A HARDWARE REALIZATION TECHNIQUE FOR N-PORT ADAPTORS IN WAVE DIGITAL FILTERS WITH TWO'S COMPLEMENT ARITHMETIC

BY

Paul R. Moon

A THESIS PRESENTED TO THE

FACULTY OF GRADUATE STUDIES

UNIVERSITY OF MANITOBA

IN PARTIAL FULFILLMENT

OF THE REQUIREMENTS FOR THE DEGREE

MASTER OF SCIENCE

WINNIPEG, MANITOBA

AUGUST 1978

A HARDWARE REALIZATION TECHNIQUE FOR N-PORT ADAPTORS IN WAVE DIGITAL FILTERS WITH TWO'S COMPLEMENT ARITHMETIC

BY

PAUL R. MOON

A dissertation submitted to the Faculty of Graduate Studies of the University of Manitoba in partial fulfillment of the requirements of the degree of

MASTER OF SCIENCE

ABSTRACT

This thesis is a study of hardware realizations for n-port adaptors in wave digital filters. The theory of wave digital filters is presented with the scattering matrix formulation for n-port adaptors. The realization of the n-port adaptor matrix, including the minimal n-port adaptor derived for state variables, is treated as a numerical problem, where the n-port adaptor is related to an equivalent structure of known adaptors, if possible, for optimization of the matrix entries. A numerical decomposition of the adaptor matrix to add and shift operations is given based on a binary series expansion in the canonical signed digit code with matrix coefficients. This decomposition furnishes a program for a hardware or computer implementation of the adaptor matrix with two's complement arithmetic which introduces no roundoff error during computation while the required wordlength in every addition operation is the same as the wordlength at the input to the matrix. That is, no increase of adder wordlengths is required to hold underflows while the roundoff error during recursive filter operation is limited to the final truncations before feedback. It is also found that the decomposition is efficient, but most likely not optimal, in terms of the number of adders required. A third order elliptic lowpass filter example has been realized by the decomposition and the hardware implementation was simulated in assembly language on a minicomputer. The number of adders required for this example compares favorably with wave digital realizations of the same filter by other methods. It is concluded that n-port adaptors present a viable method of hardware realization for wave digital filters.

ACKNOWLEDGEMENT

The author expresses his appreciation to Professor G. O. Martens for his interest and guidance during the course of this work. The author also wishes to thank Mr. A. T. Ashley and Dr. H. H. Le, with whom he had many useful discussions. Financial support from the University of Manitoba, the National Research Council of Canada, and the Canadian Council of Professional Engineers is also greatly appreciated.

CONTENTS

1. INTRODUCTION

The use of digital filtering for the processing of discrete time sequences, often derived from sampled analog signals, has become well established [4, 5, 6]. Digital filter designs have the significant advantages of precise specification of critical frequencies, freedom from component tolerance effects due to aging, and absence of limitations due to component size at low frequencies and high component Q requirements. Most digital filter structures have been based on a direct realization from the transfer function, which is quite satisfactory but exhibits high sensitivity to the multiplier coefficients for high order systems and in some cases limit cycle oscillations due to the nonlinear effects of rounding or truncation during computation. The sensitivity problem, which requires large wordlengths in the multiplier coefficients, is often circumvented by realizing a high order system as a cascade of first and second order sections. The limit cycle problem has been analyzed at length, but solutions are known only for first and second order sections.

The wave digital filter is a new structure, introduced by Fettweis [12, 13], based on a network topological translation with scattering variables of an analog prototype filter to a discrete time system using a bilinear transformation between the complex variables of the Laplace and z- transform domains. The frequency response of this wave digital structure exhibits a low sensitivity to multiplier coefficients which is related to the low sensitivity of the analog prototype to element variations. On a hardware basis this is advantageous but counterbalanced by the fact that the wave digital filter

requires more adders than the conventional design. Significantly, the suppression of limit cycle oscillations is made possible in the wave digital design by certain characteristics of the network scattering matrix. These features have occasioned considerable interest in wave digital filters, which have been realized as the interconnection of building blocks called adaptors to achieve the required topological structure implied by the network scattering matrix. The series, parallel, and lattice adaptors, introduced by Fettweis and his coworkers [13, 23], permit wave digital filter designs for a variety of network structures, e.g. the doubly terminated LC ladder, but not all structures can be realized with these adaptors. Hence, Martens and Meerkötter have formulated the generalized or n-port adaptor [16] from which Le [25] has developed adaptors for the Brune section, the Darlington C and D sections, and the twin-T structure which allow cascade synthesis. Ashley [24] has developed a canonical network adaptor from the n-port adaptor by using scattering variables as state variables for the elimination of reactive redundancies. It has also been shown that limit cycles can be suppressed in certain cases with canonical n-port adaptors [24], but a general solution has not been disclosed.

The purpose of this thesis is to explore the possibilities of obtaining hardware realizations for n-port adaptors directly from the matrix formulation of Martens and Meerkötter. Such a hardware realization would encompass all of the above mentioned adaptors. The problem of realizing the n-port adaptor matrix (scattering matrix) is treated as a numerical problem with the justified assumption that two's complement arithmetic will be used. One approach taken with favorable

results is a numerical decomposition of the adaptor matrix to add and shift operations based on a binary series expansion in the canonical signed digit code with matrix coefficients. This decomposition, given in chapter 4, furnishes a program for hardware or computer implementation of the adaptor matrix with two's complement arithmetic which introduces no roundoff error during computation while the required wordlength in every addition operation is the same as the wordlength at the input to the matrix. The decomposition is also efficient, but most likely not optimal, in terms of the number of adders. A third order elliptic lowpass filter is given as an example in chapter 3 and realized in chapter 4. The hardware count for the realization of this example compares favorably with wave digital realizations of the same filter by other methods and the hardware implementation has been simulated on a minicomputer with assembly language programming for two's complement arithmetic. It is concluded that the numerical approach to n-port adaptor realizations is viable and worthy of further investigation.

The thesis is organized in three major sections. Chapter 2 is a brief presentation of discrete time systems which is intended to provide background for wave digital filters. Chapter 3, which presents the theory of wave digital filters, may be particularly helpful to the uninitiated since the theory and its variations are somewhat dispersed throughout a number of papers and theses. Chapter 4 presents the central topic, an approach to the realization of n-port adaptors with binary arithmetic leading to hardware implementations in two's complement form.

Standard mathematical notation is used for the most part. Matrix operators are upper case characters, where the superscripts T and -1 denote the transpose and inverse, respectively. Vectors and functions are lower case characters, where the use of a vector will be clear in context. Exceptions will be adequately defined where introduced.

2. DISCRETE TIME SYSTEMS

2.1 Introduction

This chapter presents results from the theory of discrete time systems which are relevant to the development of wave digital filters. A complete development of these results can be found in the references [4,5,6] which require a background of matrix analysis, complex variables, and the Laplace transform [1,2,3].

2.2 Motivation- The Sampling Theorem

It is well known that a bandlimited function is specified by samples taken with sufficient frequency. Specifically, if f(t) is a continuous function of time where f(t)=0 for t<0, the Laplace transform of f(t), if it exists, is defined by

$$\hat{f}(s) = L[f(t)] = \int_0^\infty f(t)e^{-st}dt, \quad s = \sigma + j\omega \ . \tag{2.1}$$

An impulse train is defined by

$$\delta_T(t) = \sum_{n=-\infty}^{\infty} \delta(t-nT) = \frac{1}{T} \sum_{n=-\infty}^{\infty} e^{jn\omega_0 t} \ , \ \omega_0 = 2\pi/T \ , \tag{2.2}$$

where $\delta(t)$ is a unit impulse and the second summation in (2.2) follows from Fourier transform theory . If we define the impulsively sampled function

$$f^*(t) = f(t)\delta_T(t) \tag{2.3}$$

then

$$L[f^*(t)] = \frac{1}{T}\sum_{n=-\infty}^{\infty} \hat{f}(s+nj\omega_0) \tag{2.4}$$

from (2.2) and (2.1). Also

$$L[f^*(t)] = \sum_{n=0}^{\infty} f(nT)e^{-nTs} \tag{2.5}$$

from (2.3) and (2.1). Note the use of $\delta(t)$ is formally justified as the result of a limiting process.

The result (2.4) implies f(t) can be recovered from $f^*(t)$ if $\hat{f}(s)=0$ for $|\omega|>\omega_0/2$ by applying $f^*(t)$ to an ideal lowpass filter with cutoff at $\omega_0/2$. In practice only an approximation to impulsive sampling and the ideal lowpass filter can be obtained. If $\hat{f}(s)\neq 0$ for $|\omega|>\omega_0/2$ then the expression (2.4) or (2.5) only approximates $\hat{f}(s)$ in the interval $-\omega_0/2\leq\omega\leq\omega_0/2$, in which case "aliasing" is said to occur.

2.3 Fourier Transform of a Sample Sequence

The Fourier transform of an absolutely summable sequence f(n), n = 0,1,2,..., is defined as

$$F(e^{j\omega}) = \sum_{n=0}^{\infty} f(n)e^{-j\omega n} \quad , \tag{2.6}$$

where the sequence f(n) may represent the sampled function f(t), i.e. we assume that f(nT) is replaced by f(n), which we denote by

$$f(nT)\rightarrow f(n) \quad , \; n = 0,1,2,\ldots \quad . \tag{2.7}$$

Note $F(e^{j\omega})$ is a periodic function of ω. The definition (2.6) is closely related to the impulsively sampled function $f^*(t)$, since we see from (2.5)

$$L[f^*(t)] \Big|_{s=j\omega} = F(e^{j\omega T}) \tag{2.8}$$

where T appears as a frequency scale factor.

The Fourier transform of the convolution

$$y(n) = \sum_{k=0}^{n} h(n-k)x(k) \tag{2.9}$$

is given by

$$Y(e^{j\omega}) = H(e^{j\omega})\ X(e^{j\omega}) \tag{2.10}$$

as shown in [5], where $H(e^{j\omega})$ and $Y(e^{j\omega})$ are the Fourier transforms of the sequences h(n) and y(n), n = 0,1,2,... , respectively. The convolution in (2.9) represents the response y(n) for an input x(n) to a linear causal shift invariant discrete time system with unit sample response h(n), n = 0,1,2,... . It also follows that $H(e^{j\omega})$ gives the frequency response for a complex exponential input sequence, i.e.

$$y(n) = H(e^{j\omega})\ e^{j\omega n} \tag{2.11}$$

where $x(n) = e^{j\omega n}$. If the sampled function x(t) is bandlimited to $\omega_0/2 = \pi/T$, then from (2.8) and (2.4)

$$X(e^{j\omega T}) = \frac{1}{T}\ \hat{x}(s) \Big|_{s=j\omega}, \quad -\omega_0/2 \leq \omega \leq \omega_0/2 \ , \tag{2.12}$$

so from (2.10)

$$Y(e^{j\omega T}) = \frac{1}{T}\ H(e^{j\omega T})\ \hat{x}(s) \Big|_{s=j\omega}, \quad -\omega_0/2 \leq \omega \leq \omega_0/2 \ . \tag{2.13}$$

Thus if we replace y(nT) by y(n) from (2.9) and apply the impulse train

$$y^*(t) = \sum_{n=0}^{\infty} y(nT)\ \delta(t-nT) \tag{2.14}$$

to an ideal lowpass filter with cutoff $\omega_0/2$, by (2.8) and (2.4) the recovered signal y(t) has

$$\hat{y}(s) = \begin{cases} \frac{1}{T} H(e^{j\omega T})\ \hat{x}(s)\ \Big|_{s=j\omega}, & -\omega_0/2 \leq \omega \leq \omega_0/2 \\ 0, & |\omega| > \omega_0/2 \end{cases} \tag{2.15}$$

The basis of frequency selective behaviour in a discrete time system is expressed in (2.11) and its application to continuous time signals by interface with impulsive sampling and signal reconstruction is expressed in (2.15). The fundamental design problem for a discrete time filter is the generation of a unit sample response h(n), n = 0,1,.., such that the frequency response

$$H(e^{j\omega}) = \sum_{n=0}^{\infty} h(n)\ e^{-j\omega n} \tag{2.16}$$

meets a required specification. In the case of (2.15) this response is frequency scaled to $H(e^{j\omega T})$. Note the term digital filter applies to a discrete time filter which is realized with finite length arithmetic elements.

2.4 z-Transform of a Sample Sequence

The one sided z-transform of a sequence h(n) is

$$H(z) = \sum_{n=0}^{\infty} h(n)\ z^{-n} \tag{2.17}$$

where z is a complex variable. Hence H(z) is an analytic function in

its region of convergence. The reader is referred to [5] for the general properties of the z-transform which are useful in the analysis of discrete time systems. Here we note the z-transform evaluated on the unit circle gives the Fourier transform of a sequence, i.e.

$$H(z)\Big|_{z=e^{j\omega}} = H(e^{j\omega}) \tag{2.18}$$

from (2.16) and (2.17). If h(n) is the unit sample response previously introduced then (2.18) gives the frequency response of the discrete time system. Another important feature is that the z-transform of a complex exponential sequence can be expressed in a closed form, i.e. if

$$h(n) = h_0\, e^{n(\alpha+j\beta)} \quad , \quad n = 0,1,2,\ldots \ , \tag{2.19}$$

where h_0 may be a complex number and α is usually negative, then

$$H(z) = h_0 \sum_{n=0}^{\infty} [\, e^{(\alpha+j\beta)}]^n z^{-n} = h_0 \sum_{n=0}^{\infty} [e^{(\alpha+j\beta)} z^{-1}]^n \tag{2.20}$$

or

$$H(z) = h_0 \,/\, [\, 1 + e^{(\alpha+j\beta)} z^{-1} \,] \ , \tag{2.21}$$

where (2.21) follows if we interpret (2.20) as a geometric series in the complex variable $u = e^{(\alpha+j\beta)} z^{-1}$ which converges for $|u| < 1$. Note the sequence in (2.19) includes a simple exponential sequence and an undamped exponential sequence as special cases.

2.5 State Equations for Discrete Time Systems

A linear causal shift invariant discrete time system can be characterized by state equations of the form

$$x(k+1) = A\,x(k) + B\,u(k)$$
$$y(k) = C\,x(k) + D\,u(k) \quad , \tag{2.22}$$

$k = 0,1,2,\ldots$, where u, x, and y are the input, state, and output vectors, respectively, A is an N x N matrix, and B, C, and D are matrices of appropriate dimensions [7]. Since the system is linear and shift invariant the response of an output component $y_i(k)$ to an input component $u_j(k)$, $k= 0,1,2,\ldots$, is given by the convolution sum in (2.9) with x(k) replaced by $u_j(k)$ and h(k) replaced by $h_{ij}(k)$. Here $h_{ij}(k)$ is the response $y_i(k)$ to a unit sample input at $u_j(k)$, i.e. the input

$$u_j(k) = \begin{cases} 1 & \text{for } k = 0 \\ 0 & \text{for } k \neq 0 \end{cases} \quad . \tag{2.23}$$

Hence the frequency response from input $u_j(k)$ to output $y_i(k)$ is given by $H(e^{j\omega})$ in (2.16) or $H(z)\big|_{z=e^{j\omega}}$ in (2.18).

It can be shown $H_{ij}(z)$ has the form

$$H_{ij}(z) = \frac{c_N z^{-N} + c_{N-1} z^{-N+1} + \ldots + c_0}{a_N z^{-N} + a_{N-1} z^{-N+1} + \ldots + 1} \tag{2.24}$$

for the system in (2.22). If A has distinct eigenvalues, this follows by transforming the state equations to normal coordinates

$$v(k+1) = \Lambda\, v(k) + \hat{B}\, u(k) \tag{2.25a}$$
$$y(k) = \hat{C}\, v(k) + D\, u(k) \tag{2.25b}$$

where the diagonal matrix

$$\Lambda = T^{-1} A T \tag{2.26}$$

may contain pairs of complex conjugate eigenvalues and

$$\begin{aligned} v(k) &= T^{-1} x(k) \\ \hat{B} &= T^{-1} B \\ \hat{C} &= C T \quad . \end{aligned} \tag{2.27}$$

Note complex conjugate eigenvalues will have corresponding complex conjugate eigenvectors in the columns of T. For a unit sample input, as in (2.23), we have for any component $v_p(k)$, $p = 1,2,\ldots,N$, of the state vector

$$\begin{aligned} v_p(0) &= 0 \\ v_p(1) &= \hat{b}_{pj} \\ v_p(k+1) &= \lambda_p v_p(k) \quad , \quad k = 1,2,\ldots \ , \end{aligned} \tag{2.28}$$

where λ_p is the p^{th} eigenvalue. Hence

$$\begin{aligned} v_p(0) &= 0 \\ v_p(k) &= \lambda_p^{k-1} \hat{b}_{pj} \end{aligned} \tag{2.29}$$

or

$$\begin{aligned} v_p(0) &= 0 \\ v_p(k) &= \hat{b}_{pj} \, e^{(k-1)(\alpha+j\beta)} \quad , \quad k = 1,2,\ldots \ , \end{aligned} \tag{2.30a}$$

where we set

$$\lambda_p = e^{(\alpha+j\beta)} \quad . \tag{2.30b}$$

From (2.19), (2.20), and (2.21) we see the z-transform of $v_p(k)$ is

$$V_p(z) = \hat{b}_{pj}\, z^{-1} / (1 + e^{(\alpha+j\beta)} z^{-1}) \qquad (2.31)$$

and it is seen the z-transform of $u_j(k)$ is simply one. Since $y_i(k)$ is given in (2.25b) as a linear superposition of states, as in (2.30a), and input, as in (2.23), by the linearity property of the z-transform $Y_i(z) = H_{ij}(z)$ will be a linear combination of terms such as (2.31) with one. This gives the form in (2.24). Since $y_i(k)$, $k = 0,1,2,\ldots$, is a sequence of real numbers no complex numbers appear among the coefficients in (2.24) and from (2.29) it follows that this is an infinite impulse response (IIR) system. The other important case is the finite impulse response (FIR) system which occurs when the matrix A has N repeated eigenvalues equal to zero. We will not deal with the FIR case but mention that it offers the advantage of a linear phase characteristic.

It is apparent the system (2.22) can be hardware implemented with binary arithmetic to produce a digital filter. For simplicity it is common practice to reduce a system such as (2.22) with single input and output to phase canonical form before implementation. That is, it can be shown [7] a transformation $q(k) = P^{-1} x(k)$ exists where

$$\begin{aligned} q(k+1) &= \hat{A}\, q(k) + \hat{B}\, u(k) \\ y(k) &= \hat{C}\, q(k) \quad , \end{aligned} \qquad (2.32)$$

$$\hat{A} = P^{-1} A P = \begin{bmatrix} 0 & 1 & 0 & \cdots & 0 \\ 0 & 0 & 1 & \cdots & 0 \\ \vdots & \vdots & \vdots & & \vdots \\ 0 & 0 & 0 & \cdots & 1 \\ -a_N & -a_{N-1} & -a_{N-2} & \cdots & -a_1 \end{bmatrix}, \quad (2.33)$$

$$\hat{B} = \begin{bmatrix} 0 \\ 0 \\ \vdots \\ 0 \\ 1 \end{bmatrix}, \quad \hat{C} = \begin{bmatrix} c_N \\ c_{N-1} \\ \vdots \\ c_2 \\ c_1 \end{bmatrix}. \quad (2.34) \quad (2.35)$$

It also follows by application of z-transform theory to (2.32-2.35) that

$$\frac{Y(z)}{U(z)} = \frac{c_N z^{-N} + c_{N-1} z^{-N+1} + \cdots + c_1 z}{a_N z^{-N} + a_{N-1} z^{-N+1} + \cdots + 1} \quad (2.36)$$

and since $U(z) = 1$ for the unit sample input (2.36) gives $H(z)$ as in (2.24), where we may identify the coefficients $a_N, \ldots, a_1$ from (2.33) and $c_N, \ldots, c_1$ from (2.35) with those in (2.24). The price paid for this apparent simplicity is high sensitivity of the poles and zeroes of $H(z)$ to a_i and c_i, $i = 1,2,\ldots,N$, since it is well known the roots of a high order polynomial are extremely sensitive to its coefficients. In practice this type of digital filter realization requires

coefficients of 8 to 12 bits wordlength for sufficient accuracy in the frequency response $H(z)\Big|_{z=e^{j\omega}}$. There are many possible structures [5] for the realization of (2.32) but a typical form is shown in figure 2.1. An alternate approach which combats the sensitivity problem is to realize the transfer function in (2.36) as a cascade of first and second order systems.

2.6 Design of Frequency Response

Methods of designing the unit sample response such that the frequency response $H(e^{j\omega T})$ in (2.16) meets a required specification follow from the closed form shown for H(z) in (2.24). Essentially the problem is to determine the coefficients in (2.24) so that $H(e^{j\omega T}) = H(z)\Big|_{z=e^{j\omega T}}$ meets the required specification. The design methods most commonly used are the impulse invariance method and the bilinear transformation.

The impulse invariance method fixes H(z) as the z-transform of the sample sequence h(nT) where h(t) is a continuous time function whose Laplace transform $\hat{h}(s)\Big|_{s=j\omega}$ approximates the desired frequency characteristic for $-\omega_0/2 \leq \omega \leq \omega_0/2$. In this case we have from (2.18) and (2.8)

$$H(e^{j\omega T}) = L[\, h^*(t)\,]\Big|_{s=j\omega} \tag{2.37}$$

and with (2.4)

$$H(e^{j\omega T}) = \frac{1}{T} \sum_{n=-\infty}^{\infty} \hat{h}(s+jn\omega_0) \Big|_{s=j\omega} . \tag{2.38}$$

It is seen from (2.38) that $H(e^{j\omega T})$ will approximate the characteristic

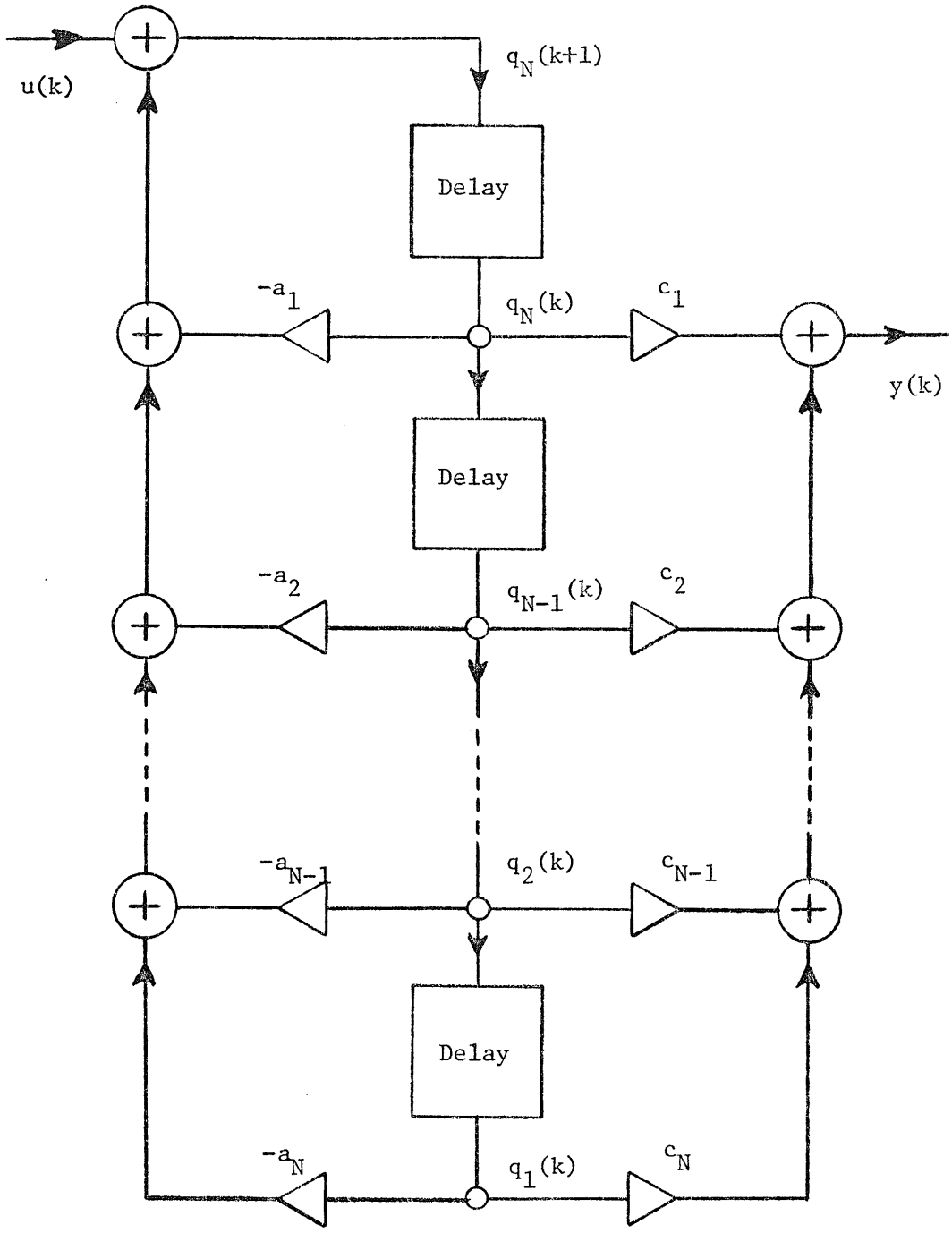


Figure 2.1 Typical Digital Filter Realization of Phase Canonical Form in (2.32)

$\hat{h}(s)\big|_{s=j\omega}$ for $-\omega_0/2 \leq \omega \leq \omega_0/2$, but a major disadvantage is aliasing of the response. The function h(t) is usually the impulse response of an appropriate analog filter, so $\hat{h}(s)$ follows from the filter design. A partial fraction decomposition of $\hat{h}(s)$ gives the complex exponential components of h(t) from which the corresponding complex exponential sequences, as in (2.19), comprising h(nT) are determined. The superposition of the z-transforms for each complex exponential sequence, as in (2.21), will give the required form in (2.24).

The bilinear transformation is the mapping

$$z = \frac{1 + s}{1 - s} \tag{2.39}$$

from the s plane to the z plane, which transforms the rational function H(z) in (2.24) to the rational function H(z(s)) of s. The transformation (2.39) maps the imaginary axis in the s plane to the unit circle in the z plane, so the frequency response

$$H(z)\Big|_{z=e^{j\omega T}} = H(z(s))\Big|_{s=j\omega'} \tag{2.40}$$

where $s = \sigma' + j\omega'$. The mapping is shown by setting

$$z = r\, e^{j\omega T} = \frac{1+j\omega'}{1-j\omega'} \tag{2.41}$$

in (2.39). Note

$$r = \frac{|1 + j\omega'|}{|1 - j\omega'|} = 1$$

and

$$\omega T = \tan^{-1}(1+j\omega') - \tan^{-1}(1-j\omega') = 2\tan^{-1}\omega' , \tag{2.43a}$$

$$\omega' = \tan(\omega T/2) , \tag{2.43b}$$

i.e. the relation between ω and ω' is nonlinear. The problem of approximating a given magnitude function on the imaginary axis for a rational function of s, as in (2.40), is the well known approximation problem in analog filter theory. Hence we may set $H(z(s))$ as a transfer function known from filter theory (e.g. a Butterworth lowpass characteristic) and reverse the transformation in (2.39) to obtain the required $H(z)$. The nonlinear relation in (2.43) requires pre-scaling of the characteristic in the s plane to compensate for the warping effect on the frequency axis, but the general shape of the characteristic will be intact. The problem of predistorting phase in the s plane to compensate for warping is open to investigation. The advantage of this method is the absence of aliasing distortion and it can also be said that any possible transfer function $H(z)\Big|_{z=e^{j\omega T}}$ corresponds to a bilinear transformation on some characteristic in the s plane. The bilinear transformation will be used in the derivation of wave digital filters.

