Analysis of Linear Digital Networks

RONALD E. CROCHIERE AND ALAN V. OPPENHEIM, SENIOR MEMBER, IEEE

Invited Paper

***Abstract*—A framework is presented for the analysis, representation, and evaluation of digital filter structures. Based on the notation of linear signal-flow graphs and their equivalent matrix representation, a set of general linear digital network properties are reviewed, including precedence relations computability, Tellegen's theorem, interreciprocity, and network transposition. These properties are then utilized in developing time and frequency domain analysis techniques and sensitivity analysis techniques. These techniques, in turn, are applied to the comparison of several basic digital filter structures.**

I. Introduction

When IMPLEMENTING a digital filter on a general-purpose computer, it is often unnecessary to pay close attention to the structure used for the implementation. In contrast, when implementing a filter in special-purpose hardware, such factors as speed, cost, wordlength, etc., are closely related to the network structure used. Because of the increasing practicality of hardware implementations of digital filters, there has been considerable effort directed over the last several years toward the development and investigation of digital filter structures. As part of this effort, there has been the development of techniques for analyzing filter structures efficiently and also the theoretical investigation of general properties of structures. Thus, in addition to the development of specific classes of structures, a set of properties and theorems for filter structures has evolved over the last several years. Much of this theory has a counterpart in classical analog network theory and, in fact, in many instances the application of specific results to digital filter structures was motivated by the success of similar techniques in analog network theory.

In this paper we suggest a framework for the analysis and representation of digital filter structures, and their evaluation with respect to several considerations. We have found it convenient to use the notation of linear signal-flow graphs or the equivalent matrix representation. This notation serves as the basis for a general network representation for digital networks, and through this representation many of the theorems and methods of signal-flow graphs can be specifically adapted for digital networks. In particular, in Section II we summarize the signal-flow graph and matrix notation, and in Section III we present a number of general network properties. Sections IV and V are directed toward a discussion of computer-aided analysis of structures. This includes a discussion of the use of the properties developed in Section III for time and frequency domain analysis and the analysis of sensitivity to coefficient quantization. A comparison of a number of structures with regard to coefficient sensitivity for one specific filter is also given.

There are a variety of considerations that must be taken into account in choosing or comparing structures. In Section VI we suggest, as one such consideration, that of the amount of parallelism inherent in a structure. The ideas discussed in that section represent only an initial, but we feel promising, attempt to quantify the consideration of parallelism in structures.

This paper is intended in part as a tutorial review and as such we have attempted to cover a broad range of topics. In many cases, details that are available through the references have been omitted. In particular, many of these details can be found in [1].

Manuscript received August 15, 1974; revised November 11, 1974. This work was supported in part by the National Science Foundation under Grant GK-31353 and in part by the Advanced Research Projects Agency of the Department of Defense, monitored by ONR under Contract N00014-67-A-0204-0064.

R. E. Crochiere is with Bell Laboratories, Murray Hill, N.J. 07974.

A. V. Oppenheim is with the Research Laboratory of Electronics and the Department of Electrical Engineering, Massachusetts Institute of Technology, Cambridge, Mass. 02139.

II. Signal-Flow Graph and Matrix Representation of Digital Networks

In this paper we will find it convenient to represent digital networks in terms of linear signal-flow graphs or the equivalent matrix representation. This is motivated in part by the properties of signal-flow graphs which are conveniently exploited in the discussion of digital networks. Thus, in this section, we first summarize the notation of signal-flow graphs and the equivalent matrix equations to be used in this paper.

A signal-flow graph is a collection of nodes and directed branches [2], [3]. Associated with each node is a node signal and associated with each branch is a branch signal. For the case of digital networks, the node and branch signals are discrete-time signals or their z transforms.

In representing digital networks with signal-flow graphs, we will find it convenient to distinguish between source branches and network branches. A *network branch* originates from a network node and terminates at a network node. The input to the branch is taken as the signal value at the node where the branch originates.

For the representation of digital filters, we restrict the network branches to correspond either to a constant gain (a coefficient branch) or to a constant gain in cascade with a unit delay (a coefficient-delay branch), and assume also that only one network branch of each kind is permitted between any two nodes. The output of the coefficient branch from node j to node k will be denoted as $w_{cjk}(n)$ and the output of the delay branch from node j to node k will be denoted as $w_{djk}(n)$. The node signal value at node j will be denoted as $y_j(n)$. The branch outputs are then related to the branch inputs by

$$w_{cjk}(n) = f_{cjk} y_j(n) \tag{1a}$$

and

$$w_{djk}(n) = f_{djk} y_j(n-1) \tag{1b}$$

where f_{cjk} and f_{djk} are constant real coefficients. Alternatively, we can express the node values and branch outputs in terms of

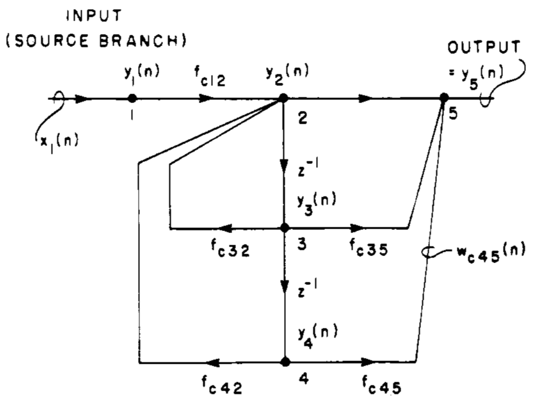


Fig. 1. Example of digital network expressed in signal-flow graph notation.

their z transforms—in which case, (1a) and (1b) are written as

$$W_{cjk}(z) = F_{cjk}(z)Y_j(z) \tag{2a}$$

$$W_{djk}(z) = F_{djk}(z)Y_j(z) \tag{2b}$$

where

$$F_{cjk}(z) = f_{cjk} \tag{3a}$$

and

$$F_{djk}(z) = z^{-1}f_{djk}\,. \tag{3b}$$

The terms $F_{cjk}(z)$ and $F_{djk}(z)$ correspond to the *branch transmittances* of the coefficient and the coefficient-delay branches, respectively. It will generally be convenient to assume that there is both a coefficient and a coefficient-delay branch between each pair of network nodes, although some of the branch coefficients may be zero.

Source branches represent the injection of external signal sources into the network. They will be assumed to have unity transmittance so that the source branch output corresponds to the value of the external signals entering the network at that point. It will be assumed that there is at most one source branch associated with each node in the network. The source branch signal value associated with the kth node is designated as $x_k(n)$ or equivalently $X_k(z)$ for its z transform. For convenience, we will assume that a source branch connects to each network node, although at some nodes the source branch output may be zero.

Using the foregoing signal-flow graph notation, linear digital networks of any arbitrary configuration can be represented. An example of such a digital network is given in Fig. 1. It corresponds to a conventional second-order direct form (canonical) structure. The input of the network is via a source branch entering node 1 and the output is taken as the node signal $y_5(n)$ of node 5. Coefficients f_{c25}, f_{d23}, and f_{d34} are assumed to be unity and all other branches not shown are assumed to have coefficient values of zero. Nodes 2 and 5 act as both summing points and branch points in the network. The branch signal $w_{c45}(n)$, for example, is obtained from the multiplication of the node signal $y_4(n)$ by the branch coefficient f_{c45}.

The node signal value at each node in a network is the sum of the branch signals entering the node. Thus

$$Y_k(z) = X_k(z) + \sum_{j=1}^{N} [W_{cjk}(z) + W_{djk}(z)], \qquad k = 1, 2, \cdots, N \tag{4}$$

where it is assumed that there are N nodes in the network. Alternatively, with the aid of (2) and (3), this can be written as

$$Y_k(z) = X_k(z) + \sum_{j=1}^{N} [f_{cjk} + f_{djk}z^{-1}]\; Y_j(z), \qquad k = 1, 2, \cdots, N. \tag{5}$$

As there are N nodes in the network, there are N corresponding equations and they constitute a total description of the network. Inputs to the network are described by means of the source branches and outputs of the network are obtained as the signal values of one or more of the network nodes.

The N equations in (5) can be expressed more compactly in matrix form as

$$Y(z) = X(z) + f_c^t Y(z) + f_d^t Y(z)z^{-1} \tag{6}$$

where

$Y(z)$ a column vector of the N node signal values $Y_k(z)$ ($k = 1, 2, \cdots, N$),

$X(z)$ a column vector of the N branch signal values $X_k(z)$ ($k = 1, 2, \cdots, N$) of the source branches,

f_c an $N \times N$ matrix of coefficients for network branches of the first kind (coefficient branches),

f_d an $N \times N$ matrix of coefficients for network branches of the second kind (coefficient-delay branches).

The superscript t in (6) denotes the matrix transpose, which is used for consistency between the subscript conventions of signal-flow graphs and matrices. An element f_{djk} of matrix f_d, for example, corresponds to the coefficient value of the coefficient-delay branch from node j to node k. If no such branch exists in the network, then $f_{djk} = 0$. As an example, the matrix representation for the digital network in Fig. 1 is

$$\begin{bmatrix} Y_1(z) \\ Y_2(z) \\ Y_3(z) \\ Y_4(z) \\ Y_5(z) \end{bmatrix} = \begin{bmatrix} X_1(z) \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} + \begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ f_{c12} & 0 & f_{c32} & f_{c42} & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & f_{c35} & f_{c45} & 0 \end{bmatrix} \begin{bmatrix} Y_1(z) \\ Y_2(z) \\ Y_3(z) \\ Y_4(z) \\ Y_5(z) \end{bmatrix} + z^{-1} \begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} Y_1(z) \\ Y_2(z) \\ Y_3(z) \\ Y_4(z) \\ Y_5(z) \end{bmatrix}.$$

The response of the node signal values $Y(z)$ of a network due to the signal inputs $X(z)$ can be expressed in terms of a matrix inversion as

$$Y(z) = T^t(z)X(z) \tag{7}$$

where

$$T^t(z) = [I - f_c^t - f_d^t z^{-1}]^{-1} \tag{8}$$

and I is defined as the $N \times N$ identity matrix. The $N \times N$ matrix $T(z)$ will be referred to as the transfer function matrix of the network. An element $T_{jk}(z)$ of this matrix can be identified as the transfer function from node j to node k and it characterizes the response of the network at node k due to a single source branch input at node j. That is, with all source branch inputs but $X_j(z)$ set to zero,

$$T_{jk}(z) = \frac{Y_k(z)}{X_j(z)}. \tag{9}$$

In general, if more than one signal is injected into the network by means of source branches, the response at a particular node k can be expressed, using superposition, as

$$Y_k(z) = \sum_{j=1}^{N} T_{jk}(z) X_j(z). \tag{10}$$

If the source branch at a specific node a is defined as the network input and a specific node signal at node b is defined to be taken as the output of the network, then the transfer function $T_{ab}(z)$ is the system function of the network. For example, in the network of Fig. 1, the system function would correspond to $T_{15}(z)$.

