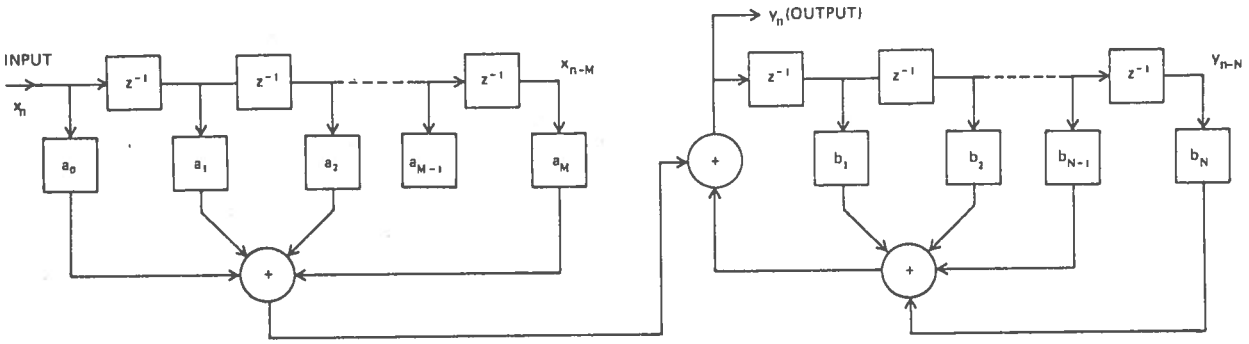


Figure 3 Digital network of equation 2.34

The operation of the digital network of Fig. 3 may be interpreted as follows: at the instant t, which is a multiple of the sampling period T, x_n forms the input to the network. The other quantities $x_{n-1}, x_{n-2}, \ldots x_{n-M}, y_{n-1}, y_{n-2}, \ldots, y_{n-N}$ are stored and used in the computation of y_n. The block z^{-1} represents a unit delay, i.e., the sampling period T, the rectangles containing the constants a_0, a_1, etc., represent multiplication by the constants, and the symbol $\oplus$ represents an adder. The block diagram also gives an estimate of the number of multipliers, adders, and delay elements needed to realize the required digital filter. There are a number of ways in which a digital filter transfer function $H(z)$ can be realized and hence the design is aimed at a configuration which uses less memory and fewer multipliers. As an illustration, Figs. 4 and 5 show different configurations of Fig. 3 which use fewer delay elements.

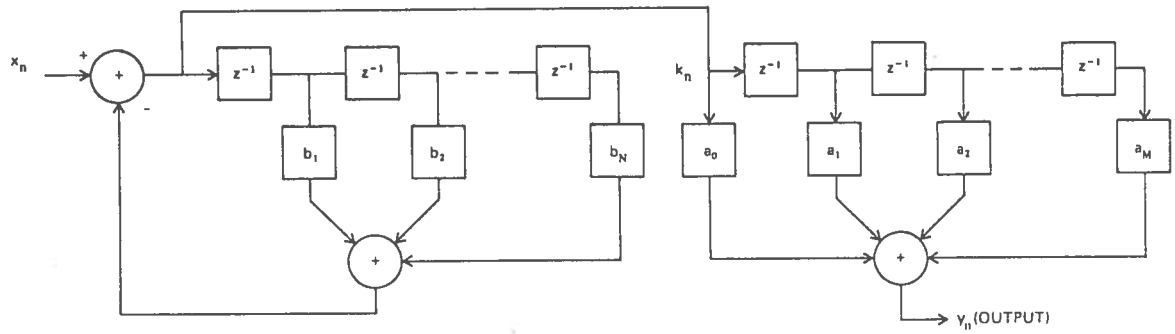


Figure 4 Modified digital network of Fig. 3

In Fig. 4

$$k_n = x_n - \sum_{i=1}^{N} b_i k_{n-i} \quad \ldots \tag{2.35}$$

$$y_n = \sum_{i=0}^{M} a_i k_{n-i} \quad \ldots \tag{2.36}$$

It can be noted that in Fig. 4, we have to store only N or M previous values of k_n, depending on which is greater. This reduces the memory required. In Fig. 5, single delay elements are used for the same output (assuming $N = M$, for convenience).

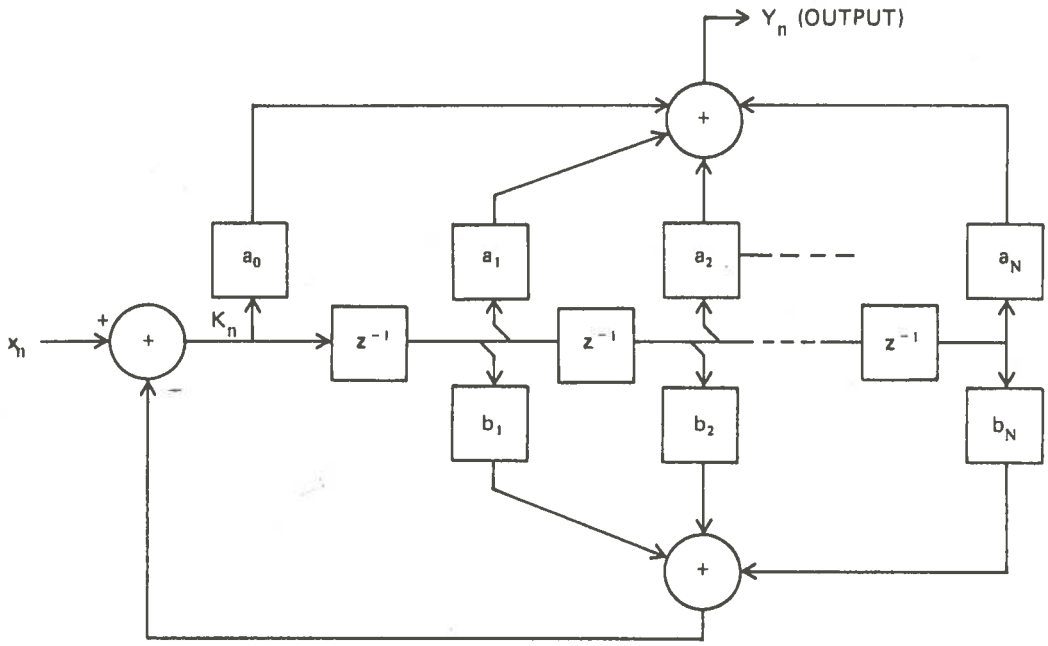


Figure 5 Another digital network version of Fig. 3

It is known that for higher-order difference equations, the direct forms tend to be more inaccurate owing to numerical inaccuracy. The other two forms, which are slightly better than the direct forms in terms of numerical errors, are shown in Fig. 6 and Fig. 7. Their transfer functions are of the form, respectively,

$$H(z) = H_1(z) \cdot H_2(z) \ldots H_i(z) \tag{2.37}$$

$$H(z) = H_1(z) + H_2(z) + \ldots + H_i(z) \tag{2.38}$$

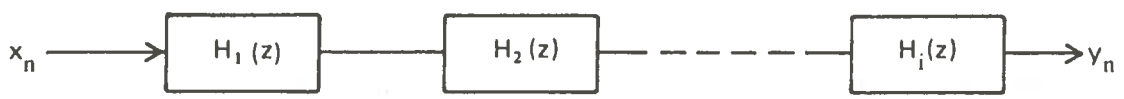


Figure 6 The graphical representation of equation 2.37

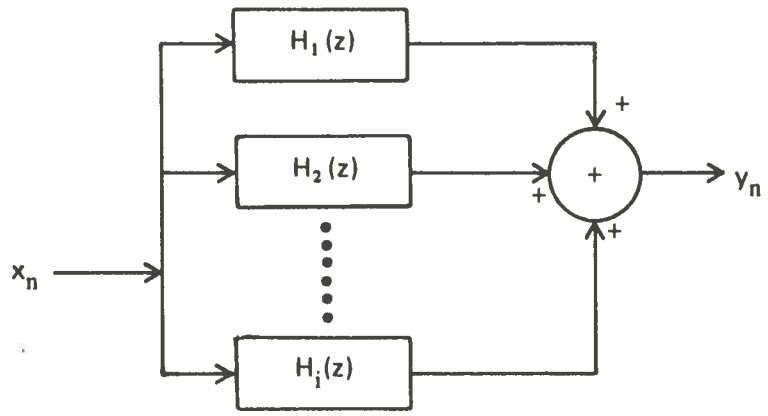


Figure 7 Graphical representation of equation 2.38

2.05 Recursive and Non-Recursive Filters

Digital filters may be synthesized by using direct convolution, linear recursive equations, or fast Fourier transforms. The mathematical methods used to design digital filters depend strongly on whether the filter impulse response is of finite duration or of infinite duration. In the case of finite-duration impulse response, the filter has only zeros and no poles, and in the case of infinite-duration impulse response, it has both zeros and poles. Generally, the term recursive implies that the computation of the output is an explicit function of previous outputs and inputs, whereas non-recursive implies that the output is an explicit function of previous inputs only. A recursive filter is one that has infinite-duration impulse response and a non-recursive filter is one with finite-duration impulse response. Classification by duration of impulse response may not necessarily be valid in all cases. It has been shown that any finite-duration impulse-response filter can be synthesized by recursive techniques [26, 48]. A finite-duration impulse response can be synthesized by fast Fourier transform techniques.

Recursive filtering can increase the speed of computation by an order of magnitude compared to non-recursive filtering. For example, if we need 100 samples to achieve good frequency shaping, we have to carry out 100 multiplication and addition operations to obtain a single output using a non-recursive filter, whereas with the recursive filtering we may require only 10 multiplication and addition operations to achieve comparable frequency shaping. However, it has been shown that, with the development of FFT algorithms, for some applications non-recursive filtering is quite competitive with recursive filtering [55, 61, 66].

3.00 Classical Numerical Approach to Digital Filter Design

General methods for designing a variety of filters of specified transfer function have been developed by M.A. Martin [49, 50]. The performance of several classical methods applied to sampled data was also evaluated by him. We will examine a few design methods in this section.

Digital filtering can be considered as a branch of numerical analysis from an electrical engineer's point of view. Basically, we can conceive of a digital filter as a set of numerical weights that are applied to the incoming signal which is assumed to be in a sampled form. The values of these weights and the number of them required will depend on the filtering requirements. The filtering operation consists in multiplying the input values by the corresponding weights and summing. Each such operation yields one output and the process is repeated for each succeeding output. We can refer to this as a numerical evaluation. As can be seen, we have to perform a large number of multiplications and additions, which could be done in a general purpose computer.

Let the filter weights be represented by h_n, the input data values by x_n and the output values y_n. Then we have the relationship

$$y_k = \sum_{n=N_1}^{N_2} h_n x_{k-n} \tag{3.01}$$

where the number of filter weights is $(N_2 - N_1 + 1)$. The digital filter is the set of $(N_2 - N_1 + 1)$ fixed weights $h_{N_1}, h_{N_1+1}, \ldots, h_n, \ldots, h_{N_2}$. To filter a set of sampled data, (3.01) is repeated by moving the set of weights along the sampled data. We can visualize this relationship qualitatively, as in Fig. 8, which illustrates the filter operation with 5 weights [74].

$$\begin{array}{ccccccccccc} X_1 & & X_2 & X_3 & X_4 & X_5 & X_6 & X_7 & X_8 & X_9 & X_{10} \\ & & \underbrace{h_2 \quad h_1 \quad h_0 \quad h_{-1} \quad h_{-2}} & & & & & & & & \end{array}$$

$$y_4 = h_2 X_2 + h_1 X_3 + h_0 X_4 + h_{-1} X_5 + h_{-2} X_6 \tag{3.02}$$

Figure 8 Filtering operation with five weights

In Fig. 8, we have computed the output y_4. To compute the output y_5, the filter is shifted one unit to the right and multiplication and summation performed as before. For an analog filter, the output is equal to the input convolved with its impulse response function and can be written as

$$y(t) = h(t) \star x(t) \tag{3.03}$$

where $\star$ represents convolution.