3. THEORY OF WAVE DIGITAL FILTERS

3.1 Introduction

This chapter presents the theory of wave digital filters, which were introduced by Fettweis [12,13], in the context of discrete time systems and network theory. An overview of discrete time systems was given in chapter 2 and a summary of the required topological and state variable approach to network theory is given as a preliminary in this chapter. The n-port adaptor approach to wave digital filters, disclosed by Martens and Meerkötter [16] and further developed by Le [25] and Ashley [24], is emphasized here as a preliminary to chapter 4, which is devoted to the realization of n-port adaptors with binary arithmetic. Finally, a third order lowpass filter is given as an example and realized in chapter 4.

3.2 Network Topology and State Variables

The state variable approach to network theory [9,11] is closely related to network topology and the reader may consult the references for background in the following discussion. We require the following key topological concepts based on the network graph.

Definition 3.1: A normal tree for an RLC network is a tree which contains the maximum number of capacitances possible in any tree and the minimum number of inductances possible in any tree.

Theorem 3.1: For any RLC network a normal tree exists which contains all of the voltage sources and none of the current sources, provided no

loops of voltage sources or cutsets of current sources exist.

Definition 3.2: An excess capacitance (inductance) is a capacitance (inductance) which corresponds to a link (twig) for the normal tree.

Theorem 3.2: Any excess capacitance (inductance) is part of a capacitive loop (inductive cutset) with the normal tree (cotree).

The state variable formulation is based on a normal tree, as in theorem 3.1, and consists of two distinct stages:

1. All topological constraints on the network variables are enforced by

$$\left[\begin{array}{c} v_\ell \\ \hline i_t \end{array}\right] = \left[\begin{array}{c|c} 0 & Q_\ell^T \\ \hline -Q_\ell & 0 \end{array}\right] \left[\begin{array}{c} i_\ell \\ \hline v_t \end{array}\right] \tag{3.1}$$

where the vectors of network voltages and currents have been partitioned as

$$v = \begin{bmatrix} v_1 \\ v_2 \\ v_3 \\ \cdot \\ \cdot \\ v_N \end{bmatrix} = \left[\begin{array}{c} v_\ell \\ \hline v_t \end{array}\right] \begin{matrix} \}\text{links} \\ \\ \}\text{twigs} \end{matrix}, \quad i = \begin{bmatrix} i_1 \\ i_2 \\ i_3 \\ \cdot \\ \cdot \\ i_N \end{bmatrix} = \left[\begin{array}{c} i_\ell \\ \hline i_t \end{array}\right] \begin{matrix} \}\text{links} \\ \\ \}\text{twigs} \end{matrix} \tag{3.2}$$

and the fundamental cutset matrix for the normal tree is

$$Q_f = \left[\begin{array}{c:c} \underbrace{Q_\ell}_{\text{links}} & \underbrace{U}_{\text{twigs}} \end{array}\right] \}\text{twigs} \tag{3.3}$$

with U an identity matrix. On this basis the link voltages and

twig currents, v_ℓ and i_t, are shown as linearly dependent on the twig voltages and link currents, v_t and i_ℓ, respectively. Hence the vector $\left[\begin{array}{c|c} i_\ell^T & v_t^T \end{array}\right]^T$ or one of its subvectors must be a set of state variables.

2. The element constraints are enforced upon each voltage-current pair corresponding to an edge in the network graph using (3.1) to eliminate the dependent variables v_ℓ and i_t. It can be shown [11] the resulting state vector will be $\left[\begin{array}{c|c} i_{\ell L}^T & v_{tC}^T \end{array}\right]^T$, where the vector $\left[\begin{array}{c|c} i_\ell^T & v_t^T \end{array}\right]^T$ has been partitioned as

$$\left[\begin{array}{c} i_\ell \\ \hline v_t \end{array}\right] = \left[\begin{array}{l} i_{\ell R} \\ i_{\ell L} \\ i_{\ell C} \\ i_{\ell I} \\ \hline v_{tR} \\ v_{tL} \\ v_{tC} \\ v_{tE} \end{array}\right] \begin{array}{l} \} \text{ resistive links} \\ \} \text{ inductive links} \\ \} \text{ capacitive links (excess)} \\ \} \text{ link current sources} \\ \} \text{ resistive twigs} \\ \} \text{ inductive twigs (excess)} \\ \} \text{ capacitive twigs} \\ \} \text{ twig voltage sources} \end{array} \qquad (3.4)$$

since $i_{\ell R}$, $i_{\ell C}$, v_{tR}, and v_{tL} are dependent on $\left[\begin{array}{c|c} i_{\ell L}^T & v_{tC}^T \end{array}\right]^T$ and $i_{\ell I}$ and v_{tE} are independent sources. The dependencies of $i_{\ell R}$ and v_{tR} follow from the linear relation enforced between voltage-current pairs by resistances. The dependencies of $i_{\ell C}$ and v_{tL} follow since, by theorem 3.2, each excess capacitance forms a C loop with the normal tree and each excess inductance forms an L cutset with the normal cotree. The time derivative of KVL for a C loop gives the form

$$i_1/C_1 + i_2/C_2 + \ldots + i_n/C_n = 0 \qquad (3.5)$$

and the time derivative of KCL for an L cutset gives the form

$$v_1/L_1 + v_2/L_2 + \ldots + v_n/L_n = 0 \qquad (3.6)$$

which contain the stated dependencies. The element constraints also furnish the time derivative terms which yield state equations in standard matrix form.

3.3 Network Characterization with Scattering Variables as State Variables

A state variable formulation for an RLC network can also be obtained with scattering variables by enforcing topological constraints followed by element constraints based on a normal tree analogous to section 3.2. For simplicity, sources in the network are restricted to series combinations of a voltage source with a resistance, where the combination is taken as a single edge in the network graph. The scattering variables a_j and b_j, $j= 1,2,\ldots,N$, are an alternate set of network coordinates related to the vectors of network voltages and currents from (3.2) by

$$\begin{aligned} a &= v + R\,i \\ b &= v - R\,i \end{aligned} \qquad (3.7)$$

where

$$R = \begin{bmatrix} R_1 & & & \\ & R_2 & & \\ & & \ddots & \\ & & & R_N \end{bmatrix} \qquad (3.8)$$

That is, for each voltage-current pair, v_j and i_j, $j= 1,2,\ldots,N$,

corresponding to an edge in the network graph we have the coordinate transformation

$$\begin{aligned} a_j &= v_j + R_j i_j \\ b_j &= v_j - R_j i_j \end{aligned} \tag{3.9}$$

where R_j, j= 1,2,...,N, is an arbitrary reference parameter and the positive current reference is from the negative to the positive voltage reference. In most applications R_j is assigned a value related to the corresponding network element, R,L, or C, in which case scattering variables are particularly useful for dealing with voltage transfer ratios. Note the transformation (3.7) is nonsingular, i.e.

$$\begin{aligned} v &= (a + b) / 2 \\ i &= R^{-1} (a - b) / 2 \quad . \end{aligned} \tag{3.10}$$

The topological constraints on the network variables a and b from (3.7) are enforced by writing a and b in terms of the independent vector $\left[i_\ell^T \mid v_t^T \right]^T$ in (3.1). This gives

(3.11)

$$a = \left[\begin{array}{c} a_\ell \\ \hline a_t \end{array}\right] = \left[\begin{array}{c} v_\ell \\ \hline v_t \end{array}\right] + \left[\begin{array}{c|c} R_\ell & 0 \\ \hline 0 & R_t \end{array}\right] \left[\begin{array}{c} i_\ell \\ \hline i_t \end{array}\right] = \underbrace{\left[\begin{array}{c|c} -R_\ell & Q_\ell^T \\ \hline R_t Q_\ell & U \end{array}\right]}_{T} \underbrace{\left[\begin{array}{c|c} -U & 0 \\ \hline 0 & U \end{array}\right]}_{D} \left[\begin{array}{c} i_\ell \\ \hline v_t \end{array}\right]$$

and similarly

$$b = \left[\begin{array}{c} b_\ell \\ \hline b_t \end{array}\right] = \underbrace{\left[\begin{array}{c|c} -R_\ell & Q_\ell^T \\ \hline R_t Q_\ell & U \end{array}\right]}_{T} \left[\begin{array}{c} i_\ell \\ \hline v_t \end{array}\right] \tag{3.12}$$

where the subscripts ℓ and t denote link and twig partitions of a, b, and R conformable with the partitions in (3.2). We prove either the components of a or the components of b can be independently specified by showing the matrix T in (3.11) and (3.12) is nonsingular. This is done by noting

$$T = \left[\begin{array}{c|c} -U & Q_\ell^T \\ \hline 0 & U \end{array}\right]\left[\begin{array}{c|c} Z & 0 \\ \hline R_t Q_\ell & U \end{array}\right] , \tag{3.13}$$

where

$$Z = R_\ell + Q_\ell^T R_t Q_\ell , \tag{3.13a}$$

and constructing the inverse

$$T^{-1} = \left[\begin{array}{c|c} Z & 0 \\ \hline R_t Q_\ell & U \end{array}\right]^{-1}\left[\begin{array}{c|c} -U & Q_\ell^T \\ \hline 0 & U \end{array}\right]^{-1} . \tag{3.14}$$

Since the second matrix in (3.14) is self-inverse and

$$\left[\begin{array}{c|c} Z & 0 \\ \hline R_t Q_\ell & U \end{array}\right]^{-1} = \left[\begin{array}{c|c} Z^{-1} & 0 \\ \hline -R_t Q_\ell Z^{-1} & U \end{array}\right] , \tag{3.15}$$

where it can be shown Z^{-1} exists, we have finally

$$T^{-1} = \left[\begin{array}{c|c} Z^{-1} & 0 \\ \hline -R_t Q_\ell Z^{-1} & U \end{array}\right]\left[\begin{array}{c|c} -U & Q_\ell^T \\ \hline 0 & U \end{array}\right] . \tag{3.16}$$

Hence, from (3.11),

$$\left[\begin{array}{c} i_\ell \\ \hline v_t \end{array}\right] = D^{-1} T^{-1} a = D T^{-1} a \tag{3.17}$$

where D is self-inverse and we have chosen a as the independent vector. Note a must contain a state vector since $\left[\begin{array}{c|c} i_\ell^T & v_t^T \end{array}\right]^T$ contains a state vector. The components of a are free of topological constraints, but b is completely specified by the topological constraints in (3.12), i.e. from (3.17) and (3.12)

$$b = \underbrace{T\,D\,T^{-1}}_{S}\,a = S\,a \quad . \tag{3.18}$$

The matrix S in (3.18), known as the scattering matrix for the network topology or interconnections, expresses the topological constraints between the dependent vector b and the independent vector a. Since D is diagonal the form $S = T\,D\,T^{-1}$ in (3.18) gives the eigenvalues of S directly from (3.11) as -1's and +1's of multiplicities equal to the number of links and number of twigs, respectively. The eigenvectors of S are the columns of T. The form

$$S = \left[\begin{array}{c|c} -U & Q_\ell^T \\ \hline 0 & U \end{array}\right]\left[\begin{array}{c|c} -U & 0 \\ \hline -2K & U \end{array}\right]\left[\begin{array}{c|c} -U & Q_\ell^T \\ \hline 0 & U \end{array}\right] \tag{3.19}$$

results from substituting T from (3.11) and T^{-1} from (3.16) in (3.18) and multiplying out the inner matrices where

$$K = R_t\,Q_\ell\,Z^{-1} \quad . \tag{3.20}$$

From (3.13a) it is seen that Z is the network impedance matrix (on a loop basis) if all network elements are replaced by resistances equal to the corresponding reference values R_j, j= 1,2,...,N, in (3.8). Hence it follows from (3.20) that K is the matrix of voltage transfer ratios from sources in series with the link resistances to the twig resistances. It can also be shown from the definitions (3.9) that S/2 must be the

matrix of voltage transfer ratios defined in a similar way for the entire resistive network. An alternate form is

$$K = Y^{-1} Q_\ell G_\ell \tag{3.21}$$

where

$$G = R^{-1}, \quad G_\ell = R_\ell^{-1}, \quad G_t = R_t^{-1}, \tag{3.22}$$

and

$$Y = G_t + Q_\ell G_\ell Q_\ell^T \tag{3.23}$$

is the network admittance matrix (on a node basis). The form for S in (3.19) is one of several given in [16].

The element constraints are enforced on the components of a and b with the assumption

$$R_j = Z_j \Big|_{j\omega=1}, \tag{3.24}$$

j= 1,2,...,N, in (3.8), i.e. $R_j = R$, $R_j = L$, or $R_j = 1/C$ where the j^{th} element may be a resistance, inductance, or capacitance, respectively. The state vector is determined from a by the dependencies which follow from the element constraints. If a is partitioned as

$$\left[\begin{array}{c} a_\ell \\ \hline a_t \end{array}\right] = \left[\begin{array}{l} a_{\ell R} \\ \hline a_{\ell L} \\ \hline a_{\ell C} \\ \hline a_{tR} \\ \hline a_{tL} \\ \hline a_{tC} \end{array}\right] \begin{array}{l} \}\text{resistive links with voltage sources} \\ \}\text{inductive links} \\ \}\text{capacitive links (excess)} \\ \}\text{resistive twigs with voltage sources} \\ \}\text{inductive twigs (excess)} \\ \}\text{capacitive twigs} \end{array} \tag{3.25}$$

which is analogous to (3.4) then the resistive element constraints with (3.9) and (3.24) imply

$$\begin{aligned} a_{\ell R} &= e_{\ell R} \\ a_{tR} &= e_{tR} \end{aligned}, \tag{3.26}$$

where $e_{\ell R}$ and e_{tR} are vectors of the source voltages in series with the resistances. Hence $a_{\ell R}$ and a_{tR} are taken as independent inputs. From section 3.2, the only other linear dependencies implied by element constraints are of form (3.5) for each excess capacitance and form (3.6) for each excess inductance. From (3.5) with KVL we have the relations

$$\begin{aligned} a_1 + a_2 + \ldots + a_n &= 0 \\ b_1 + b_2 + \ldots + b_n &= 0 \end{aligned} \tag{3.27}$$

for the fundamental C loop corresponding to an excess capacitance and from (3.6) with KCL we have

$$\begin{aligned} a_1/L_1 + a_2/L_2 + \ldots + a_n/L_n &= 0 \\ b_1/L_1 + b_2/L_2 + \ldots + b_n/L_n &= 0 \end{aligned} \tag{3.28}$$

for the fundamental L cutset corresponding to an excess inductance. The dependencies expressed in (3.27) and (3.28) for each excess capacitance and excess inductance, respectively, imply $a_{\ell C}$ and a_{tL} are linearly dependent on a_{tC} and $a_{\ell L}$, respectively, i.e. $\left[a_{\ell L}^T \,\vdots\, a_{tC}^T\right]^T$ is a state vector. Note a transformation

$$\begin{bmatrix} i_{\ell L} \\ \hdashline v_{tC} \end{bmatrix} = P \begin{bmatrix} a_{\ell L} \\ \hdashline a_{tC} \end{bmatrix} \tag{3.29}$$

must exist where P is a nonsingular matrix since the two vectors in (3.29) are state vectors of the same system. The resistive element constraints with (3.9) and (3.24) also imply

$$\begin{aligned} b_{\ell R} &= v_{\ell R} \\ b_{tR} &= v_{tR} \quad , \end{aligned} \tag{3.30}$$

i.e. the components of $b_{\ell R}$ and b_{tR} may be taken as outputs corresponding to voltages across each series combination of a resistance with voltage source where the voltage source may be zero.

The scattering matrix S in (3.18) or (3.19) becomes $\tilde{S}$ by the operations of row and column interchange to conform with the reordering

$$\tilde{a} = \begin{bmatrix} a_{\ell L} \\ a_{tL} \\ a_{tC} \\ a_{\ell C} \\ a_{\ell R} \\ a_{tR} \end{bmatrix} = \left[\begin{array}{c} \tilde{a}_1 \\ \hline \tilde{a}_2 \end{array}\right] \qquad \tilde{b} = \begin{bmatrix} b_{\ell L} \\ b_{tL} \\ b_{tC} \\ b_{\ell C} \\ b_{\ell R} \\ b_{tR} \end{bmatrix} = \left[\begin{array}{c} \tilde{b}_1 \\ \hline \tilde{b}_2 \end{array}\right] \tag{3.31}$$

of a and b from (3.25) which gives the system equations input-output form. That is

$$\tilde{b} = \tilde{S}\,\tilde{a} \tag{3.32a}$$

or

$$\left[\begin{array}{c} \tilde{b}_1 \\ \hline \tilde{b}_2 \end{array}\right] = \left[\begin{array}{c|c} \tilde{S}_{11} & \tilde{S}_{12} \\ \hline \tilde{S}_{21} & \tilde{S}_{22} \end{array}\right] \left[\begin{array}{c} \tilde{a}_1 \\ \hline \tilde{a}_2 \end{array}\right] \tag{3.32b}$$

or

$$\begin{aligned} \tilde{b}_1 &= \tilde{S}_{11}\,\tilde{a}_1 + \tilde{S}_{12}\,\tilde{a}_2 \\ \tilde{b}_2 &= \tilde{S}_{21}\,\tilde{a}_1 + \tilde{S}_{22}\,\tilde{a}_2 \end{aligned} \tag{3.32c}$$

Further, since $a_{\ell C}$ and a_{tL} are dependent on the state vector $\left[a_{\ell L}^T \mid a_{tC}^T\right]^T$, the matrix $\tilde{S}$ can be reduced with dependency relations to $\hat{S}$ where

$$\hat{b} = \hat{S}\,\hat{a} \tag{3.33}$$

and

$$\hat{a} = \begin{bmatrix} a_{\ell L} \\ a_{tC} \\ a_{\ell R} \\ a_{tR} \end{bmatrix} = \begin{bmatrix} \hat{a}_1 \\ \hline \hat{a}_2 \end{bmatrix} \qquad \hat{b} = \begin{bmatrix} b_{\ell L} \\ b_{tC} \\ b_{\ell R} \\ b_{tR} \end{bmatrix} = \begin{bmatrix} \hat{b}_1 \\ \hline \hat{b}_2 \end{bmatrix}, \qquad (3.34)$$

i.e. $\hat{a}$ contains only the state vector and inputs. This gives the reduced system equations

$$\begin{bmatrix} \hat{b}_1 \\ \hline \hat{b}_2 \end{bmatrix} = \left[\begin{array}{c|c} \hat{S}_{11} & \hat{S}_{12} \\ \hline \hat{S}_{21} & \hat{S}_{22} \end{array}\right] \begin{bmatrix} \hat{a}_1 \\ \hline \hat{a}_2 \end{bmatrix} \qquad (3.35a)$$

or

$$\begin{aligned} \hat{b}_1 &= \hat{S}_{11}\,\hat{a}_1 + \hat{S}_{12}\,\hat{a}_2 \\ \hat{b}_2 &= \hat{S}_{21}\,\hat{a}_1 + \hat{S}_{22}\,\hat{a}_2 \end{aligned} \qquad (3.35b)$$

The reduction from $\tilde{S}$ to $\hat{S}$, i.e. the reduction to state variables, which uses dependency relations of the form (3.27) and (3.28) for fundamental C loops and L cutsets, respectively, is given with proofs for the general case in [24]. This reference also treats the removal of zero eigenvalues from $\hat{S}$, which otherwise result in unobservable modes in the state equations. Since the proof is lengthy we give here only the result for $\hat{S}$ when C loops exist without L cutsets, as in a later example. If $\overline{S}$ represents $\hat{S}$ rearranged by row and column operations to conform to link-twig ordering then $\overline{S}$ is given by

$$\overline{S} = \left[\begin{array}{c|c} -U & \overline{Q}_\ell^T \\ \hline 0 & U \end{array}\right] \left[\begin{array}{c|c} -U & 0 \\ \hline -2\overline{K} & U \end{array}\right] \left[\begin{array}{c|c} -U & \overline{Q}_\ell^T \\ \hline 0 & U \end{array}\right] \qquad (3.36)$$

where, referring to (3.19), $\overline{Q}_\ell^T$ is Q_ℓ^T with all rows corresponding to link capacitances deleted and $\overline{K}$ is K with all columns corresponding to link

capacitances deleted. $\hat{S}$ can be obtained from $\bar{S}$ by reordering.

Finally, element constraints are imposed between $\tilde{a}_1$ and $\tilde{b}_1$ in (3.32c) or $\hat{a}_1$ and $\hat{b}_1$ in (3.35b) by the differential equations for the inductive and capacitive network elements. For an inductance

$$v = -L\, s\, i \tag{3.37}$$

so by (3.9) and (3.24)

$$\begin{aligned} a &= v + L\, i = -L\, s\, i + L\, i = L\, i\, (1 - s) \\ b &= v - L\, i = -L\, s\, i - L\, i = -L\, i\, (1 + s) \end{aligned} \tag{3.38}$$

or

$$b = -\frac{1 + s}{1 - s}\, a \quad . \tag{3.39}$$

For a capacitance

$$i = -C\, s\, v \tag{3.40}$$

so by (3.9) and (3.24)

$$\begin{aligned} a &= v + i/C = v - s\, v = v\, (1 - s) \\ b &= v - i/C = v + s\, v = v\, (1 + s) \end{aligned} \tag{3.41}$$

or

$$b = \frac{1 + s}{1 - s}\, a \quad . \tag{3.42}$$

The inductive and capacitive constraints (3.39) and (3.42) can be denoted on a system basis by

$$\begin{bmatrix} b_{\ell L} \\ b_{tL} \\ b_{tC} \\ b_{\ell C} \end{bmatrix} = \frac{1 + s}{1 - s} \begin{bmatrix} -U_1 & & & \\ & -U_2 & & \\ & & U_3 & \\ & & & U_4 \end{bmatrix} \begin{bmatrix} a_{\ell L} \\ a_{tL} \\ a_{tC} \\ a_{\ell C} \end{bmatrix} \tag{3.43}$$

where U_1, U_2, U_3, and U_4 are identity matrices conformable with the corresponding vectors in (3.43). Defining

$$\tilde{\Sigma} = \begin{bmatrix} -U_1 & & & \\ & -U_2 & & \\ & & U_3 & \\ & & & U_4 \end{bmatrix} \tag{3.44a}$$

$$\hat{\Sigma} = \begin{bmatrix} -U_1 & \\ & U_3 \end{bmatrix} \tag{3.44b}$$

where we note

$$\begin{aligned} \tilde{\Sigma} &= \tilde{\Sigma}^{-1} \\ \hat{\Sigma} &= \hat{\Sigma}^{-1} \end{aligned} \tag{3.44c}$$

we have from (3.43)

$$\tilde{b}_1 = \frac{1+s}{1-s}\,\tilde{\Sigma}\,\tilde{a}_1 \tag{3.45}$$

and

$$\hat{b}_1 = \frac{1+s}{1-s}\,\hat{\Sigma}\,\hat{a}_1 \quad . \tag{3.46}$$

Substituting (3.45) and (3.46) in (3.32c) and (3.35b), respectively, and using (3.44c) we have the corresponding system equations

$$\frac{1+s}{1-s}\,\tilde{a}_1 = \tilde{\Sigma}\,\tilde{S}_{11}\,\tilde{a}_1 + \tilde{\Sigma}\,\tilde{S}_{12}\,\tilde{a}_2 \tag{3.47a}$$

$$\tilde{b}_2 = \tilde{S}_{21}\,\tilde{a}_1 + \tilde{S}_{22}\,\tilde{a}_2 \tag{3.47b}$$

and reduced system equations

$$\frac{1+s}{1-s}\,\hat{a}_1 = \hat{\Sigma}\,\hat{S}_{11}\,\hat{a}_1 + \hat{\Sigma}\,\hat{S}_{12}\,\hat{a}_2 \tag{3.48a}$$

$$\hat{b}_2 = \hat{S}_{21}\,\hat{a}_1 + \hat{S}_{22}\,\hat{a}_2 \tag{3.48b}$$

where all variables are Laplace transforms and zero initial conditions have been assumed. Both (3.47) and (3.48) are valid descriptions of the network, but (3.47) contains more variables than necessary.