The matrix equation (6) can alternatively be expressed in the time domain. Specifically, the inverse z transform of (6) corresponds to the time domain matrix difference equation

$$y(n) = x(n) + f_c^t y(n) + f_d^t y(n-1) \tag{11}$$

which is similar in some respects to a state-space description of a digital network [4]. It differs, however, in a number of important respects. The number of variables in (11) corresponds to the number of nodes in the network. This is in contrast to a state-space description where the number of variables is equal to the number of essential states in the network, i.e., the number of network nodes with one or more delay branches entering them. The matrix difference equation (11) has the property that the coefficients in the matrices f_c^t and f_d^t correspond in a one-to-one fashion with the branch transmittances in the flow graph. In a state-space representation, this is not generally true.

III. General Network Properties

In the preceding section, we presented the representation of digital networks in terms of flow graph and matrix notation. Based on this representation, there are a number of network properties which we will want to make use of in the remaining section. The first of these is the notion of node precedence relations. As we will see, there is inherent in any given network structure an order of precedence in which node values are computed. The node precedence relations become particularly important in Section VI where we will discuss parallelism in structures.

A second general property that we discuss is Tellegen's theorem for digital networks. This simple but elegant theorem provides a useful framework for developing a number of other properties of digital networks. We conclude this section with a discussion of the notions of reciprocity, interreciprocity, and network transposition. These ideas are useful in computer-aided analysis of network structures and play a particularly important role in the sensitivity analysis of structures as presented in Section V.

Precedence Relations for Digital Networks

The implementation of a digital network generally corresponds to a direct evaluation of the difference equation (11). We observe that (11) expresses each node value as a linear combination of the inputs and present and past values of the node values. If we do not order the computation of the nodes, it is possible that the computation of one node value requires other node values that have not yet been computed. Thus there is an implied set of precedence relations associated with this set of N scalar equations and they are solely a consequence of the coefficient branch topology in the structure.

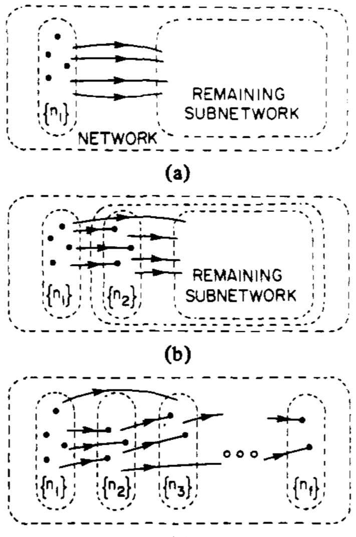


Fig. 2. Algorithmic procedure for generating precedence form of a network. (a) Extracting first node set. (b) Extracting second node set. (c) Final representation of network in precedence form (only coefficient branches are shown).

It is of interest to investigate the implications of these precedence relations. We first note that the precedence relations are solely a consequence of the coefficient branch topology in the structure and do not depend at all on the coefficient-delay branches. An algorithmic procedure for laying out a structure in a form which reveals these precedence relations will be given. The procedure begins by searching the network for all nodes which do not have coefficient branches entering them. These nodes are then separated from the network and assigned to node set $\{n_1\}$, as depicted in Fig. 2(a). It is apparent that these node values can be computed without knowledge of any of the node values in the network for time index n. Their evaluation may require only past node values for time $n-1$ due to coefficient-delay branches, or present source input values at time n.

Next, the subnetwork is considered which contains all nodes and branches in the network *except* those nodes which belong to the node set $\{n_1\}$ and those branches which connect these nodes. A search is made over this subnetwork for all nodes which do not have coefficient branches from other nodes in the subnetwork entering them. These nodes are assigned to node set $\{n_2\}$, as indicated in Fig. 2(b). It is apparent that each of the nodes in set $\{n_2\}$ must have at least one coefficient branch entering it which comes from node set $\{n_1\}$ when considering the overall network, since otherwise the node would have been contained in set $\{n_1\}$. This procedure is repeated, as depicted in Fig. 2(c), until at the last stage none of the remaining nodes of the final subnetwork have coefficient branches entering them. These nodes are assigned to node set $\{n_f\}$. The resulting configuration of the network, as given in Fig. 2(c), is referred to as the *precedence form* of the network as it depicts all of the precedence requirements of the network in terms of node evaluations. An example of the precedence form of the

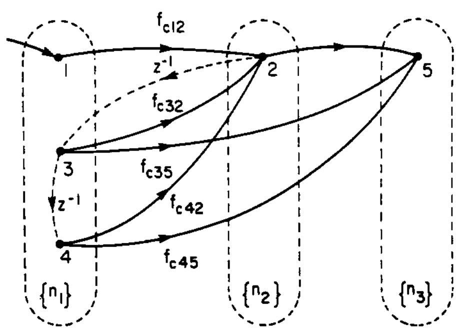


Fig. 3. Precedence form of network in Fig. 1.

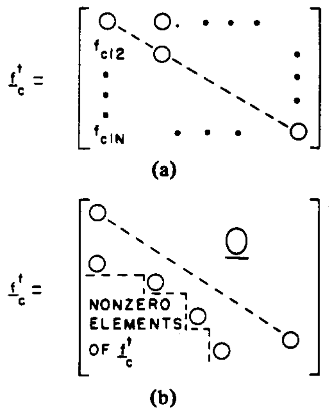


Fig. 4. (a) Form of matrix term f_c^t for matrix representation in a computable form. (b) General form of f_c^t when node sets contain more than one node each and nodes are numbered according to algorithmic procedure.

network in Fig. 1 is given in Fig. 3. In this figure, we have indicated the coefficient branches as solid lines and the coefficient-delay branches as dotted lines to emphasize the fact that the precedence relations are a function only of the coefficient branch topology.

If the foregoing algorithmic procedure does not proceed to completion, then the precedence relations are contradictory and the network is defined as being *noncomputable*. That is, if at some stage all nodes in the remaining subnetwork have coefficient branches entering them, then this subnetwork contains one or more closed loops without delay in it and none of the nodes can be evaluated without requiring the value of at least one other node in the subnetwork. If the algorithm does proceed to completion and produces a network in the form depicted in Fig. 2(c), then there are no closed loops without delay in the network and it is defined as being *computable*. Generally, computable networks are the only type of digital networks that are of interest in digital signal processing. It should be noted that if a network is noncomputable, the network equation (11) may still be solvable. However, they are not solvable by computing successive node values in the network.

The network property of computability can be represented explicitly in terms of the matrix representation. Specifically, the matrix representation in (6) or (11) for a computable network can be written in a form such that the matrix f_c^t is zero on and above the main diagonal [5], as depicted in Fig. 4(a). Furthermore, it can be shown that if a network is noncomputable, then the matrix representation cannot be expressed in this form [1]. A procedure for numbering the nodes of a computable network in order to obtain a matrix representation

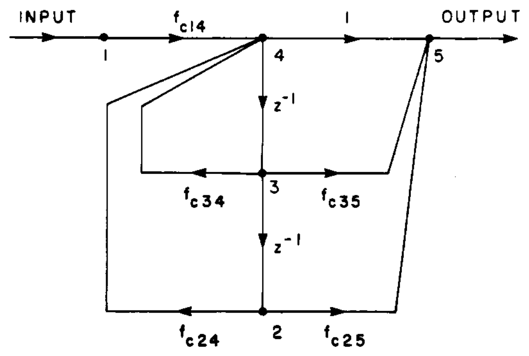


Fig. 5. Network of Fig. 1 with nodes numbered according to precedence form numbering scheme.

of this form can be determined with the aid of the precedence form of the network. To obtain this form, the network nodes can be numbered from 1 to N consecutively, first numbering all nodes in set $\{n_1\}$, then all nodes in sets $\{n_2\}$, $\{n_3\}, \cdots,$ $\{n_f\}$, until all of the nodes have been numbered. With the node numbers corresponding to the subscripts of the matrix equation, this leads to a form of the matrix f_c^t given in Fig. 4(b). If the network is computable, then there is at least one node in each node set and consequently f_c^t is at least of the form given in Fig. 4(a) and can generally be expressed in a form depicted in Fig. 4(b).

If this numbering scheme is applied to the network example of Fig. 1, it is apparent from the precedence form in Fig. 3 that the original node numbers 2 and 4 must be interchanged. This results in the new node numbering scheme given in Fig. 5. The matrix representation for this network then becomes

$$\begin{bmatrix} Y_1(z) \\ Y_2(z) \\ Y_3(z) \\ Y_4(z) \\ Y_5(z) \end{bmatrix} = \begin{bmatrix} X_1(z) \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} + \begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ f_{c14} & f_{c24} & f_{c34} & 0 & 0 \\ 0 & f_{c25} & f_{c35} & 1 & 0 \end{bmatrix} \begin{bmatrix} Y_1(z) \\ Y_2(z) \\ Y_3(z) \\ Y_4(z) \\ Y_5(z) \end{bmatrix}$$
$$+ z^{-1} \begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} Y_1(z) \\ Y_2(z) \\ Y_3(z) \\ Y_4(z) \\ Y_5(z) \end{bmatrix}$$

where f_c^t is now of the form such that it is zero on and above the main diagonal.

If a network is nonrecursive and computable, i.e., if it has no loops (with or without delay) in its topology, then the matrix representation can be expressed in a form such that both f_c^t and f_d^t are simultaneously zero on and above their main diagonals [1], as expressed in Fig. 4(a). The procedure for generating this form of the matrix representation is similar to that for generating the computable form of the matrix, with the exception that both coefficient and coefficient-delay branches are used in the construction of a network form analogous to that of Fig. 2(c). An interesting consequence of this form of the matrix representation is that it leads to a transposed transfer function matrix $T^t(z)$ which is lower triangular. This follows from (8) since the inverse of a lower triangular matrix is lower triangular.

Tellegen's Theorem for Digital Networks

Tellegen's theorem for analog networks has played an important role in developing and understanding network properties [6], [7]. Because digital networks, and more generally signal-flow graphs, are not subject to Kirchhoff's laws, Tellegen's theorem in its most general form does not apply. It does apply, however, in a somewhat restricted form. Tellegen's theorem, as it applies to digital networks, provides a useful framework for developing and interpreting a number of important and useful network properties. In particular, we will utilize Tellegen's theorem to show that the matrix of transfer functions for a network and its transpose are interreciprocal. As we will see, this property is useful for computer-aided analysis of filter structures. Tellegen's theorem also leads to a number of useful relations which we utilize in Section V for sensitivity analysis of digital filter structures. While in many cases these properties can be developed in alternate ways without utilizing Tellegen's theorem directly, Tellegen's theorem is appealing in part because it provides a unifying basis.

Several variations of Tellegen's theorem for signal-flow graphs have been proposed, including those by Seviora and Sablatash [8], Fettweis [9], Blostein [10], and Lee [11], [12]. The form of the theorem and its related concepts which are presented here is perhaps most similar to that proposed by Fettweis, and is stated as follows.