Comparing the analog and digital version, we can write

$$y_k = \sum_{m=0}^{M} h_m x_{k-m} \quad \text{. . (digital)} \tag{3.04}$$

$$y(\tau) = \int_0^{\infty} h(t)\, x(\tau - t)\, dt \quad \text{(analog)} \tag{3.05}$$

Extending this similarity to the frequency domain, for the analog case we have

$$Y(\omega) = H(\omega)\, X(\omega) \ , \tag{3.06}$$

which means that the Fourier transform of the output is equal to the product of the Fourier transform of the input and the system transfer function. Extending this to the sampled representation, we can write

$$Y_N(\omega) = H_N(\omega)\, X_N(\omega) \tag{3.07}$$

where the subscript N represents the sampled value of the function. To convert a continuous representation to a sampled representation, the function is multiplied by an infinite sequence of equally spaced delta functions. For instance, one way of computing the weights of a digital filter is to multiply the impulse response of the equivalent analog filter $h(t)$ by the impulse train $\sum_n \delta(t - nT)$,

i.e.,
$$h_N(t) = h(t) \sum_n \delta(t - nT) \tag{3.08}$$

where T is the spacing between the impulses. From the above equation, we derive the transfer function of a digital filter. Taking Fourier transforms, we have the digital filter transfer function as

$$H_N(\omega) = \sum_n h(nT)\, e^{-jn\omega T} \tag{3.09}$$

where $h(nT)$ represents the weights.

Equation 3.09 is a periodic function with period $2\pi/T$.

If we design the basic low-pass filter, the other filters like high-pass, band-pass, and band-elimination could be easily derived from the basic low-pass version. For example, we can derive a band-pass filter by subtracting the weights of two low-pass filters with different cutoff frequencies. Similarly, the high-pass filter is derived by subtracting the weights of a low-pass filter from the all-pass filter. The digital all-pass filter has simply one weight equal to unity and all the others are zero. It is also possible to perform integration and differentiation by digital techniques as presented by Martin. Martin has also illustrated the different techniques used to compute the weights of a digital filter.

Considering a low-pass central filter (for a central filter $N_2 = N_1 = N$), we have the transfer function given by

$$H_N(\omega) = \sum_{n=-N}^{N} h_n e^{-jn\omega T} \tag{3.10}$$

and hence we have now [- (-N) + N + 1] weights, i.e., (2N + 1) weights. For a linear phase we have

$$h_{-n} = h_n \tag{3.11}$$

Therefore,

$$H_N(\omega) = h_0 + 2 \sum_{n=1}^{N} h_n \cos(n\omega T) \tag{3.12}$$

which is the transfer function of a low-pass filter. It is to be noted here that since $H_N(\omega)$ is a real function of ω, the phase shift introduced is zero for the case when $H_N(\omega)$ is positive, and $\pm\pi$ when $H_N(\omega)$ is negative.

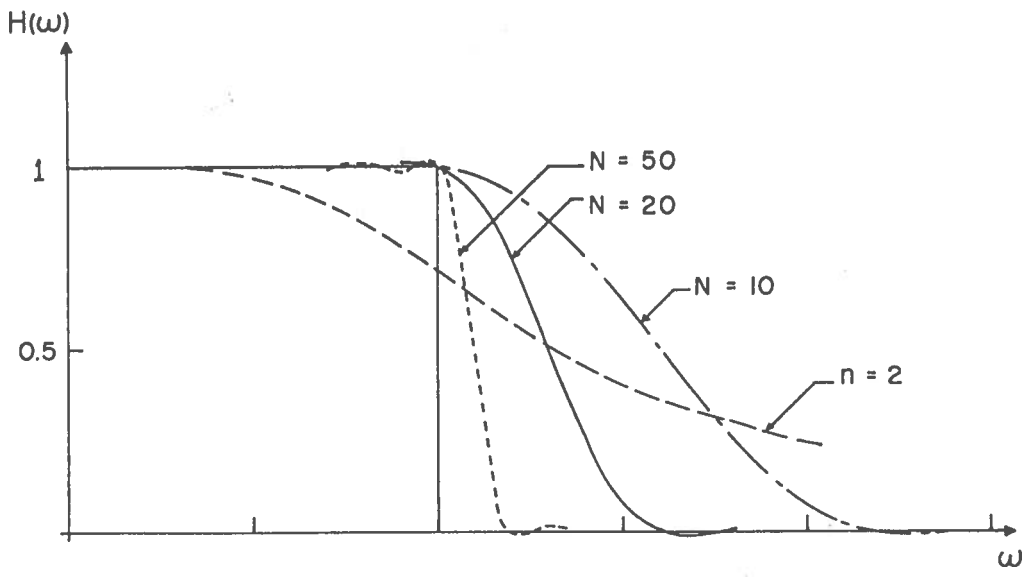


Figure 9 Sine-terminated digital filters

Figure 9 shows the transfer functions of some low-pass filters designed by Martin for N = 10, 20, 50.

4.00 Frequency Domain Synthesis of Digital Filters

4.01 Synthesis of Digital Filters

Design of analog filters involves finding a practical transfer function that approximates closely the ideal filter response. For example, Fig. 10 shows the amplitude characteristic of an ideal low-pass filter, and Fig. 11 the practically realizable filter characteristic (Butterworth type).

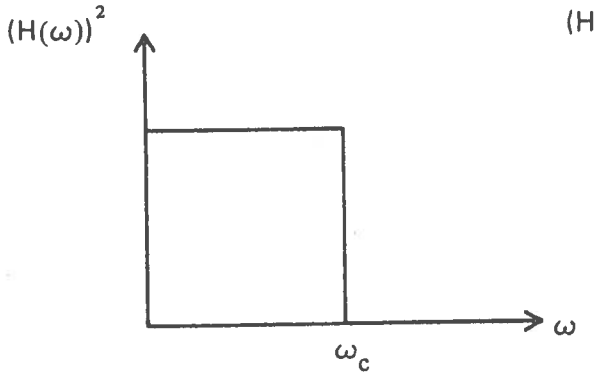


Figure 10 Ideal low-pass filters

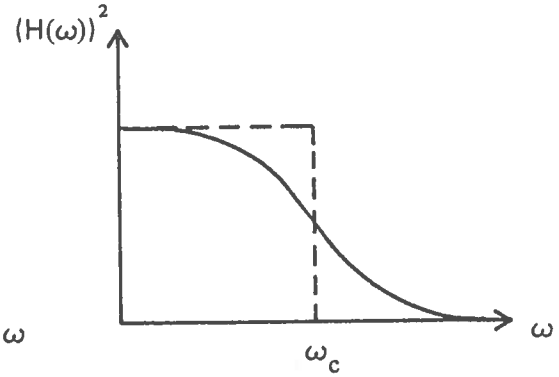


Figure 11 Butterworth filter response

It is evident that the response of Fig. 10 is not realizable using passive components and hence the approximation of Fig. 11.

Once the transfer function is chosen we have to realize the transfer function using real components. One of the important problems in synthesis is to isolate each pair of complex poles and zeros and this is generally achieved using operational amplifiers. In digital filter analysis we do not need to worry about this problem, owing to the inherent isolation of digital logic elements. In view of the fact that analog filter design techniques are very well established, the digital filter design problem is often converted into analog form. A number of such techniques are available now, namely, the bilinear transformation, impulse invariance, frequency sampling, etc. In this section we will examine some of the aspects of the bilinear technique which is fairly straightforward.

4.02 Bilinear Transformation Technique

The sampling theorem dictates that we sample at least twice the rate of the highest signal frequency to be able to recover the signal completely. It is also fairly well known that sampling introduces an aliasing effect in frequency and this occurs at the folding frequency, also called the Nyquist frequency. Moreover, the folding frequency f_f is equal to one-half the sampling frequency, i.e., $f_f = \frac{1}{2T}$, T being the sampling period. Hence

in a sampled data system, all frequencies above the folding frequency f_f will be folded back and appear to be in the range 0 to f_f. The aliasing effect that is caused by the sampling limits the useful frequency range of a digital system to $0 \leqslant \omega \leqslant \pi/T$, whereas in the analog case the range extends to $0 \leqslant \omega \leqslant \infty$.

The linear transformation technique for the design of a digital filter involves mapping the fundamental frequency range of a digital system into a pseudo-frequency ω_p having a range $0 \leqslant \omega_p \leqslant \infty$, which is similar to the analog system and facilitates the use of transfer functions of analog filters for the digital filter design.

The bilinear transformation is defined by

$$s = \frac{z - 1}{z + 1} \tag{4.01}$$

where s is the Laplacian operator and z is the discrete operator defined by

$$z = e^{j\omega T} \tag{4.02}$$

T being the sampling period.

Any exponent raised to the operator z indicates the number of sampling periods a signal is delayed. z^{-n} for example indicates nT seconds delay. From (4.01) and (4.02) we can write

$$s = \frac{e^{j\omega T} - 1}{e^{j\omega T} + 1} = \left[\frac{e^{\frac{j\omega T}{2}} - e^{\frac{-j\omega T}{2}}}{e^{\frac{j\omega T}{2}} + e^{\frac{-j\omega T}{2}}} \right] \left[\frac{e^{\frac{j\omega T}{2}}}{e^{\frac{j\omega T}{2}}} \right] \tag{4.03}$$

$$= j \frac{\sin \frac{\omega T}{2}}{\cos \frac{\omega T}{2}} = j\omega_p \tag{4.04}$$

where

$$\omega_p = \tan \frac{\omega T}{2} \tag{4.05}$$

Hence the transformation $j\omega_p$ maps the fundamental frequency range $0 \leqslant \omega \leqslant \pi/T$ into the infinite pseudo-frequency range $0 \leqslant \omega_p \leqslant \infty$. This effect is shown in Fig. 12.

As we can see, the bilinear transformation is a powerful tool since it permits the use of Bode-plot techniques and continuous transfer-function methods to write difference equations as applied to a continuous linear filter.

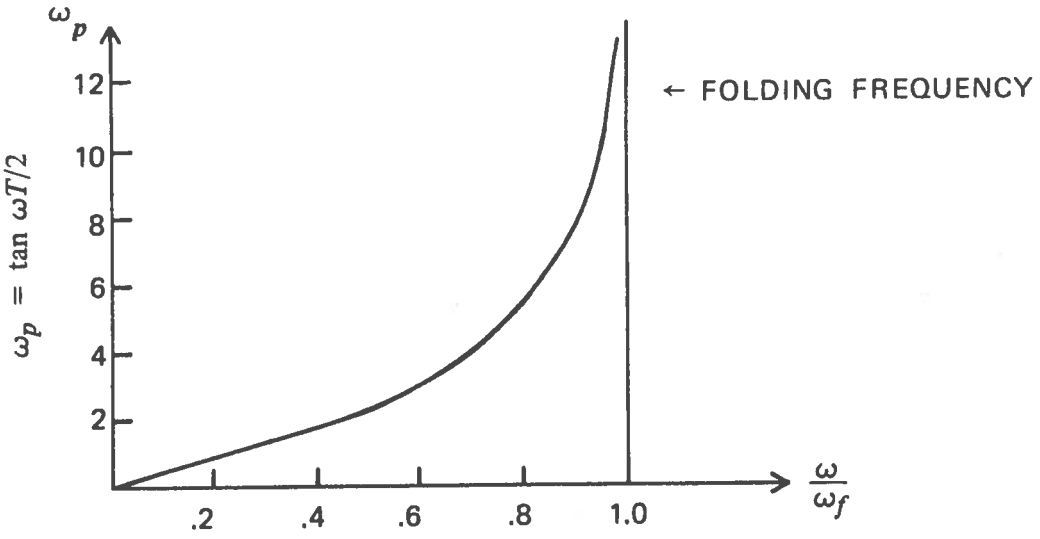


Figure 12 Digital filter frequency transformation

In the bilinear transformation, the transfer function of a continuous signal $G(s)$ is converted into a transfer function of a discrete sampled signal $G(z)$. $G(z)$ is then in turn converted into the difference equation form.

If, for example,
$$G(z) = \frac{X(z)}{Y(z)} = \frac{z+1}{z-1} = \frac{1+z^{-1}}{1-z^{-1}} \tag{4.06}$$

where $X(z)$ is a filter input and $Y(z)$ filter output, then from (4.06)

$$X(z)\ (1 - z^{-1}) = Y(z)\ (1 + z^{-1}) \tag{4.07}$$

If x_n and y_n are the filter input and output at time n, we can write,

$$x_n(1 - z^{-1}) = y_n(1 + z^{-1}) \tag{4.08}$$

Representing
$$x_n z^{-1} = x_{n-1} \quad \text{and} \quad y_n z^{-1} = y_{n-1} \tag{4.09}$$

$$x_n = y_n + y_{n-1} + x_{n-1} \tag{4.10}$$

which we identify as the difference equation obtained from the transfer function $G(z)$. A computer program could be written for the difference equation (4.10).