State equations in standard form can be obtained for the network from the reduced system equations in (3.48) which contain as variables only the state vector $\hat{a}_1$, the input vector $\hat{a}_2$, and the output vector $\hat{b}_2$. A straightforward algebraic rearrangement of (3.48a) gives

$$(I + \hat{\Sigma}\hat{S}_{11})s\hat{a}_1 = -(I - \hat{\Sigma}\hat{S}_{11})\hat{a}_1 + (1 - s)\hat{\Sigma}\hat{S}_{12}\hat{a}_2 \quad (3.49)$$

which will yield the required form for the state equations if $(I + \hat{\Sigma}\hat{S}_{11})$ is nonsingular. The requirement that $(I + \hat{\Sigma}\hat{S}_{11})$ be nonsingular is equivalent to $(I + \hat{\Sigma}\hat{S}_{11})$ having no zero eigenvalues, which in turn means $\hat{\Sigma}\hat{S}_{11}$ must have no -1 eigenvalues, since the eigenvalues of $(I + \hat{\Sigma}\hat{S}_{11})$ are the eigenvalues of $\hat{\Sigma}\hat{S}_{11}$ increased by one. We show $\hat{\Sigma}\hat{S}_{11}$ has no -1 eigenvalues in practical networks by introducing the bilinear transformation

$$z = \frac{1 + s}{1 - s} \quad (3.50)$$

in the reduced system equations of (3.48). That is, we define the system

$$\begin{aligned} z\,\hat{a}_1(z) &= \hat{\Sigma}\,\hat{S}_{11}\,\hat{a}_1(z) + \hat{\Sigma}\,\hat{S}_{12}\,\hat{a}_2(z) \\ \hat{b}_2(z) &= \hat{S}_{21}\,\hat{a}_1(z) + \hat{S}_{22}\,\hat{a}_2(z) \end{aligned} \quad (3.51)$$

whose solution $\hat{a}_1(z)$ is a mapping through (3.50) of the solution $\hat{a}_1(s)$ for the system (3.48). Consider any component of the input $\hat{a}_2(s)$ as one and the others as zero. Since $\hat{a}_1(s)$ is related by (3.29) to a state vector with a known transform solution for which it can be shown no poles lie in the right half plane if the network is passive, no poles of $\hat{a}_1(s)$ lie in the right half plane. Also no poles of $\hat{a}_1(s)$ lie at infinity since the state vector as defined in (3.34) contains only finite modes . Hence by the mapping properties of the bilinear transformation shown in section 2.6 the poles of $\hat{a}_1(z)$ lie within or on the unit

circle, but not at -1. Since from (3.51)

$$(zI - \hat{\hat{\Sigma}}\hat{S}_{11}) \hat{a}_1(z) = \hat{\hat{\Sigma}}\hat{S}_{12} \hat{a}_2(z) \qquad (3.52)$$

the poles of $\hat{a}_1(z)$ must lie at the eigenvalues of $\hat{\hat{\Sigma}}\hat{S}_{11}$, which shows $\hat{\hat{\Sigma}}\hat{S}_{11}$ has no eigenvalues at -1 and hence the desired inverse $(I + \hat{\hat{\Sigma}}\hat{S}_{11})$ exists. Thus from (3.49) and (3.48b) we have the network state equations

$$\begin{aligned} s\,\hat{a}_1 &= -(I+\hat{\hat{\Sigma}}\hat{S}_{11})^{-1}(I-\hat{\hat{\Sigma}}\hat{S}_{11})\hat{a}_1 + (1-s)(I+\hat{\hat{\Sigma}}\hat{S}_{11})^{-1}\hat{\hat{\Sigma}}\hat{S}_{12}\hat{a}_2 \\ \hat{b}_2 &= \hat{S}_{21}\hat{a}_1 + \hat{S}_{22}\hat{a}_2 \end{aligned} \qquad (3.53)$$

3.4 Derivation of Wave Digital Filters

The canonic wave digital filter structure derives directly from (3.51), the bilinear transformation of the reduced system equations (3.48). The wave digital filter is defined by the recursive relations

$$\begin{aligned} \hat{a}_1(n+1) &= \hat{\hat{\Sigma}}\hat{S}_{11}\hat{a}_1(n) + \hat{\hat{\Sigma}}\hat{S}_{12}\hat{a}_2(n) \\ \hat{b}_2(n) &= \hat{S}_{21}\hat{a}_1(n) + \hat{S}_{22}\hat{a}_2(n) \end{aligned} \qquad (3.54)$$

where one observes (3.51) is the z-transform representation of (3.54). Since (3.51) and (3.48) are related by the bilinear transformation the frequency response from any input of (3.54) to any state or output is the corresponding transfer ratio in the original, or prototype, network with the nonlinear mapping (2.43) between frequency variables, as explained in section 2.6.

A wave digital structure based on the prototype network also derives from the bilinear transformation of the unreduced system equations in (3.47). The corresponding wave digital filter is defined by

$$\begin{aligned} \tilde{a}_1(n+1) &= \tilde{\tilde{S}}_{11}\tilde{a}_1(n) + \tilde{\tilde{S}}_{12}\tilde{a}_2(n) \\ \tilde{b}_2(n) &= \tilde{S}_{21}\tilde{a}_1(n) + \tilde{S}_{22}\tilde{a}_2(n) \end{aligned} \tag{3.55}$$

which will exhibit the same properties regarding frequency response as the canonic wave digital filter. However (3.55) is not canonic in the number of variables unless C loops and L cutsets are absent in the prototype network. The wave digital filters originally introduced by Fettweis [12,13] were not necessarily canonic, but canonic structures have since been investigated [21,22,24].

The wave digital filter given by (3.54) or (3.55) implies the following element transformations from the prototype network to the discrete time domain:

a) A resistance with series voltage source e transforms to a wave source a(n) = e(nT) and a wave sink b(n) corresponding to 2v -e where v is the voltage across the series combination.

b) An inductance transforms to a delay with sign inversion, i.e. a(n+1) = -b(n).

c) A capacitance transforms to a delay, i.e. a(n+1) = b(n).

These element transformations are shown in figure 3.1. The transformed elements are combined with a hardware realization of $\hat{S}$ or $\tilde{S}$, which represents the frequency independent topological structure of the prototype network, to produce the wave digital filter. Figure 3.2 shows the general structure of the wave digital filter.

3.5 Alternate Structures for the Scattering Matrix

The scattering matrix $\tilde{S}$, which is central to the realization of a wave digital filter (WDF), as shown in figure 3.2, can be composed

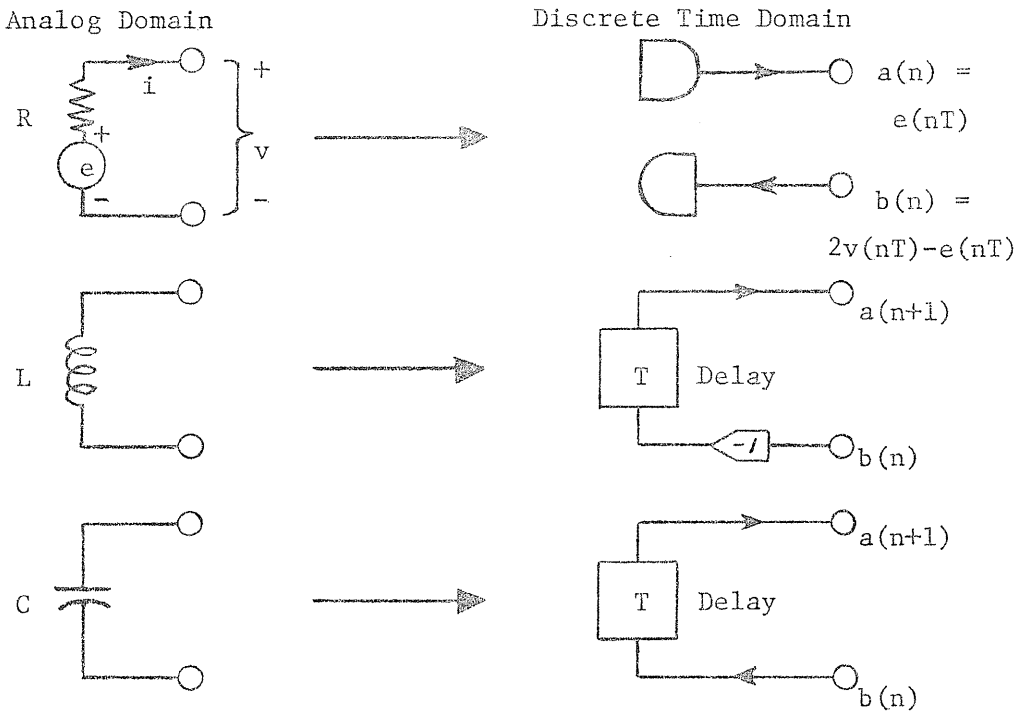


Figure 3.1 Wave Digital Element Transformations

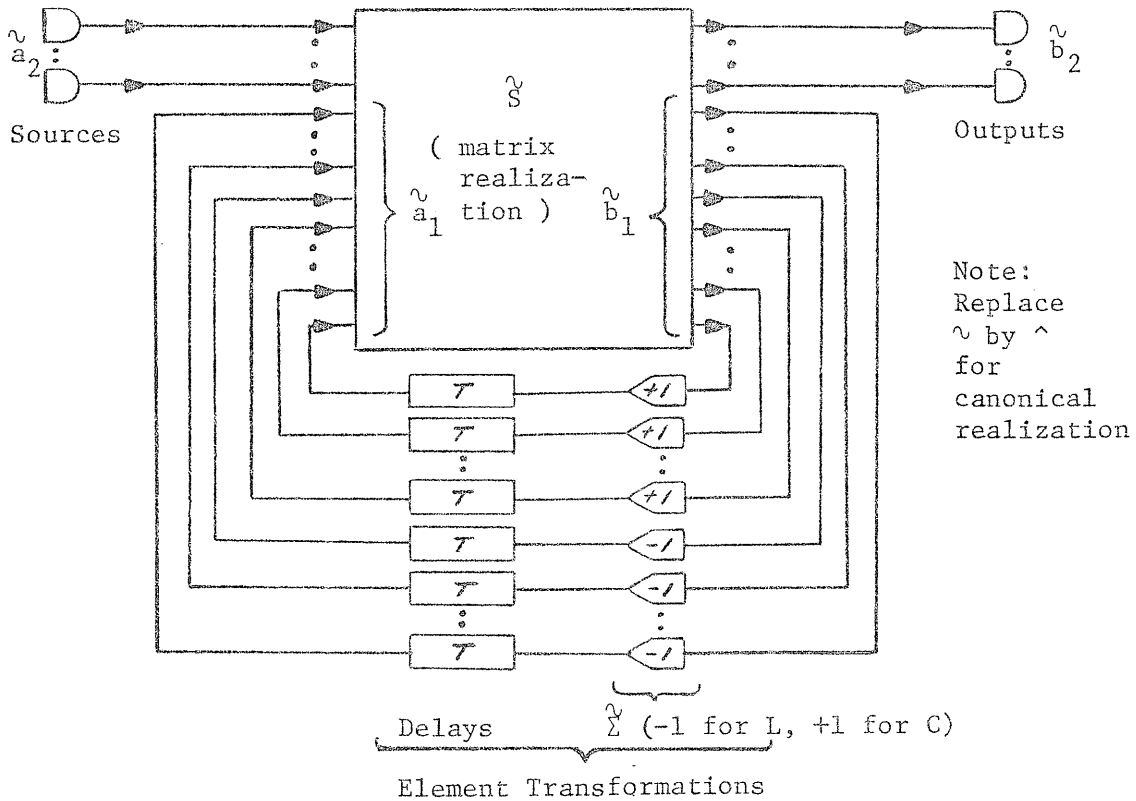


Figure 3.2 General Structure of Wave Digital Filter

of building blocks called adaptors. That is, $\tilde{S}$ corresponds to a network topology or graph which can be represented by the interconnection of topological subsections or subgraphs. At an interconnection of terminal pairs from two subsections 1 and 2 we observe

$$\begin{aligned} v_1 &= v_2 \\ i_1 &= -i_2 \end{aligned} \tag{3.56}$$

which gives

$$\begin{aligned} a_1 &= b_2 \\ b_1 &= a_2 \end{aligned} \tag{3.58}$$

where the corresponding reference resistances R_1 and R_2 must be equal. The interconnection of adaptors representing topological substructures is in fact the original method of realizing WDF's and adaptors have been derived for series [13,14], parallel [13,14], lattice [23], Darlington D [25], and Brune sections [25].

The generalized or n-port adaptor is embodied in (3.19) which can also be applied to any topological subsection of the network graph. Thus the previously mentioned adaptor types can all be derived with (3.19). The application of (3.19) to the entire network yields the n-port adaptor for $\tilde{S}$ in the realization of figure 3.2. It is therefore of interest to investigate possible hardware realizations of n-port adaptors directly from (3.19). This is the central topic of the thesis which is treated in chapter 4. The approach taken there is numerical since the specification of $\tilde{S}$ by a matrix of binary numbers computed with (3.19) allows us to consider all possible realizations. Similarly, direct realizations of the reduced scattering matrix $\hat{S}$ are possible for canonic WDF's. A method of achieving $\tilde{S}$ or $\hat{S}$ with binary entries

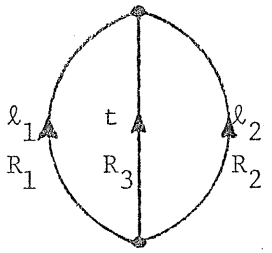
required for hardware realization is given below.

The decomposition of $\tilde{S}$ as an interconnection of basic adaptors, where possible, provides valuable information about the entries of $\tilde{S}$. In WDF designs it is necessary that one of the two ports at an interconnection between adaptors be reflection-free, i.e. the corresponding entry on the diagonal of one adaptor scattering matrix must be zero. It can be shown this is equivalent to defining the reference resistance at the reflection-free port as the resistance seen looking into the remaining network of reference resistances for the adaptor. This reflection-free requirement is necessary to guarantee the computability of the structure, i.e. the absence of delay free loops [13]. This same arrangement of reflection-free ports means each entry of $\tilde{S}$ can be written as a sum of products of appropriate entries of the individual adaptor scattering matrices by following an orderly procedure, i.e. the structure is also "computable by hand". Further, from (3.19), the entries of each adaptor scattering matrix will be finite binary numbers if the entries of K are finite binary numbers and we exclude the possibility of ideal transformers. The entries of K in the adaptor scattering matrices are the multipliers in the individual adaptors, so if the multipliers are chosen as binary numbers $\tilde{S}$ will contain only binary numbers. Thus the prototype filter optimization which is normally performed to achieve binary adaptor multipliers required for hardware realization will also furnish $\tilde{S}$ with binary entries. It is sufficient to determine K by "hand computing" the adaptor structure since S, and hence $\tilde{S}$, follows with (3.19). Also by (3.36) this procedure will furnish $\overline{S}$, and hence $\hat{S}$, with binary entries when loops of C exist without cutsets of L since $\overline{K}$ is a submatrix of K. An

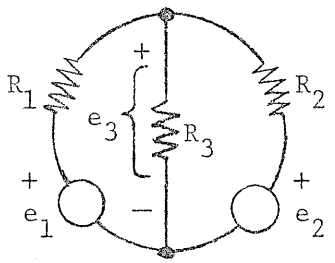
alternate procedure for determining K given in [16] with essentially the same results is the Thevenin reduction of the network of reference resistances for $\tilde{S}$.

An analog prototype filter with low sensitivity of its transfer function to element variations is most desirable for an optimization which requires binary multipliers in the adaptors of the corresponding WDF. For this reason the doubly terminated LC ladder is often chosen as the prototype filter. The ladder structure can be realized as a suitable combination of series and parallel adaptors [13]. We now derive the scattering matrices of the three port series and parallel adaptors which will be of use in the network example of section 3.7:

A. Parallel Adaptor



Network Topology
(oriented graph)



Equivalent Circuit
for Determination of K

$$Q = \left[\begin{array}{c|c} Q_\ell & U \end{array}\right] = \left[\begin{array}{cc|c} 1 & 1 & 1 \end{array}\right]$$

$$Q_\ell = \left[\begin{array}{cc} 1 & 1 \end{array}\right]$$

$$b_t = 2 K a_\ell$$

$$K = \left[\begin{array}{cc} e_3/e_1 & e_3/e_2 \end{array}\right] = \left[\begin{array}{cc} \alpha_1/2 & \alpha_2/2 \end{array}\right]$$

$$G_1 = 1/R_1 \ , \ G_2 = 1/R_2 \ , \ G_3 = 1/R_3 \ , \ G_T = G_1 + G_2 + G_3$$

By nodal analysis

$$\alpha_1 = 2G_1/G_T \ , \ \alpha_2 = 2G_2/G_T \ .$$

From (3.19)

$$S = \left[\begin{array}{cc:c} -1 & 0 & 1 \\ 0 & -1 & 1 \\ \hdashline 0 & 0 & 1 \end{array}\right]\left[\begin{array}{cc:c} -1 & 0 & 0 \\ 0 & -1 & 0 \\ \hdashline -\alpha_1 & -\alpha_2 & 1 \end{array}\right]\left[\begin{array}{cc:c} -1 & 0 & 1 \\ 0 & -1 & 1 \\ \hdashline 0 & 0 & 1 \end{array}\right]$$

If ℓ_2 is a reflection-free port $G_2 = G_1 + G_3$ or

$$\alpha_1 = G_1/(G_1+G_3)$$

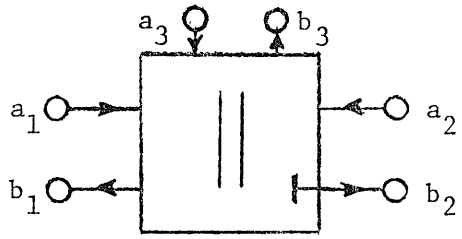
$$\alpha_2 = 1$$

Hence

$$S = \begin{bmatrix} -(1-\alpha_1) & 1 & 1-\alpha_1 \\ \alpha_1 & 0 & 1-\alpha_1 \\ \alpha_1 & 1 & -\alpha_1 \end{bmatrix} \begin{array}{l} \leftarrow \ell_1 \\ \leftarrow \ell_2 \\ \leftarrow t \end{array}$$

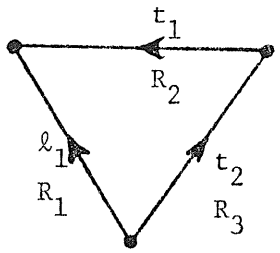
$$\begin{array}{ccc} \uparrow & \uparrow & \uparrow \\ \ell_1 & \ell_2 & t \end{array}$$

From [14] S can be realized with 1 multiplier, 4 adders, and 2 inverters, where t corresponds to the dependent port. The symbol for this adaptor is

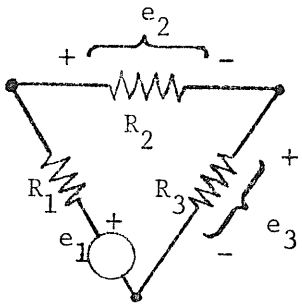


where the stroked output b_2 indicates the reflection-free port.

B. Series Adaptor



Network Topology
(oriented graph)



Equivalent Circuit
for Determination of K

$$Q = \left[\begin{array}{c|cc} Q_\ell & U \end{array}\right] = \left[\begin{array}{c|cc} 1 & 1 & 0 \\ 1 & 0 & 1 \end{array}\right]$$

$$Q_\ell = \begin{bmatrix} 1 \\ 1 \end{bmatrix}$$

$$b_t = 2\,K\,a_\ell$$

$$K = \begin{bmatrix} e_2/e_3 \\ e_3/e_1 \end{bmatrix} = \begin{bmatrix} \alpha_1/2 \\ \alpha_2/2 \end{bmatrix}$$

By loop analysis

$$\alpha_1 = 2R_2/(R_1+R_2+R_3) \quad , \quad \alpha_2 = 2R_3/(R_1+R_2+R_3)$$

From (3.19)

$$S = \left[\begin{array}{c|cc} -1 & 1 & 1 \\ \hline 0 & 1 & 0 \\ 0 & 0 & 1 \end{array}\right]\left[\begin{array}{c|cc} -1 & 0 & 0 \\ \hline -\alpha_1 & 1 & 0 \\ -\alpha_2 & 0 & 1 \end{array}\right]\left[\begin{array}{c|cc} -1 & 1 & 1 \\ \hline 0 & 1 & 0 \\ 0 & 0 & 1 \end{array}\right]$$

If t_1 is a reflection-free port $R_2 = R_1 + R_3$ or

$$\alpha_1 = 1$$

$$\alpha_2 = R_3/(R_1+R_3)$$

Hence

$$S = \begin{bmatrix} \alpha_2 & 1-\alpha_2 & 1-\alpha_2 \\ 1 & 0 & -1 \\ \alpha_2 & -\alpha_2 & 1-\alpha_2 \end{bmatrix} \begin{matrix} \leftarrow \ell_1 \\ \leftarrow t_1 \\ \leftarrow t_2 \end{matrix}$$

$$\begin{matrix} \uparrow & \uparrow & \uparrow \\ \ell_1 & t_1 & t_2 \end{matrix}$$

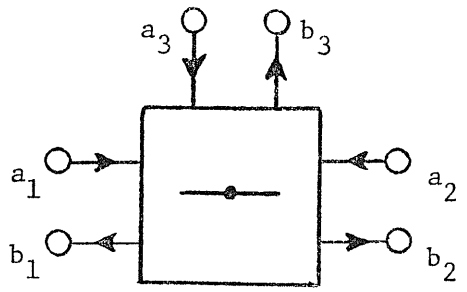
From [14] S can be realized with 1 multiplier, 4 adders, and 2 inverters,where ℓ_1 corresponds to the dependent port. If t_1 is not reflection-free, but instead $R_1 = R_2$, i.e. the adaptor is symmetric, then

$\alpha_1 = \alpha_2$.

Hence

$$S = \begin{bmatrix} -1+2\alpha_1 & 2-2\alpha_1 & 2-2\alpha_1 \\ \alpha_1 & 1-\alpha_1 & -\alpha_1 \\ \alpha_1 & -\alpha_1 & 1-\alpha_1 \end{bmatrix}$$

In this case S can be realized with 1 multiplier, 6 adders, and 2 inverters. The symbol for the series adaptor is



where a stroked output would indicate a reflection-free port.

3.6 Properties of the Scattering Matrix and Related Properties of Wave Digital Filters

The WDF characterized in (3.54) is a discrete time system expressed with state equations, as discussed in section 2.5. Here the state transition matrix A corresponds to $\tilde{\tilde{S}}_{11}$ and the WDF is an infinite impulse response (IIR) system. Since a prototype analog filter with low sensitivity to element variations can be optimized to give fairly simple binary multiplier coefficients in the adaptors, it follows from section 3.5 that the entries of $\tilde{S}_{11}$ will be binary numbers with a moderate number of digits. This is in contrast to the high sensitivity of system response to multiplier coefficients in the phase canonical form described in section 2.5. That is, although the phase canonical form requires a minimum number of entries in the matrix A, the WDF can be hardware competitive since it does not require extremely long word-lengths in $\tilde{\tilde{S}}_{11}$. It has also been shown that the low sensitivity of the WDF design to adaptor multipliers implies favorable roundoff noise characteristics following each multiplier [20].

The scattering matrix S or $\tilde{S}$, derived in section 3.3, possesses the following properties:

a) $S = S^{-1}$. This is seen directly from (3.19).

b) $S^T G S = G$. This is the lossless property which follows from conservation of energy in the network, i.e.

$$v^T i = 0. \tag{3.58}$$

Substituting

$$v = (a + b)/2 \quad , \quad i = G (a - b)/2 \tag{3.59}$$

gives

$$(a + b)^T G (a - b) = 0$$

or

$$a^T G a - b^T G b = 0 \quad . \tag{3.60}$$

Substituting $b = S a$ in (3.60) gives

$$a^T G a = (Sa)^T G Sa = a^T S^T G S a \tag{3.61}$$

which must hold for an arbitrary vector a. Hence

$$G = S^T G S \quad . \tag{3.62}$$

c) $S^T G = G S$. This is the reciprocity property which follows from b) by postmultiplying by S^{-1} and substituting a).

Note that any two of a), b), and c) imply the third.

The lossless property in b) implies zero input limit cycles, due to the inevitable nonlinear effects of digit truncations, can be eliminated in hardware realizations of WDF's [18]. It also implies overflow oscillations in the forced response of WDF's with two's complement arithmetic can be eliminated [19]. Since limit cycles have posed considerable problems in conventional digital filters the above are significant characteristics of WDF's. Specifically, zero input limit cycles are suppressed by requiring

$$\tilde{a}_1(n+1) = [\, \tilde{\tilde{S}}_{11} \tilde{a}_1(n) + \tilde{\tilde{S}}_{12} \tilde{a}_2(n) \,]\# \tag{3.63}$$

in (3.55) where [ ]# denotes sign magnitude truncation of the enclosed quantity. Sign magnitude truncation means the components of $\tilde{a}_1(n+1)$ are rounded down in magnitude to the nearest discrete value available in the finite binary wordlength alloted for $\tilde{a}_1(n)$ in a hardware implementation. Some form of digit truncations is always required prior to feedback since fractional values in $\tilde{S}_{11}$ generate excess digits to the right during computation of $\tilde{a}_1(n+1)$. The absence of zero input limit cycles is shown by defining a Lyapunov function which is the positive definite form

$$V(n) = \underset{\sim}{a}_1^T(n)\, \underset{\sim}{G}_{11}\, \underset{\sim}{a}_1(n) \quad . \tag{3.64}$$

Since

$$\underset{\sim}{S}^T\, \underset{\sim}{G}\, \underset{\sim}{S} = \underset{\sim}{G} \tag{3.65}$$

it can be shown

$$x^T\, \underset{\sim}{S}_{11}^T\, \underset{\sim}{G}_{11}\, \underset{\sim}{S}_{11}\, x \leq x^T \underset{\sim}{G}_{11} x \tag{3.66}$$

for any conformable vector x. Also

$$(\underset{\sim}{\Sigma S}_{11})^T \underset{\sim}{G}_{11} (\underset{\sim}{\Sigma S}_{11}) = \underset{\sim}{S}_{11}^T \underset{\sim}{\Sigma}^T \underset{\sim}{G}_{11} \underset{\sim}{\Sigma S}_{11} = \underset{\sim}{S}_{11}^T \underset{\sim}{G}_{11} \underset{\sim}{S}_{11} \tag{3.67}$$

since $\underset{\sim}{\Sigma} = \underset{\sim}{\Sigma}^T = \underset{\sim}{\Sigma}^{-1}$ is a diagonal matrix. Hence

$$x^T\, (\underset{\sim}{\Sigma S}_{11})^T\, \underset{\sim}{G}_{11}\, (\underset{\sim}{\Sigma S}_{11})\, x \leq x^T\, \underset{\sim}{G}_{11}\, x \quad . \tag{3.68}$$

For zero input

$$\underset{\sim}{a}_1(n+1) = \underset{\sim}{\Sigma}\, \underset{\sim}{S}_{11}\, \underset{\sim}{a}_1(n) \tag{3.69}$$

so

$$\begin{aligned} V(n+1) &= \underset{\sim}{a}_1^T(n+1)\, \underset{\sim}{G}_{11}\, \underset{\sim}{a}_1(n+1) = (\underset{\sim}{\Sigma S}_{11} \underset{\sim}{a}_1(n))^T \underset{\sim}{G}_{11} (\underset{\sim}{\Sigma S}_{11} \underset{\sim}{a}_1(n)) \\ &= \underset{\sim}{a}_1^T(n) (\underset{\sim}{\Sigma S}_{11})^T \underset{\sim}{G}_{11} (\underset{\sim}{\Sigma S}_{11}) \underset{\sim}{a}_1(n) \\ &\leq \underset{\sim}{a}_1^T(n)\, \underset{\sim}{G}_{11}\, \underset{\sim}{a}_1(n) = V(n) \quad . \end{aligned} \tag{3.70}$$

With sign magnitude truncation (3.69) becomes

$$\underset{\sim}{a}_1(n+1) = [\, \underset{\sim}{\Sigma}\, \underset{\sim}{S}_{11} \underset{\sim}{a}_1(n)\,]\# \tag{3.71}$$

and (3.70) becomes

$$V(n+1) < V(n) \quad . \tag{3.72}$$

Now since the WDF is a finite state machine, with zero input the state vector eventually either becomes zero or assumes a previous state, in which case a periodic limit cycle exists. Since the second possibility is incompatible with (3.72) no limit cycles exist. It has also been shown that WDF's with adaptor constructions for $\underset{\sim}{S}$ have the property (3.72) and hence no zero input limit cycles if the adaptors are

individually "passified" by the application of sign magnitude truncation [18]. Unfortunately, the reduced scattering matrix $\hat{S}$ required for canonic WDF's does not necessarily have the property in b) for a diagonal matrix G so alternate procedures have been suggested for ensuring a positive definite function, such as (3.64), is strictly decreasing with zero input [24,25]. In the case of the third order network example in section 3.7,it has been shown a satisfactory Lyapunov function exists for the canonical form.