Tellegen's Theorem: Consider two networks, each of which contains N nodes. For the first network, the source branch signals, coefficient branch signals, coefficient-delay branch signals, and node signals are denoted as $X_k(z)$, $W_{cjk}(z)$, $W_{djk}(z)$, and $Y_k(z)$, respectively; for the second network, they are denoted as $X'_k(z)$, $W'_{cjk}(z)$, $W'_{djk}(z)$, and $Y'_k(z)$, respectively. Then Tellegen's theorem states that

$$\sum_{j=1}^{N}\sum_{k=1}^{N} [Y_k(z)\,(W'_{cjk}(z) + W'_{djk}(z)) - Y'_k(z)\,(W_{cjk}(z) + W_{djk}(z))] + \sum_{k=1}^{N} [Y_k(z)X'_k(z) - Y'_k(z)X_k(z)] = 0 \quad (12)$$

which follows in a straightforward manner by substituting (4) into the identity

$$\sum_{k=1}^{N} [Y_k(z)\,Y'_k(z) - Y'_k(z)\,Y_k(z)] = 0. \quad (13)$$

Alternatively, Tellegen's theorem can be expressed in the time domain as

$$\sum_{j=1}^{N}\sum_{k=1}^{N} [y_k(n)\,(w'_{cjk}(n) + w'_{djk}(n)) - y'_k(n)\,(w_{cjk}(n) + w_{djk}(n))] + \sum_{k=1}^{N} [y_k(n)x'_k(n) - y'_k(n)x_k(n)] = 0. \quad (14)$$

Three closely associated concepts which follow directly from Tellegen's theorem are those of reciprocity, interreciprocity, and network transposition. These concepts have long histories in classical network theory. They have been developed specifically for linear digital networks through the efforts of Jackson [13] and Fettweis [9]. They are also closely related to Mason's theorem on flow graph reversal [2], [3] and to the adjoint signal-flow graph concepts proposed by Seviora and Sablatash [8], Blostein [10], and Lee [11], [12]. The discussion in the next two sections follows most closely the approach taken by Fettweis.

Reciprocity and Interreciprocity in Digital Networks

A network is defined to be reciprocal if, for any two signal distributions in the network, denoted by primed and unprimed variables, it is possible to write [9]

$$\sum_{k=1}^{N} [Y_k(z)X'_k(z) - Y'_k(z)X_k(z)] = 0. \quad (15)$$

For linear shift-invariant networks, it follows by substituting (10) into (15) that this is equivalent to the condition

$$T_{jk}(z) = T_{kj}(z) \quad (16)$$

for all pairs of nodes j and k in the network. That is, the matrix of transfer functions $T(z)$ for a reciprocal network is symmetric, so that for any pair of nodes j and k, the transfer function from j to k is identical to the transfer function from k to j. In practice, digital networks generally do not have this property.

A more important, but closely related, concept is that of interreciprocity between two networks [9]. If two networks are considered, each with N nodes, the first with signal variables $X_k(z)$ and $Y_k(z)$ and the second with signal variables $X'_k(z)$ and $Y'_k(z)$, then the two networks are interreciprocal if

$$\sum_{k=1}^{N} [Y_k(z)X'_k(z) - Y'_k(z)X_k(z)] = 0. \quad (17)$$

This relation is identical to that of (15), except that the primed and unprimed variables in this case correspond to sets of variables in two different networks. By substituting (10) into (17), it follows that for linear shift-invariant networks, this condition is equivalent to the condition

$$T_{jk}(z) = T'_{kj}(z) \quad (18)$$

for all pairs of nodes j and k in the two networks. That is to say, the transfer function matrix of the unprimed network is equal to the transpose of the transfer function matrix of the primed network. The applications of this concept will become apparent shortly.

Network Transposition

Another important concept in network theory is that of network transposition [9], [13]. The transposition of a network is defined as the operation of reversing the direction of all of the branches in the network. The resulting network is referred to as the transpose network. In the matrix representation, this corresponds to the operations

$$f'_c = f^t_c \quad (19a)$$

and

$$f'_d = f^t_d \quad (19b)$$

where the primed matrices correspond to those of the transpose network.

An important relationship between networks and their transposes is that they are interreciprocal. This can be shown by utilizing (19), together with Tellegen's theorem (12). Specifically, for the original network,

$$W_{cjk}(z) = F_{cjk}(z)\, Y_j(z) \tag{20a}$$

$$W_{djk}(z) = F_{djk}(z)\, Y_j(z) \tag{20b}$$

while, for the transposed (primed) network,

$$W'_{cjk}(z) = F'_{cjk}(z)\, Y'_j(z) = F_{ckj}(z)\, Y'_j(z) \tag{20c}$$

$$W'_{djk}(z) = F'_{djk}(z)\, Y'_j(z) = F_{dkj}(z)\, Y'_j(z). \tag{20d}$$

Equation (17) then follows from substitution of (20) into (12) and appropriate manipulations of the resulting summations.

IV. Time and Frequency Domain Analysis of Digital Networks

A particularly important area of application of digital network theory is in the development of general techniques for the analysis of arbitrary digital networks. The need for such techniques can be recognized when one considers the large variety of possible structures recently proposed as candidates for filter implementation. Some of these types of structures are illustrated in Section VI. With a computer-aided network analysis package, the process of analyzing and comparing such structures can be reduced to a much simpler and more manageable task.

The general matrix representation for digital networks presented in Section II provides for a convenient way of uniquely expressing networks of any configuration on a computer. The process of describing these networks and analyzing them can then be interpreted in terms of performing appropriate manipulations on the computer.

Frequency Response Analysis

One approach to obtaining the frequency response of an arbitrary network is to perform the matrix inversion in (8) at each frequency. In general, this approach is not very desirable as it requires an excessive amount of computation and can be prone to numerical errors [14]. A more desirable approach for computing the frequency response can be realized by expressing the matrix representation (6) in the form

$$[I - f_c^t - f_d^t z^{-1}]\, Y(z) = X(z) \tag{21}$$

where I is the $N \times N$ identity matrix. This matrix equation corresponds to N simultaneous linear scalar equations in the N unknowns $Y_i(z)$, $i = 1, 2, \cdots, N$. By assuming that the source branch inputs are all zero, except for that connected to node a, the solution to this set of equations provides the transfer functions $T_{aj}(z)$, $j = 1, 2, \cdots, N$. The solution of the N simultaneous linear equations can be obtained by a procedure such as Gaussian elimination using complex arithmetic. Generally, the matrices f_c^t and f_d^t will be relatively sparse. Various techniques were recently proposed for taking advantage of this sparsity to minimize the amount of computation necessary to solve equations of this type [15], [16]. Although many of these techniques have been developed for computer-aided analysis of conventional analog circuits, many of them apply equally well for the case of digital networks.

Time Domain Analysis

One approach for computation of the unit-sample response is to first express the time domain matrix representation (11) in a computable form and then solve the N scalar equations corresponding to this matrix equation consecutively at each time increment n. As discussed in Section III, the computable form of the matrix representation is that form in which f_c^t is zero on and above the main diagonal. To perform the network analysis, the initial conditions $y(-1) = 0$ are chosen and the input signals

$$x_j(n) = \begin{cases} 1, & j = a \text{ and } n = 0 \\ 0, & \text{otherwise} \end{cases}$$

are used. The unit-sample responses from a given input node a to all other nodes in the network can then be computed by solving the N scalar equations in the computable form of (11) for $y_j(n)$ for $j = 1, 2, \cdots, N$ consecutively and for $n = 0, 1, 2, 3, \cdots$ consecutively. As this set of calculations corresponds directly to that which must be performed in the actual implementation of the structure, simulation of effects such as limit cycles and overflow can also be incorporated into this procedure by duplicating on the computer the actual arithmetic operations which would be performed in the hardware implementation of the network.

An alternate approach to time domain analysis is to transform (11) into the form

$$y(n) = Ay(n-1) + Bx(n) \tag{22}$$

where

$$B = [I - f_c^t]^{-1} \tag{23}$$

and

$$A = Bf_d^t. \tag{24}$$

This can be accomplished if the initial matrix inversion in (23) can be performed as it can for computable networks since in that case $\det [I - f_c^t] = 1$. The solution of the unit sample response can then be obtained by solving (22) successively for $y(n)$ with the same initial conditions and input signals already used. As this is no longer the same set of equations as that represented by the structure, the simulation of limit cycles or overflow effects cannot be made with this method.

Analysis of Transpose Networks

The preceding frequency response and time response analyses yield the transfer functions or unit-sample responses, respectively, from the network input to all points internal to the network with a single analysis. In some cases, it is desired to analyze such responses from various points internal to the network to the output. For example, in the analysis of the effects of computational roundoff noise, the roundoff errors occur at various points within the network and can be modeled as an injection of white noise sources at these points in the network [13], [17], [18]. To determine the effect of these errors on the network output, it is necessary to determine the transfer functions from various internal points of the network to the output. With the analysis procedures described, this would require several analyses at each frequency. Alternatively, this can be performed with a single analysis at each frequency of interest by exploiting the interreciprocal property of transpose networks discussed in Section III. For example, the frequency response analysis of the transpose network corresponds to the analysis of the N simultaneous linear equations given by

$$[I - f_c - f_d z^{-1}]\, Y'(z) = X'(z) \tag{25}$$

where the unprimed variables correspond to those of the original network and the primed variables correspond to those of the transpose network. With $X'_b(z)$ chosen to be unity and $X'_j(z)$ chosen to be zero for $j \neq b$, the solution gives all of the transfer functions

$$\begin{aligned} T_{jb}(z) &= T'_{bj}(z) \\ &= Y_j(z), \qquad j = 1, 2, \cdots, N. \end{aligned}$$

That is, it gives all of the transfer functions from all internal points of the network to the output for a single analysis of the transpose network. A similar computational savings can be obtained in a time domain analysis where the unit-sample responses from more than one internal point of the network to the output are desired.

V. Sensitivity and Derivative Analysis of Networks

The implementation of digital filters is subject to the constraint of finite register length, the effects of which manifest themselves in two ways. One is the introduction of error due to arithmetic roundoff and the second is a change in the overall filter characteristics due to coefficient quantization. As indicated in the previous section, the analysis of the effects of arithmetic roundoff can utilize the properties of transpose networks to simultaneously compute the transfer functions from all of the roundoff noise sources to the output.

The effects of coefficient quantization can also be analyzed utilizing many of the ideas introduced in the previous section. The sensitivity of filter characteristics to quantization of the coefficients is highly dependent on the particular structure used to represent the filter and consequently, in evaluating filter structures, it is important to carry out an analysis of the sensitivities of the system function with respect to the branch coefficients or branch transmittances in the network. In this section, techniques for such a sensitivity analysis will be discussed. In addition, two other types of analysis, which are sometimes desired in general computer-aided analysis systems, are considered. The first of these is the evaluation of the group delay associated with the system function of a network and the second is the slope of the magnitude of the frequency response.