The following procedure indicates the digital filter design by the bilinear transformation technique.

1. Specify the filter cutoff frequency ω_c required in the range $0 \leqslant \omega_c \leqslant \frac{\pi}{T}$.

2. Convert this frequency into the pseudo-frequency ω_p using the relationship $\omega_p = \tan \frac{\omega_c T}{2}$.

3. Design a transfer function $G(s)$ with the properties of the digital filter at the new frequencies and ranges. There is no need to synthesize $G(s)$.

4. Replace s by $\frac{z-1}{z+1}$ and express the transfer function in the form of a difference equation which yields the desired digital filter.

As an example, let us consider the design of a low-pass digital filter with the following specification.

Sampling rate	=	2 kHz
Corner frequency	=	100 Hz (3 db point)
Roll-off rate	=	10 db down at 200 Hz and over.

The filter has to be monotonic in pass-band and stop-band. As we know, in the analog domain, this calls for a Butterworth type filter.

The critical frequencies are $\omega_{f_1} T = 2\pi \cdot \frac{100}{2000}$

$$\omega_{f_2} T = 2\pi \cdot \frac{200}{2000}$$

$$\omega_p = \tan \frac{\omega_f T}{2}$$

$$\omega_{p_1} = \tan \frac{2\pi \times 100}{2 \times 2000} = \tan 9^\circ = 0.158$$

$$\omega_{p_2} = \tan \frac{2\pi \times 200}{2 \times 2000} = \tan 18^\circ = 0.325.$$

Design a Butterworth filter with the corner frequency

$$\omega_c = 0.158, \text{ and } \frac{\omega_{p_2}}{\omega_{p_1}} = \frac{0.325}{0.158} = 2.111.$$

Hence $$1 + (2.111)^{2n} = 10$$

$$\therefore n = 2 \text{ (2nd order filter).}$$

A second-order Butterworth filter with $\omega_c = 0.158$ has poles at

$$S = 0.158 (-0.707 \pm j0.707)$$

$$= -0.108 \pm j0.108 \text{ and no zeros.}$$

Therefore we can write the transfer function as

$$G(s) = \frac{s_1 s_2}{(s + s_1)(s + s_2)}$$

$$= \frac{2\,(0.108)^2}{(s + 0.108)^2 + (0.108)^2}$$

$$= \frac{0.0234}{s^2 + 0.216s + 0.0234}$$

Substituting $s = \dfrac{z - 1}{z + 1}$,

$$G(z) = \frac{0.0234}{\dfrac{(z-1)^2}{(z+1)^2} + 0.216\,\dfrac{(z-1)}{(z+1)} + 0.0234}$$

$$= \frac{0.0234\,(z^2 + 2z + 1)}{(z - 1)^2 + 0.216\,(z^2 - 1) + 0.0234\,(z + 1)^2}$$

$$= \frac{0.0234\,(z^2 + 2z + 1)}{1.2394z^2 - 1.9532z + 0.8074}$$

$$\therefore \quad \frac{Y(z)}{X(z)} = \frac{0.0234\,(z^{-2} + 2z^{-1} + 1)}{0.8074z^{-2} - 1.9532z^{-1} + 1.2394}$$

Normalizing the above equation, we obtain the difference equation

$$y_n = 0.0189x_{n-2} + 0.0378x_{n-1} + 0.0189x_n - 0.652y_{n-2} + 1.58y_{n-1}$$

which is the digital filter for the given specification.

5.00 Fast Fourier Transform Technique

5.01 The Fast-Fourier Transform

The fast Fourier transform, first reported by Cooley and Tukey, is a powerful tool for computing the coefficients of the discrete Fourier transform (DFT). It is now widely used in signal analysis, power spectrum analysis and simulation of filter characteristics [7, 8]. The solutions to many of the above problems are now obtainable more economically than by using conventional techniques.

The discrete Fourier transform is a powerful, reversible, mapping operation for time series. The usefulness of the discrete Fourier transform arises from the fact that it has properties analogous to the Fourier integral transform. The DFT essentially defines a spectrum of a time series.

In order to analyze a continuous signal by means of a digital system, we generally sample the continuous signal and produce a time series of discrete samples which are fed to the digital system. It is well known that when the sampling is done according to the sampling theorem, the discrete time series represents the continuous signal completely, taking into account the fact that the signal is band limited. The discrete Fourier transform of such a series, the samples being taken at equal intervals, is closely related to the Fourier transform of a continuous signal. This property makes the DFT a very efficient tool in power spectrum analysis and filter simulation in digital computers.

The FFT is an efficient algorithm for computing the coefficients of the DFT of a time series and is carried out iteratively, saving a considerable amount of computing time. For example, if the time series consists of $N = 2^n$ samples, then only about $2nN = 2N \log_2 N$ arithmetic operations are required to evaluate all N associated DFT coefficients, whereas with the conventional straightforward method, N^2 operations are required. As can easily be seen there is a tremendous saving in computation time, especially when N is large. As an illustration, it has been reported that for $N = 8{,}192$ samples, the computation by FFT requires 5 seconds to compute all the DFT coefficients on an IBM computer, whereas the conventional method takes as long as half an hour. By virtue of the reduction in computation time, the FFT finds extensive application in the following areas:

1. Filter simulation
2. Computation of power spectra and autocorrelation functions of sampled data
3. Pattern recognition
4. Computation of cross-variance functions
5. Decomposing convolved functions.

5.02 The Discrete Fourier Transform

Let us consider a time function $f(t)$ represented by the sequence of N samples $f(nT)$, $0 \leqslant n \leqslant N - 1$, where T is the sampling period in the time domain. Let $F(k\nu)$ represent the spectrum $F(\omega)$, $0 \leqslant k \leqslant N - 1$, where ν is the chosen increment between samples in the frequency domain. Taking the discrete form we have

$$F^*(k\nu) = \sum_{n=0}^{N-1} f(nT)\, e^{-j\nu Tkn} \tag{5.01}$$

The DFT is defined by

$$F_k = \sum_{n=0}^{N-1} f_n\, e^{-j\nu Tnk} \tag{5.02}$$

where F_k is the kth coefficient of the DFT, f_n denotes the nth sample of the time series which consists of N samples. The f_n's and $\nu = \frac{2\pi}{NT}$ can be complex numbers and the F_k's are almost always complex.

Rewriting (5.02)

$$F_k = \sum_{n=0}^{N-1} f_n W^{nk}, \qquad k = 0, 1, \ldots . N - 1 \tag{5.03}$$

where $W = e^{-(2\pi j/N)}$.

The discrete Fourier transform is also called the discrete-time finite-range Fourier transform. The inverse of the DFT does exist, is very similar to the DFT, and can be computed by the fast Fourier transform methods. The inverse of (5.02) is

$$f_\ell = (\frac{1}{N}) \sum_{n=0}^{N-1} F_k W^{-k\ell}, \quad \ell = 0, 1, \ldots . N - 1 \tag{5.04}$$

We also have the relationship

$$F_k = F_{N+k} = F_{2N+k} = \ldots . \tag{5.05}$$

and similarly,

$$f_\ell = f_{N+\ell} = f_{2N+\ell} = \ldots . \tag{5.06}$$

The DFT has properties that are quite similar to those of the Fourier integral transform. One of the important properties of the DFT is the convolution relationship. That is, the inverse DFT of the product of the two DFT's is the periodic mean convolution of the two time series of the DFT's.

5.03 Comparison of Computation Time

As mentioned earlier, the fast Fourier transform is a powerful algorithm for the efficient computation of the coefficients of the DFT of a time series. The FFT reduces the computation time considerably. Table I compares the computation time between the direct and the FFT methods for a few operations.

The FFT not only reduces the computation time, but also reduces round-off errors associated with these computations. As seen earlier, the computation time and round-off errors are reduced by a factor of $\frac{\log_2 N}{N}$. The FFT is a computation technique of sequentially combining progressively larger weighted sums of data samples so as to produce the DFT coefficients defined by

$$F_k = \sum_{n=0}^{N-1} f_n W^{nk} .$$

TABLE I

Comparison of the number of multiplications required using direct and FFT methods

Operation	Formula	Approximate Number of Multiplications	
		Direct	FFT
Discrete Fourier transform (DFT)	$\sum_{k=0}^{N-1} X_k e^{-2\pi jrk/N}$ where $r = 1, 2, \ldots . N-1$	N^2	$2N \log_2 N$
Filtering (convolution)	$\sum_{k=0}^{N-1} X_k Y_{u-k}$ where $u = 0, 1, \ldots . N-1$	N^2	$3N \log_2 N$
Autocorrelation functions	$\sum_{k=0}^{N-1-r} X_k X_{r+k}$ where $r = 0, 1, \ldots . N-1$	$\frac{N}{4}(\frac{N}{2}+3)$	$3N \log_2 N$
Two-dimensional Fourier transform (pattern analysis)	$\sum_{k=0}^{N-1}\sum_{\ell=0}^{N-1} X_{k,\ell} e^{-2\pi j[kq+(r/N)]}$ where $q = 0, 1, \ldots . N-1$	N^4	$4N^2 \log_2 N$
Two-dimensional filtering	$\sum_{k=0}^{N-1}\sum_{\ell=0}^{N-1} X_{k,\ell} Y_{q-k,\, r-\ell}$ where $q,r = 1, 2, \ldots . N-1$	N^4	$3N^2 \log_2 N$

There are two types of fast Fourier transform algorithms, namely, decimation in time and decimation in frequency, and each has several modifications [20].

5.04 Decimation in Time

As we have seen earlier the DFT and its inverse are of the same form and hence the same procedure could be used for computing either of them just by exchanging the roles of f_ℓ and F_k and applying proper scale factors and sign changes. The decimation in time was first used by Cooley and Tukey. The decimation in frequency is obtained by changing the role of f_ℓ and F_k. We will first consider the decimation in time.

Suppose we have a time series of N samples and divide them into two functions, g_n consisting of the even-numbered samples and h_n consisting of odd-numbered samples. Each function now has $N/2$ samples – the even-numbered function has $g_0, g_2, g_4 \ldots .$ and the odd-numbered function has $h_1, h_3, h_5 \ldots . .$ We can write

$$g_n = f_{2n} \tag{5.07}$$

$$h_n = f_{2n+1} \qquad n = 0, 1, \ldots ., \frac{N}{2} - 1 \tag{5.08}$$

This situation is illustrated in Fig. 13.