3.7 Network Example for Realization with n-Port Adaptor

We choose the network in figure 3.3, which is a third order elliptic filter, as a prototype for a WDF realization with an n-port adaptor. Since the network contains a C loop the reduced scattering matrix $\hat{S}$ will be used for a canonic realization. The element values are first optimized to give binary multipliers in the adaptor construction of S from which the corresponding S or $\hat{S}$ with binary entries is obtained as previously described.

The oriented network graph is shown in figure 3.4a with the reference tree chosen for the calculation of S via (3.19). An interconnection of series and parallel adaptor topologies which produces the network topology is shown in figure 3.4b and the corresponding adaptor construction of the WDF with transformed elements is shown in figure 3.5. In figures 3.4b and 3.5 the edges in the adaptor topologies and the ports of the adaptors are numbered to correspond with the ordering of the rows and columns of the adaptor scattering matrices derived in section 3.5. A multiplier is indicated at each adaptor port which is neither dependent nor reflection-free. It is seen from the

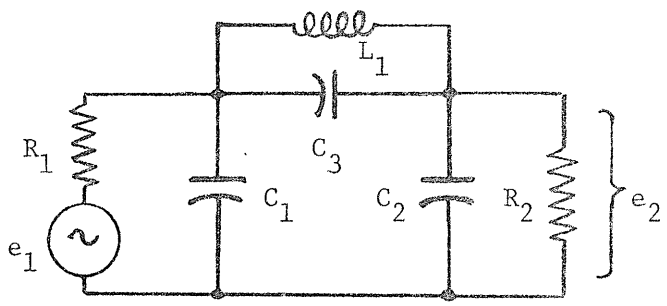
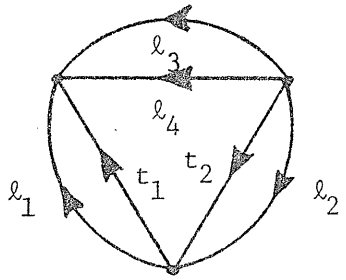
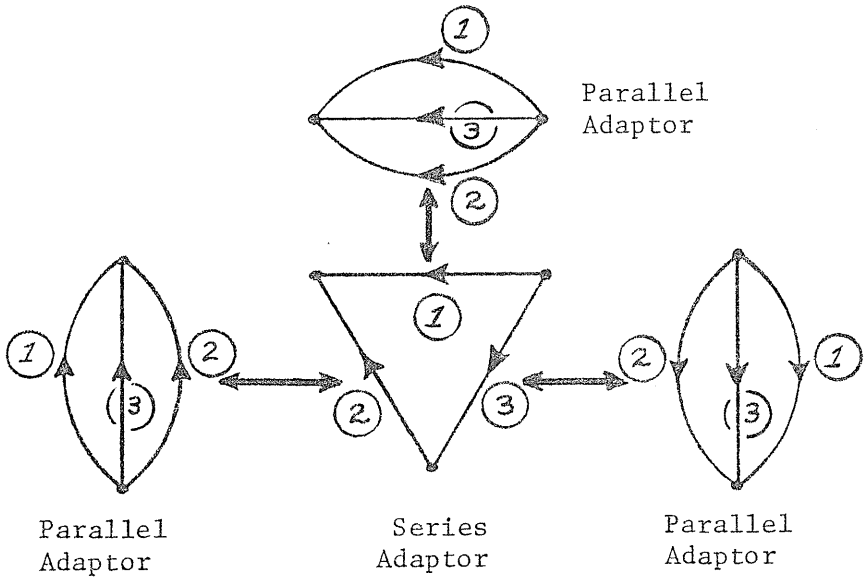


Figure 3.3 Analog Prototype for Wave Digital Filter Design



a) Oriented Network Graph with Chosen Tree



b) Network Topology Constructed from Adaptor Topologies

Figure 3.4 Network Topology

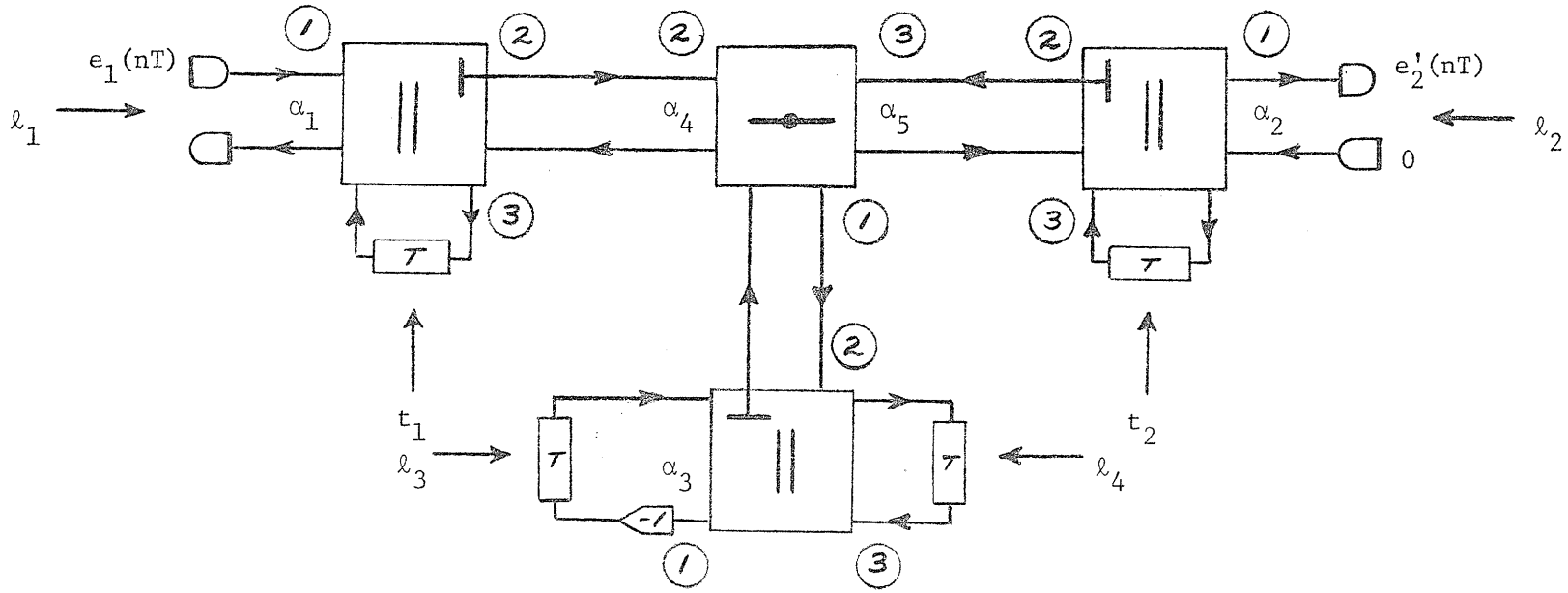


Figure 3.5 Adaptor Realization for Analog Prototype

adaptor derivations that

$$\alpha_1 = (1/R_1)/(1/R_1+C_1) = 1/(1+R_1C_1)$$

$$\alpha_2 = (1/R_2)/(1/R_2+C_2) = 1/(1+R_2C_2)$$

$$\alpha_3 = (1/L_1)/(1/L_1+C_3) = 1/(1+L_1C_3) \quad .$$

Since the ports of the three parallel adaptors interconnected to the series adaptor are reflection-free, the reference resistances for ports (1), (2), and (3) of the series adaptor are $\alpha_3 L_1$, $\alpha_1 R_1$, and $\alpha_2 R_2$, respectively. Thus

$$\alpha_4 = 2\alpha_1 R_1/(\alpha_1 R_1 + \alpha_2 R_2 + \alpha_3 L_1)$$

$$\alpha_5 = 2\alpha_2 R_2/(\alpha_1 R_1 + \alpha_2 R_2 + \alpha_3 L_1) \quad .$$

The matrix 2K corresponding to S is determined from the adaptor realization as the set of transfer constants between appropriate ports, i.e. $b_\ell = 2\ K\ a_t$. The links and twigs for the network reference tree are labelled in the network graph of figure 3.4 and the corresponding ports of the adaptor realization in figure 3.5 are indicated by ℓ_1, ℓ_2, ℓ_3, ℓ_4, t_1, and t_2. The required transfer constants from ports ℓ_1, ℓ_2, ℓ_3, and ℓ_4 to ports t_1 and t_2 are obtained by applying the scattering matrices for series and parallel adaptors given in section 3.5 with substitution of the appropriate multipliers α_1, α_2, α_3, α_4, or α_5 from figure 3.5. The computability of the structure in figure 3.5 resulting from the arrangement of reflection-free ports guarantees an orderly pattern can be developed for determining the transfer constants by assuming an input at any one of the link ports and calculating the response at each one of the twig ports through the cascade of adaptors. For this structure we obtain

$$2K = \begin{bmatrix} \alpha_1(2-\alpha_4) & -\alpha_2\alpha_4 & \alpha_3\alpha_4 & (1-\alpha_3)\alpha_4 \\ -\alpha_1\alpha_5 & \alpha_2(2-\alpha_5) & \alpha_3\alpha_5 & (1-\alpha_3)\alpha_5 \end{bmatrix} .$$

The cutset matrix Q is determined from figure 3.4a as

$$Q = \left[\, Q_\ell \,\vdots\, U \,\right] = \left[\begin{array}{cccc:cc} 1 & 0 & 1 & 1 & 1 & 0 \\ 0 & 1 & 1 & 1 & 0 & 1 \end{array}\right]$$

(Q_ℓ: the first four columns)

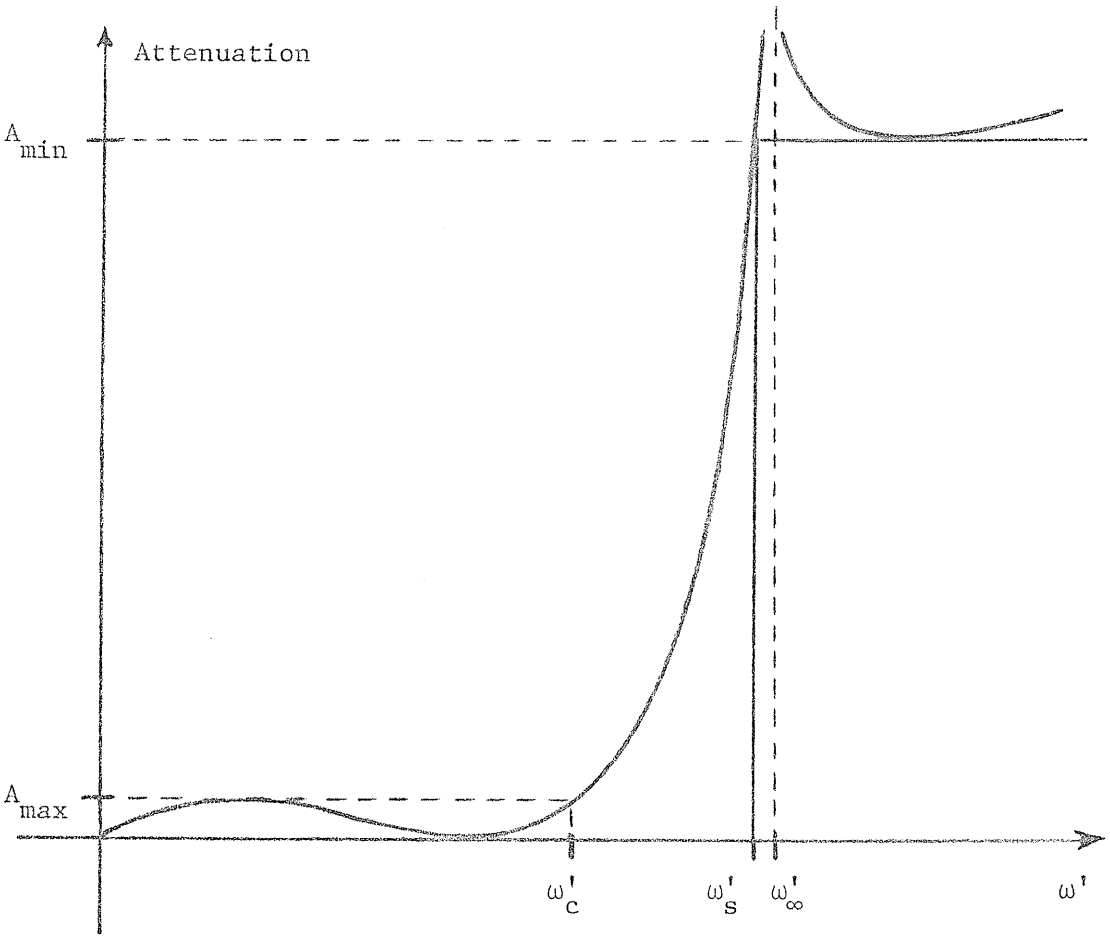
The analog prototype filter in figure 3.3 has been optimized by K. Meerkötter for the specification shown in figure 3.6 to the values

$$R_1 = 1$$
$$R_2 = 1$$
$$C_1 = 1$$
$$C_2 = 1$$
$$C_3 = 1/16$$
$$L_1 = 16/15$$

which give the binary multipliers

$$\alpha_1 = 1/2$$
$$\alpha_2 = 1/2$$
$$\alpha_3 = 15/16$$
$$\alpha_4 = 1/2$$
$$\alpha_5 = 1/2 \quad .$$

The resulting lowpass characteristic meets the requirements specified in figure 3.6. Since the excess capacitance C_3 lies in link ℓ_4 we delete the fourth column of 2K to obtain $2\overline{K}$ for use in (3.36), which for the above multipliers is



A_{max} = .12 db

A_{min} = 39 db

ω'_c = 1.02

ω'_s = 3.4

$\omega'_\infty = \sqrt{15}$ = 3.87 ..

Figure 3.6 Frequency Response of Optimized Prototype Filter

$$2\overline{K} = \begin{bmatrix} 3/4 & -1/4 & 15/32 \\ -1/4 & 3/4 & 15/32 \end{bmatrix} .$$

Likewise we delete the fourth column of Q_ℓ to obtain

$$\overline{Q}_\ell = \begin{bmatrix} 1 & 0 & 1 \\ 0 & 1 & 1 \end{bmatrix}$$

for use in (3.36). From (3.36) the reduced scattering matrix is

$$\overline{S} = (1/32)\begin{bmatrix} -8 & -8 & 15 & 25 & -7 \\ -8 & -8 & 15 & -7 & 25 \\ 16 & 16 & -2 & 18 & 18 \\ 24 & -8 & 15 & -7 & -7 \\ -8 & 24 & 15 & -7 & -7 \end{bmatrix} \begin{matrix} \leftarrow \ell_1 \\ \leftarrow \ell_2 \\ \leftarrow \ell_3 \\ \leftarrow t_1 \\ \leftarrow t_2 \end{matrix} .$$

$$\begin{matrix} \uparrow & \uparrow & \uparrow & \uparrow & \uparrow \\ \ell_1 & \ell_2 & \ell_3 & t_1 & t_2 \end{matrix}$$

$\overline{S}$ may be reordered to give $\hat{S}$, which is based on the enumeration of inductive, capacitive, and resistive ports in that order for state equations. However this is not necessary since the WDF can be realized simply by observing the proper element types at the terminals of $\overline{S}$. Since input $a_{\ell 2}$ and output $b_{\ell 1}$ are not required in the filter the corresponding column and row of $\overline{S}$ may be deleted to realize the filter as

$$\bar{S}_R = \begin{bmatrix} -8 & 15 & -7 & 25 \\ 16 & -2 & 18 & 18 \\ 24 & 15 & -7 & -7 \\ -8 & 15 & -7 & -7 \end{bmatrix} \begin{matrix} \leftarrow \ell_2 \\ \leftarrow \ell_3 \\ \leftarrow t_1 \\ \leftarrow t_2 \end{matrix}$$

$$\begin{matrix} \uparrow & \uparrow & \uparrow & \uparrow \\ \ell_1 & \ell_3 & t_1 & t_2 \end{matrix}$$

For this example it is seen that

$$\hat{S}_{11} = \begin{bmatrix} -2 & 18 & 18 \\ 15 & -7 & -7 \\ 15 & -7 & -7 \end{bmatrix}$$

and it can be shown [24] $G - \hat{S}_{11}^T G \hat{S}_{11}$ is positive semidefinite for

$$G = \begin{bmatrix} 5 & & \\ & 6 & \\ & & 6 \end{bmatrix} .$$

From this it follows that sign magnitude truncation suppresses zero input limit cycles in this example by a proof similar to that given in section 3.6. A method of hardware realization for n-port adaptors, and reduced n-port adaptors such as $\bar{S}$ and $\bar{S}_R$, is given in chapter 4.

We observe that the series/parallel adaptor realization of this filter requires 5 multipliers, 18 adders, and 9 inverters, where two adders and one inverter can be deleted by eliminating the unused input and output. The multipliers α_1, α_2, α_4, and α_5 require no hardware since they are simple shifts, while α_3 requires only a single adder. The delay corresponding to the excess capacitance can be eliminated by

the technique described in [21]. However, since this example is a symmetric filter it has been pointed out [24] that a lattice adaptor realization is possible requiring essentially 8 adders and some inverters.

4. REALIZATION OF N-PORT ADAPTORS WITH BINARY ARITHMETIC

4.1 Introduction

The scattering matrix, S or $\tilde{S}$, or the reduced scattering matrix, $\bar{S}$ or $\hat{S}$, which represents an n-port adaptor derived from the topology of a prototype analog filter must be realized to produce the corresponding wave digital filter when terminated with appropriate sources, sinks, and feedback through delays. Hereafter we will simply use S to denote any of these matrices and a and b to denote the corresponding vectors of scattering variables since the methods of this chapter apply to any of these cases which were given in chapter 3. The port reference values for the unreduced scattering matrix correspond to the element values of the prototype filter, which have been adjusted within a frequency response constraint to yield matrix elements which are binary numbers of finite length. Given the above assumptions, the final step in the design of a wave digital filter is a translation from the matrix specification for S to a flow graph or block diagram realization containing only certain binary arithmetic operations. We may regard this translation as a purely numerical process without any special relation to the filter theory.

We desire a translation from the matrix operator S to a realization which is optimal in some sense, where the criteria for optimality are dictated by practical considerations. The first (and often strongest) criterion is minimization of the number of arithmetic operations required in the realization, or at least minimization of

those arithmetic operations deemed most costly. A second criterion is the relative difficulty of computing a full precision result with the realization, i.e. the number of underflow bits required during internal calculations. Often a third criterion is the computation time set by a particular structure. Since one may not assume a single realization can be optimal by more than one criterion, it is only reasonable to attempt optimization by an individual criterion, although we may examine the result by multiple criteria.

In the remainder of this chapter we consider the problem of translation from the matrix operator S to a structure which is directly realizable with binary arithmetic. A method of decomposition of the matrix S to add and shift operations is presented which uses a binary series expansion in the canonical signed digit code with matrix coefficients. The decomposition is reasonably hardware efficient and yields a two's complement arithmetic structure in which addition operations of the same wordlength as the inputs to the matrix are adequate for calculating all bits in the outputs. Sign magnitude truncation is then applied during recursive filter operation. The decomposition can be further applied in reducing the number of adders required but with the loss of the above property regarding full precision results. The general problem of minimizing the number of arithmetic operations in the realization has not been solved, either here or apparently in the literature, and warrants further study.

4.2 Reduction of the Problem to Integer Arithmetic

The scattering matrix S consists of elements which may be regarded as integers under suitable scaling since we have restricted

the elements to binary numbers of finite length. This allows a simple statement of the realization problem in terms of integer arithmetic. The necessary scale factor is 2^M, where M is the maximum number of bits to the right of the radix point in any matrix element, and we set

$$S_I = 2^M S \tag{4.1}$$

which gives the matrix of integers S_I. Since the elements of S are less than two in magnitude the elements of S_I must lie between -2^{M+1} and 2^{M+1}, but often, as in the example of section 3.7, the elements of S_I lie between -2^M and 2^M. Hence in the following we assume that the elements of S_I lie between -2^M and 2^M, and note that the more general case follows simply by increasing M by one in the appropriate places. The problem is now the realization of the matrix operator S_I consisting of integer elements, where the outputs will be divided by the scale factor 2^M to give results corresponding to the matrix S. The division by 2^M is obviously a shift of the radix point M places to the left.

The components of the vectors a(n) and b(n) may also be regarded as integers due to the finite wordlength requirement in any hardware realization. These vectors enter into the computation of (3.54) or (3.55) using the hardware realization for S during recursive operation of the WDF. We assume the components of a(n) are fixed point integers of length m bits and note that the components of $a_1(n+1)$ computed from (3.54) or (3.55) will have non-integer parts since the elements of S have fractional parts. Hence the implied feedback relation must be implemented by taking integers for $a_1(n+1)$ which only approximate the true values. For wave digital filters it was shown that sign magnitude truncation in the feedback path suppresses limit cycle oscillations, so relation (3.63) will be used.

4.3 Restrictions on the Realization

The realizations considered for the matrix operator S_I will contain only the operations of addition, sign inversion, and transposition of the radix point. A single addition may combine only two operands, but the result of an operation may branch to any number of points. Subtraction is performed as a composite of the two basic operations of sign inversion and addition, while multiplication is performed efficiently as a composite of all three basic operations. Hence the above class of realizations contains a minimal set of operations adequate for the efficient realization of matrix operators for recursive digital filters. Because transfer function designs require extremely long multiplier coefficients in their state transition matrix nearly all their addition operations occur within the multipliers, which has lead to the common assumption that digital filter complexity is indicated by the number of multipliers. However, wave digital filters have relatively short multiplier coefficients, as well as a "full" state transition matrix, which means the previous assumption is not accurate. In fact the specification of all operations in terms of additions, sign inversions, and shifts removes the difficulty of specifying one or more multiplication operations whose complexities depend entirely on the number and pattern of bits in the multipliers. This allows an accurate comparison of the actual number of arithmetic operations required in filter realizations produced by different techniques, such as the wave digital and the transfer function designs.

Any possible realization for S_I may be generated from a corresponding set of expressions which are arithmetically equivalent to S_I. This follows since we may obtain such a set of expressions from the

realization itself. Thus we will denote the translation from S_I to a block diagram realization with a set of arithmetic expressions. In view of the previous restrictions on the realization these expressions need only contain the operations of addition, multiplication by negative one to symbolize sign inversion, and multiplication by a power of two to symbolize a shift of the radix point.

4.4 Binary Decomposition of the Integer Matrix

We utilize a decomposition of the integer matrix S_I based on a binary series expansion of each matrix element. The matrix is denoted by

$$S_I = \begin{bmatrix} I_{11} & I_{12} & \cdots & I_{1\ell} \\ I_{21} & I_{22} & \cdots & I_{2\ell} \\ \vdots & \vdots & & \vdots \\ I_{\ell 1} & I_{\ell 2} & \cdots & I_{\ell\ell} \end{bmatrix} \tag{4.2}$$

and I_{ij}, for $i,j = 1,2,\ldots,\ell$, is represented by a series of the form

$$I_{ij} = C_{M_{ij}} 2^M + C_{M-1_{ij}} 2^{M-1} + \ldots + C_{1_{ij}} 2^1 + C_{0_{ij}} 2^0 \tag{4.3}$$

where

$$C_{k_{ij}} \in \{ -1, 0, 1 \} \tag{4.4}$$

for $k=0,1,2,\ldots,M$. From the above follows the expression

$$S_I = C_M 2^M + C_{M-1} 2^{M-1} + \ldots + C_1 2^1 + C_0 2^0 \tag{4.5}$$

where

$$C_k = \begin{bmatrix} C_{k_{11}} & C_{k_{12}} & \cdots & C_{k_{1\ell}} \\ C_{k_{21}} & C_{k_{22}} & \cdots & C_{k_{2\ell}} \\ \vdots & \vdots & & \vdots \\ C_{k_{\ell 1}} & C_{k_{\ell 2}} & \cdots & C_{k_{\ell\ell}} \end{bmatrix} \qquad (4.6)$$

for k=0,1,2,...,M, which gives S_I as a series in powers of two with matrix coefficients C_k, k=0,1,2,...,M. Note that the binary series expansion specified in (4.3) and (4.4) for the integers I_{ij} allows only the coefficients -1, 0, and 1, so likewise the coefficient matrices C_k, k=0,1,2,...,M, contain only -1, 0, and 1 as elements. For each integer I_{ij} we choose a canonical series expansion from those possible expansions in (4.3), where the term canonical means the number of non-zero coefficients among $C_{k_{ij}}$, k=0,1,2,...,M, is a minimum. The lexicographic notation corresponding to a canonical series expansion is known as the canonical signed digit code. Since the standard binary representation corresponds to one possible expansion of an integer I_{ij} in (4.3) the number of non-zero digits in the canonical signed digit code is equal to or less than the number of non-zero digits in the standard representation. The canonical signed digit code is chosen in (4.3) because it minimizes the number of non-zero entries in the coefficient matrices C_k, k = 0, 1,2,...,M, of (4.5) and hence reduces the number of arithmetic operations required in the realization. The derivation of the canonical signed digit code is presented in section 4.5.

With the power series expansion of S_I in (4.5) each component of b(n) can be expressed as a summation of weighted integers which can be computed as a sequence of partial sums with floating radix point.

From (4.1) and (3.18) we have

$$b(n) = (1/2^M)\ S_I\ a(n) \tag{4.7}$$

which gives

$$b(n) = (1/2^M)\ (\ C_M 2^M + C_{M-1} 2^{M-1} + \ldots + C_1 2^1 + C_0 2^0\ )\ a(n) \tag{4.8}$$

by substituting S_I from (4.5). Considering the i^{th} component of b(n) for any i = 1,2,...,ℓ, we have from (4.8)

$$2^M b_i = 2^M d_{iM} + 2^{M-1} d_{i,M-1} + \ldots + 2^1 d_{i1} + 2^0 d_{i0} \tag{4.9}$$

where d_{ik} for k=0,1,2,...,M, is the dot product of the i^{th} row vector of C_k in (4.6) with a, i.e.

$$\begin{aligned} d_{ik} &= (\ C_{k_{i1}}\ \ C_{k_{i2}}\ \ \ldots\ \ C_{k_{i\ell}}\ ) \cdot a \\ &= C_{k_{i1}} a_1 + C_{k_{i2}} a_2 + \ldots + C_{k_{i\ell}} a_\ell \quad . \end{aligned} \tag{4.10}$$

The dot products d_{ik}, k=0,1,2,...,M, are linear combinations of the components of a with the coefficients restricted to -1, 0, and 1, so it is apparent that $2^M b_i$ is an integer. To compute $2^M b_i$ in (4.9) we define the partial sum S_{ik}, for k = 1,2,...,M, as the sum of the first k+1 terms in (4.9) in the order of ascending powers, i.e.

$$\begin{aligned} S_{i0} &= d_{i0} \\ S_{i1} &= S_{i0} + 2^1 d_{i1} \\ S_{i2} &= S_{i1} + 2^2 d_{i2} \\ &\vdots \\ S_{iM} &= S_{i,M-1} + 2^M d_{iM} \end{aligned} \tag{4.11}$$

Note $S_{iM} = 2^M b_i$ is the desired result. In order to operate with the

scaled integers $2^M d_{ik}$, k = 1,2,...,M, we scale each partial sum S_{ik} in (4.11) by 2^{M-k}, i.e. we define the floating point partial sums

$$\begin{aligned} S'_{i0} &= 2^M S_{i0} \\ S'_{i1} &= 2^{M-1} S_{i1} \\ S'_{i2} &= 2^{M-2} S_{i2} \\ &\vdots \\ S'_{i,M-1} &= 2^1 S_{i,M-1} \\ S'_{iM} &= 2^0 S_{iM} \end{aligned} \qquad (4.12)$$

From (4.11) and (4.12) follow the recursive relations

$$\begin{aligned} S'_{i0} &= 2^M d_{i0} \\ S'_{i1} &= 2^M d_{i1} + 2^{-1} S'_{i0} \\ S'_{i2} &= 2^M d_{i2} + 2^{-1} S'_{i1} \\ S'_{i3} &= 2^M d_{i3} + 2^{-1} S'_{i2} \\ &\vdots \\ S'_{iM} &= 2^M d_{iM} + 2^{-1} S'_{i,M-1} \end{aligned} \qquad (4.13)$$

which give the operational structure required in section 4.8 to compute the result $S'_{iM} = 2^M b_i$ in two's complement form.