A First-Order Network Sensitivity Relation

A convenient expression for evaluating the sensitivities of the system function, with respect to the branch coefficients, has been proposed by various authors, including Seviora and Sablatash [8], Fettweis [9], and Lee [11], [12]. The proof of this expression is very elegant in that it involves the use of Tellegen's theorem and the concepts of interreciprocity and network transposition [8], [9], [19]. It is a rather lengthy proof, however, and will not be repeated here. The sensitivity expression can be defined by considering the single-input-single-output network indicated in Fig. 6. The system function of interest in this network, $H(z)$, is equal to the transfer function $T_{ab}(z)$ from node a to node b:

$$H(z) = T_{ab}(z). \tag{26}$$

The sensitivity of $H(z)$ with respect to a branch transmittance $F_{nm}(z)$ of a branch (either a coefficient or a coefficient-delay branch) from some node n to some node m can be given as

$$\left\{\begin{matrix}\text{sensitivity of } H(z) \\ \text{with respect to } F_{nm}(z)\end{matrix}\right\} \triangleq \frac{\partial T_{ab}(z)}{\partial F_{nm}(z)} = T_{an}(z)\, T_{mb}(z). \tag{27}$$

Thus the sensitivity of $H(z)$ with respect to $F_{nm}(z)$ can be expressed in terms of the product of two transfer functions in

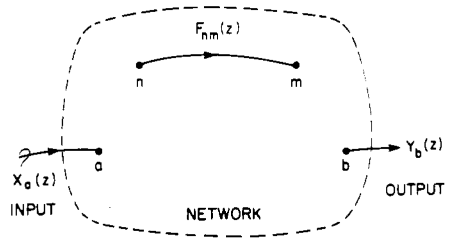


Fig. 6. Single-input-single-output network.

the network—the transfer function from node a to node n and the transfer function from node m to node b.

In some cases, the sensitivities of the magnitude of the system function with respect to the branch coefficients f_{cnm} and f_{dnm} may be desired. These sensitivities can be derived with the aid of (27) and are given as [1]

$$\frac{\partial |H(z)|}{\partial f_{cnm}} = \text{Re}\left[\frac{|H(z)|}{H(z)}\, T_{an}(z)\, T_{mb}(z)\right] \tag{28}$$

and

$$\frac{\partial |H(z)|}{\partial f_{dnm}} = \text{Re}\left[\frac{|H(z)|}{H(z)}\, T_{an}(z)\, T_{mb}(z)\, z^{-1}\right] \tag{29}$$

where it has been assumed that f_{cnm} and f_{dnm} are real. The sensitivity relations (27)–(29) are expressed in terms of the system function and two transfer functions $T_{an}(z)$ and $T_{mb}(z)$ in the network. The first transfer function is always one which is directed from the input of the network to an internal point of the network and the second transfer function is always one which is directed from an internal point in the network to the output. Thus the evaluation of network sensitivities is performed via the computation of network transfer functions and they can be determined with only two analyses of the network at each frequency. The first set of transfer functions $T_{an}(z)$ is obtained from the analysis of the original network and the second set of transfer functions $T_{mb}(z)$ is obtained from the analysis of the transpose network as discussed in Section IV. The system function $T_{ab}(z)$ can be obtained from either of the two analyses. If the sensitivity of the system function with respect to more than one coefficient in the network is desired, as is often the case, these sensitivities can all be obtained from the results of the same two network analyses with essentially no extra cost in computation. This then constitutes a highly efficient technique for the analysis of multiparameter network sensitivities. The technique is similar to the adjoint network approach by Director and Rohrer [7] for evaluating sensitivities of analog networks.

Higher Order and Large-Change Network Sensitivities

In studying the effects of variations of the system function $H(z)$ due to variations of coefficients or branch transmittances, we may, in some cases, desire to know the effect of changes in $H(z)$, denoted as $\Delta H(z)$, due to changes $\Delta F_{nm}(z)$ in a branch transmittance $F_{nm}(z)$, where $\Delta F_{nm}(z)$ may be more than an incremental change. These changes can be expressed with the aid of the Taylor's series expansion in the form

$$\Delta H(z) = \frac{\partial H(z)}{\partial F_{nm}(z)}\, \Delta F_{nm}(z) + \frac{1}{2}\frac{\partial^2 H(z)}{\partial F_{nm}^2(z)}\, (\Delta F_{nm}(z))^2 + \cdots \tag{30}$$

where $H(z) = T_{ab}(z)$, as denoted in Fig. 6. The higher order sensitivities in this expression can be obtained with the aid of the first-order network sensitivity relation in (27) and the chain rule for differentiation. This approach leads to a general expression for the higher order network sensitivities, which for the rth-order sensitivity can be given in the form [5]

$$\frac{\partial^r H(z)}{\partial F_{nm}^r(z)} = r!\, T_{mn}^{r-1}(z)\, T_{an}(z)\, T_{mb}(z). \tag{31}$$

By applying this expression to the Taylor's series expansion of (30), a closed-form solution can be obtained:

$$\Delta H(z) = \frac{T_{an}(z)\, T_{mb}(z)\, \Delta F_{nm}(z)}{1 - T_{mn}(z)\, \Delta F_{nm}(z)} \tag{32}$$

where the transfer functions $T_{an}(z)$, $T_{mn}(z)$, and $T_{mb}(z)$ are evaluated for $\Delta F_{nm}(z) = 0$. Various forms of equations similar to (32) have been widely used in feedback control theory involving the use of signal-flow graphs [20], [21]. It is sometimes referred to as the bilinear theorem.

An interesting consequence of the large-change sensitivity relation (32) can be observed for the class of nonrecursive computable networks. For these structures it has been observed in Section III that if $F_{nm}(z) \neq 0$, then $T_{mn}(z) = 0$. Consequently, for this class of structures, (32) takes the form

$$\begin{aligned}\Delta H(z) &= T_{an}(z)\, T_{mb}(z)\, \Delta F_{nm}(z)\\ &= \frac{\partial H(z)}{\partial F_{nm}(z)}\, \Delta F_{nm}(z).\end{aligned} \tag{33}$$

Thus the change in the system function varies linearly with respect to a change in a branch transmittance in nonrecursive computable networks [5].

Derivative Analysis of Networks

Two other types of analysis, which are sometimes desired in general computer-aided analysis systems, are the evaluation of the group delay of the system function of a network and the slope of the magnitude of the frequency response. For these analyses, the frequency response can be defined as

$$\begin{aligned}H(z)\big|_{z=e^{j\omega}} &= T_{ab}(e^{j\omega})\\ &= |H(e^{j\omega})|\, e^{j\theta(\omega)}\end{aligned} \tag{34}$$

where ω is the angular frequency and $\theta(\omega)$ is the phase. The group delay τ is then given by

$$\tau = -\frac{\partial \theta(\omega)}{\partial \omega} \tag{35}$$

and the slope of the magnitude of the frequency response is given by $\partial |H(e^{j\omega})| / \partial \omega$. General expressions for these quantities can be derived in terms of appropriate system functions in the network and can be expressed in the forms [1], [22]

$$\tau = \sum_{n=1}^{N} \sum_{m=1}^{N} \mathrm{Re}\left[\frac{f_{dnm}\, T_{an}(z)\, T_{mb}(z)\, z^{-1}}{H(z)}\right]\Bigg|_{z=e^{j\omega}} \tag{36}$$

$$\frac{\partial |H(e^{j\omega})|}{\partial \omega} = \frac{1}{|H(e^{j\omega})|} \sum_{n=1}^{N} \sum_{m=1}^{N} \cdot \mathrm{Im}\left[\frac{f_{dnm}\, T_{an}(z)\, T_{mb}(z)\, z^{-1}}{H(z)}\right]\Bigg|_{z=e^{j\omega}} \tag{37}$$

where Re [·] and Im [·] correspond to the operations of taking the real and the imaginary parts, respectively. The double sums in (36) and (37) correspond to the operation of summing over all possible coefficient-delay branches in the network. Alternatively, this sum may be carried out only over those coefficient-delay branches which have nonzero coefficient values f_{dnm} as all other terms in the summation for which $f_{dnm} = 0$ will be zero. Similar analysis techniques for the analysis of group delay and the slope of the magnitude of the frequency response for analog networks have been proposed by Pinel and Blostein [23], [24].

Some of these general analysis techniques have been utilized in a digital network analysis package called CADNAP [1]. It is capable of analyzing networks of arbitrary configuration for system functions, unit-sample responses, and coefficient sensitivities. This package was used for the analysis of the networks given in Section VI.

A Statistical Measure of Word Length

In evaluating and comparing various structures, an important consideration is the required word length. In principle, for any given design, the required word length can be determined exactly by evaluating the frequency response for successively smaller values of coefficient word length until the filter specifications are exceeded. An alternative approach is to base the analysis of coefficient errors on a statistical representation. One of the first attempts to use a statistical measure of coefficient errors for comparison of digital filters was made by Knowles and Olcayto [25]. More recently, a statistical approach for estimating the coefficient word length necessary to keep the magnitude of the system function of a digital filter within a prescribed error bound was proposed by Avenhaus [26] and modified by Crochiere [27]. A similar measure for finite impulse response filters in direct form has been proposed by Chan and Rabiner [28]. The resulting measure of word length by these methods, referred to as statistical word length, is a conservative estimate and can generally predict the necessary coefficient word length to within a couple of bits. This measure is generally more efficient to compute than the actual word length. Consequently, it is useful in studies where general comparisons of coefficient sensitivity properties of structures are desired but where exact values are unnecessary. Such a comparison is presented in Section VI.

A second application of statistical word length is as an aid in optimizing the design of a digital filter for minimum coefficient word length. The statistical word length is useful in this application because it is a more convenient function to minimize than the actual word length which is a nonlinear integer function of the filter design parameters. Improvements of one to three bits have been observed using this approach [27], [29].

To define the statistical word length, assume that it is of interest to design a filter for which the magnitude of the ideal system function is $H_I(\omega)$ with a given tolerance function $\delta(\omega)$. When the filter coefficients are specified to infinite precision, the magnitude of the system function will be denoted as $|H_0(\omega)|$. The magnitude error incurred when finite precision coefficients are actually used will be denoted by $\Delta|H(\omega)|$ so that the true magnitude response $|H(\omega)|$ will be

$$|H(\omega)| = \Delta|H(\omega)| + |H_0(\omega)|.$$

The filter must be designed in such a way that $|H(\omega)|$ approximates the ideal transfer function to within the allowed error deviation.