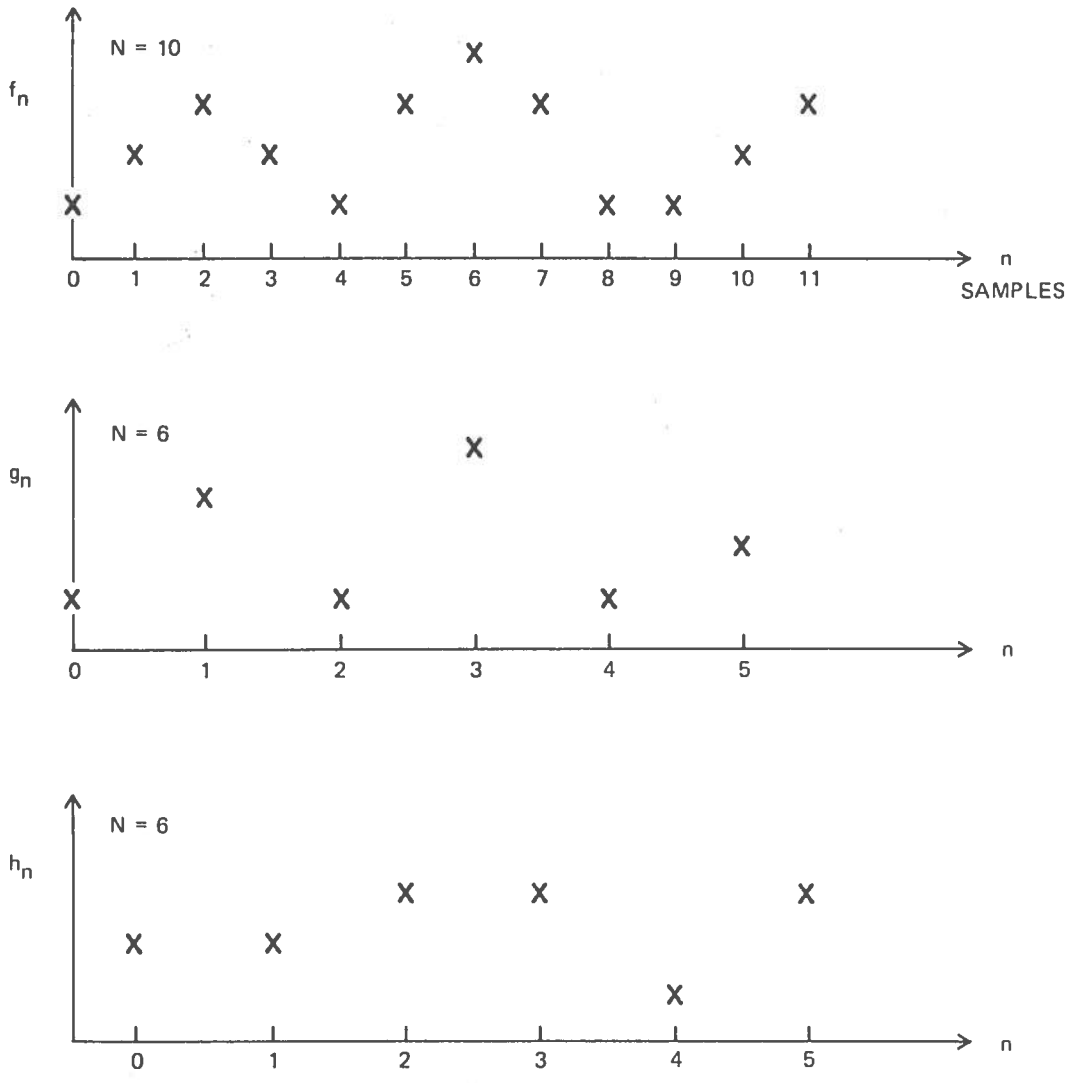


Figure 13 Decomposition of time-series into odd and even-numbered samples

Since g_n and h_n are sequences of $\frac{N}{2}$ samples each, they have discrete Fourier transforms defined by

$$G_k = \sum_{n=0}^{\frac{N}{2}-1} g_n (W^2)^{nk} \tag{5.09}$$

$$H_k = \sum_{n=0}^{\frac{N}{2}-1} h_n (W^2)^{nk} \tag{5.10}$$

where $k = 0, 1, 2, \ldots, \frac{N}{2} - 1$.

Writing the discrete Fourier transform of the sequence F_k in terms of the odd and even numbered sequences,

$$F_k = \sum_{n=0}^{\frac{N}{2}-1} (g_n W^{2nk} + h_n W^{(2n+1)k}) \tag{5.11}$$

or

$$F_k = \sum_{n=0}^{\frac{N}{2}-1} g_n (W^2)^{nk} + W^k \sum_{n=0}^{\frac{N}{2}-1} h_n (W^2)^{nk} \tag{5.12}$$

$$= G_k + W^k H_k , \qquad 0 \leqslant k \leqslant \frac{N}{2} \tag{5.13}$$

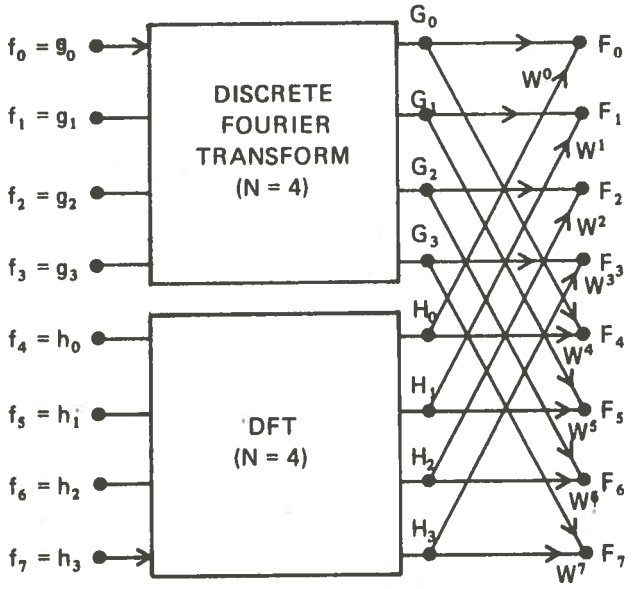
The implication of equation (5.13) in terms of computational speed is worth noting at this point. The computation of G_k and H_k by the 'direct' method requires $(\frac{N}{2})^2$ operations each and an additional N operations are required to combine them to give F_k. Thus the total number of computational operations required is $N + N^2/2$. 'Direct' computation of F_k would have required N^2 operations and hence FFT reduces the computation by a factor of 2 for large N.

For values greater than $N/2$, the discrete Fourier transforms G_k and H_k repeat periodically the values obtained when $k < N/2$. Hence substituting $k + \frac{N}{2}$ for k in equation (5.13),

$$F_{k+\frac{N}{2}} = G_k + W^{(k+\frac{N}{2})} H_k , \qquad 0 \leqslant k \leqslant N/2 \tag{5.14}$$

$$= G_k - W^k H_k, \qquad 0 \leqslant k \leqslant N/2 \tag{5.15}$$

From equations (5.13) and (5.15), the first $N/2$ and last $N/2$ samples of the DFT of f_n (a sequence of n samples) can be simply obtained from the DFT of g_n and h_n, both sequences of $N/2$ samples. This situation is dipicted graphically in Fig. 14. Now the computation of G_k (or H_k) in turn can be further reduced to the computation of sequences of $\frac{(N/2)}{2} = \frac{N}{4}$, samples. These reductions can be carried out as long as the number of



$$W^n = -W^{n-(N/2)}$$

therefore $W^5 = -W^{5-(8/2)} = -W^1$

Also, $F_5 = F_{1+4} = G_1 - W^1 H_1$, by equation (4.13)

$= G_1 + W^5 H_1$.

Figure 14 Signal flow graph of decimation in time. Eight point DFT reduced to two DFT's

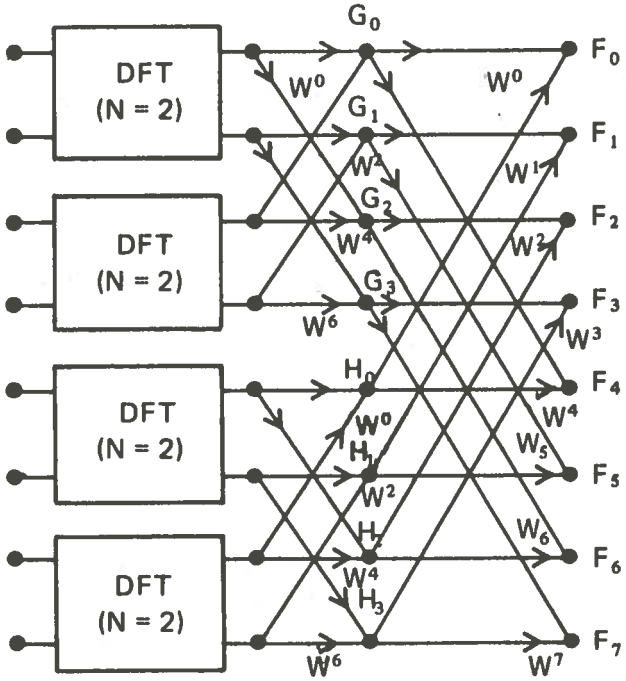


Figure 15 Eight point DFT of Fig. 14 reduced to four two-point DFT's

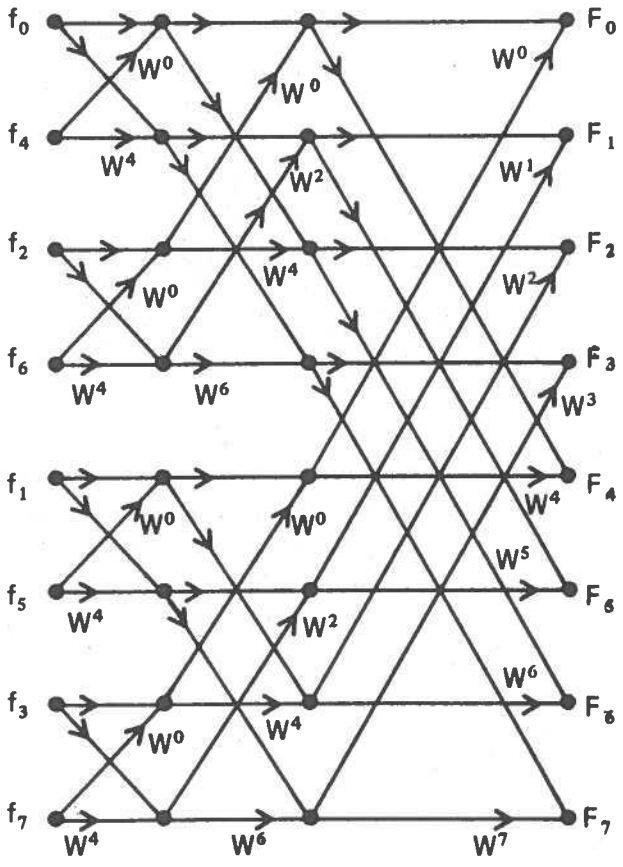


Figure 16 Eight point DFT of Fig. 13 completely reduced to complex multiplication and addition

samples of the function is a multiple of 2. Thus if the function f_n has 2^n samples, n such reductions could be made. Of course the DFT of one point function is the sample itself. These reduction procedures are illustrated in Figs. 15 and 16.

In Fig. 16 the whole operation is reduced to complex multiplications and additions. From the signal flow graph of Fig. 16 we note that there are 8 X 3 terminal nodes corresponding to 24 additions and 2 X 8 X 3 arrows corresponding to 48 multiplications. This amounts to $N \log_2 N + 2N \log_2 N$ operations. Half of the multiplications can be omitted since the transmission indicated by the arrow is unity. Half of the remaining multiplications are also easily eliminated as we will see, by using the fact that $W^{N/2} = -1$. Thus for N samples, a power of 2, we see that we need $N \log_2 N$ additions and, at most, $\frac{1}{2} N \log_2 N$ multiplications for the computation of the DFT of an N point sequence.

If we assume that input data are stored in the computer in the order $f_0, f_4, f_2, f_6, f_1, f_5, f_3, f_7$, the computation of the DFT can be done 'in-place'. To see the iteration process, suppose that each node corresponds to two memory locations (two memory locations are required due to the fact that the quantities are in general complex). Then the left eight

nodes represent memory locations corresponding to the input data in the order specified above. The computation of the DFT is done 'in-place'; that is, by writing all intermediate results over the original data sequence, and then writing the final answer over the intermediate results. The advantage of this 'in-place' computation is that no storage is required beyond that required for the original N complex numbers. The first step in the computation is then to compute the contents of the memory locations corresponding to the left-most eight nodes. Since each pair of input nodes affects only the corresponding nodes, the newly computed value could be stored in the same memory locations from which the input was taken, as they are no longer needed for further computation. The computation is now carried to the next vertical array of nodes to the right and this iteration process is continued till the last pair of nodes.

We can note at this point that the shuffling of the input data shown in Fig. 16 was necessary for the 'in-place' computation. This shuffling is called bit reversal. A number of modifications are suggested by various authors to keep the input and output data in the serial sequence order and still carry out the in-place computation [26]. These are illustrated in Figs. 17 and 18.