4.5 Canonical Signed Digit Code for an Integer

The canonical signed digit code (CSDC) for an integer I is the lexicographic notation $c_M c_{M-1} \cdots c_2 c_1 c_0$ for the expansion

$$I = c_M 2^M + c_{M-1} 2^{M-1} + \cdots + c_1 2^1 + c_0 2^0 \qquad (4.14)$$

with the following requirements on the coefficients c_k :

a) $c_k \in \{ -1, 0, 1 \}$, k = 0,1,2,...,M ; (4.15)

b) if

$$T_M = \sum_{k=0}^{M} |c_k| \qquad (4.16)$$

then T_M (i.e. the number of non-zero coefficients) is a minimum among all sets of M+1 coefficients satisfying (4.14) and (4.15);

c) $c_k c_{k-1} = 0 \ , \quad k = 1,2,\ldots,M, \qquad (4.17)$

i.e. two non-zero digits may not be adjacent in the CSDC.

It has been shown that a set of coefficients satisfying (4.14) with a) and b) is not unique for every integer I. On the other hand,a unique set of coefficients exists satisfying (4.14) with a) and c) for every integer I and this unique set will also have the minimum property b), as we will show following Reitwiesner [26]. For these reasons it is conventional to define the unique CSDC by (4.14) with a) and c), whereupon follows the minimal representation property in b). A method of determining the CSDC for any integer is given below as part of theorem 4.0.

Theorem 4.0 :

A. There exists a set of coefficients c_k, k = 0,1,2,...,M, satisfying (4.14), (4.15), and (4.17) for any integer I.

B. The set of coefficients in A is unique.

C. The set of coefficients in A has the minimal representation property defined in b) above.

D. The canonical signed digit code for a positive integer is determined in the following way:

1) Partition the standard binary form into naturally occurring strings of ones and zeroes which must alternate.

2) Categorize each string as

a) type 0 - a string of a single zero,

b) type 0..0 - a string of two or more zeroes,

c) type 1 - a string of a single one, or

d) type 1..1 - a string of two or more ones.

3) Begin step 4) at the least significant digit.

4) Proceed from right to left until a type 1..1 string is encountered. If a 1..1 string is not found the integer is now in the canonical signed digit code. If a 1..1 string is found replace it with the equivalent signed digit string 10..-1. The leading digit in the 10..-1 string will occupy the position of the least significant digit in the adjacent 0 or 0..0 string to the left of the replaced string. If the adjacent string is type 0 the leading digit will be appended to the 1 or 1..1 string to the left of the type 0 string. The enlarged 1 or 1..1 string should be categorized as a 1..1 string before proceeding.

5) From the present position repeat step 4).

For a negative integer determine the CSDC of the magnitude of the integer in standard binary form and invert the sign of each bit in the result.

Proof:

A. Existence

Existence is shown constructively by applying the reduction procedure given below to the standard binary form for a positive integer and interpreting the resulting finite string of signed digits as the lexicographic notation $c_M c_{M-1} \cdots c_2 c_1 c_0$ for the expansion (4.14). The reduction procedure may be applied to any signed digit code, so a negative integer may be treated as a positive magnitude integer with the sign of each bit inverted.

The coefficients c_K, k = 1,2,...,M, obtained from the reduction procedure will satisfy (4.15) since the only digit changes which occur are the replacement of strings with other strings containing the digits -1, 0, and 1, where the digits of the given code are already restricted to -1, 0, and 1. The coefficients resulting after each replacement step, including the final step, will also satisfy (4.14) since the essential operation in each step is the interchange of a ±(10..-1) or ±(1 -1) string with a ±(1..1) or ±(1) string, respectively, which leaves the value of the number unchanged. This follows from

$$\sum_{k=P}^{Q} 2^k = 2\sum_{k=P}^{Q} 2^k - \sum_{k=P}^{Q} 2^k = \sum_{k=P+1}^{Q+1} 2^k - \sum_{k=P}^{Q} 2^k = 2^{Q+1} - 2^P \quad (4.18)$$

where P and Q, $Q \geq P$, are integers. Finally, the coefficients will satisfy (4.17) because it will be seen that the reduction process eliminates adjacencies between non-zero digits progressively from right to left.

The reduction procedure consists of the following steps.

1) Partition the given signed digit code as the following strings:

   a) 0 - a single zero;

   b) 0..0 - two or more zeroes;

   c) 1 - a single one;

   d) 1..1 - two or more ones;

   e) -1 - a single negative one;

   f) -1..-1 -two or more negative ones;

   where no string in a) through f) may be a substring of a larger string of like digits in the code.

2) Proceed from right to left until the first pair of adjacent 1, or 1..1, and -1, or -1..-1, strings is encountered. Reduce this adjacent pair of strings as in table 4-1, which lists the possible

adjacencies and the desired reductions via string substitutions.

POSSIBLE ADJACENCY	REDUCTION
1 -1	0 1
1 -1..-1	0 0..1
1..1 -1	1..0 1
1..1 -1..-1	1..0 0..1
-1 1	0 -1
-1 1..1	0 0..-1
-1..-1 1	-1..0 -1
-1..-1 1..1	-1..0 0..-1

Table 4-1: Possible Adjacencies between Non-Zero Strings and Reductions

Continue from right to left and reduce each adjacency of the type in table 4-1 until they have all been removed. Note the remaining adjacent non-zero digits can now be found only in those 1..1 or -1..-1 strings which may exist.

3) Proceed again from right to left until the first 1..1 or -1..-1 string is encountered and replace it with a 10..-1 or -10..1 string, respectively . The leading digit of the replacement string will lie in the least significant digit of the adjacent 0 or 0..0 string to the left of the replaced string. If the adjacent string was a 0..0 string the leading digit will not be adjacent to a non-zero digit. If the adjacent string was a 0 string the leading digit will be adjacent to a string containing non-zero digits which are either like or unlike the leading digit. If the adjacency is with a string of like digits then a

new or enlarged string of type 1..1 or -1..-1 will be produced to the left of the replaced string. We will treat this new or enlarged string just as any other 1..1 or -1..-1 string. If the adjacency is with a string of unlike digits then either the first, third, fifth, or seventh possibility in table 4-1 occurs, which should be reduced according to the table before proceeding. At this point note adjacent non-zero digits can be found only in those 1..1 or -1..-1 strings which may exist.

4) If 1..1 or -1..-1 strings still exist to the left of the present position repeat step 3). If not, the reduction is complete and there are no adjacent non-zero digits in the result.

This completes the existence proof.

B. Uniqueness

We prove this by contradiction. Assume the existence of two distinct sets of coefficients, c_k, $k = 0,1,2,\ldots,M$, and c'_k, $k = 0,1,2,\ldots,M$, satisfying (4.14), (4.15), and (4.17). From (4.14) we have

$$I = \sum_{k=0}^{M} c_k 2^k = \sum_{k=0}^{M} c'_k 2^k \quad . \tag{4.19}$$

Since the sets of coefficients are distinct

$$c_k \neq c'_k \tag{4.20}$$

for some $k \leq M$. Let $L \leq M$ be the smallest k for which (4.20) is true. Then

$$\sum_{k=0}^{L} c_k 2^k - \sum_{k=0}^{L} c'_k 2^k = c_L 2^L - c'_L 2^L = (c_L - c'_L) 2^L$$

or transposing

$$\sum_{k=0}^{L} c_k 2^k = \sum_{k=0}^{L} c'_k 2^k + (c_L - c'_L)\, 2^L \quad . \tag{4.21}$$

If L=M subtracting (4.21) from (4.19) gives

$$0 = (c_M - c'_M) \, 2^M \tag{4.22}$$

which is an obvious contradiction. If L < M subtracting (4.21) from (4.19) gives

$$\sum_{k=L+1}^{M} c_k 2^k = \sum_{k=L+1}^{M} c'_k 2^k - (c_L - c'_L) \, 2^L \quad . \tag{4.23}$$

The possible combinations of values for c_L and c'_L in (4.23) are covered by two mutually exclusive cases:

$$\text{Case 1 - } c_L \neq 0 \neq c'_L \quad ; \tag{4.24}$$

$$\text{Case 2 - either } c_L = 0 \text{ or } c'_L = 0 \quad . \tag{4.25}$$

In each case, from (4.20)

$$c_L \neq c'_L \tag{4.26}$$

and, from (4.15),

$$\begin{aligned} c_L &\in \{ -1, 0, 1 \} \\ c'_L &\in \{ -1, 0, 1 \} \quad . \end{aligned} \tag{4.27}$$

In case 1 we have, from (4.24), (4.26), and (4.27),

$$(c_L - c'_L) \, 2^L = (\pm 2) \, 2^L = \pm 2^{L+1} \tag{4.28}$$

and applying (4.17) in case 1 gives

$$c_{L+1} = 0 = c'_{L+1} \quad . \tag{4.29}$$

Using (4.28) and (4.29) in (4.23) gives

$$\sum_{k=L+2}^{M} c_k 2^k - \sum_{k=L+2}^{M} c'_k 2^k = \pm 2^{L+1} \tag{4.30}$$

which is a contradiction since the summations in (4.30) can differ only by a positive or negative integer multiple of 2^{L+2}. In case 2 we have, from (4.25), (4.26), and (4.27),

$$(c_L - c'_L) \, 2^L = (\pm 1) \, 2^L = \pm 2^L \tag{4.31}$$

which, with (4.23), gives the contradiction

$$\sum_{k=L+1}^{M} c_k 2^k - \sum_{k=L+1}^{M} c'_k 2^k = \pm 2^L \qquad (4.32)$$

since the summations can differ only by a positive or negative integer multiple of 2^{L+1}. This completes the uniqueness proof.

C. Minimality

This proof is constructive in nature. Examine the reduction method in part A of the proof. At each step a string substitution is made which introduces a number of non-zero digits equal to or less than the number of non-zero digits removed. Hence the result cannot contain more non-zero digits than the signed digit string given for reduction, where the string given for reduction is any code for the integer I. Also by part B of the proof the result is unique, so this unique result must have the minimality property.

D. Method of Determining CSDC

This method is a special case of the method in part A of the proof, which has already been established.

This completes the proof.

4.6 Use of Two's Complement Form for Binary Arithmetic

Any realization for a matrix operator with arbitrary integer elements consists of binary adders, binary shifts, and sign inverters where a binary number appearing at any point in the realization during computation may have positive or negative sign since the inputs are arbitrary integers. Hence, a hardware implementation requires a suitable method of representing signed numbers and as a further consequence all

adders, shifts, and sign inverters must operate on signed numbers. Methods of representing signed numbers and the design of hardware adders, shifters, and inverters are closely related problems which have been considered at length in the computer literature [26,27,28]. Here we briefly outline the reasons for choosing the complement form of signed number representation.

A set of m+1 binary devices in a hardware implementation assumes 2^{m+1} distinct states which may be assigned to the representation of 2^{m+1} binary numbers. With no loss of generality we may assume the numbers are integers since all other cases result by shifting the radix point. For signed numbers the state assignment must allocate certain states for positive numbers and other states for negative numbers. Normally a representation of 2^m positive numbers and 2^m negative numbers is chosen, where the set of negative numbers includes the sign inverses of the positive numbers (except possibly zero). For computational purposes there are two practical methods of state assignment, the sign and magnitude representation and the complement representation. Sign and magnitude representation is the obvious method where the most significant bit (the $m+1^{th}$ bit) represents the sign (0 for positive and 1 for negative) and the remaining m bits represent the magnitude of the binary number. The complement representation, on the other hand, encodes positive numbers in sign and magnitude form but encodes negative numbers by adding to them 2^{m+1} (or $2^{m+1}-1$ in an alternate version) .

Hardware considerations dictate a preference for the complement representation of signed numbers. With the complement representation a hardware adder capable of adding two positive numbers is sufficient for adding any combination of positive and negative numbers.

However, with the sign and magnitude representation the addition of signed numbers requires an "adder/subtractor", i.e. a hardware device providing carry (adder) or borrow (subtractor) mode of operation, conditional on the sign combination of the two inputs, as well as sign control logic for the result. Hence the complement representation minimizes the number of gates required to realize the adders in a hardware implementation and reduces the time required to perform an addition by eliminating the carry/borrow switching inherent with the sign and magnitude representation. The sign inversion operation is more complex with the complement representation than with the sign and magnitude representation, which requires the logical inversion of only a single bit, but this drawback is minor relative to the advantages mentioned. Shifting operations have no practical difference in complexity between the two representations. The advantages of the complement representation also extend to the hardware implementation of array multipliers where it is desirable to recode the multiplier canonically with signed digits (known as Booth's algorithm). We will not pursue the topic of array multipliers since they are not required in this work.

The complement representation has two variations: two's complement form, which has become the widely accepted standard, and one's complement form, which is found only occasionally in practice. These two forms differ only in the complementing value which is added to negative numbers when encoding them. The complementing value with two's complement form is 2^{m+1} and with one's complement form it is $2^{m+1}-1$, where we have chosen the represented numbers as a set of m bit positive integers and their sign inverses. It is also common to scale the signed m bit integers to the range between -1 and +1 by dividing by

the factor 2^m (i.e. shifting the radix point m places to the left). The corresponding complement forms result when the complementing values are also divided by 2^m, in which case the original complementing values 2^{m+1} and $2^{m+1}-1$ become 2 and $2-(1/2^m)$, respectively, or in binary form $10_{\wedge}00....00$ and $01_{\wedge}11....11$. The names two's complement form and one's complement form are derived from these complementing values which result from the viewpoint of scaling the integers. (Note the term one's complement is meant to imply all one's complement.) Since the location of the radix point is arbitrary and does not affect the arithmetic properties of the complement representations, we will apply the name two's complement form or the name one's complement form to any form which can be obtained from the corresponding integer form by shifting the radix point.

The two's complement form is preferred to one's complement form because addition in one's complement form requires an " end around" carry output from the most significant bit to the carry input of the least significant bit, whereas addition in two's complement form simply discards the carry output. This end around carry must propagate through the adder to complete the addition, which significantly increases the time required for an addition. Also, one's complement form results in two possible codings for zero, while two's complement form has a single unambiguous representation for zero. The only possible advantage of one's complement over two's complement form is the simpler implementation of sign inversion with one logical inverter for the sign bit. Thus we will consider only the standard two's complement form for design purposes.

4.7 Theory of Two's Complement Arithmetic

This section presents definitions and theorems from two's complement arithmetic necessary for the n-port adaptor realization. The proofs for the theorems are given in appendix A. The numbers represented in two's complement form are assumed to be signed integers which may be taken as fixed point numbers or as the mantissas of floating point numbers. The latter interpretation is useful where it is necessary to rescale results to the allowed range.

Definition 4.1: (Number of Bits) m is the number of bits allowed for the positive integers to be represented.

Definition 4.2:(Domain of Integers) X is any integer in the domain $-2^m \leq X \leq 2^m-1$, i.e. it is one of the following:

a) the m bit positive integers $0,1,2, \ldots,2^m-2,2^m-1$;

b) the m bit negative integers $-1,-2,\ldots,-2^m+2,-2^m+1$;

c) the m+1 bit negative integer -2^m.

Note 0 is in the set of positive integers only. Note X is allowed the extreme negative value -2^m which is one greater in magnitude than the extreme positive value 2^m-1.

Definition 4.3: (Two's Complement Form) $T_{(m)}(X)$ denotes the two's complement representation for X which is defined by

$$T_{(m)}(X) = \begin{cases} X & \text{if } 0 \leq X < 2^m \\ 2^{m+1} + X & \text{if } -2^m \leq X < 0 \end{cases} .$$

Hence $2^{m+1}-1 \geq T_{(m)}(X) \geq 0$, i.e. $T_{(m)}(X)$ is an m+1 bit positive integer. Note $T_{(m)}(X)$ maps negative values of X in inverted order to the integers between 2^m and $2^{m+1}-1$ inclusive, maps positive values of X identically to themselves, and has a unique inverse. Figure 4.1 illustrates the

mapping $T_{(m)}(X)$. In cases where there is no ambiguity we will drop the subscript (m) and use T(X) for $T_{(m)}(X)$.

Definition 4.4: (Lexicographic Notation) The lexicographic notation $X_0X_1X_2\ldots.X_{m-1}X_m$ for T(X) represents the binary expansion $T(X)=\sum_{i=0}^{m} X_i 2^{m-i}$. Note the numbering convention for two's complement form begins with X_0 for the high order bit.

Theorem 4.1: (Sign Bit) A necessary and sufficient condition for $X \geq 0$ ($X < 0$) is $X_0 = 0$ ($X_0 = 1$).(Hence we call X_0 the sign indicating bit or sign bit.)

Theorem 4.2: (Signed Digit Code for X) $X = -X_0 2^m + \sum_{i=1}^{m} X_i 2^{m-i}$,i.e. in lexicographic notation $X = (-X_0)X_1X_2\ldots.X_m$.

Theorem 4.3: (Complementing) $T(X) + T(-X) = 2^{m+1}$.

Theorem 4.4: (Algorithm for Complementing) $T(-X) = \overline{X}_0\overline{X}_1\overline{X}_2\ldots.\overline{X}_m +$ 000....1, in lexicographic notation, where $\overline{X}_i$ denotes the logical complement of X_i.

Theorem 4.5: (Arithmetic Shift to the Right)

a) $T_{(m)}(2^{-k}X)$ is defined (i.e. $2^{-k}X$ is an integer) if and only if $X_{m-k+1}=X_{m-k+2}= \ldots =X_m=0$, where $T_{(m)}(X) = X_0X_1X_2\ldots X_{m-k+1}X_{m-k+2}\ldots X_m$ and $1 \leq k \leq m$.

b) $T_{(m)}(2^{-k}X) = 2^{-k}T_{(m)}(X) + X_0 \sum_{i=0}^{k-1} X_i 2^{m-i}$,or in lexicographic notation $T_{(m)}(2^{-k}X) = X_0^{(1)}X_0^{(2)}X_0^{(3)}\ldots X_0^{(k)}X_0X_1X_2\ldots X_{m-k}$ where $X_0^{(1)}=X_0^{(2)}=X_0^{(3)}= \ldots = X_0^{(k)}= X_0$, i.e. $X_0^{(1)}X_0^{(2)}X_0^{(3)}\ldots X_0^{(k)}$ represents k replications or "fill-ins" of the sign bit X_0.

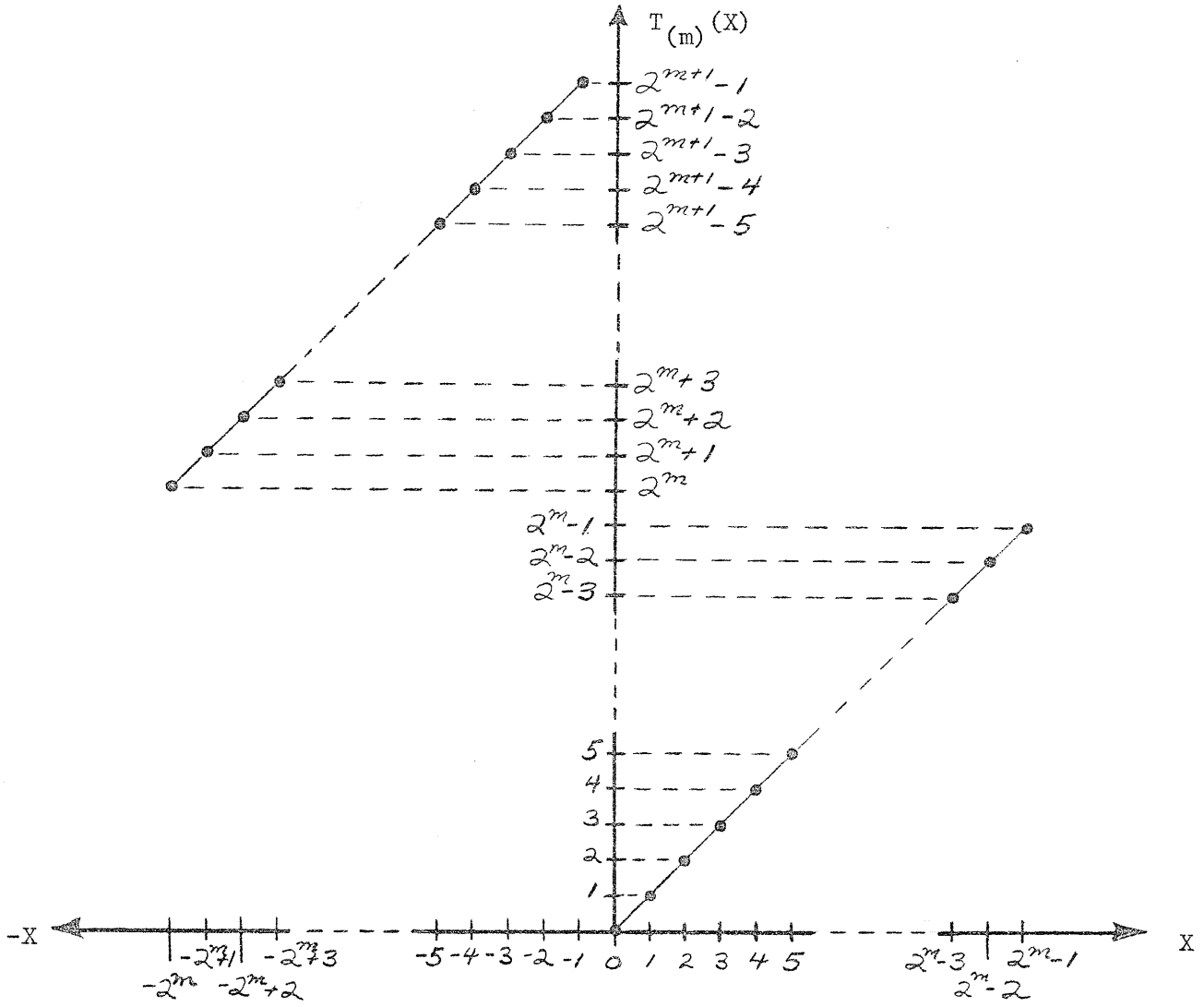


Figure 4.1 The Mapping $T_{(m)}(X)$ for Integers

Corollary to Theorem 4.5: (Arithmetic Shift to the Left)

a) $T_{(m)}(2^k X)$ is defined (i.e. $2^k X$ is in the domain of $T_{(m)}$) if and only if $X_0 = X_1 = \ldots = X_k$ where $T_{(m)}(X) = X_0 X_1 X_2 \ldots X_m$ and $m \geq k \geq 1$.

b) In case a) $T_{(m)}(2^k X) = X_k X_{k+1} \ldots X_m 0^{(1)} 0^{(2)} \ldots 0^{(k)}$ in lexicographic notation.

Theorem 4.6: (Scaling Up) If $T_{(m)}(X) = X_0 X_1 \ldots X_m$ then $T_{(m+M)}(2^M X) = X_0 X_1 \ldots X_m 0^{(1)} 0^{(2)} \ldots 0^{(M)}$, where $M \geq 1$.

Definition 4.5: (Notation for Sign Magnitude Truncation) $[U]\#$ will denote the sign magnitude truncation of a number U to the nearest integer, where U may consist of an integer part and a non-integer part. Formally

$$[U]\# = \begin{cases} +I & \text{if } U \geq 0 \\ -I & \text{if } U < 0 \end{cases}$$

where I is a positive integer such that

$$|I| \leq |U| \quad \text{and} \quad |I + 1| > |U| \quad .$$

Definition 4.6: (Notation for Modulo N Arithmetic) $[U]^*$ will denote the modulo N value of U (where U may be an expression). Unless otherwise specified $N = 2^{m+1}$.

Theorem 4.7: (Scaling Down with Sign Magnitude Truncation) If $T_{(m)}(X) = X_0 X_1 \ldots X_m$ then $T_{(m-M)}([2^{-M} X]\#) = [\ [2^{-M} T_{(m)}(X)]\# + t\ ]^*$ where $t = 1$ if $X_0 = 1$ and $X_{m-M+1} X_{m-M+2} \ldots X_m \neq 0$, $t = 0$ otherwise, $m > M \geq 1$, and $[\ ]^*$ indicates modulo 2^{m+1-M}.

Theorem 4.8: (Addition) $T(X+Y) = [\ T(X) + T(Y)\ ]^*$ if $-2^m \leq X, Y, X+Y \leq 2^m - 1$.

Theorem 4.9: (Discard of Carry from Sign Bit)

$[\ T(X) + T(Y)\ ]^* = [\ T(X) + T(Y)\ ]\not{C}$ where $[\ ]\not{C}$ signifies the discard of the carry from the $m+1^{th}$ or sign bit during addition.

Definition 4.7: (Overflow) An overflow is the condition $X+Y > 2^m-1$ or $X+Y < -2^m$ where $-2^m \leq X, Y \leq 2^m-1$. Note $T(X+Y)$ is undefined in this case.

Theorem 4.10: (Detection of Overflows)

a) An overflow may exist only if X and Y have like signs.

b) An overflow exists if and only if $X_0 = Y_0$ and $T_0 \neq X_0$ where T_0 is the sign bit of $T = [\ T(X) + T(Y)\ ]^*$.

Definition 4.8: (Carries) C represents the carry out of the sign bit. V represents the carry into the sign bit. In each case 1 represents a carry and 0 represents the absence of a carry.

Theorem 4.11: (Alternate Detection of Overflows) An overflow exists if and only if $C \oplus V = 1$ where $\oplus$ means "exclusive or".

Theorem 4.12: (Eliminating Overflow by Pre-Scaling) If an overflow exists then prescaling X and Y to X/2 and Y/2 eliminates the overflow.

Theorem 4.13: (Obtaining Pre-Scaled Result from Overflow) If an overflow exists for $Z = X + Y$ and X/2 and Y/2 are integers then $T(Z/2) = \overline{T}_0 T_0 T_1 T_2 .. T_{m-1}$ where $T = [\ T(X) + T(Y)\ ]^*$.