For a given filter structure, the error $\Delta|H(\omega)|$ can be expressed in terms of a linear first-order approximation:

$$\Delta|H(\omega)| \cong \sum_{i=1}^{m} \frac{\partial|H_0(\omega)|}{\partial c_i} \Delta c_i \tag{38}$$

where the values $c_i (i = 1, 2, \cdots, m)$ represent the ideal infinite precision values for the filter. Now let Q define the quantization step size for rounding of the coefficients and assume that the quantization errors Δc_i have a uniform probability of lying between $-Q/2$ and $Q/2$. The statistical errors in the coefficients then have zero mean and variance $\sigma^2_{\Delta c_i} = Q^2/12$. Assuming that the errors Δc_i are independent, the variance of $\Delta|H(\omega)|$ is

$$\sigma^2_{\Delta|H(\omega)|} \cong \frac{Q^2 S^2(\omega)}{12} \tag{39}$$

where

$$S^2(\omega) = \sum_{i=1}^{m} \left[\frac{\partial|H_0(\omega)|}{\partial c_i}\right]^2. \tag{40}$$

Assuming that $\Delta|H(\omega)|$ has approximately a Gaussian distribution with variance given by (39), we can choose a value $x\sigma_{\Delta|H(\omega)|}$ such that

$$P[|\Delta|H(\omega)|| \leqslant x\sigma_{\Delta|H(\omega)|}] = y$$

or

$$P\left[|\Delta|H(\omega)|| \leqslant \frac{xQS(\omega)}{\sqrt{12}}\right] = y. \tag{41}$$

We will be assured with probability y that

$$|\Delta|H(\omega)|| \leqslant \delta(\omega) - ||H_0(\omega)| - H_I(\omega)|$$

if we choose Q so that at all frequencies of interest

$$\frac{xQS(\omega)}{\sqrt{12}} \leqslant \delta(\omega) - ||H_0(\omega)| - H_I(\omega)|. \tag{42}$$

Then with $q(\omega)$ denoting the maximum allowable quantization step size as a function of frequency,

$$q(\omega) = \frac{\sqrt{12}\,(\delta(\omega) - ||H_0(\omega)| - H_I(\omega)|)}{xS(\omega)}. \tag{43}$$

The coefficient word length of the filter can be defined as

$$W = 1 + i_M - i_L \tag{44}$$

where 2^{i_M} is the value of the most significant bit and 2^{i_L} the least significant bit (the sign bit is not counted in (44)); i_M is dictated by the magnitude of the coefficients and i_L by the coefficient step size. In particular, using (43), i_L is approximated as

$$i_L \cong \log_2 \left[\min_{\omega} \frac{\sqrt{12}\,(\delta(\omega) - ||H_0(\omega)| - H_1(\omega)|)}{xS(\omega)}\right] \tag{45}$$

and the statistical word length W is given by

$$W = 1 + i_M - \log_2 \left[\min_{\omega} \frac{\sqrt{12}\,(\delta(\omega) - ||H_0(\omega)| - H_I(\omega)|)}{xS(\omega)}\right]. \tag{46}$$

To compute the statistical word length, it is necessary to compute the function $S(\omega)$ as given by (40). This can be done using the techniques discussed previously in this section.

For the case of equiripple filters, it is generally possible to make use of the fact that $q(\omega)$ will usually be a minimum at those frequencies for which $(\delta(\omega) - ||H_0(\omega)| - H_I(\omega)|)$ is a minimum, i.e., the frequencies at which the ripples occur. In this case, it is generally sufficient to evaluate (43) only at those frequencies in order to evaluate (46).

VI. Digital Filter Structures and Sensitivity Comparisons

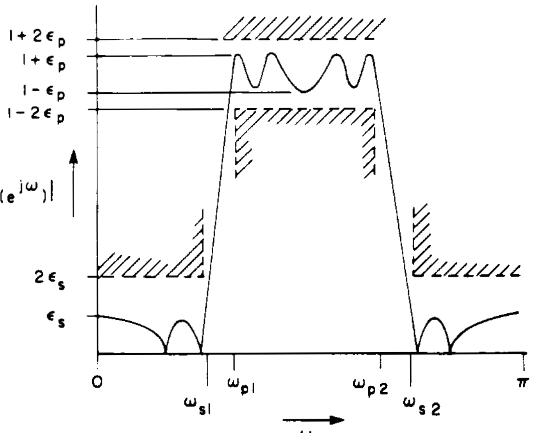
The choice of a filter structure is an important consideration in the design of a digital filter. This choice directly affects the word-length requirements for the signals and coefficients in the filter and determines its characteristics such as roundoff noise, limit cycle behavior, and dynamic range. In response to this problem, many types of structures were recently proposed by various authors as candidates for filter implementation. Some of these structures are examined in this section with regard to coefficient sensitivity for the case of a specific bandpass filter example used by Avenhaus [26]. The comparison is made on the basis of statistical word length and the number of required multiplies and adds. In Section VII, a further comparison will be made on the basis of parallelism and serialism. The results should not be taken as final or conclusive for all examples or applications but, rather, they should be used to gain insight into the types of structures that appear to be desirable and look promising for further investigation.

The Structures

The filter specifications used for the comparison are given in Fig. 7. Six different structures were synthesized for this example. No attempt will be made to explain the details for synthesis of each of these different types of structures as these details are readily available in the references. Further examples also can be found in [1] and [30].

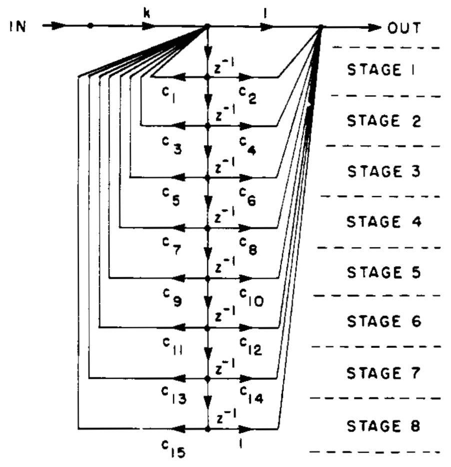
The first three types of structures which were examined are the well-known direct-form II structure, the cascade-form structure (with second-order direct-form II sections), and the parallel-form structure, given in Figs. 8, 9, and 10, respectively. These structures correspond to representations of the system function as a ratio of polynomials, a product of ratios of second-order polynomials, and a partial-fraction expansion, respectively. They were perhaps the first classes of structures to be considered in digital signal processing. Their properties have been studied extensively and are well known [19], [30]-[32]. Another way of expressing system functions is by means of continued-fraction expansions. Mitra and Sherwood [33], [34] have proposed several classes of structures which can be realized by this approach. Fig. 11 gives one such structure.

A major category of structures which have received attention recently are the ladder or lattice structures. Unlike the previous categories, these structures are not synthesized by representing the system function in terms of some factorization or expansion. Instead, the mechanisms behind the synthesis procedures for these structures are closely related to the conventional two-port network procedures used in analog circuit theory. As a consequence, these structures are composed of building blocks which take the form of two-input, two-output networks. An example of this class of structures is given in Fig. 12. It is synthesized according to procedures suggested by Gray and Markel [35].



$\epsilon_p = 0.0157816$	$\epsilon_s = 0.00526063$
$\omega_{p1} = 0.4111111\ \pi$	$\omega_{s1} = 0.3847015\ \pi$
$\omega_{p2} = 0.4666667\ \pi$	$\omega_{s2} = 0.4944444\ \pi$

Fig. 7. Filter design specifications for eighth-order bandpass example.



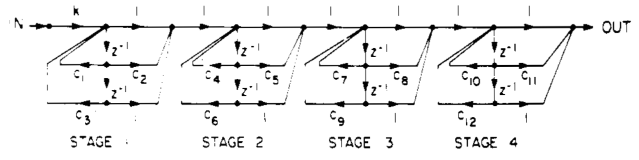
IN
k
1
OUT
z^{-1}
c_1 c_2 c_3 c_4 c_5 c_6 c_7 c_8 c_9 c_{10} c_{11} c_{12} c_{13} c_{14} c_{15}
1
STAGE 1
STAGE 2
STAGE 3
STAGE 4
STAGE 5
STAGE 6
STAGE 7
STAGE 8

$k = 0.005656462366$

$c_1 = 1.479757145$	$c_2 = -1.387818861$
$c_3 = -4.548129731$	$c_4 = 3.990834175$
$c_5 = 4.352241828$	$c_6 = -3.801090558$
$c_7 = -6.802822262$	$c_8 = 5.978059098$
$c_9 = 4.094876357$	$c_{10} = -3.801090558$
$c_{11} = -4.026604318$	$c_{12} = 3.990834175$
$c_{13} = -0.7833447265$	$c_{14} = -1.387818861$
$c_{15} = 1.232007442$	

Fig. 8. Direct-form II structure (Example 1).

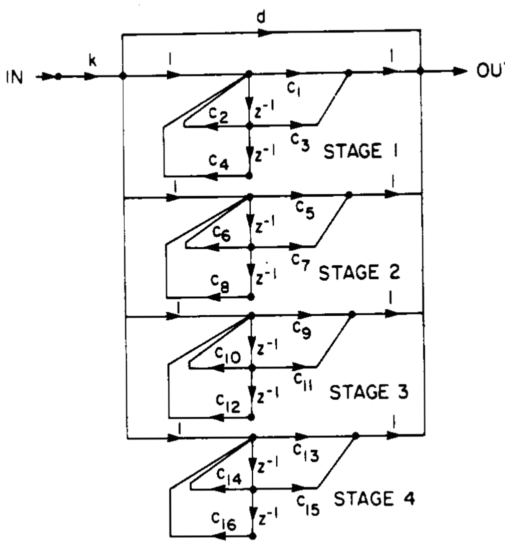
Another class of structures synthesized in a manner similar to that of classical analog structures are the wave digital filters of Fettweis [36], [37]. This class of structures recently received considerable attention [38]-[41]. The synthesis procedure is similar to that of classical analog circuits; in fact, at present, the easiest way to design them is via the design of appropriate analog prototype structures. The digital structures are obtained by converting these analog designs to corresponding digital designs. The resulting digital structures are discrete simulations of the wave-flow diagrams of the analog structures. As such, digital structures designed in this way exhibit properties similar to their analog counterparts. Just as there are many ways to synthesize analog structures, there are also many



$k = 0.005656462366$

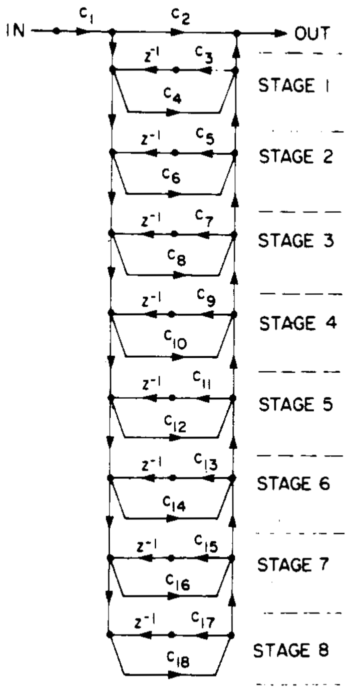
$c_1 = 0.2855823838$	$c_4 = 0.4457342317$	$c_7 = 0.196901665$	$c_{10} = 0.5515388641$
$c_2 = 0.451259755$	$c_5 = -1.09869455$	$c_8 = -0.0099665157$	$c_{11} = -0.7304169706$
$c_3 = -0.9116860616$	$c_6 = -0.913078673$	$c_9 = -0.9695353354$	$c_{12} = -0.9705899009$

Fig. 9. Cascade-form (direct-form II sections) structure (Example 2).