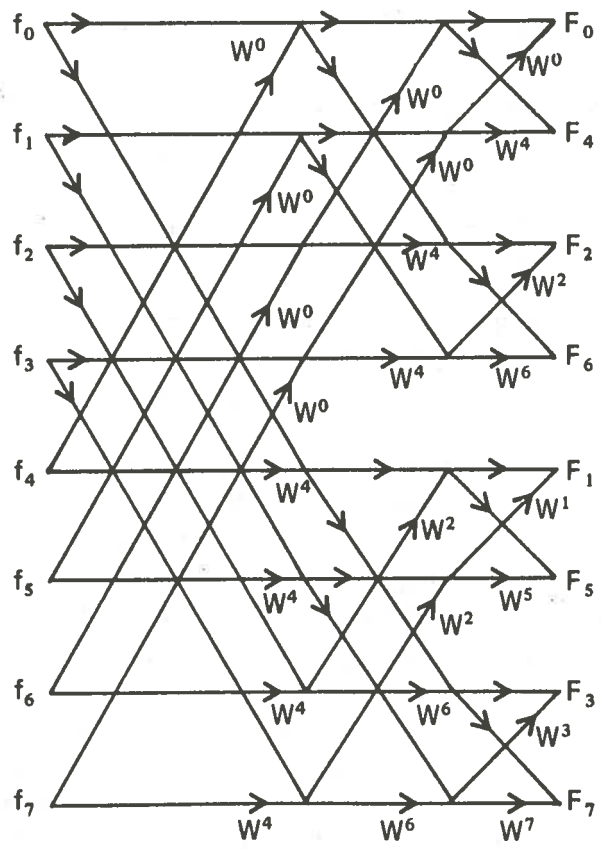


Figure 17 Rearrangement of the flow graph of Fig. 16 gives naturally ordered time samples

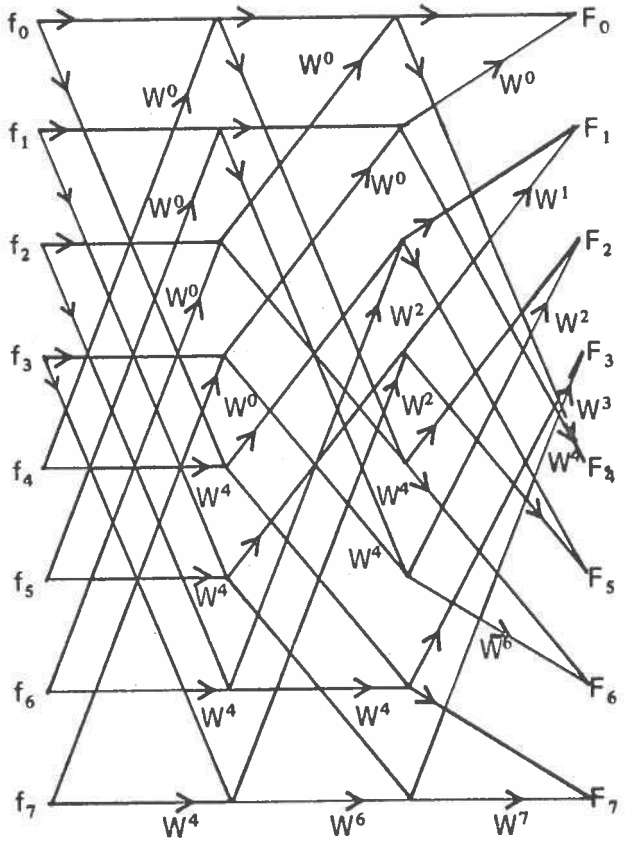


Figure 18 Rearrangement of the flow graph of Fig. 16 gives DFT computation without bit reversal

5.05 Decimation in Frequency

Another quite distinct form of the fast Fourier transform algorithm known as decimation in frequency was found independently by Sande, Cooley, and Stockham [8, 72]. Suppose the time series f_n, having N sampling points, has a DFT F_k and let g_n and h_n represent the two sequences derived from f_n as before. However, in this case g_n consists of the first $N/2$ points in f_n, and h_n the last $N/2$ points in f_n. Then we have

$$g_n = f_n \tag{5.16}$$

$$h_n = f_{n+\frac{N}{2}} \quad , \tag{5.17}$$

where $n = 0, 1, 2, \ldots, \frac{N}{2} - 1$.

Writing the N-point DFT of f_n in terms of g_n and h_n

$$F_k = \sum_{n=0}^{\frac{N}{2}-1} \left[g_n W^{nk} + h_n W^{k(n+\frac{N}{2})} \right] \tag{5.18}$$

$$= \sum_{n=0}^{\frac{N}{2}-1} \left[\left(g_n + e^{-j\pi k} h_n \right) W^{nk} \right] \tag{5.19}$$

We now consider the even- and odd-numbered points of F_k separately (hence the name decimation in frequency). Replacing k by $2k$ and $2k + 1$ in (5.19) we get

$$F_{2k} = \sum_{n=0}^{\frac{N}{2}-1} (g_n + h_n)(W^2)^{nk} \tag{5.20}$$

$$F_{2k+1} = \sum_{n=0}^{\frac{N}{2}-1} \left[(g_n - h_n) W^n \ (W^2)^{nk} \right] \tag{5.21}$$

which we recognize as the $(N/2)$ point DFT's of the functions $(g_n + h_n)$ and $(g_n - h_n) W^n$. From equations (5.20) and (5.21) we can conclude that the DFT of a N-sample sequence f_n may be determined as follows. For even-numbered transform points, it may be computed as an $N/2$ point DFT of a simple combination of the first $N/2$ and last $N/2$ samples of f_n. For odd-numbered transform points it may be computed as another $N/2$ point DFT of a different simple combination of the first and last $N/2$ samples of f_n. This is illustrated in Fig. 19 for an 8-point function. Figures 20 and 21 illustrate the replacement of each of the DFT's indicated in Fig. 19, by two 2-point DFT's and each of the 2-point DFT's by two single-point transforms, these last being equivalency operations.

Figure 21 gives much information about the method of decimation in frequency and allows us to compare it with the decimation in time form. As can be seen, both forms require $\frac{N}{2} \log N$ complex additions, complex subtractions, and complex multiplications. Both computations can be done 'in-place'. It is interesting to note that Fig. 21 has the same geometry as Fig. 17, decimation in time form. The different rearrangements are given in Figs. 22 and 23. It is to be noted here that the arrangement of Fig. 22 needs no bit reversal of input, output, or coefficients but 'in-place' computation cannot be done.

The forms of Figs. 16, 17, 18, 21, 22 and 23 constitute a set of canonic forms of the FFT. We may choose among these forms, to find an algorithm with the 'in-place' computation properties, normally ordered input, normally ordered output, or normally ordered coefficients, but not all at once. If we need 'in-place' computation we have to

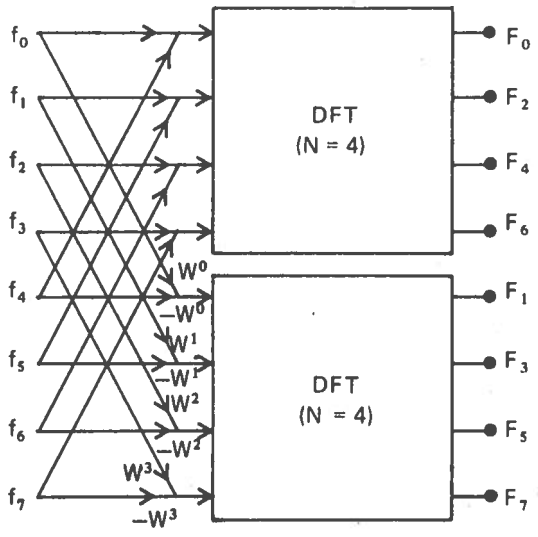


Figure 19 Eight-point DFT reduced to 2 four-point DFT's by decimation in frequency

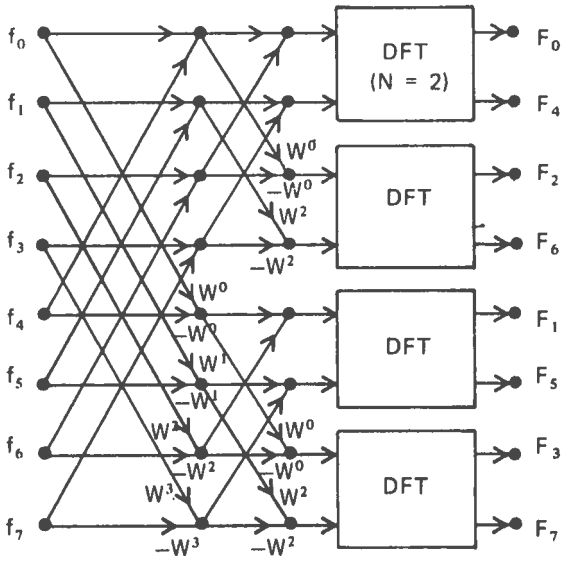


Figure 20 Eight-point DFT further reduced to 4 two-point DFT's

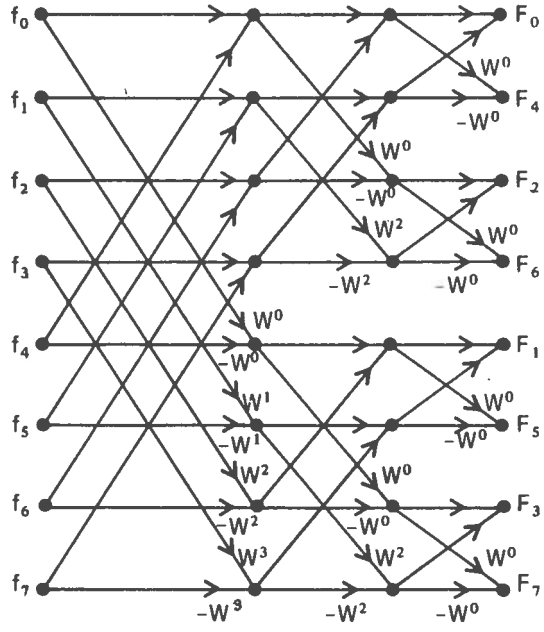


Figure 21 Complete reduction of the eight-point DFT's

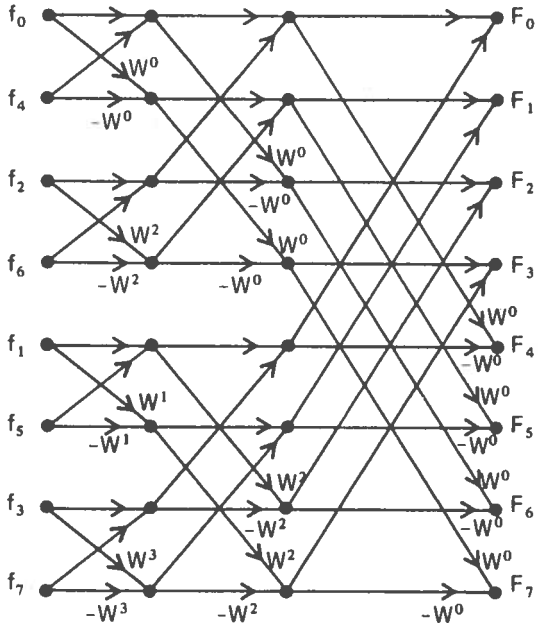


Figure 22 Rearrangement of Fig. 21 with naturally ordered DFT coefficients

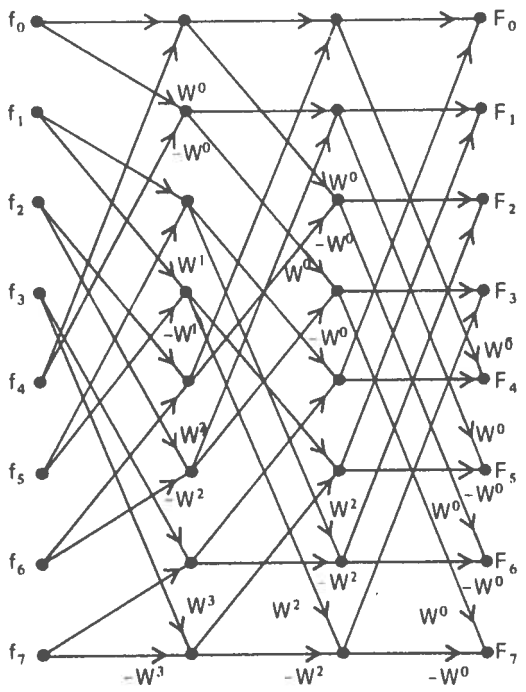


Figure 23 Rearrangement of Fig. 21 to give DFT computation without bit reversal

accommodate bit reversal and to eliminate bit reversal we have to sacrifice 'in-place' computation. All the methods described above are equally useful and the particular method to be adopted depends on the particular problem.