Definition 4.9: (Positive Overflow) A positive overflow is an overflow condition for which $X > 0$ and $Y > 0$.

Definition 4.10: (Negative Overflow) A negative overflow is an overflow condition for which $X < 0$ and $Y < 0$.

Theorem 4.14: (Result of a Positive or Negative Overflow)

If a positive (negative) overflow exists then $Z = X + Y \underset{(+)}{-} 2^{m+1}$ where $T = [\, T(X) + T(Y) \,]^*$ and $Z = T^{-1}(T)$, i.e. Z is obtained from the unique inverse $T^{-1}(\ )$ for $T(\ )$.

Theorem 4.15: (Multiple Overflows)

If

a) $S = X_1 + X_2 + \ldots + X_n$,

b) $T(X_k)$ exists for $k = 1,2,\ldots,n$,

c) $T_k = [\, T_{k-1} + T(X_k) \,]^*$ for $k = 2,3,\ldots,n$, where $T_1 = T(X_1)$, i.e. we perform a sequence of n-1 additions in two's complement form attempting to determine T(S),

d) N and P are the numbers of negative and positive overflows, respectively, in the sequence of c),

then $T^{-1}(T_n) = S + (N-P)2^{m+1}$.

Corollary 1 to Theorem 4.15: (Zero Net Overflows) Under the conditions of the theorem the result T_n equals T(S) if and only if the number of positive overflows equals the number of negative overflows.

Corollary 2 to Theorem 4.15: (Arithmetic Sum in Range) Under the conditions of the theorem if T(S) exists, i.e. if $-2^m \leq S \leq 2^m-1$, then $T^{-1}(T_n) = S$.

4.8 Realization of the Binary Decomposition with Two's Complement Arithmetic

The following considerations apply to the realization for the binary decomposition of S_I with two's complement arithmetic. We assume the "inputs" are $T_{(m)}(a_i)$, $i = 1,2,\ldots,\ell$, which, by definition 4.3, implies m+1 bits are alloted for each input $T_{(m)}(a_i)$ and a_i is an m bit integer for $i = 1,2,\ldots,\ell$. The integer $2^M b_i$, for $i = 1,2,\ldots,\ell$, given by (4.9) with (4.10) must be restricted to m+M bits since we require that the integer part of b_i is m bits for feedback to a_i, as implied by (3.54) or (3.55), with a fixed point representation. Hence the desired "outputs" are $T_{(m+M)}(2^M b_i)$, $i = 1,2,\ldots,\ell$, from which we obtain

$$T_{(m)}([b_i]\#) = [\ [2^{-M} T_{(m+M)}(2^M b_i)]\# + t\ ]* \qquad (4.33)$$

for $i = 1,2,\ldots,\ell$ by application of theorem 4.7, i.e. if

$$T_{(m+M)}(2^M b_i) = T_0 T_1 T_2 \ldots T_m T_{m+1} \ldots T_{m+M} \quad , \qquad (4.34)$$

as in definition 4.4, then

$$T_{(m)}([b_i]\#) = [\ T_0 T_1 T_2 \ldots T_m + t\ ]* \qquad (4.35)$$

where

$$t = T_0 \bullet \overline{(\ \overline{T}_{m+1} \bullet \overline{T}_{m+2} \bullet \ldots \bullet \overline{T}_{m+M}\ )} \quad . \qquad (4.36)$$

Referring to definition 4.6 and theorem 4.9 any required modulo m+1 additions, as in (4.35), result from discarding the carry out with an m+1 bit adder. $T_{(m)}(\Sigma_{ii}[b_i]\#)$ is fedback through a delay to $T_{(m)}(a_i)$ to satisfy (3.54) or (3.55), i.e. we have

$$T_{(m)}(a_i(n+1)) = T_{(m)}(\Sigma_{ii}[b_i(n)]\#) \qquad (4.37)$$

where

$$T_{(m)}(\Sigma_{ii}[b_i(n)]\#) = \begin{cases} T_{(m)}([b_i(n)]\#) & \text{if } \Sigma_{ii} = 1 \\ \text{INV}\{T_{(m)}([b_i(n)]\#)\} & \text{if } \Sigma_{ii} = -1 \end{cases} .$$

Here INV{ } represents the operation of complementing the enclosed two's complement form, as in theorem 4.4.

The realization for S_I with two's complement arithmetic is derived from (4.10) and (4.13). The application of addition theorem 4.8 to (4.10) gives

$$T_{(m)}(d_{ik}) = [\, T_{(m)}(C_{k_{i1}} a_1) + T_{(m)}(C_{k_{i2}} a_2) + \ldots + T_{(m)}(C_{k_{i\ell}} a_\ell)\,]^* \qquad (4.38)$$

for $i = 1,2,\ldots,\ell$ and $k=0,1,2,\ldots,M$ where d_{ik} is restricted to an m bit integer as in corollary 2 to theorem 4.15. The terms $T_{(m)}(C_{k_{ij}} a_j)$, $j = 1,2,\ldots,\ell$, in (4.38) are given by

$$T_{(m)}(C_{k_{ij}} a_j) = \begin{cases} T_{(m)}(a_j) & \text{if } C_{k_{ij}} = 1 \\ 0 & \text{if } C_{k_{ij}} = 0 \\ \text{INV}\{T_{(m)}(a_j)\} & \text{if } C_{k_{ij}} = -1 \end{cases} . \qquad (4.39)$$

The terms d_{ik} in (4.10), which are simple linear combinations of $a_1, a_2, \ldots, a_\ell$, may have certain sums in common which can be utilized to reduce the total number of adders and inverters in forming $T_{(m)}(d_{ik})$ for $i = 1,2,\ldots,\ell$ and $k=0,1,2,\ldots,M$. These common sums are best seen by examining the row vectors of the matrices C_k, $k=0,1,2,\ldots,M$, in (4.6), where favorable results have been obtained in specific examples. Unfortunately, no general method has been found for minimizing the number of adders and inverters here. The terms $T_{(m)}(d_{ik})$, $i = 1,2,\ldots,\ell$ and $k=0,1,2,..,M$, are computed in the input section of the realization and made available for further use.

We now introduce two's complement forms for the computations in (4.13). Since d_{ik} is an m bit integer for i = 1,2,...,ℓ and k= 0,1,2, ...,M, $2^M d_{ik}$ is an m+M bit integer and it is also seen from (4.11) and (4.12) that S'_{ik} is an integer for k = 0,1,2,...,M. The general term in (4.13) is

$$S'_{ik} = 2^M d_{ik} + 2^{-1} S'_{i,k-1} \qquad (4.40)$$

for k = 0,1,2,...,M, where $S'_{i,-1} = 0$, so if $S'_{i,k-1}$ is at most an m+M+1 bit integer then S'_{ik} is at most an m+M+1 bit integer. Since S'_{i0} is an m+M bit integer, by induction S'_{ik} is at most an m+M+1 bit integer for k = 1,2,...,M. Hence $2^{-1}S'_{ik}$, or $2^{-1}S'_{i,k-1}$, is at most an m+M bit integer and following (4.40) we may set

$$T_{ik} = [\, T_{(m+M)}(2^M d_{ik}) + T_{(m+M)}(2^{-1} S'_{i,k-1}) \,]^* \qquad (4.41)$$

for k = 1,2,...,M where [ ]* indicates modulo m+M+1 addition as in definition 4.6 and theorem 4.9. From addition theorem 4.8 we see $T_{ik} = T_{(m+M)}(S'_{ik})$ if no overflow condition exists. Except for k = M we require only $T_{(m+M)}(2^{-1}S'_{ik})$ for use in the next calculation (i.e. for k+1), so if an overflow exists we apply scaling theorem 4.13 and if no overflow exists we apply shifting theorem 4.5 to obtain $T_{(m+M)}(2^{-1}S'_{ik})$ from T_{ik} in either case. That is, if

$$T_{ik} = T_0^{(k)} T_1^{(k)} T_2^{(k)} \cdots T_{m+M}^{(k)} \quad , \qquad (4.42)$$

where i is fixed for the moment, then

$$T_{(m+M)}(2^{-1}S'_{ik}) = T_0^{(k)} T_0^{(k)} T_1^{(k)} T_2^{(k)} \cdots T_{m+M-1}^{(k)} \qquad (4.43)$$

if no overflow occurs and

$$T_{(m+M)}(2^{-1}S'_{ik}) = \overline{T}_0^{(k)} T_0^{(k)} T_1^{(k)} T_2^{(k)} \cdots T_{m+M-1}^{(k)} \qquad (4.44)$$

if an overflow occurs. Combining (4.43) and (4.44) we have

$$T_{(m+M)}(2^{-1}S'_{ik}) = S_0^{(k)} T_0^{(k)} T_1^{(k)} T_2^{(k)} \ldots\ldots T_{m+M-1}^{(k)} \qquad (4.45)$$

with

$$S_0^{(k)} = (D_0^{(k)} \cdot \bar{S}_0^{(k-1)} + \bar{D}_0^{(k)} \cdot S_0^{(k-1)}) \cdot T_0^{(k)} + D_0^{(k)} \cdot S_0^{(k-1)} \qquad (4.46)$$

where

$$T_{(m+M)}(2^{-1}S'_{i,k-1}) = S_0^{(k-1)} S_1^{(k-1)} \ldots\ldots S_{m+M}^{(k-1)} \qquad (4.47)$$

and

$$T_{(m+M)}(2^{M}d_{ik}) = D_0^{(k)} D_1^{(k)} \ldots\ldots D_{m+M}^{(k)} \quad . \qquad (4.48)$$

If $T_{(m+M)}(2^{M}d_{ik}) = 0$, i.e. if $d_{ik} = 0$, then (4.46) reduces to

$$S_0^{(k)} = S_0^{(k-1)} \quad . \qquad (4.46a)$$

Expression (4.46) follows from theorem 4.10 on detection of overflows which gives the following table 4-2.

$D_0^{(k)}$	$S_0^{(k-1)}$	$T_0^{(k)}$	$S_0^{(k)}$	
0	0	0	0	
0	0	1	0	(overflow)
0	1	0	0	
0	1	1	1	
1	0	0	0	
1	0	1	1	
1	1	0	1	(overflow)
1	1	1	1	

Table 4-2: Correction of Sign Bit $S_0^{(k)}$

Note $S_0^{(k)} \neq T_0^{(k)}$ only in the second and seventh rows of the table which are overflow cases. In summary, the sequence of additions in (4.40) is performed in two's complement form by using (4.41) and (4.45) with (4.46). If d_{ik}, $k = 1,2,\ldots,P$, $P \leq M$, are the same for two or more indices i the sequence (4.40) for $k=0,1,2,\ldots,P$ can be used in common for those indices to reduce the total of adders and inverters in the realization. $T_{(m+M)}(2^M d_{ik})$ in (4.41) is obtained from $T_{(m)}(d_{ik})$ in (4.38) by a shift of digits, as given in theorem 4.6, which also implies

$$D_j = 0 , \quad j = m+1, m+2, \ldots, m+M , \tag{4.49}$$

in (4.48). Also note overflows are eliminated at all but the M^{th} sum , where we must assume an overflow cannot occur if $2^M b_i$ is to be an m+M bit integer as previously required.

Each sum in the sequence of (4.41) requires the actual addition of only the first m+1 bits in the addends $T_{(m+M)}(2^M d_{ik})$ and $T_{(m+M)}(2^{-1} S'_{i,k-1})$. That is, from (4.48) and (4.49)

$$T_{(m+M)}(2^M d_{ik}) = D_0^{(k)} D_1^{(k)} \cdots D_m^{(k)} 0_{(m+1)} 0_{(m+2)} \cdots 0_{(m+M)} \tag{4.50}$$

so with (4.47) in (4.41) we have

$$T_{ik} = T_0^{(k)} T_1^{(k)} \cdots T_m^{(k)} S_{m+1}^{(k-1)} S_{m+2}^{(k-1)} \cdots S_{m+M}^{(k-1)} \tag{4.51}$$

where

$$T_0^{(k)} T_1^{(k)} \cdots T_m^{(k)} = [\, D_0^{(k)} D_1^{(k)} \cdots D_m^{(k)} + S_0^{(k-1)} S_1^{(k-1)} \cdots S_m^{(k-1)} \,]^* \tag{4.52}$$

since both addends are positive numbers. We also see from (4.45) that T_{ik} in (4.51) is given by

$$T_{ik} = T_0^{(k)} T_1^{(k)} \cdots T_m^{(k)} T_m^{(k-1)} T_{m+1}^{(k-1)} \cdots T_{m+M-1}^{(k-1)} \tag{4.53}$$

or, by induction through k, k-1, k-2, ... , 1,

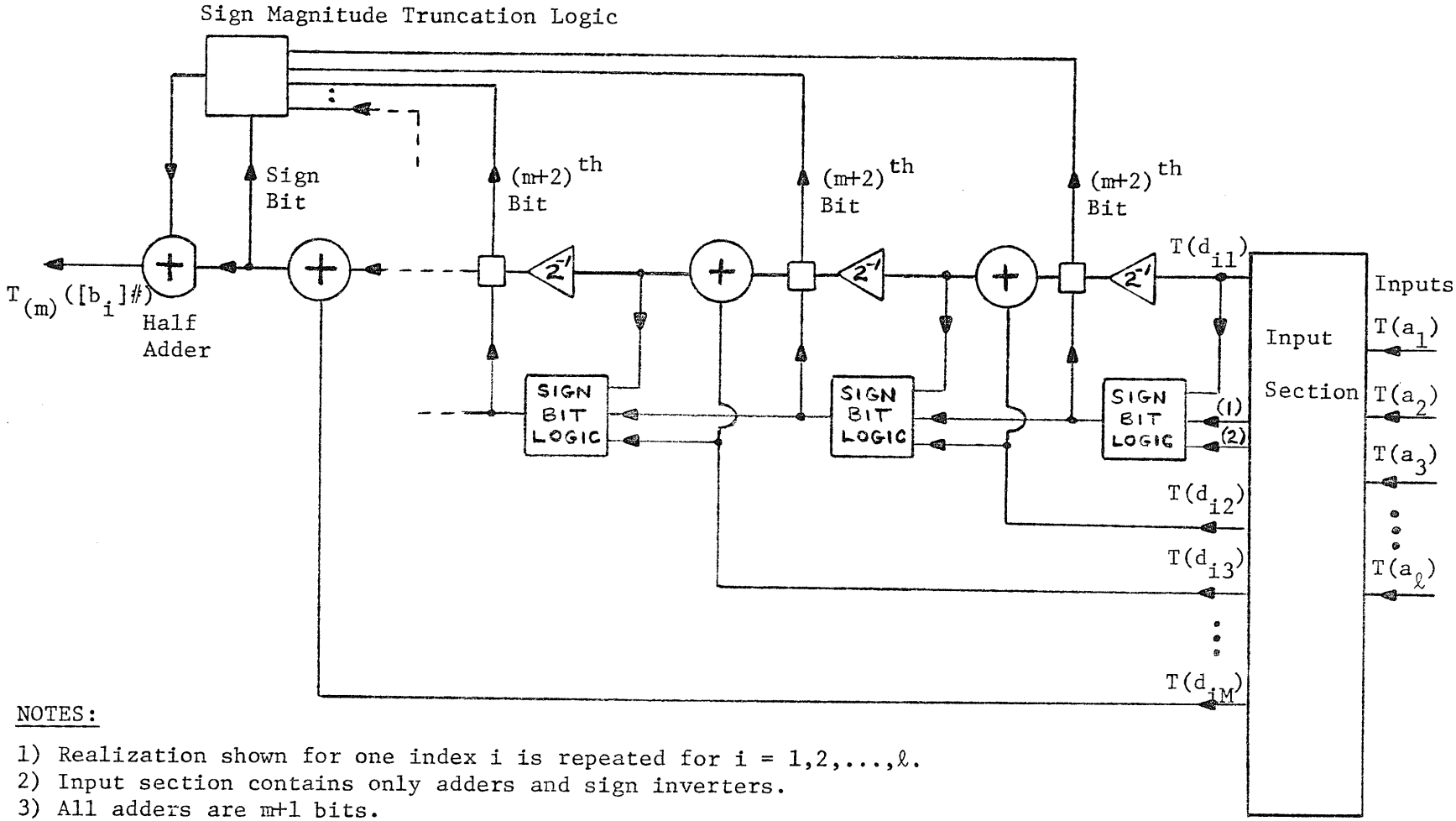
$$T_{ik} = T_0^{(k)} T_1^{(k)} \cdots T_m^{(k)} T_m^{(k-1)} T_m^{(k-2)} \cdots T_m^{(1)} T_m^{(0)} T_{m+1}^{(0)} \cdots T_{m+M-k}^{(0)} . \quad (4.54)$$

For the final sum T_{iM} we have from (4.54)

$$T_{iM} = T_0^{(M)} T_1^{(M)} \cdots T_m^{(M)} T_m^{(M-1)} T_m^{(M-2)} \cdots T_m^{(1)} T_m^{(0)} \quad (4.55)$$

which means $T_{m+1}^{(M)} T_{m+2}^{(M)} \cdots T_{m+M}^{(M)}$ is determined simply by feeding forward the unused m^{th} bit of each partial sum.

The general form of the block diagram realization incorporating the above development with two's complement arithmetic is shown in figure 4.2a, where figure 4.2b gives bit level detail of the adders with shift and sign bit logic. The blocks labelled sign bit logic and sign magnitude truncation logic represent realizations of (4.46) and (4.36), respectively. If an input $T(d_{ik})$ to an adder is identically zero for some k, i.e. if d_{ik} has all zero coefficients, then the adder is eliminated and the sign bit relation for the following shift reduces to (4.46a) which is a simple feedforward of the previous sign bit. Hence the resulting cascaded shifts may be combined. The realization produces full precision results (i.e. m+M+1 bits) for $T_{(m+M)}(2^M b_i)$, i = 1,2, ...,ℓ, before sign magnitude truncation for feedback while all adders have the uniform and minimal length of m+1 bits. Hence roundoff noise is introduced only from sign magnitude truncation at the delays during filter operation. Uncorrected overflows may occur only in the section computing $T_{(m)}(d_{ik})$ or in the final adder computing $T_{(m+M)}(2^M b_i)$ for i = 1,2,...,ℓ and k=0,1,2,...,M.



NOTES:

1) Realization shown for one index i is repeated for i = 1,2,...,ℓ.
2) Input section contains only adders and sign inverters.
3) All adders are m+1 bits.
4) Inputs (1) and (2) are input sign bits at the adder with output $T(d_{i1})$.

Figure 4.2a Block Diagram for Realization of S_I with Two's Complement Arithmetic

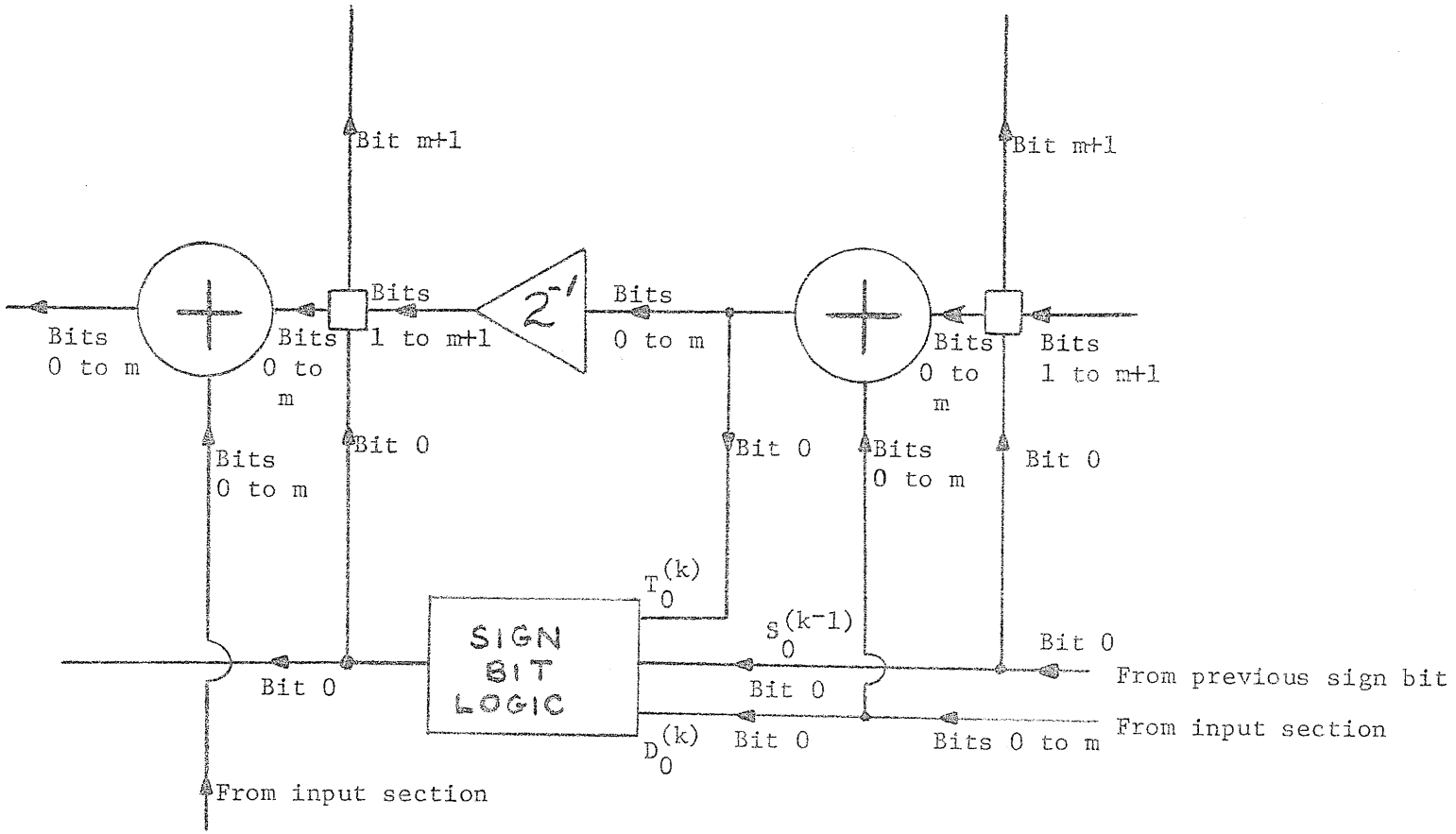


Figure 4.2b Bit Level Detail around Adders in Figure 4.2a

4.9 Hardware Implementation of the Two's Complement Realization

The realization of S_I in the block diagram of figure 4.2 is amenable to hardware implementation since the required elements are composed of functions available in standard medium scale integrated (MSI) logic circuits, such as transistor-transistor logic (TTL). Here we will consider a dedicated hardware implementation with parallel arithmetic operations for high speed, although serial arithmetic operations may be preferred for economy in some applications. This discussion will not include general design methods for logic circuits or gate level design of arithmetic circuits since many references are available [29]. One may consider the dedicated hardware approach as a hardwired or fixed program of computation in contrast to the stored program concept used in computers and microprocessors. The stored program concept economizes by using the same arithmetic elements repeatedly, but with a loss of operating speed. Hence it is worthwhile noting the realization of S_I in figure 4.2 also furnishes an efficient program for computer or microprocessor implementation of S_I. Since two's complement arithmetic has become standard in computers, an assembly language repertoire of two's complement operations is usually available for a direct computer simulation of the structure in figure 4.2 . Assembly language programming, which offers maximum operating speed with the computer, will be used with a PDP-11 minicomputer to simulate the dedicated hardware design for the example of section 4.10.

The adders required for two's complement arithmetic are standard binary adders, as seen in section 4.7. A single bit binary full adder is a combinational logic circuit (i.e. a network of gates) which provides the switching functions Σ_k (sum bit) and C_k (carry out)

for input bits X_k, Y_k, and C_{k-1} (carry in), as given in table 4-3.

C_{k-1}	X_k	Y_k	Σ_k	C_k
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

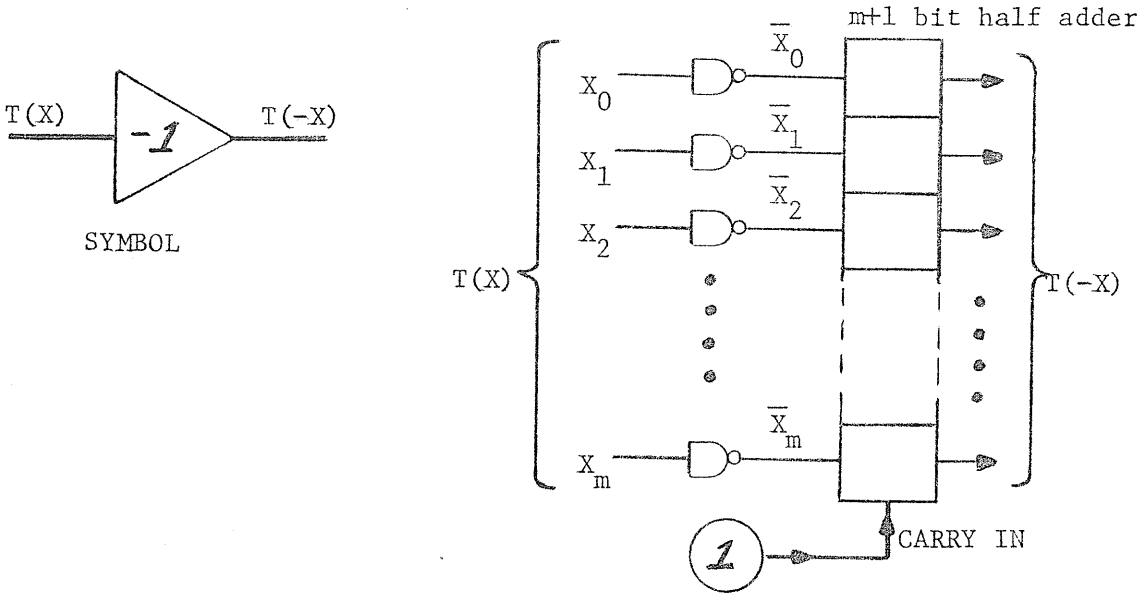
Table 4-3: Switching Function for a Single Bit Adder

Essentially a full adder provides three inputs (including carry) per bit. An m+1 bit full adder is a logic circuit equivalent to m+1 single bit full adders with carries C_k interconnected for k = 0,1,2,...,m. TTL full adders are commonly available in multiples of four bits, where the MSI logic circuit is not usually a simple replication of four single bit full adders since more efficient designs are known. The speed of a TTL adder, and hence of a network of adders, is determined by the gate propagation delays from the inputs to the carry and sum bits. Since carry delays accumulate, for increased speed a look ahead carry circuit is often included to directly determine the carry into the next four bit adder.