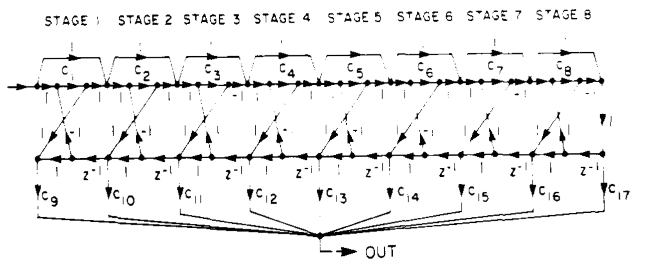
$k = 0.10000$	$d = 0.07220910762$
$c_1 = 0.2034508401$	$c_9 = -0.2052671178$
$c_2 = 0.2855823838$	$c_{10} = 0.196901665$
$c_3 = 0.9026252904$	$c_{11} = -0.2413054652$
$c_4 = -0.9116860616$	$c_{12} = -0.9695353354$
$c_5 = 0.182701196$	$c_{13} = -0.1965294023$
$c_6 = 0.4457342317$	$c_{14} = 0.5515388641$
$c_7 = -0.9457352763$	$c_{15} = 0.298888793$
$c_8 = -0.913078673$	$c_{16} = -0.9705899009$

Fig. 10. Parallel-form structure (Example 3).



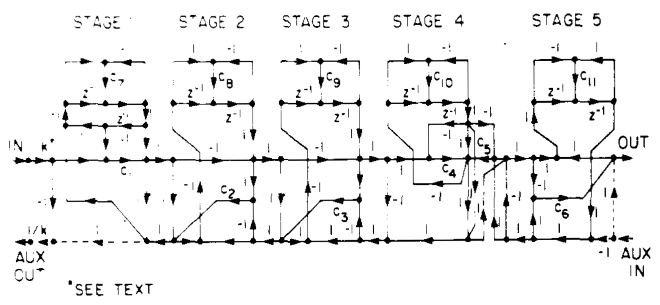
c_1	$= 0.0002828231183$
c_2	$= 25.53154355$
c_3	$= -0.2117906668$
c_4	$= 1.017015349$
c_5	$= 0.1736820909$
c_6	$= -12.7837888$
c_7	$= 0.4649449432$
c_8	$= -1.606209516$
c_9	$= -0.2767502193$
c_{10}	$= 7.610142205$
c_{11}	$= 3.939247317$
c_{12}	$= -0.4301416747$
c_{13}	$= -0.9905202523$
c_{14}	$= 0.6779657016$
c_{15}	$= -21.82668197$
c_{16}	$= -0.4193825017$
c_{17}	$= 0.2481853405$
c_{18}	$= 0.4028556869$

Fig. 11. Continued-fraction expansion structure type 1B (Example 4).


STAGE 1
STAGE 2
STAGE 3
STAGE 4
STAGE 5
STAGE 6
STAGE 7
STAGE 8
c_1 c_2 c_3 c_4 c_5 c_6 c_7 c_8
z^{-1}
c_9 c_{10} c_{11} c_{12} c_{13} c_{14} c_{15} c_{16} c_{17}
OUT

c_1 = 0.7833447265	c_9 = 0.005656462366	c_{17} = 0.8446328267
c_2 = -0.1885428229	c_{10} = 0.0002916124024	
c_3 = 0.9843741951	c_{11} = -0.001699657657	
c_4 = -0.1920962931	c_{12} = -0.06980454614	
c_5 = 0.9936376827	c_{13} = 0.2031623313	
c_6 = -0.1890252997	c_{14} = 0.01061747099	
c_7 = 0.9914182038	c_{15} = -0.01641846145	
c_8 = -0.1964790194	c_{16} = -1.031187912	

Fig. 12. Ladder structure by Gray and Markel [35] (Example 5).



k = 3.1194961	c_5 = 0.007805575218	c_9 = -0.3698270573
c_1 = 0.06502370945	c_6 = 0.0574957168	c_{10} = -0.1915378541
c_2 = 0.2235692269	c_7 = -0.1915378541	c_{11} = 0.1915378541
c_3 = 0.5809284606	c_8 = 0.0003565321277	
c_4 = 0.1626985222		

Fig. 13. One type of wave digital filter (Example 6).

TABLE I

Example	Network Description	Statistical Word Length (bits) Passband W_p	Stopband W_s	i_M	No. of Multiplies	No. of Adds	Bit × Multiply Product
1	direct-form II	20.86	10.85	2	16	16	334
2	cascade (direct-form II)	11.33	5.76	0	13	16	147
3	parallel form	10.12	7.95	−1	18	16	182
4	continued-fraction 1B	22.61	14.46	4	18	16	408
5	ladder (Gray and Markel [35])	13.97	10.81	0	17	32	238
6	wave digital filter (Fettweis [36], [37])	11.35[a]	5.67	−1[a]	12/11[a]	31	136/125

[a]See text.

ways to synthesize the wave digital filters. An example is given in Fig. 13.

Comparisons of the Structures

Comparisons of these structures have been made on the basis of statistical word length and the number of additions and multiplications required. The results are tabulated in Table I. The statistical word lengths were evaluated according to (46), with $x = 2$ (e.g., $y = 0.95$). This measure represents a conservative estimate of the coefficient word length (for fixed-point arithmetic) necessary to meet the filter specifications in Fig. 7.

Separate evaluations of the statistical word lengths were made for the passband and stopband specifications in order to obtain a more general picture of the sensitivity characteristics of each type of structure. Obviously, the largest of the two word-length estimates must be chosen to meet the specifications at all frequencies.

The statistical word lengths are generally conservative estimates of the necessary word length (for coefficient rounding) by approximately 2 bits. For example, the cascade (direct-form II) structure of Example 2 requires 10-bit coefficients to meet the specifications in Fig. 7, whereas the statistical word length predicts 11.33 bits. These word lengths do not include the extra bit required for representing the sign. One exception to this occurs for the case of the wave digital filters. For this class of structures, the statistical word length for meeting the passband constraints appears, from observations, to be conservative by approximately 4 bits. For example, the actual word length required for the wave digital filter to meet the passband requirements was 7 bits, whereas the statistical word length was 11.35 bits. This can be attributed to the special pseudopassive and pseudolossless properties of these structures which are inherited from their analog counterparts [40], [41].

The value of i_M for each structure (see Table I) corresponds to the power of 2 represented by the most significant bit in the binary representation of the coefficients. It is chosen such that the largest coefficient magntiude (except the scale factor) in the structure is within the range 2^{i_M} and 2^{i_M+1}.

The number of multiplications and additions required for each structure are tabulated in Table I. It should be emphasized that these numbers are for the specific bandpass elliptic filter discussed and are not necessarily representative for arbitrary system functions. Also indicated in Table I is the bit-multiplier product for each structure. It represents a rough measure of the total amount of computation which must be performed in each structure.

The comparisons of the structures in this section have been made primarily on the basis of coefficient sensitivity and the number of multiplies and adds. These attributes are clearly important in the choice of a structure, particularly since they vary so widely from one structure to another, as can be seen in Table I. Of equal importance in the choice of a structure are the issues of roundoff noise, limit cycle effects, and dynamic range. Although these characteristics have not been considered in this comparison, the authors feel that it is important to mention them in order that these results may be viewed in a proper perspective to the overall issues of filter structure design.

VII. Parallelism and Serialism in Structures

In the previous section we carried out a comparison of some structures based on coefficient sensitivity. There are, however, many other considerations to be taken into account in choosing a filter structure. In this section we suggest one such additional consideration, that of the potential parallelism inherent in the structure. In some applications it is of interest to implement a digital filter at extremely high sampling rates, requiring parallel computation; in such cases, a structure must be chosen which permits the required degree of parallelism. These considerations are similar in some respects to those of computation structures for multiprocessor design [42]-[44].

Our approach to discussing parallelism in structures is through the notion of precedence, as presented in Section III. The precedence form of a structure reveals the order in which nodes must be evaluated and, consequently, the smaller the number of precedence levels, the higher the degree of parallelism in the structure. By beginning from the node precedence relations, it is possible to develop a coefficient precedence graph which specifies the number of multiplier precedence levels. One of the major fundamental operations generally involved in the implementation of a digital filter structure is that of multiplication. Since the time needed to perform a multiply is often much greater than that needed for an addition or for other operations in the structure, the time necessary to compute one filter cycle can be roughly approximated in terms of multiplier cycle times (e.g., the time necessary to compute a multiply). For this situation, the parallel/serial tradeoffs of a structure (e.g., speed/cost tradeoffs) can be roughly approximated in terms of its multiplier precedence relations. To develop the multiplier precedence relations, it is convenient to draw the network such that all delay branches have unity coefficients and, consequently, do not require multiplies. All multiplications necessary for the implementation of the structure are then represented by the coefficient branches in the structure and appear in the precedence form of the network, as illustrated in Fig. 14. This form of network is identical to that in Fig. 2(c) and the precedence relations for coefficients are interpreted in a manner similar to that for nodes.

A precedence graph for those multiplies which must be performed in the implementation of the structure can now be extracted from the foregoing coefficient precedence relations. It must be recognized that some of the coefficients associated with the coefficient sets in Fig. 14 may not be required to be implemented as multiplies, namely, those which are unity. Consequently, the coefficient precedence relations must be modified to account for these situations. For example, if a coefficient which belongs to coefficient set $\{c_2\}$ is preceded only by unity gain coefficients in set $\{c_1\}$, then the multiply associated with this coefficient can be performed in the same time slot as the multiplies for coefficients in $\{c_1\}$. With these situations taken into account, the procedure leads to a set of multiplier precedence relations, as depicted by the multiplier precedence graph of Fig. 15. The circles in this graph represent the multiplies which must be performed and the arrows merely indicate the precedence relations. The multiplies associated with set $\{m_1\}$ can all be performed simultaneously without conflict in approximately the same time slot. Once these multiplies have been performed, all of the multiplies in set $\{m_2\}$ can be performed simultaneously in the next time slot, etc. Coefficients in set $\{m_1\}$ correspond to all of the coefficients in $\{c_1\}$ which must be implemented as multiplies plus all of the coefficients in sets $\{c_2\}$, $\{c_3\}$, $\cdots$, $\{c_{f-1}\}$ which are preceded only by coefficients that do not have to be implemented as multiplies. Coefficients in set $\{m_2\}$ correspond to those remaining coefficients in set $\{c_2\}$ plus any

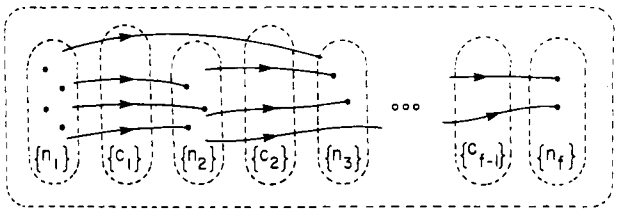


Fig. 14. Precedence form of a network illustrating precedence of coefficients.