One of the important applications of the FFT is to compute correlations and convolutions [7]. In this application it is necessary to compute a transform and an inverse transform and it is possible to use an algorithm for the inverse transform which accepts bit-reversed inputs. In this way it is possible to avoid bit reversing altogether. Another interesting application of the FFT is interpolation. The FFT has certainly modified the economics of transform methods and more and more interesting and feasible applications are being found [23].

6.00 Digital Hardware Modules and the Impact of LSI

6.01 Hardware Implementation of a Digital Filter

The earlier view that a digital filter is essentially a software oriented device is no longer valid. With the rapid growth of digital technology, the digital filter is becoming more and more hardware oriented, rather than a pure software routine. Due to the revolution in large-scale integration (LSI), the time is rapidly approaching when it will

be economically attractive to carry out on-line digital information processing. The availability of fast computers at very low prices and the concept of time-sharing are also changing the trend towards hardware implementation of digital filters. In the last few years, considerable effort has been spent on the theory and design of digital filters. However, little effort has been devoted to developing hardware building blocks for the implementation of digital filters. Some of the design aspects are illustrated in this section.

The high accuracy and precision demanded by digital filters makes their design more complicated. Once the accuracy of the filter response is specified, the word lengths required to implement this accuracy in the hardware can be determined. The filter design considered in this report will be oriented towards implementation in the form of LSI chips. The choice of the LSI building block will depend to a great extent on multipliers and the input sample points. If we have a basic digital filter building block, it can be combined in different configurations to achieve any desired filter response. This can be thought of as similar to using standard NAND or NOR gates to achieve any desired logic function.

For example, it has been shown that any multiple-pole transfer function can be broken up into 2-pole groupings by using either a partial fraction expansion or straight factoring of the original transfer function. The partial-fraction expansion technique leads to the parallel 2-pole filter representation of Fig. 24 [6] .

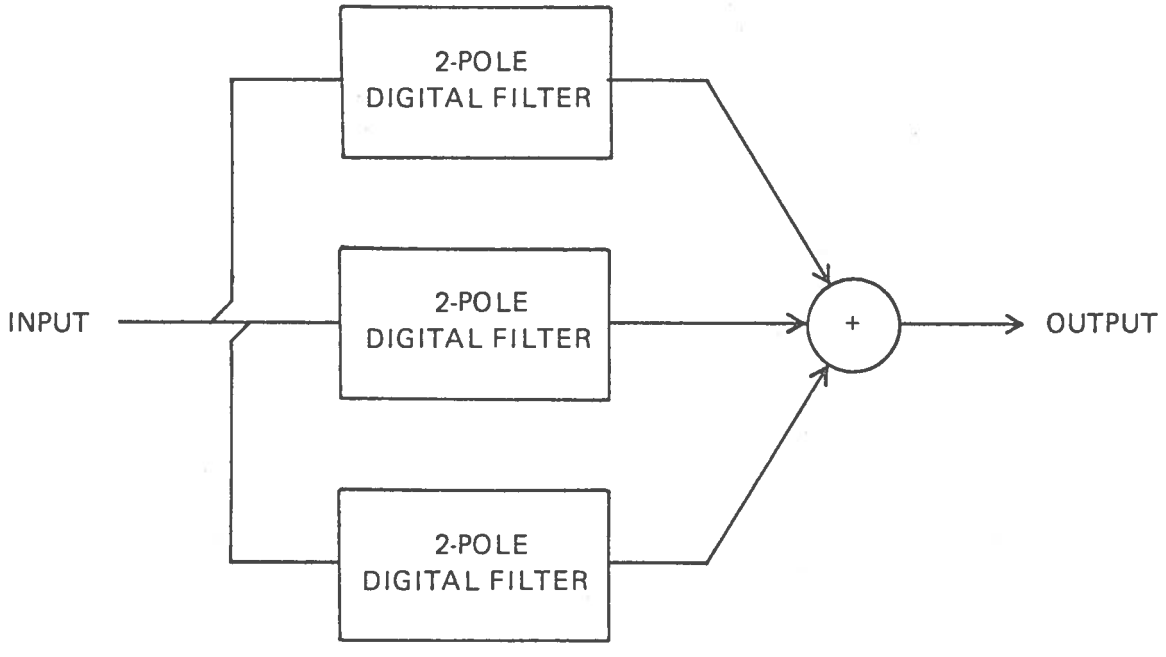


Figure 24 Serial-order parallel digital filter configuration

Figure 25 Sixth-order cascaded digital filter configuration

Figure 25 shows the result of factoring the original transfer function. Since complex filters can be realized from either parallel or cascade connections of the basic 2-pole version, the basic hardware building block should be in terms of the 2-pole filter. The 2-pole filter has to solve the difference equation given by

$$Y(t+2T) = \sum_{k=1}^{2} C_{2,k}\, X\,[t+(2-k)T] - \sum_{k=1}^{4} D_{2,k}\, X\,[t+(3-k)T]$$

where X and Y are input and output amplitudes, respectively, t is the time, T the sampling period, and C and D are constants determined solely by the pole and zero locations of the filter and the sample period.

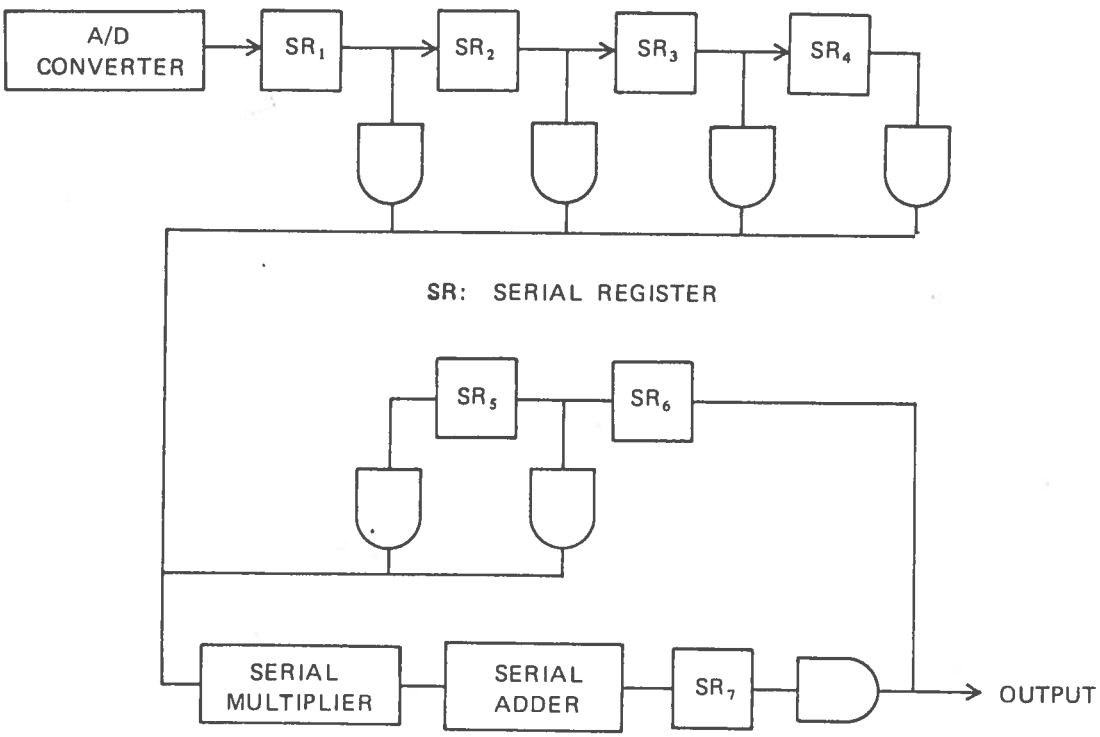


Figure 26 Serial-sequential 2-pole filter block schematic

Figure 26 illustrates the filter hardware configuration of a serial sequential-mode operation. In the serial sequential mode, the hardware design and the useful sampling frequency obtainable from a given configuration depend on the number of bits per sample.

Moreover, because of the serial operation, the internal processing must be faster than the input sampling rate. The bit lengths of the sample registers and multipliers must be carefully chosen. For a given configuration, the maximum achievable frequency is inversely proportional to the bit length of the operands due to the serial operation. It is worth noting here that this configuration requires minimum logic implementation at the expense of speed. It was estimated that with the current P-MOS technology, a maximum filter frequency of 2 kHz could be obtained. Filter frequencies of 110 kHz are feasible using emitter-coupled logic (ECL) technology with this serial configuration.

The block schematic of Fig. 27 illustrates the implementation of a 2-pole filter in the parallel simultaneous mode. The entire operation is done in a parallel mode and the number of bits per input sample influences the hardware design more than the sampling frequency, in contrast to the serial configuration. In this case the hardware design depends on the bit length, but the maximum achievable filter frequency is not affected as much. Even though this configuration requires more logic for implementation, high frequencies are feasible. With the present P-MOS technology, frequencies of about 150–200 kHz are possible, and up to 10 MHz using ECL technology [70]. Presently, the limitation in achieving higher frequencies for digital filter implementation seems to be the maximum conversion speed attainable for the A/D and D/A converters. But it is only reasonable to assume that with the present state of the art we can easily hope to achieve very high speeds for A/D conversion at much lower costs in the near future. Ultrahigh or video speed A/D converters have conversion rates in excess of 100 Mega samples per second, but usually with low resolution and accuracy (usually 6 bits or less). The high conversion rate is obtained by trading off accuracy for speed, that is by using ultrahigh-speed components [10, 67].

Digital filter implementation becomes economically attractive only when one filter configuration system is used to filter at least a few inputs, using the time-sharing technique. Often, not one, but several analog signals must be filtered. The question then is whether to use one filter configuration system for each analog input or one over-all system, time-shared or multiplexed between analog inputs. It is obvious, due to the requirement of large numbers of digital components, that it is not economically practical to do the filtering individually. We can use the time-sharing technique by employing a single A/D converter, multiplexer, and a digital processor as illustrated in Fig. 1 in the introduction. But there are certain problems involved with time-sharing and it is a question of trade-off between economics and accuracy. Every time an analog signal is processed by some circuit, no matter how simple it is, an error is introduced. As this is also the case with time-sharing, it deteriorates the over-all system accuracy to some extent and the effect increases with the number of signals being multiplexed.

Another penalty paid for time-sharing is a reduced conversion rate. When N analog signals are sequentially converted in time, each signal is being converted at a rate of only $1/N$ and, hence, the conversion speed is reduced. For example, if 20 analog inputs are being converted at 10,000 per second, then essentially each input is converted at 500 per second. This is one of the reasons why higher filtering frequencies are not now obtainable, keeping the system cost to reasonable limits.