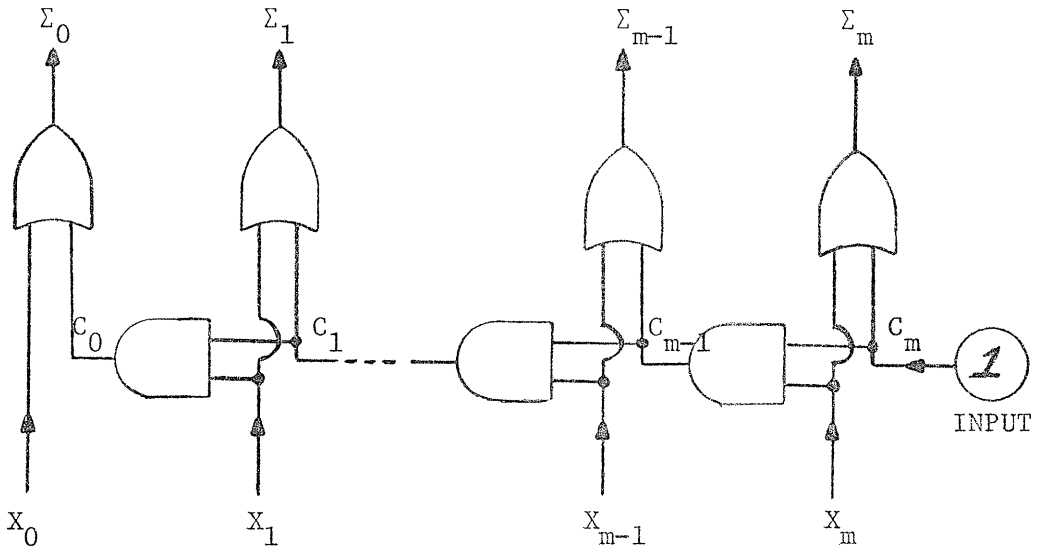
The sign inverters are implemented according to the algorithm for two's complementing in theorem 4.4. Hence a sign inverter for an m+1 bit two's complement form consists of a logical inverter for each bit

followed by the addition of one to the result, as shown in figure 4.3a. The required addition can be implemented with an m+1 bit half adder, i.e. an adder with only two inputs (including carry) per bit, as shown in figure 4.3b, which offers a large saving in gates relative to a full adder. The half adder consists of a train of AND gates which determine carries $C_m, C_{m-1}, \ldots, C_1, C_0$ into the sum bits $\Sigma_m, \Sigma_{m-1}, \ldots, \Sigma_1, \Sigma_0$, respectively, where exclusive OR gates give $\Sigma_k = X_k \oplus C_k$ for $k = m, m-1, \ldots, 1, 0$. An alternate approach to sign inversion incorporates the required addition with an existing adder following the sign inverter as shown in figure 4.3c. That is, since the carry input to the least significant bit of each m+1 bit adder is free, at this carry we add one to the result obtained by logically inverting each bit of an input to the adder. If the adder following a sign inverter requires sign inversion of both inputs then a reduction to a single sign inverter following the adder can be used, as in figure 4.3d. Continued application of this reduction usually leads to an adder requiring sign inversion in only one input, where the schemeof figure 4.3c applies. In this case the essential cost of a sign inverter is one logical inverter per bit. Note all sign inverters in the block diagram of figure 4.2a are in the input section, which computes $T_{(m)}(d_{ik})$ for $i = 1,2,\ldots,\ell$ and k 1,2,...,M, and the feedback paths. Also, if a shift exists at an adder input a sign inverter cannot be incorporated with that adder input unless a half adder segment is appended to the carry input to accomodate the excess bits created by the shift.

Any shift (2^{-1}) with overflow correction for the preceeding two's complement addition, as shown in figure 4.2 , is implemented simply as a hardwired shift of digits one place to the right combined with the

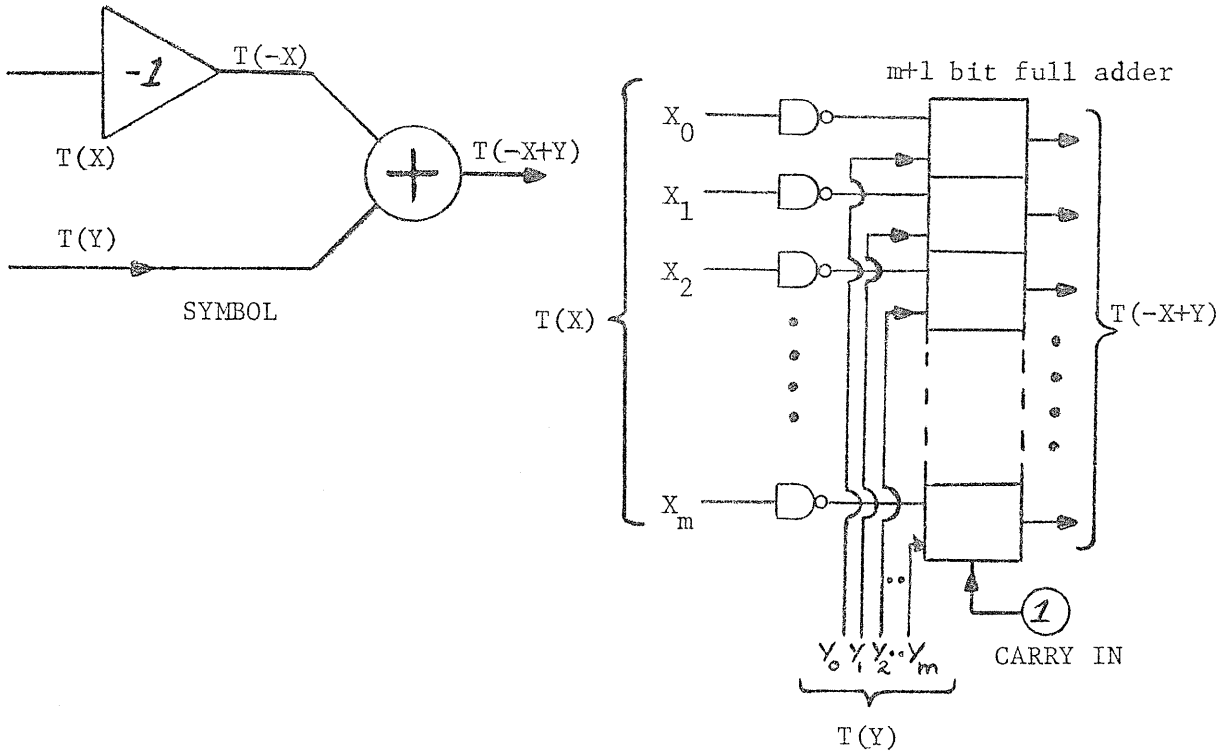


a) Implementation of Complementing Algorithm

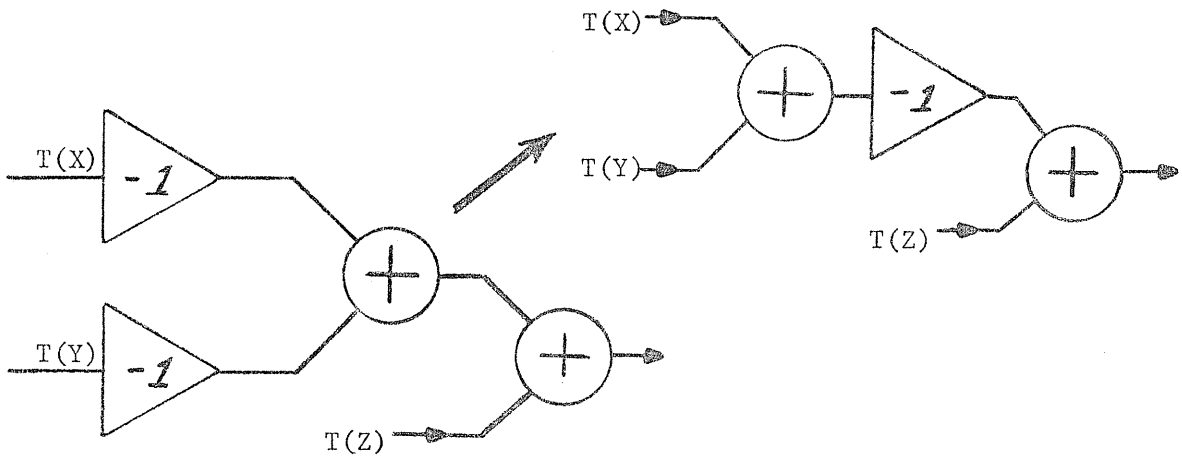


b) Implementation of Half Adder

Figure 4.3 Sign Inversion with Two's Complement Arithmetic



c) Combination of Sign Inverter with Existing Adder



d) Reduction of Sign Inverters for Application of b)

Figure 4.3(cont.) Sign Inversion with Two's Complement Arithmetic

gates required for sign bit logic. The rightmost digit in the shift (always a zero) is dropped. (In a serial program the hardwired shift would be simulated with a shift register.) As seen in table 4-2 the sign bit resulting in the preceding addition is replicated if no overflow occurred and reversed if overflow occurred. The sign bit logic is implemented according to relation (4.46), which gives the structure shown in figure 4.4a. If a sign inversion is combined with an adder, as in the preceding paragraph, and the adder precedes a shift then the additional logic in figure 4.4b is required for input to the sign bit logic which accompanies the shift. That is, if $T(d_{ik})$ is the proper input to the adder, where $T(-d_{ik}) = T_0'^{(k)} T_1'^{(k)} \ldots T_m'^{(k)}$ is actually available, then $T_0^{(k)} = \overline{T}_0'^{(k)} \oplus (\overline{T}_1'^{(k)} \cdot \overline{T}_2'^{(k)} \cdot \ldots \overline{T}_m'^{(k)})$, where $T_0^{(k)}$ is the sign bit of $T(d_{ik})$.

The sign magnitude truncation logic is implemented with logic gates according to relation (4.36), as shown in figure 4.5. Here again a half adder suffices for adding a one if required for magnitude correction. Each feedback delay is implemented with two data latches which transfer and hold the output data bits to the inputs during recursive operation of the filter.

Typical choices of TTL circuits for adders, logical inverters, gates, data latches, and multiplexers are listed in table 4-4 with their propagation delays. These choices are representative of economical types in common use, where the gates are available in standard or high speed versions. We will use this hardware in the design example of section 4.10.

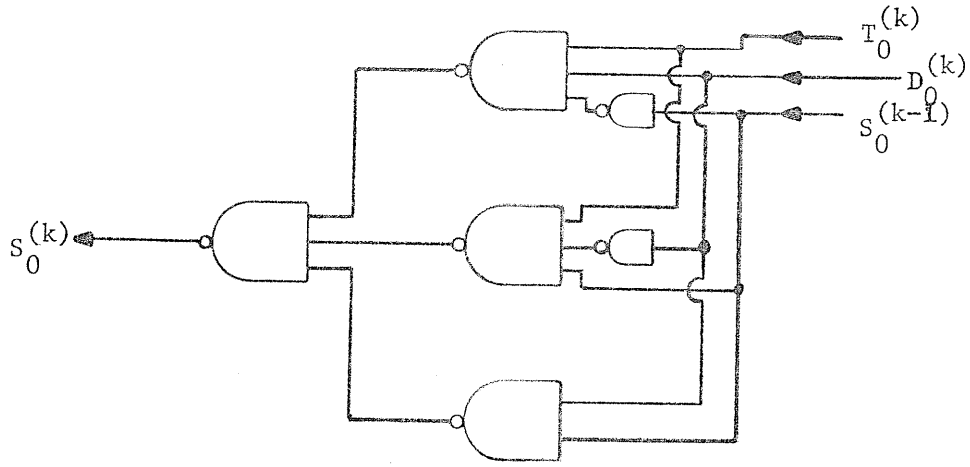
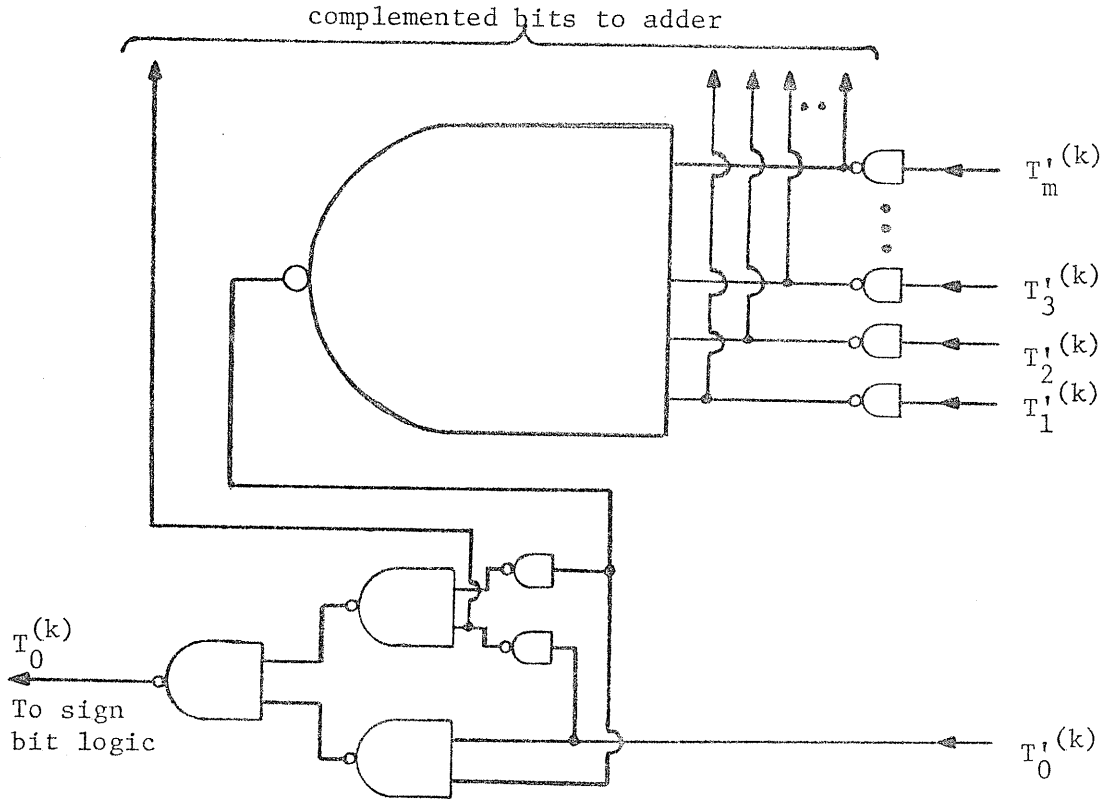


Figure 4.4a Sign Bit Logic



Note: $T(-d_{ik}) = T_0'^{(k)} T_1'^{(k)} \ldots T_m'^{(k)}$

Figure 4.4b Logic for Determining $T_0^{(k)}$ at Adder with Combined Sign Inversion

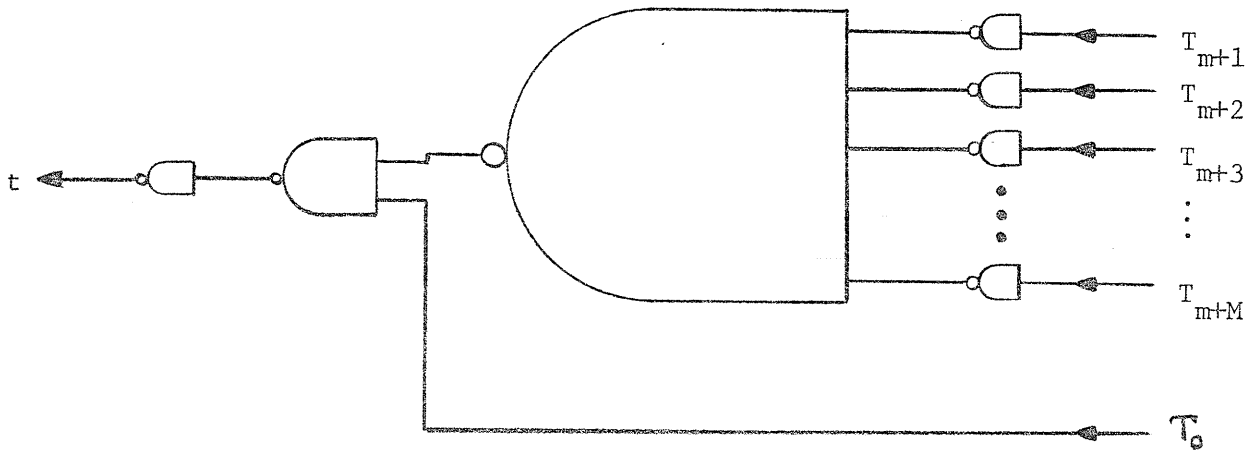


Figure 4.5 Sign Magnitude Truncation Logic

Function	TTL Type	Propagation Delay (max.)
6-input logical inverter	SN5404	22 ns
	SN54H04	13 ns
2-input NAND gate (quadruple)	SN5400	22 ns
	SN54H00	10 ns
3-input NAND gate (triple)	SN5410	22 ns
	SN54H10	10 ns
4-input NAND gate (dual)	SN5420	22 ns
	SN54H20	10 ns
8-input NAND gate	SN5430	22 ns
	SN54H30	10 ns
4 bit binary full adder with look ahead carry	SN5483	40 ns (bit sum) 34 ns (bit carry) 12 ns (look ahead carry)
4 bit data latch with clock input and complementary outputs	SN5475	40 ns
4 bit 2 input multiplexer	S74157	27 ns

Table 4-4: Representative TTL Circuits

The computation time required from inputs to outputs in the block diagram of figure 4.2 can be determined from the resulting structure which will resemble figure 4.6a. This significant factor limits the operating speed of the filter. As shown in [6], the required computation time is essentially the maximum of accumulated bit sum, bit carry, and logic propagation delays along any possible path from the least significant input bit to the most significant output bit. From figure 4.4 and table 4-4 the delay for $T_0^{(k)}$ through the sign bit logic is 44 ns, which can be reduced to 20 ns by using high speed (series H) gates (e.g. SN54H10). Since this reduced delay is less than a carry delay which is always found paralleling the sign bit logic we need not consider paths which pass through the sign bit logic. The worst case path would pass through an input section of maximum length followed by the entire struc-

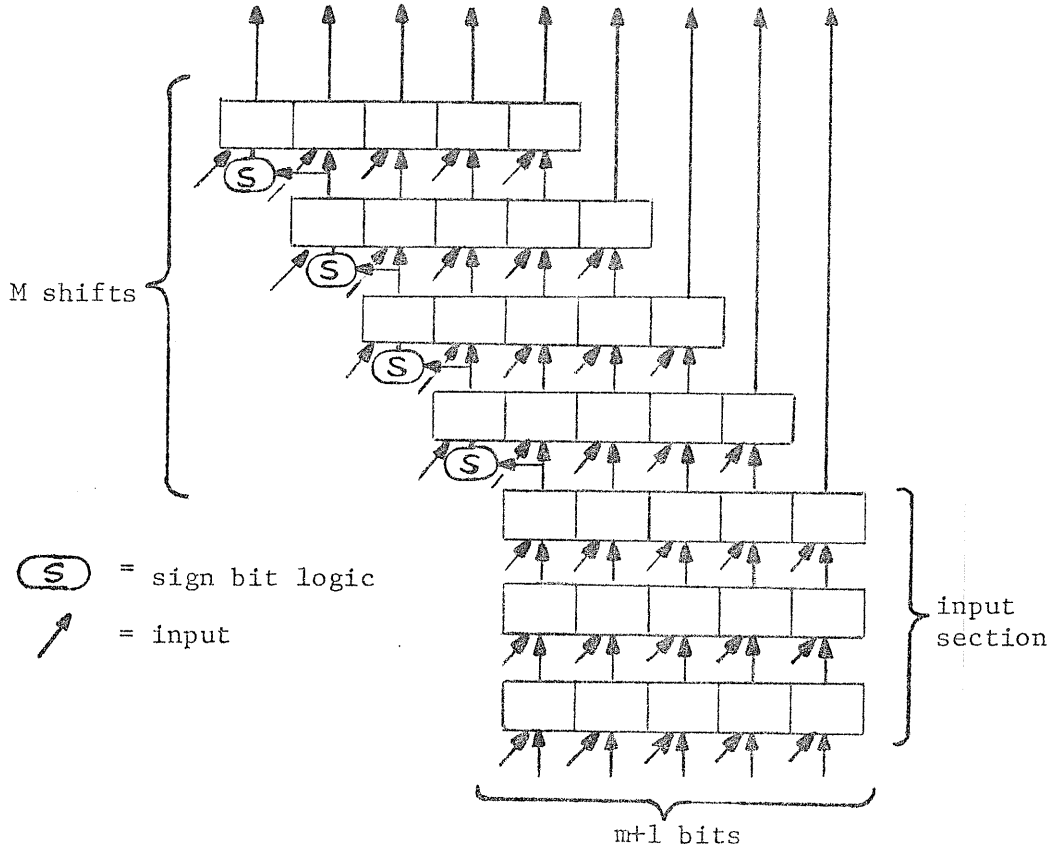


Figure 4.6a Bitwise Structure of Adders in Block Diagram of Fig. 4.2a

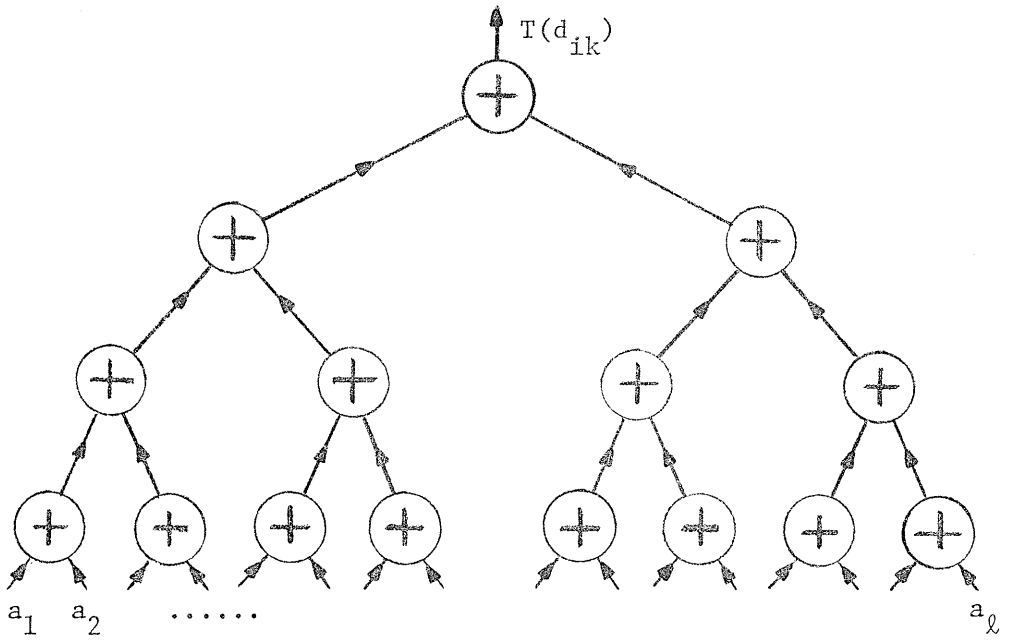


Figure 4.6b Pyramidal Adder Structure Useful for Input Section

ture of shifted adders. It is seen that such a path must traverse m+M+1 carry delays in total and M+1 sum delays in the structure of shifted adders including the half adder for sign magnitude truncation. If we assume a pyramidal adder structure for the input section which sums at most ℓ variables from a_i, $i = 1,2,\ldots,\ell$, as shown in figure 4.6b, then the worst case path encounters $[\log_2 \ell]\# + 1$ sum delays in the input section with a few delays in logical inverters for sign inversion. Hence the delay on the worst case path is roughly

$$t_p = ([\log_2 \ell]\# + M + 2) t_s + (m + M + 1) t_c \qquad (4.56)$$

where t_s and t_c are the sum and carry propagation delays, respectively. Unfortunately, the benefits of look ahead carry are lost as the number of cascaded adders becomes large. We have also assumed the addition of one in the half adder for sign magnitude truncation correction begins as soon as the lowest order bit becomes available at the output. This means the carry input to the half adder must be independent of the sign bit, which is the last bit to become available. If minimum computation time is desired, this can be achieved by permanently setting the carry input to one and using a multiplexer operated by the sign magnitude truncation logic to select either the corrected or the uncorrected output. The combination of sign magnitude truncation logic (with high speed gates) and multiplexer will require about 50 ns operating time. Finally the two data latches require 80 ns operating time.

4.10 Realization of the Network Example from Chapter 3

The scaled reduced scattering matrix, S_I, derived for the third order example in section 3.7 is now realized by the methods of this chapter. From $\bar{S}_R$ in section 3.7 we have

$$S_I = \begin{bmatrix} -8 & 15 & -7 & 25 \\ 16 & -2 & 18 & 18 \\ 24 & 15 & -7 & -7 \\ -8 & 15 & -7 & -7 \end{bmatrix} \qquad (4.57)$$

where the scale factor is

$$2^M = 32 \quad , \qquad (4.58a)$$

i.e.

$$M = 5 \quad . \qquad (4.58b)$$

As in section 4.4 we replace each element of S_I with its canonical signed digit code, which gives

$$S_I = \begin{bmatrix} \bar{1}000 & 1000\bar{1} & \bar{1}001 & 10\bar{1}001 \\ 10000 & \bar{1}0 & 10010 & 10010 \\ 10\bar{1}000 & 1000\bar{1} & \bar{1}001 & \bar{1}001 \\ \bar{1}000 & 1000\bar{1} & \bar{1}001 & \bar{1}001 \end{bmatrix} \qquad (4.59)$$

where $\bar{1}$ represents -1. From (4.59) follows

$$C_0 = \begin{bmatrix} 0 & -1 & 1 & 1 \\ 0 & 0 & 0 & 0 \\ 0 & -1 & 1 & 1 \\ 0 & -1 & 1 & 1 \end{bmatrix} \qquad C_3 = \begin{bmatrix} -1 & 0 & -1 & -1 \\ 0 & 0 & 0 & 0 \\ -1 & 0 & -1 & -1 \\ -1 & 0 & -1 & -1 \end{bmatrix}$$

$$C_1 = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & -1 & 1 & 1 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \qquad C_4 = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 1 & 0 & 1 & 1 \\ 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix} \qquad (4.60)$$

$$C_2 = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \qquad C_5 = \begin{bmatrix} 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

where it is seen the only non-zero linear combinations of a_i, i = 1, 2, 3, 4, are

a) $d_{10} = d_{30} = d_{40} = d_{21} = (\ 0 \quad -1 \quad 1 \quad 1\ ) \cdot a$,

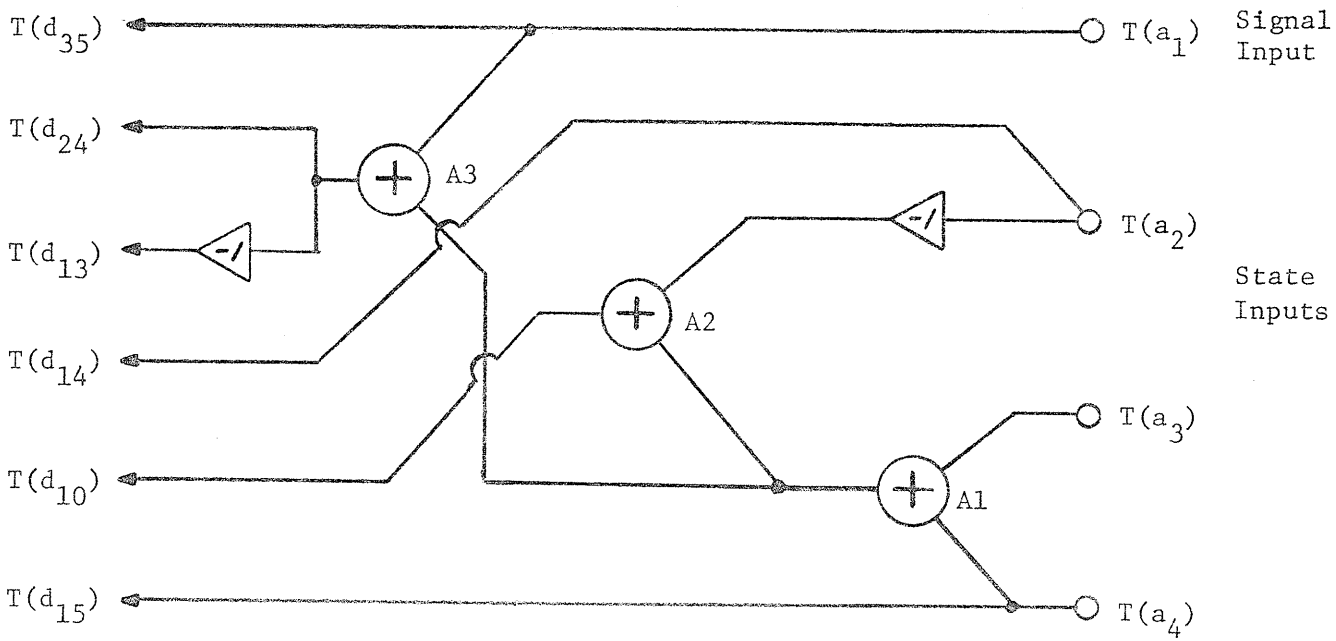
b) $d_{13} = d_{33} = d_{43} = -d_{24} = (\ -1 \quad 0 \quad -1 \quad -1\ ) \cdot a$,

c) $d_{14} = d_{34} = d_{44} = (\ 0 \quad 1 \quad 0 \quad 0\ ) \cdot a$, (4.61)

d) $d_{15} = (\ 0 \quad 0 \quad 0 \quad 1\ ) \cdot a$, and

e) $d_{35} = (\ 1 \quad 0 \quad 0 \quad 0\ ) \cdot a$.