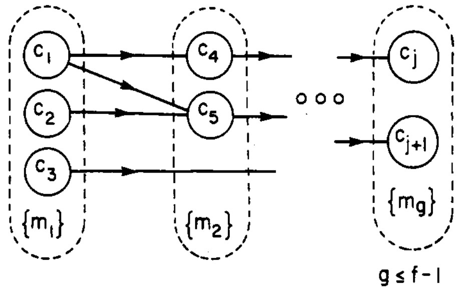


Fig. 15. Multiplier precedence graph.

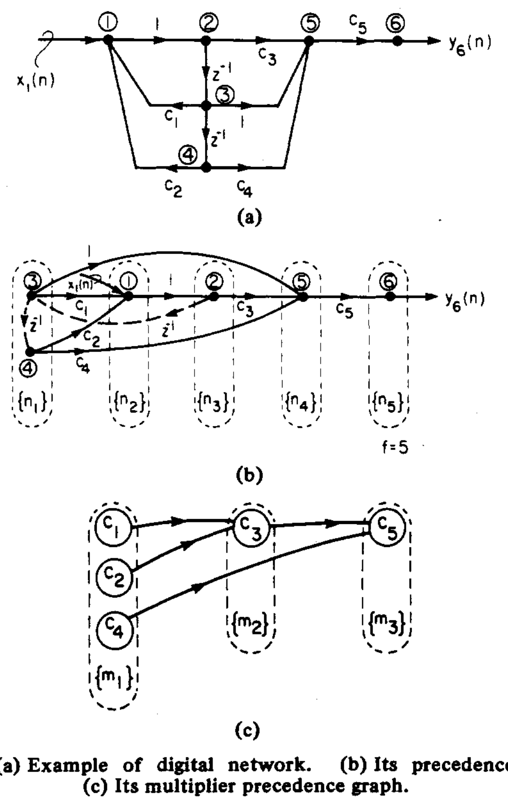


Fig. 16. (a) Example of digital network. (b) Its precedence form. (c) Its multiplier precedence graph.

coefficients in sets $\{c_3\}$, $\{c_4\}$, $\cdots$, $\{c_{f-1}\}$ which can be moved up without conflict in the multiplier precedence relations. By continuing this procedure, the multiplier precedence graph can be generated.

An example of a network, its precedence form, and its multiplier precedence graph are illustrated in Fig. 16. There are five levels of serialism with respect to nodes in this network and three levels of serialism with respect to the multiplies.

The multiplier precedence graph is a useful device for analyzing the tradeoffs between the number of multipliers M used in the implementation and the approximate minimum computation time t_m (in multiplier cycles) that it takes to

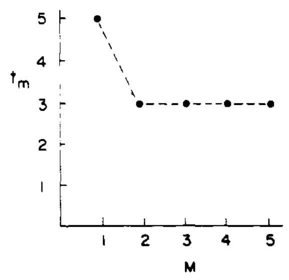


Fig. 17. Minimum multiplier cycle time t_m versus number of multiplies M for structure in Fig. 16.

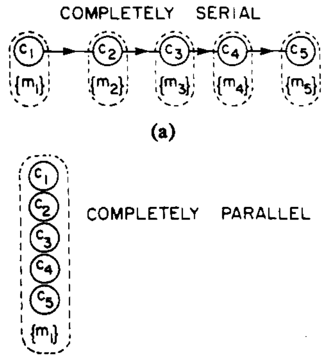


Fig. 18. Examples of multiplier precedence graphs. (a) Completely serial. (b) Completely parallel.

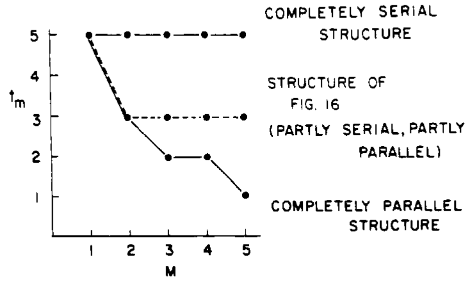


Fig. 19. Comparison between t_m versus M graphs for completely serial and completely parallel structures.

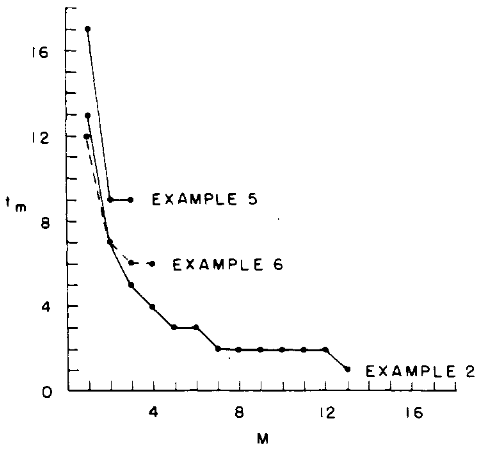


Fig. 20. Comparisons of t_m versus M graphs for some circuit examples given in Section VI.

compute one filter cycle. A useful way to illustrate this tradeoff is to plot t_m against M. As an example (in the network of Fig. 16), if $M = 1$, then all of the multiplies must be computed serially and consequently $t_m \simeq 5$. If $M = 2$, coefficients c_1 and c_2 can be implemented in the first multiplier time slot, c_3 and c_4 in the second time slot, and c_5 in the third time slot. Therefore, $t_m \simeq 3$. As this is the inherent level of serialism in this particular structure, it is impossible to do better than this by using more multipliers in the implementation. The corresponding plot of t_m versus M is given in Fig. 17.

It is interesting to compare the t_m versus M plot for a structure against other possible structures that can realize the same system function. In particular, two extremes can be considered. A structure will be defined to be *completely serial* with respect to its multiplies if every level $\{m_i\}$ in its multiplier precedence graph contains only one possible multiply, as illustrated in Fig. 18(a). Alternatively, a structure will be defined to be *completely parallel* with respect to its multiplies if its multiplier precedence graph contains only one level $\{m_1\}$ as indicated by Fig. 18(b). Plots of the t_m versus M for these two extremes and for the network of Fig. 16 are given in Fig. 19.

For a completely serial structure, the t_m versus M plots correspond to the relation

$$t_m \simeq C \tag{47}$$

where C is the total number of multiplies that must be performed in the structure. For a completely parallel structure, this relation is

$$t_m \simeq \operatorname{Int}\left[\frac{C}{M}\right] \tag{48}$$

where Int [·] corresponds to the operation of rounding up the number in brackets to the next largest integer. As suggested by Fig. 19, many practical structures may have t_m versus M plots that lie somewhere between these two limits. Fig. 20 presents a comparison of t_m versus M plots for a number of the structures discussed in Section VI.

In some cases it may be possible to increase the amount of parallelism in the computation of a network by appropriately computing different node values (or multiplies) in the network simultaneously for different time indices n, i.e., by using *pipeline processing*. This can permit a speedup in the effective cycle time in the computation of a structure at the expense of introducing delay into some of the transfer functions of the network. This can be achieved, for example, in the network of Fig. 16 by performing the multiplies for c_1, c_2, and c_4 for time index $n + 2$ in the same time slot that multiply c_3 is performed for time index $n + 1$ and multiply c_5 is performed for time index n. This technique appears, at first, to violate the preceding concepts of inherent serialism and parallelism of a network but, in fact, it does not.

An alternate way of viewing pipelining is that it is a technique by which delays are inserted into appropriate paths in the structure in order to increase its parallelism or reduce its serialism. In effect, by using the pipeline scheme in the example given, the actual structure being computed is not that of Fig. 16 but instead the structure depicted in Fig. 21(a), where the computations for all of the multiplies are being performed in the same time slot for time index n. It is easy to see from Fig. 21(b) and (c) that this form of the structure has a greater amount of parallelism than the structure of Fig. 16 and that, in fact, it is completely parallel with respect to its multiplies. Its system function, however, has an extra delay of two unit delays associated with it over that for the network in Fig. 16.

As already illustrated, pipelining can be interpreted as a process by which extra unit delays are added in appropriate parts

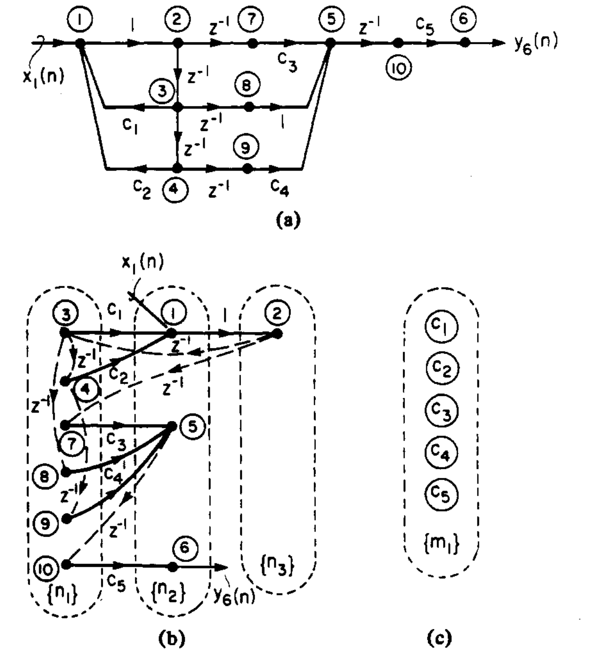


Fig. 21. (a) Pipelined form of network in Fig. 16. (b) Its precedence form. (c) Its multiplier precedence graph.

of a structure in order to reduce its inherent serialism at the expense of introducing extra delay in certain transfer functions of the structure. These delays can be inserted only into those parts of a structure which are not associated with loops or feedback paths. Also, if a structure contains several feedforward paths in parallel and a unit delay is added in one of these paths, then a unit delay must also be added in each of the paths which are in parallel with it in order to preserve the feedforward cancellations in the structure. As a result, nonrecursive structures can always be completely pipelined as they contain no loops of any kind. Recursive structures may be only partially pipelined because they contain loops. Only those parts of a recursive structure which are not associated with loops may be pipelined.

VIII. Conclusions

We have reviewed the analysis, representation, and evaluation of digital filter structures. We have centered the discussion around the notation of linear signal-flow graphs and the associated matrix representation. Using this notation, we have identified a number of basic network properties and have applied them to an analysis of filter structures.

The basic network properties presented can in some sense be interpreted as contributing to a network theory applicable to digital filters. In some instances, the parallel with analog networks is very strong; in other instances, it is not. However, it is our feeling that just as network theory for analog filters has been an important framework for the analysis and synthesis of filter structures, a rich network theory exists which can serve the same purpose for digital filters. Hopefully, the ideas presented in this paper represent a start in this direction.

In Section VI we have presented a comparison of a number of filter structures based on coefficient sensitivity. The development of this comparison represents a useful application of many of the theoretical ideas presented in the paper. It is important to stress, however, that this comparison is based on only one example. Furthermore, it is important to keep in mind that coefficient sensitivity is only one of many factors to be taken into account in comparing structures. In Section VI we have tried to address ourselves to another factor, that of the degree of parallelism inherent in a structure. Our discussion represents only a preliminary attempt to include this consideration in the evaluation of structures. Over the next several years, we anticipate considerable activity directed toward a refinement and quantification of the parameters involved in comparing digital filter structures.