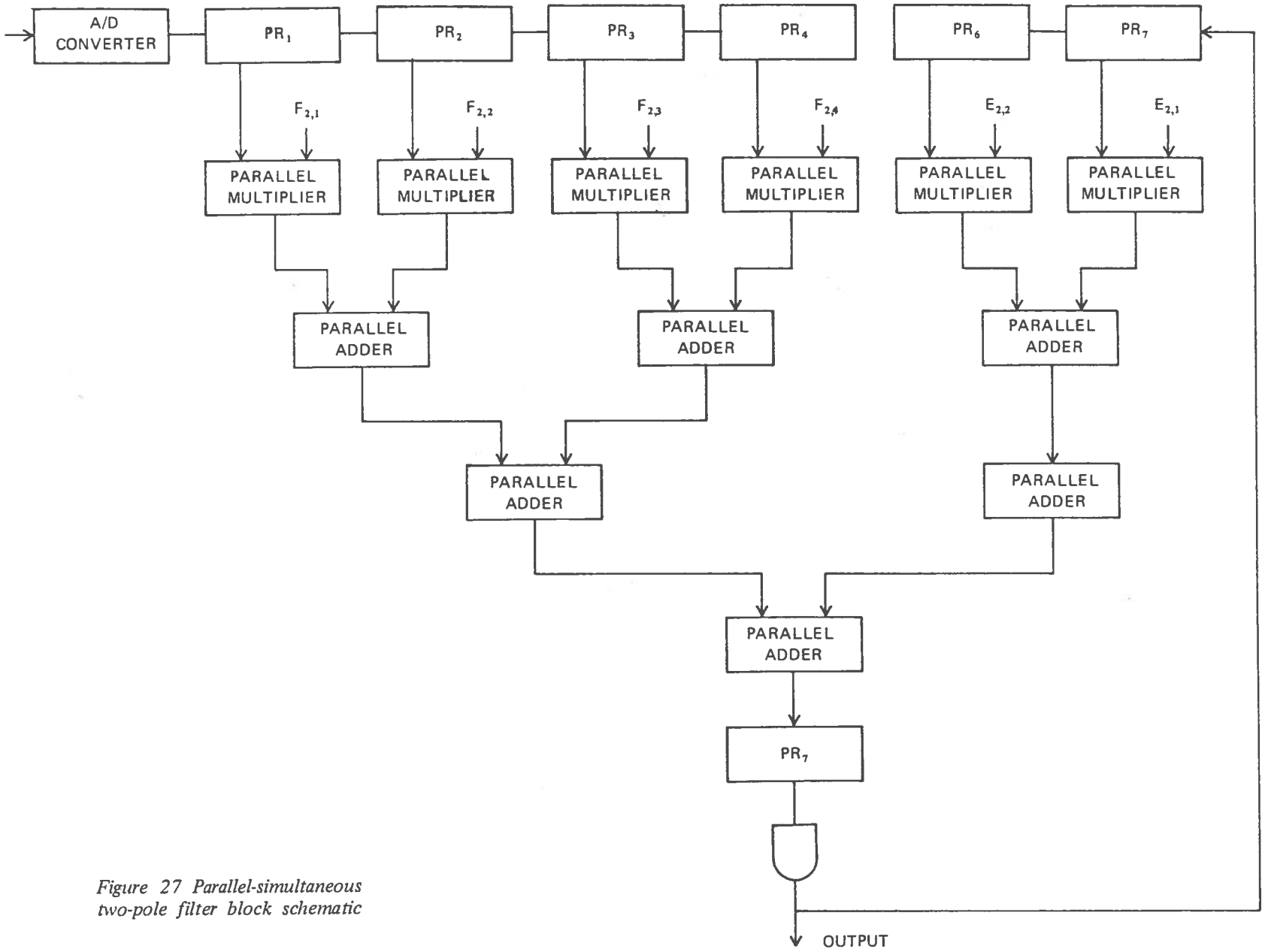
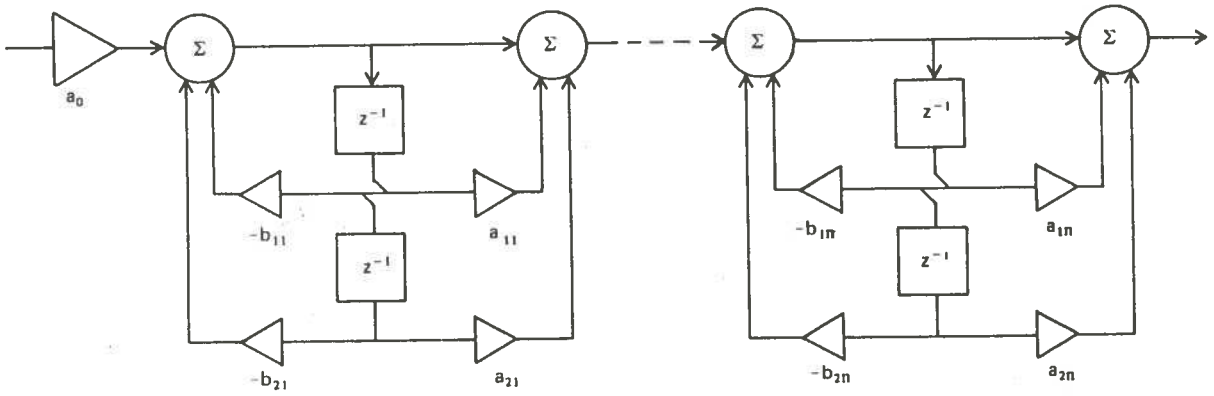


Figure 27 Parallel-simultaneous two-pole filter block schematic

6.02 Impact of LSI Technology and Cost Evaluation

In evaluating the cost of digital hardware for implementing the digital filter response, it is not correct to look at it as a single filter element. It has to be assessed from the point of view of using a single system to filter a number of analog inputs. Moreover, digital filters are economically feasible only when effectively using LSI technology, and hence the evaluation should be based on the use of LSI chips. Let us look at a specific example.

Voice-frequency communications equipment basically consists of filters, amplifiers, and signal processing circuits like modulators, clippers, etc. If LSI technology is to be applied, these functions should be implemented digitally. Of these, filtering is the most important and difficult operation. We have seen in the earlier sections the different configurations and ways of designing filters. One of the promising realizations is the form in which a series of second-order sections are cascaded, as shown in Fig. 28 [51].



$$H^*(z) = a_0 \prod_{i=1}^{n} \frac{a_{2i}z^{-2} + a_{1i}z^{-1} + 1}{b_{2i}z^{-2} + b_{1i}z^{-1} + 1}$$

Figure 28 Cascade realization of digital filter

Each section has four multipliers, four 2-input adders and two units of delay. If time multiplexing is used, one set of adders and multipliers suffices for all the filter sections. Also, since there is no cross-talk involved in digital systems, one arithmetic unit can serve many channels as well as many sections. This is illustrated in Fig. 29. Of course, when sections differ the filter coefficients differ, and hence storage of the coefficient values is imperative.

The implication of using LSI circuits can be appreciated at this point. The large number of shift registers required to store the two samples of signal per filter section and the read-only memories to store the coefficients could be produced more

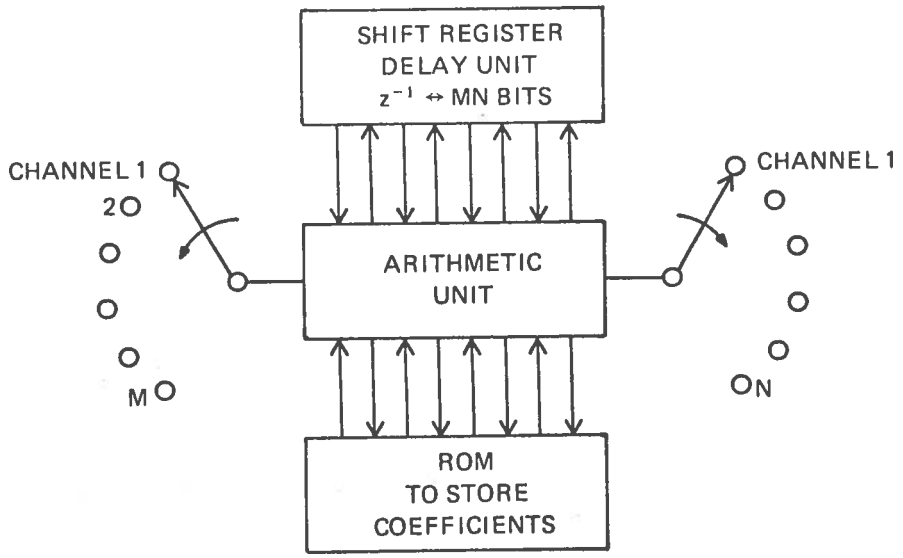


Figure 29 Multiplexing for multiple channels

economically using LSI technology. The number of chips required for a filter with the following specification was estimated by the Bell Telephone Laboratories.

System Specification

1. 12-bit accuracy for the signals
2. 8,000 samples per second signal sampling rate
3. digital filter arithmetic speed 10^7 bits per second
4. 10-bit accuracy in coefficients.

The arithmetic unit for the filter (in general there is only one per system) can be constructed of 20 chips, assuming a circuit density of present commercially available bipolar IC's, and this is medium-scale integration. If we assume 100-bit shift register chips and 200-bit read-only memory chips, then for an application such as 25 channels of identical 8^{th} order filters (4 second-order sections) commonly used in telephone switching communication systems, the number of chips required can be estimated approximately as:

Arithmetic unit	:	20 chips
Delay unit $\frac{2 \times 12 \times 25 \times 4}{100}$	:	24 chips
Coefficient store $\frac{4 \times 10 \times 4}{200}$	:	1 chip
Total		45 chips

Building 25 filters, each of 8th order, out of discrete components cannot compete with 46 chips, when all but one are standard and one custom chip is a read-only memory. As stated earlier, in applications where many channels, many sections and low frequencies are involved, digital filters seem to be more economical at present.

TABLE II

The present state of the art of MSI/LSI devices*

Company	Number of different types of MSI/LSI devices available off-the-shelf:		MOST COMPLEX OFF-THE-SHELF MSI/LSI DEVICE						
			Catalog #	Description	Number of Pins or Leads	Selling Price	Approximately equivalent to		
	MSI	LSI					Gates	Discrete Components	ICs
FAIRCHILD	16	–	9328	Dual 8 bit shift register	16	$15.40 (100+)	105	–	–
GENERAL INSTRUMENT	34	10	ROM2048	2048 bit ROM	24	$90 (100+) $20 (20,000+)	–	2750	15
HUGHES AIRCRAFT	23	20	HDSR4064	Quad 64 bit dynamic shift register	10	$25 (1000+)	–	1200	–
INTEL INC.	0	3	i-1101	256 bit RAM	16	$65 (100+)	50	1850	–
MOSTEK	–	6	MK4003P	Synchronous buffer memory (32-8 bit words)	36	$180	–	4000	100
MOTOROLA	44	0	MC1141G	Triple 66 bit shift register	10	$29.70 (100+)	300	more than 1200	–
NATIONAL SEMI-CONDUCTOR CORP.	18	6	MM423	2048 bit ROM	24	$120 (100+)	–	about 2500	–
PHILCO-FORD CORP.	–	10	PL4516C	16 channel multiplexer	34	$38.50 (100+)	–	700	(36 14-pin DTL DIPS)
RADIATION, INC.	1	0	RS-1000	16 channel analog multiplexer	28	$360 (25+)	–	400	10
RAYTHEON	22	1	RR5100	64 bit RAM	16	$38 (100+)	–	400	30
RCA	7	0	CD4006	18 bit, static shift register	14	$17.25 (1000+)	–	–	–
SIGNETICS	42	0	5822R	8 bit content addressable memory	16	$44.50 (100+)	100	–	–
SYLVANIA ELECTRIC PRODUCTS INC.	45	0	SM180	Binary up-down counter	14	$18.15 (100+)	–	220	13
TEXAS INSTRUMENTS	46	2	DRA-1001	Digital differential analyser	156	$500 (25 to 99)	253	2088	–
TRANSITRON ELECTRONIC CORP.	92	0	TMC6464	64 bit bipolar memory cell	16	$30 to $40 (100+)	–	600	50

**From* Electronics Products, December 1969.

Table II reveals the present state of the art of MSI/LSI circuits and the chart is specifically limited to off-the-shelf devices. It may be noted from the chart that a 2048-bit ROM could be obtained for as low as $100 and a triple 66-bit shift register for $30. Projecting this a little further, it will be reasonable to assume that each chip in our above example will on the average cost a maximum of $100. With this assumption, we find that 45 chips will cost $4,500 and, putting an estimate of $500 for the A/D and D/A converters, we find that 25 channels of 8th order filters will cost approximately $4,900. A similar 24-channel filtering system generally used in the telephone multiplexing system employing electronic active filters, makes use of 24 channel filters, 8 pre-group filters and a few active circuits. Each channel filter consists of 2 low pass and 2 band pass filters.

The break-down cost for the above system obtained from a telephone company is given below.

	1 channel filter = 2 LP + 2 BP	=	\$ 100	(Approx.)
Hence	24 channel filters	=	\$2,400	
	8 pre-group filters	=	\$ 800	
	associated active circuits	=	\$3,200	
	Total cost of 24 channels		\$6,500	

Hence, it is obvious that digital filters which employ LSI technology are very promising in terms of accuracy and economics.

7.00 Conclusion

At this time, one foresees increasing application of digital filtering in many signal processing operations, particularly at frequencies below a few kilohertz. The new technology of large-scale integration is making an impact on the hardware implementation of digital filters. The characteristics of digital filtering, namely, high degree of accuracy, stability, and repeatability, makes it highly competitive with analog filtering in applications such as speech synthesis and telephone switching communication circuits.