Acknowledgment

The authors wish to thank Prof. H. W. Schuessler, Dr. L. R. Rabiner, and Dr. J. F. Kaiser for their careful reading of this manuscript during its preparation and their helpful comments.

References

[1] R. E. Crochiere, "Digital network theory and its application to the analysis and design of digital filters," Ph.D. dissertation, Dep. Elec. Eng., M.I.T., Cambridge, Mass., May 1974.
[2] S. J. Mason, "Feedback theory–Some properties of signal flow graphs," *Proc. IRE*, vol. 41, pp. 1144-1156, Sept. 1953.
[3] ——, "Feedback theory–Further properties of signal flow graphs," *Proc. IRE*, vol. 44, pp. 920-926, July 1956.
[4] P. M. DeRusso, R. J. Roy, and C. M. Close, *State Variables for Engineers*. New York: Wiley, 1965.
[5] R. E. Crochiere, "Some network properties of digital filters," Res. Lab. Electron., M.I.T., Cambridge, Mass., Quart. Progr. Rep. 107, pp. 103-112, Oct. 15, 1972; also in *Proc. 1973 IEEE Int. Symp. Circuit Theory*, pp. 146-148.
[6] P. Penfield, Jr., R. Spence, and S. Duinker, *Tellegen's Theorem and Electric Networks*. Cambridge, Mass.: M.I.T. Press, 1970.
[7] S. W. Director and R. A. Rohrer, "The generalized adjoint network and network sensitivities," *IEEE Trans. Circuit Theory*, vol. CT-16, pp. 318-323, Aug. 1969.
——, "Automated network design–The frequency-domain case," *ibid.*, vol. CT-16, pp. 330-337, Aug. 1969.
[8] R. E. Seviora and M. Sablatash, "A Tellegen's theorem for digital filters," *IEEE Trans. Circuit Theory*, vol. CT-18, pp. 201-203, Jan. 1971.
[9] A. Fettweis, "A general theorem for signal-flow networks, with applications," *Arch. Elek. Übertragung*, vol. 25, pp. 557-561, Dec. 1971; also in *Dig. Tech. Papers, 1971 IEEE Int. Symp. Electrical Network Theory*, pp. 3-4.
[10] M. L. Blostein, "On the application of certain network concepts to arbitrary systems," in *Proc. 1972 IEEE Int. Symp. Circuit Theory*, pp. 213-217.
[11] A. Y. Lee, "Topological properties of the summing and branching matrices of signal-flow graphs and the application to sensitivity analysis," in *Proc. 6th Asilomar Conf. Circuits and Systems*, Nov. 1972.
[12] ——, "Computer-aided equation formulation and sensitivity analysis for discrete systems," in *Proc. 16th Midwest Symp. Circuit Theory*, Apr. 1973.
[13] L. B. Jackson, "On the interaction of roundoff noise and dynamic range in digital filters," *Bell Syst. Tech. J.*, pp. 159-184, Feb. 1970.
[14] G. E. Forsythe and C. B. Moler, *Computer Solution of Linear Algebraic Systems*. Englewood Cliffs, N.J.: Prentice-Hall, 1967.
[15] *Proc. IEEE* (*Special Issue on Computer-Aided Design*), vol. 55, Nov. 1967.
[16] *IEEE Trans. Circuit Theory* (*Special Issue on Computer-Aided Circuit Design*), vol. CT-18, Jan. 1971.
[17] A. V. Oppenheim and C. J. Weinstein, "Effects of finite register length in digital filtering and the fast Fourier transform," *Proc. IEEE*, vol. 60, pp. 957-976, Aug. 1972.
[18] L. B. Jackson, "Roundoff-noise analysis for fixed-point digital filters realized in cascade or parallel form," *IEEE Trans. Audio Electroacoust.*, vol. AU-18, pp. 107-122, June 1970.
[19] A. V. Oppenheim and R. W. Schafer, *Digital Signal Processing*, in press.
[20] J. Truxal, *Automatic Feedback Control Synthesis*. New York: McGraw-Hill, 1955.
[21] I. M. Horowitz, *Synthesis of Feedback Systems*. New York: Academic Press, 1963.
[22] R. E. Crochiere, "Computational methods for sensitivity analysis of digital filters," Res. Lab. Electron., M.I.T., Cambridge, Mass., Quart. Progr. Rep. 109, pp. 113-123, Apr. 15, 1973.
[23] J. F. Pinel and M. L. Blostein, "Computer techniques for the frequency analysis of linear electrical networks," *Proc. IEEE*, vol. 55, pp. 1810-1819, Nov. 1967.
[24] M. L. Blostein, "Some bounds on sensitivity in RLC networks,"

in *Proc. 1st Allerton Conf. Circuits and Systems*, Oct. 1963.

[25] J. B. Knowles and E. M. Olcayto, "Coefficient accuracy and digital filter response," *IEEE Trans. Circuit Theory*, vol. CT-15, pp. 31-41, Mar. 1968.

[26] E. Avenhaus, "On the design of digital filters with coefficients of limited word length," *IEEE Trans. Audio Electroacoust.*, vol. AU-20, pp. 206-212, Aug. 1972.

[27] R. E. Crochiere, "A new statistical approach to the coefficient word length problem for digital filters," *IEEE Trans. Circuits Syst.* (*Special Issue on Digital Filtering and Image Processing*); also in *Proc. 1974 IEEE Int. Symp. Circuits and Systems*, pp. 1-3.

[28] D. S. K. Chan and L. R. Rabiner, "Analysis of quantization errors in the direct form for finite impulse response digital filters," *IEEE Trans. Audio Electroacoust.*, vol. AU-21, pp. 354-366, Aug. 1973.

[29] G. Dehner, "On the design of digital Cauer filters with coefficients of limited word length," submitted to *Arch. Elek. Ubertragung*.

[30] R. E. Crochiere, "A comparison of digital filter structures on the basis of coefficient word length," Res. Lab. Electron., M.I.T., Cambridge, Mass., Progr. Rep. 115, Jan. 1975.

[31] J. F. Kaiser, "Digital filters," in *Systems Analysis by Digital Computer*, F. F. Kuo and J. F. Kaiser, Eds. New York: Wiley, 1966, ch. 7.

[32] B. Gold and C. M. Rader, *Digital Processing of Signals*. New York: McGraw-Hill, 1969.

[33] S. K. Mitra and R. J. Sherwood, "Canonic realizations of digital filters using the continued fraction expansion," *IEEE Trans. Audio Electroacoust.*, vol. AU-20, pp. 185-194, Aug. 1972.

[34] ——, "Digital ladder networks," *IEEE Trans. Audio Electroacoust.*, vol. AU-21, pp. 30-36, Feb. 1973.

[35] A. H. Gray and J. D. Markel, "Digital lattice and ladder filter synthesis," *IEEE Trans. Audio Electroacoust.*, vol. AU-21, pp. 491-500, Dec. 1973.

[36] A. Fettweis, "Some principles of designing filters imitating classical filter structures," *IEEE Trans. Circuit Theory*, vol. CT-18, pp. 314-316, Mar. 1971.

[37] ——, "Digital filter structures related to classical filter networks," *Arch. Elek. Übertragung*, vol. 25, pp. 79 ff, Feb. 1971.

[38] A. Sedlmeyer and A. Fettweis, "Realization of digital filters with true ladder configuration," in *Proc. 1973 IEEE Int. Symp. Circuit Theory*, pp. 149-152; also *Int. J. Circuit Theory Appl.*, vol. 1, no. 1.

[39] A. Fettweis and K. Meerkötter, "Suppression of parasitic oscillations in wave digital filters," in *Proc. 1974 IEEE Int. Symp. Circuits and Systems*, pp. 682-686.

[40] R. E. Crochiere, "Digital ladder structures and coefficient sensitivity," *IEEE Trans. Audio Electroacoust.*, vol. AU-20, pp. 240-246, Oct. 1972.

[41] A. Fettweis, "Pseudopassivity, sensitivity and stability of wave digital filters," *IEEE Trans. Circuit Theory*, vol. CT-19, pp. 668-673, Nov. 1972.

[42] J. L. Baer, "A survey of some theoretical aspects of multiprocessing," *Computing Surveys (ACM)*, vol. 5, pp. 31-80, Mar. 1973.

[43] J. Allen and R. Gallager, *Computation Structures*, Notes for a course at M.I.T. on Computation Structures, Course 6.032.

[44] R. T. Prosser, "Applications of Boolean matrices to the analysis of flow diagrams," Lincoln Lab., M.I.T., Lexington, Mass., Tech. Rep. 217, Jan. 22, 1960.

FIR Digital Filter Design Techniques Using Weighted Chebyshev Approximation

LAWRENCE R. RABINER, MEMBER, IEEE, JAMES H. McCLELLAN, MEMBER, IEEE, AND THOMAS W. PARKS, MEMBER, IEEE

Invited Paper

***Abstract*—This paper discusses the various approaches to designing FIR digital filters using the theory of weighted Chebyshev approximation. The different design techniques are explained and compared on the basis of their capabilities and limitations. The relationships between filter parameters are briefly discussed for the case of low-pass filters. Extensions of the theory to the problems of magnitude and complex approximation are also included, as are some recent results on the design of two-dimensional FIR filters by transformation.**

I. Introduction

IN THE PAST few years, powerful computer optimization algorithms have been developed to solve the design problem for finite-duration impulse response (FIR) digital filters. It is the purpose of this paper to review these techniques in the light of Chebyshev approximation theory and to describe some of the extensions of this theory.

Manuscript received June 10, 1974; revised September 25, 1974. This work was supported in part by NSF Grant GK-23697.

L. R. Rabiner is with Bell Laboratories, Murray Hill, N.J. 07974.
J. H. McClellan is with the M.I.T. Lincoln Laboratory, Lexington, Mass. 02173.
T. W. Parks is with the Department of Electrical Engineering, Rice University, Houston, Tex. 77001.

FIR digital filters possess certain desirable properties which make them attractive for digital signal processing applications. Among these are the ability to have exactly linear phase and the absence of stability problems in nonrecursive realizations. While long sequences are sometimes necessary to achieve sharp cutoff filters, use of the fast Fourier transform (FFT) can make the realization of such filters computationally competitive even with sharp cutoff infinite-duration impulse response (IIR) elliptic filters.

The process of designing and realizing a digital filter to meet some desired specifications consists of five basic steps.

1) Choose a design technique and convert the desired specifications into a precise mathematical formulation in order to approximate the ideal filter shape.

2) Solve the approximation problem to determine the filter coefficients which minimize a performance measure.

3) Choose a specific structure in which the filter will be realized and quantize the resulting filter coefficients to a fixed word length.

4) Quantize the digital filter variables, i.e., the input, output, and intermediate variable word lengths.