There are various ways to realize the same digital filter transfer function. Practical techniques for designing both recursive and non-recursive types of digital filters are fairly well developed. Currently, recursive filters are considered to be more practical for meeting hardware requirements for many applications. However, both types are finding wide application in signal processing operations. Fast Fourier transform and high-speed convolution techniques are being applied extensively in digital filtering operations and special purpose computers are being made available to perform fast Fourier transforms.

8.00 Acknowledgment

The authors would like to acknowledge the generous assistance of the members of the Radio and Electrical Engineering Division during the period this work was carried out and in helping in the preparation of this report. They would also like to thank Mr. J. Valihora of the Northern Electric Co. for useful discussions. Mr. K.R. Srinivasan would especially like to thank Mr. F.V. Cairns, head of the Data Systems Section, for his encouragement and advice while he was with this Section. Mr. H. Greer of the Radio Library was particularly helpful in tracking down numerous references.

9.00 Bibliography

1. Blackman, R.B. Linear data-smoothing and prediction in theory and practice. Addison-Wesley Pub. Co., Inc., Mass, 1965.
2. Blackman, R.B. and Tukey, J.W. The measurement of power spectra. Dover Pub. Inc., New York, 1959.
3. Blum, M. On experimental digital filters. J. Assoc. for Computing Machinery, 6: 283–304; 1959.
4. Capps, J.W., Link, W.F. *et al.* Study and experimental investigation on sampling rate and aliasing in time-division telemetry systems. Tech. Rept. No. ASD-TR-61-553, Aerometronic, (a division of Ford Motor Co.), Newport Beach, California, June 1962.
5. Carney, R. Design of a digital notch filter with tracking requirements. IEEE Trans., SET-9: 109–114; 1963.
6. Clapp, W.A. Digital filter building blocks for LSI technology. 1970 IEEE International Convention Digest, pp. 182–183, 1970.
7. Cooley, J.W. Applications of the fast Fourier transform method. Proc. of the IBM Scientific Computing Symposium, June 1966.
8. Cooley, J.W. and Tukey, J.W. An algorithm for the machine calculation of complex Fourier series. Math Comput., 19: 297–301; 1965.
9. Corrington, Murlan S. Improving the accuracy and speed of digital filters. Proc. Second Biennial Cornell Elec. Eng. Conf on Computerized Electronics, Vol. 2, August 1969.
10. Cotton, R., Frangipane, J., Vernot, R. Analog-to-digital converters at hundreds of megabits. 1970 IEEE International Convention Digest, pp. 260–261, 1970.
11. Crystal, T.H. and Ehrman, L. The design and application of digital filters with complex coefficients. IEEE Trans., AV-16 (3): 303–314; 1968.
12. Davenport, W.B. and Root, W.L. Random signals and noise. Chapter 11, McGraw-Hill Book Co., New York, 1958.
13. Duncan, P.H. A new approach to effective digital filter design. Proc. International Telemetry Conf., pp. 1–26, 1965.
14. Ferguson, M.J. and Mantey, P.E. Preliminary investigation of digital filter automatic frequency control. Proc. Hawaii International Conf. System Sciences, pp. 209–212, 1968.
15. Ferguson, M.J. and Mantey, P.E. Automatic frequency control via digital filtering. IEEE Trans., AV-16: 392–393; 1968.
16. Finch, J.R. LSI-digital electronics. International Solid-State Circuits Conference Digest Tech. Papers, 10: 32–33; 1967.
17. Fleck, J.T. and Freyer, W.D. An exploration of numerical filtering techniques. CAL Rept. No. XA-869-P-1, May 1953. Cornell Aero Lab. Inc., Cornell University, Buffalo, N.Y.
18. Friant, R.J., Jr. Practical digital data smoothing. GE Technical Information Series R58 ENH29, June 1958.
19. Fullenwider, E.D. and McNamee, B.I. Filtering sampled functions. Tech. Memo No. 64-104, 12 July 1956, U.S. Naval Ordnance Lab, California.

20. G-AE Concepts Subcommittee. On digital filtering. IEEE Trans., AU-16: 315–320, 1968.
21. Ganzel, J.R. A communication engineer looks at IC's. Proc. of National Telemetering Conference, pp. 124–130; 1969.
22. Gardenhire, L.W. The use of digital data systems. Proc. National Telemetering Conference, 1963.
23. Gentleman, W.M. and Sande, G. Fast Fourier transforms for fun and profit. 1966 Fall Joint Computer Conference. American Federation of Information Processing Societies Proceedings, 29: 563–578; 1966.
24. German, E.H. Digital filtering and processing by transform techniques. Vol. I. Bendix Corporation, June 30, 1968.
25. German, E.H. Digital filtering and processing by transform techniques. Vol. II. Bendix Corporation, June 30, 1968.
26. Gold, B. and Rader, C.M. Digital processing of signals. McGraw-Hill Book Co., New York, 1969.
27. Gold, B. and Rader, C.M. Effects of quantization noise in digital filters. 1966 Spring Joint Computer Conference, American Federation of Information Processing Societies Proc., 28: 213–219; 1966.
28. Golden, R.M. Digital filter synthesis by sampled data transformation. IEEE Trans., AU-16 (3): 1968.
29. Golden, R.M. and Kaiser, J.F. Design of wideband sampled data filters. Bell Systems Technical Journal, 43 (Part 2): 1533–1546; 1964.
30. Golden, R.M. and White, S.A. A holding technique to reduce number of bits in digital transfer functions. IEEE Trans., AU-16 (3): 433–436; 1968.
31. Goodman, N.R. Measuring amplitude and phase. Space Technology Laboratories Report, TR-59-0000-00691, May 1959.
32. Guillemin, E.A. Synthesis of passive networks. John Wiley and Sons, Inc., New York, 1957.
33. Helms, H.D. Non-recursive digital filters: design methods for achieving specifications on frequency response. IEEE Trans., AU-16 (3): 336–342; 1968.
34. Helms, H.D. Applications of digital waveform processing in radars. IEEE International Convention Digest, pp. 180–181, 1970.
35. Hendrickson, R.M. Frequency response functions of certain numerical filters. Space Technology Laboratories Report, GM 433-281, April 1959.
36. Holtz, H. and Leondes, C.T. The synthesis of recursive digital filters. J. of the Association for Computer Machinery, 13 (2): 262–280; 1966.
37. Houts, R.C. and Weatherman, J.E. Design problems associated with recursive digital filters. Gateway to future, IEEE Region 3 Convention Record, New Orleans, IEEE Publication 68C17-REG-3, 1968.
38. Hsia, T.C. A technique for synthesizing digital filters. IEEE Trans., IM-18: 93–96; 1969.
39. Jackson, L.B., Kaiser, J.F. and McDonald, H.S. An approach to implementation of digital filters. IEEE Trans., AU-16 (3): 413–421; 1968.

40. Jin, B. and Kaneko, T. Error analysis of digital filters realized with floating-point arithmetic. Proc. IEEE, 57: 1735–1747; 1969.
41. Jury, E.I. Theory and application of the *z*-transform method. Wiley, New York, 1958.
42. Kaiser, J.F. The digital filter and speech communication. IEEE Trans., AU-16 (2): 180–183; 1968.
43. Kaiser, J.F. and Kuo, F. System analysis by digital computers. John Wiley and Sons, Inc., New York, 1966.
44. Knowles, J.B. and Olacayto, E.M. Coefficient accuracy and digital filter response. IEEE Trans., CT-15: 31–41; 1968.
45. Korn, G.A. and Korn, T.M. Mathematical handbook for scientists and engineers. McGraw-Hill, see 4.11-3 to 4.11-5, table 4.11-1; and 21.2-13, 1961.
46. Leuthold, P. Filter networks with digital shift registers. Philips Research Reports Supplement, No. 5, 1967.
47. Lowenschuss, O. Efficient digital filters for signal detection. Proc. Hawaii International Conference on System Sciences, Honolulu, Hawaii, pp. 205–208, January 1968.
48. Lynn, P.A. Economic linear-phase recursive digital filters. Electron. Let., 6 (5): 1970.
49. Martin, M.A. Digital filters for data processing. Tech. Information Series. Doc. No. 62 SD 484 GEC Missiles & Space Div., Philadelphia, Pa. Oct. 1962.
50. Martin, M.A. Frequency domain applications to data processing. Tech. Information Series. Doc. No. 57SD340 GEC Missiles and Ordnance Systems Dept. Philedelphia, Pa. May 1957.
51. McDonald, H.S. Impact of LSI circuits on communication equipment. Proc.of the National Electronic Conference NM XXIV, pp. 569–571, 1968.
52. McRae, D.D. Interpolation errors, advanced telemetry study. Tech. Report 1, pts 1 and 2, Radiation Inc., May 1961.
53. Nagle, H.D. Jr. and Carroll, C.C. Organizing a special-purpose computer to realize digital filters for sampled-data systems. IEEE Trans., AU-16 (3): 398–412; 1968.
54. Nowak, D.J. Linear phase digital filter design. Proc. IEEE (Letters), 57: 850–851; 1969.
55. Nowak, D.J. and Schmid, P.E. A non-recursive digital filter for data transmission. IEEE Trans., AU-16 (3): 343–349; 1968.
56. Olnes, R.K. An elementary design procedure for digital filters. IEEE Trans., AU-16 (3): 330–335; 1968.
57. Oppenheim, A.V., Schafer, R.W. and Stockham, T.G. Jr. Non-linear filtering of multiplied and convolved signals. IEEE Trans., AU-16 (3): 437–465; 1968.
58. Ormsby, Joseph, F.A. Design of numerical filters with applications to missiles data filter design. Assoc. for Comp. Mach. J., 8: 440; 1961.
59. Papoulis, A. The Fourier integral and its applications. McGraw-Hill, New York, 1962.

60. Petritz, R.L. Current status of large scale integration. Fall Joint Computer Conference, American Federation of Information Processing Societies, Proc., 31: 65–85; 1967.
61. Rabiner, L.R. A design technique for non-recursive digital filters. IEEE International Convention Digest, pp. 176–177, 1970.
62. Rader, C.M. A new method of generating Gaussian random variables by computer. IEEE International Convention Digest, pp. 178–179, 1970.
63. Rader, C.K. and Gold, B. Digital filter design techniques in the frequency domain. Proc. IEEE., 55 (2): 149–171; 1967.
64. Ragazzini, J.R. and Franklin, G.R. Sampled-data control systems. McGraw-Hill, New York, 1958.
65. Roberts, P.D. Self-adjustable orthogonal digital filters for system identification and optimization. Proc. IEEE, 114 (5): 666–670; 1967.
66. Rossi, C. Window functions for non-recursive digital filters. Electon. Let., 3 (12): 559–561; 1967.
67. Schmid, P.E. A/D converter Part III. Electron. Design, pp. 97–112, January 4, 1969.
68. Scientific Data Systems. Correlation and filtering unit. CFE 1.
69. Special Issue on Digital Filtering. IEEE Trans., AU-18 (2): June 1970.
70. Steiglitz, K. Design considerations for digital tracking filters. Proc. of Hawaii International Conference on System Sciences, p. 201, 1968.
71. Steiglitz, K. The equivalence of digital and analog signal processing. Information and Control, 8: 455–467; 1965.
72. Stockham, T.G. Jr. High speed convolution and correlation. Proc. Spring Joint Computer Conference, pp. 229–233; 1966.
73. Tavares, S.E. A comparison of integration and low pass filtering. IEEE Trans., IM-15 (1 and 2): 33–38; 1966.
74. Tavares, S.E. Nature and application of digital filters. The Engineering J., Engineering Institute of Canada, pp. 23–27, January 1967.
75. Tierney, J. A method of digital frequency synthesis. IEEE International Convention Digest, pp. 174–175, 1970.
76. Tou, J. Digital and sampled data control systems. McGraw-Hill, New York, 1959.
77. Weaver, C.S., Groeben, J. Vender, *et al.* Digital filtering with applications to electrocardiogram processing. IEEE Trans., AU-16 (3): 350–391; 1968.
78. Weinberg, L. Network analysis and synthesis, McGraw-Hill, New York, 1962.
79. Whittlesey, J.R.B. A rapid method for digital filtering. Assoc. for Computing Machinery, Vol. 9: 552–556; 1964.
80. Zverev, A.I. Digital MTI radar and filters. IEEE Trans., AU-16 (3): 422–432; 1968.