The linear combinations in (4.61) are realized according to (4.38) and (4.39), as shown in figure 4.7a, where we have used the common sum $(\ 0 \quad 0 \quad 1 \quad 1\ ) \cdot a$ to reduce the number of adders. From (4.61a,b,c) we see $d_{1k} = d_{3k} = d_{4k}$ for k = 0,1,2,3,4, where $d_{11} = d_{12} = 0$, so the sequence of partial sums S'_{1k}, k = 0,1,2,3,4, as defined by (4.40), is



a) Realization for linear combinations in (4.61)

Figure 4.7 Realization of S_I for Example

identical to the sequences S'_{3k} and S'_{4k}, k = 0,1,2,3,4, and will be used in common. The remaining part of the realization, which also follows the block diagram in figure 4.2 , is shown in figure 4.7b.

The hardware requirements for the wave digital filter in this example are indicated by the block diagrams in figure 4.7. Eight m+1 bit adders are required, where the sign inverters preceeding A2 and A4 can be combined with these two adders. Hence the hardware required for each of these two inverters is one logical inverter per bit, where the extra sign bit logic of figure 4.4b is also required preceding A4. The only other sign inverter precedes the delay for feedback to $T(a_2)$. Since the delay elements will be clocked latches which furnish complementary outputs, this inverter requires only a half adder following the delay. Four half adders are required for sign magnitude truncation, with associated logic as in figure 4.5. Four blocks of sign bit logic are required, where each block follows figure 4.4a. Note sign bit correction for overflow has been extended to adder A2 which also precedes a shift, so an uncorrected overflow can occur only in adders A1, A3, A6, and A7. Each of the four delays for feedback consists of two m+1 bit latches which operate sequentially following a clock pulse. The first latch acquires the output and buffers the second latch against a changing output while its contents are transferred to the second latch. The filter is constructed with a multiple of four bits for m+1 since TTL arithmetic circuits are designed in four bit modules. Note the total hardware requirement for this example compares favorably with the requirement via the adaptor approach given in chapter 3.

The computation time required by the realization in figure 4.7 based on m+1 = 8 is somewhat better than the conservative estimate of

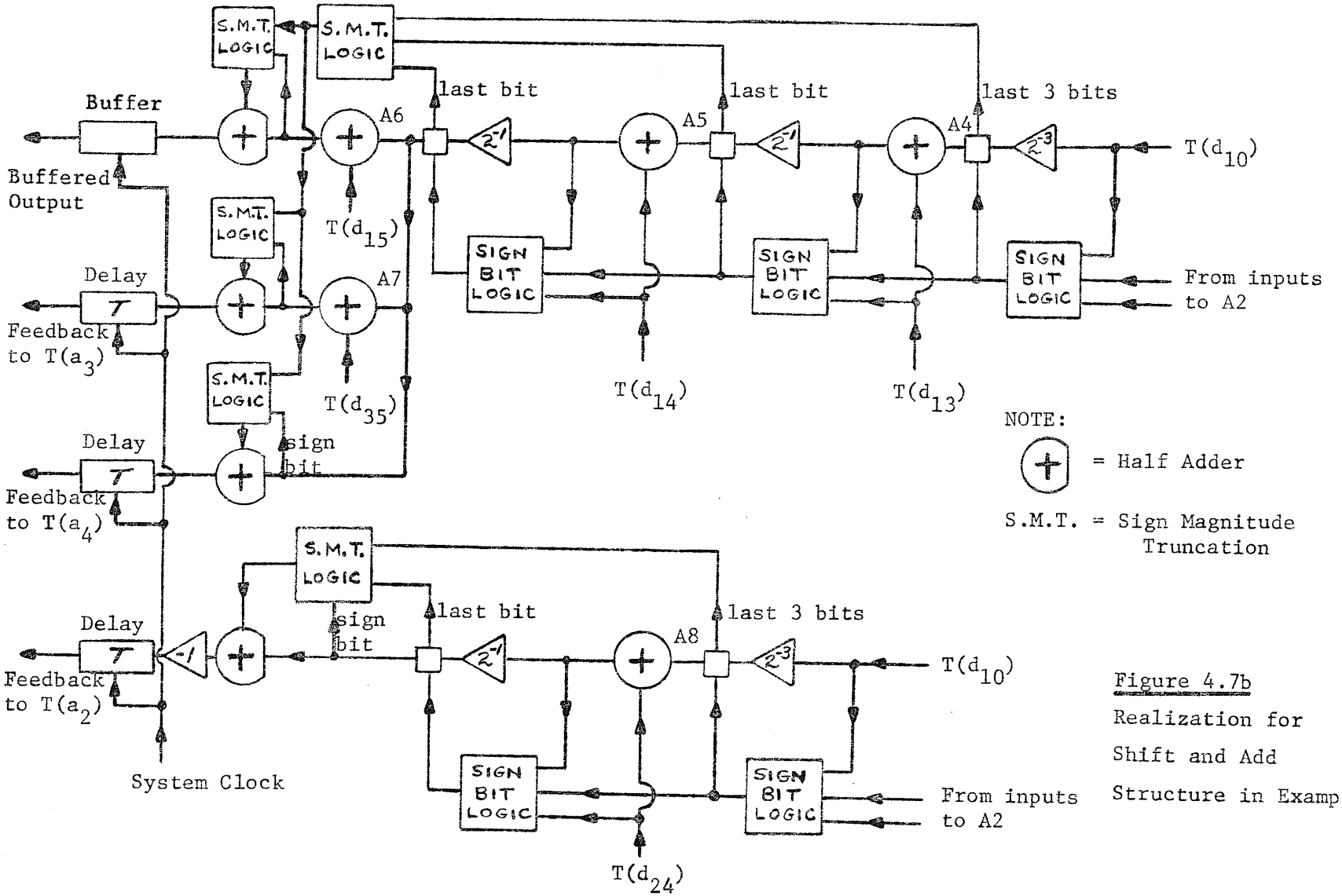
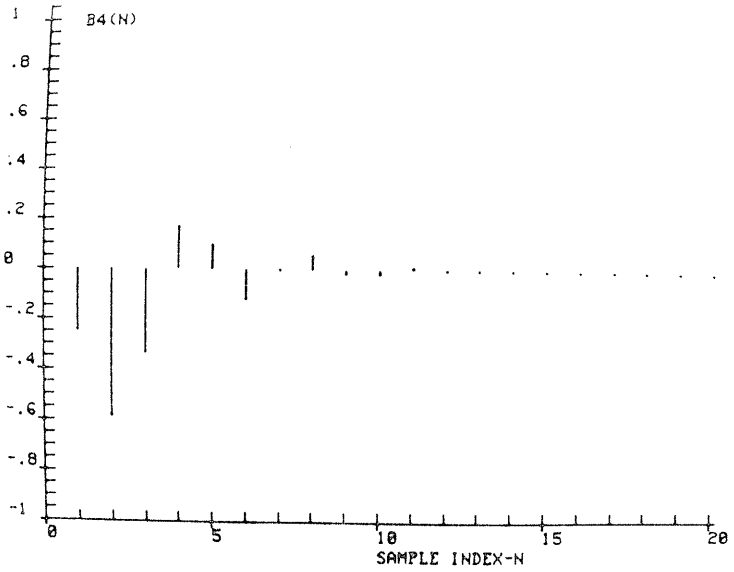


Figure 4.7b

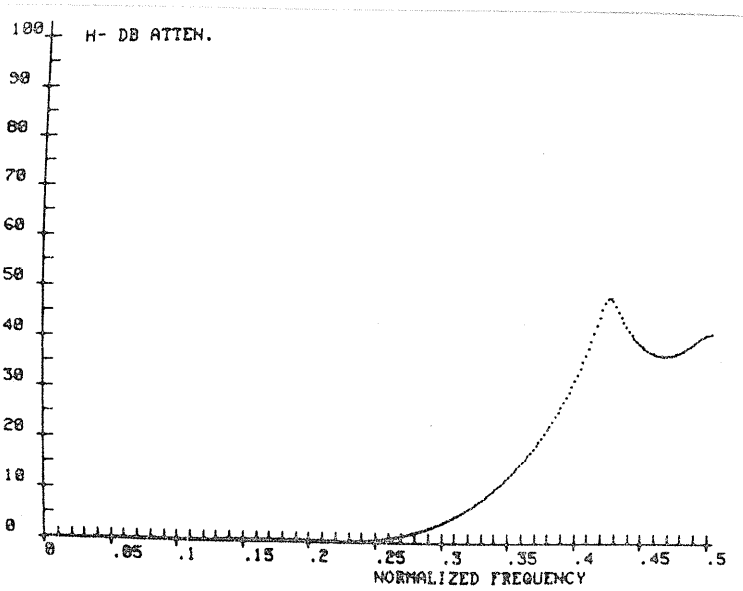
Realization for Shift and Add Structure in Example

(4.56). Here we see the longest path requires 5 sum delays, 1 logical inverter delay, and m+M+1 = 14 carry delays, so $t_p = 5\ t_s + 14\ t_c +$ 22 ns = 498 ns. Adding 130 ns operating time for sign magnitude truncation and data latches gives a total operating time of 628 ns, which corresponds to a maximum repetition rate of 1.59 Mhz. This figure is based on maximum specifications for t_s and t_c, while typical values for t_s and t_c are about 60% of maximum. The operating time could be further reduced by substituting adders with reduced t_c and t_s.

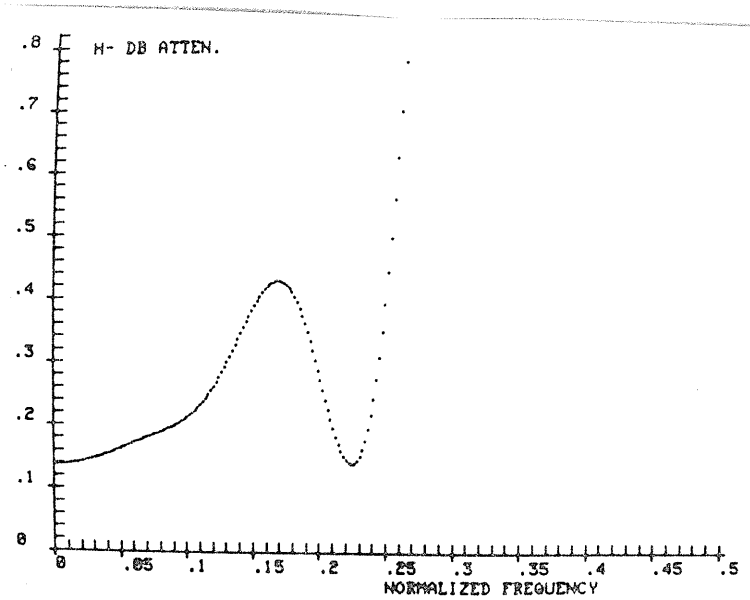
The example of figure 4.7 was simulated for m+1 = 8, 10, 12, 14, and 16 bits with assembly language programming on a PDP-11 computer. Each operation in the block diagram was performed with the corresponding assembly language command for two's complement arithmetic. The unit sample response and the magnitude and phase of its Fourier transform were displayed on the graphics terminal. Photographs 4-1 through 4-8 show the results for m+1 = 8 bits and 16 bits. The results for 16 bits agree closely with the expected warped frequency response of the analog prototype filter in figure 3.6. As predicted by the theory no limit cycle oscillations were observed in any case simulated. The response to a sinusoidal sequence at various frequencies was also observed on the graphics terminal. In this case overload due to arithmetic overflows was observed at a peak input of 75% of the arithmetic range.



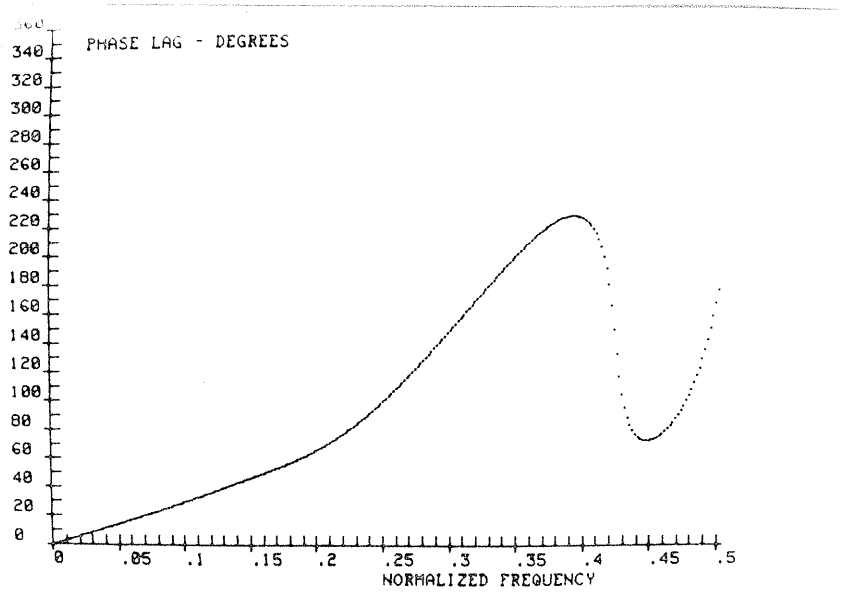
Photograph 4-1: Unit Sample Response (8 Bit Simulation)



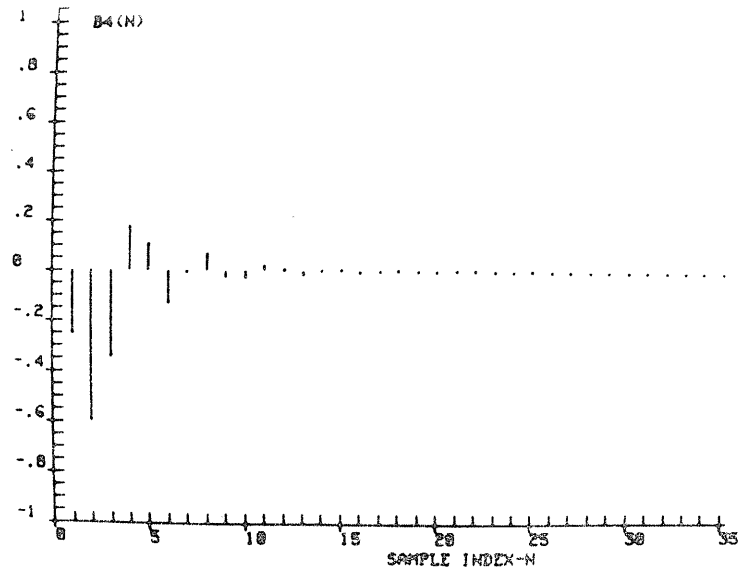
Photograph 4-2: Magnitude of Frequency Response (8 Bit Simulation)



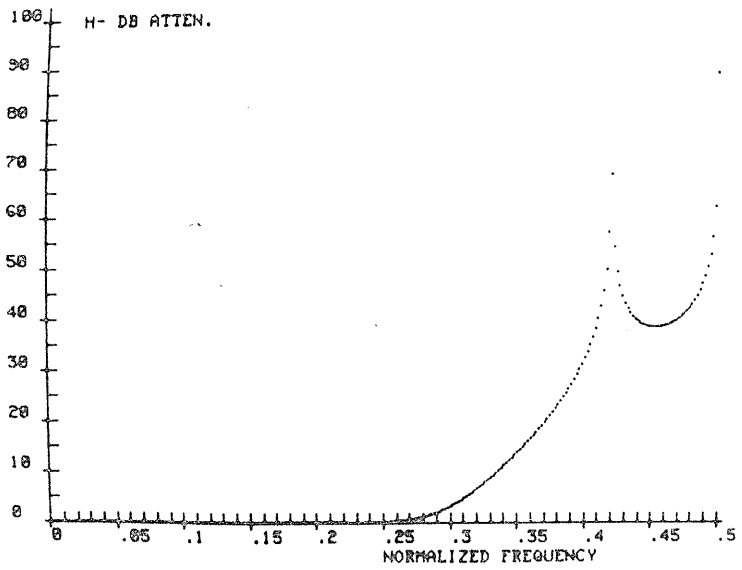
Photograph 4-3: Magnitude of Frequency Response Expanded in Passband (8 Bit Simulation)



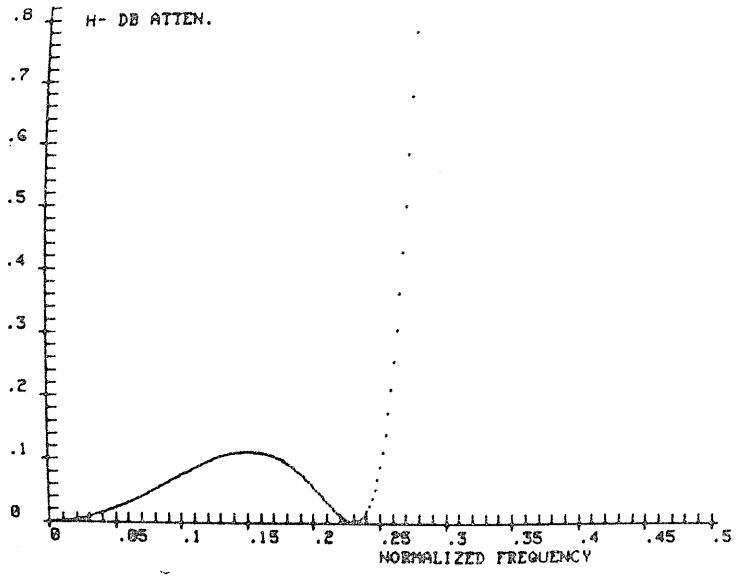
Photograph 4-4: Phase of Frequency Response (8 Bit Simulation)



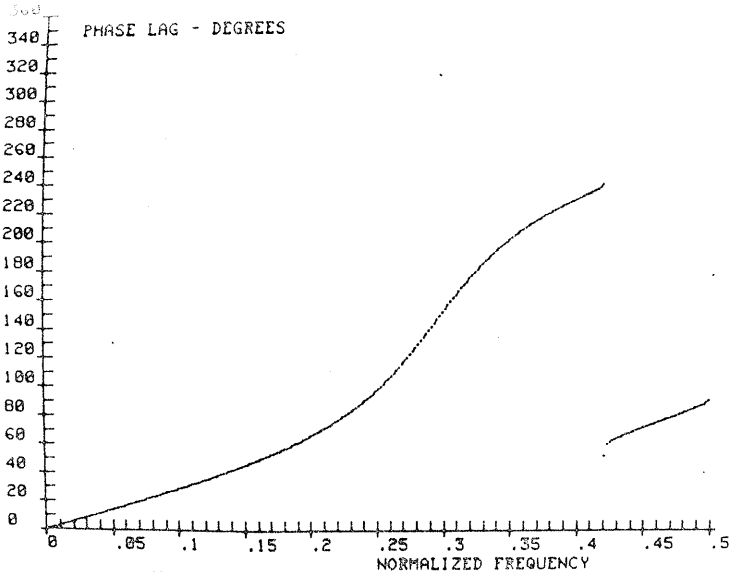
Photograph 4-5: Unit Sample Response (16 Bit Simulation)



Photograph 4-6: Magnitude of Frequency Response (16 Bit Simulation)



Photograph 4-7: Magnitude of Frequency Response Expanded in Passband (16 Bit Simulation)



Photograph 4-8: Phase of Frequency Response (16 Bit Simulation)

5. CONCLUSION AND FURTHER PROBLEMS

A numerical approach to hardware realization of the generalized or n-port adaptor with two's complement arithmetic has been given. It is found that the given method introduces no roundoff error during computation with the same wordlength as the input wordlengths. Hence sign magnitude truncation of the correct output, required for suppression of limit cycle oscillations, is readily implemented and the resulting wave digital filter will exhibit the minimum roundoff noise for a given signal wordlength. Also the method presents no drawbacks where the computation time of the hardware structure is concerned. The method appears to be reasonably hardware efficient, as verified in the example of chapter 4, but the question of hardware efficiency should be further investigated for a variety of prototype filters. It may be possible to develop a more hardware efficient variation of the method in certain cases or to utilize certain peculiarities in the numerical structure of the scattering matrix. A further search for alternate numerical methods is also desirable.

APPENDIX A

PROOFS FOR THEOREMS OF TWO'S COMPLEMENT ARITHMETIC

IN SECTION 4.7

Proof 4.1:

This is obvious from the definition of T(X) in 4.3.

Proof 4.2:

For $X \geq 0$, $T(X) = X$ and $T(X) = \sum_{i=0}^{m} X_i 2^{m-i}$. Hence

$$X = \sum_{i=0}^{m} X_i 2^{m-i} = X_0 2^m + \sum_{i=1}^{m} X_i 2^{m-i}.$$

Since $X_0 = 0$ we also have

$$X = -X_0 2^m + \sum_{i=1}^{m} X_i 2^{m-i}.$$

For $X < 0$, $T(X) = 2^{m+1} + X$ and $T(X) = \sum_{i=0}^{m} X_i 2^{m-i}$. Hence

$$2^{m+1} + X = \sum_{i=0}^{m} X_i 2^{m-i} = X_0 2^m + \sum_{i=1}^{m} X_i 2^{m-i} \quad \text{or}$$

$$X = -2^{m+1} + X_0 2^m + \sum_{i=1}^{m} X_i 2^{m-i}.$$

Since $X_0 = 1$ we have

$$X = -X_0 2^m + \sum_{i=1}^{m} X_i 2^{m-i}.$$

Proof 4.3:

For $X \geq 0$, $T(X) = X$ and $T(-X) = 2^{m+1} - X$. Hence

$$T(X) + T(-X) = X + 2^{m+1} - X = 2^{m+1}.$$

For $X < 0$, $T(X) = 2^{m+1} + X$ and $T(-X) = -X$. Hence

$$T(X) + T(-X) = 2^{m+1} + X - X = 2^{m+1}.$$

Proof 4.4:

From theorem 4.3 $T(X) + T(-X) = 2^{m+1}$ or

$T(-X) = 2^{m+1} - T(X)$. Since

$T(X) = \sum_{i=0}^{m} X_i 2^{m-i}$ and

$2^{m+1} = \sum_{i=0}^{m} 2^{m-i} + 1$ we have

$$T(-X) = \sum_{i=0}^{m} 2^{m-i} + 1 - \sum_{i=0}^{m} X_i 2^{m-i} = \sum_{i=0}^{m} (1-X_i) 2^{m-i} + 1 .$$

Now $1-X_i = \overline{X}_i$ for $i = 0,1,2,\ldots,m$ where $\overline{X}_i$ is the logical complement of X_i, so

$T(-X) = \sum_{i=0}^{m} \overline{X}_i 2^{m-i} + 1$ or in lexicographic notation

$T(-X) = \overline{X}_0\overline{X}_1\overline{X}_2\ldots\overline{X}_m + 000\ldots1.$

Proof 4.5:

a) Necessity- Assume $T_{(m)}(2^{-k}X)$ is defined, i.e. $2^{-k}X$ is an integer.

For $X \geq 0$, $T_{(m)}(X) = X$ so

$2^{-k}T_{(m)}(X)$ is an integer. Hence

$X_{m-k+1} = X_{m-k+2} = \ldots = X_m = 0.$

For $X < 0$, $T_{(m)}(X) = 2^{m+1} + X$ so

$2^{-k}T_{(m)}(X) = 2^{m+1-k} + 2^{-k}X$ which is an integer since it is the sum of two integers. Hence

$X_{m-k+1} = X_{m-k+2} = \ldots = X_m = 0.$

Sufficiency- Assume $X_{m-k+1} = X_{m-k+2} = \ldots = X_m = 0.$

For $X \geq 0$, $T_{(m)}(X) = X$ and since $2^{-k}T_{(m)}(X)$ is an integer $2^{-k}X$ is an integer.

For $X < 0$, $T_{(m)}(X) = 2^{m+1} + X$ and since $2^{-k}T_{(m)}(X)$ is an integer $2^{m+1-k} + 2^{-k}X$ is an integer. Since 2^{m+1-k} is an integer $2^{-k}X$ must be an integer.

b) For $X \geq 0$, $T(X) = X$,

$T(2^{-k}X) = 2^{-k}X = 2^{-k}T(X)$. Since $X_0 = 0$ we also have

$$T(2^{-k}X) = 2^{-k}T(X)+X_0 \sum_{i=0}^{k-1} 2^{m-i}.$$

For $X < 0$, $T(X) = 2^{m+1} + X$ so

$T(2^{-k}X) = 2^{m+1} + 2^{-k}X$. Since

$$X = -X_0 2^m + \sum_{i=1}^{m} X_i 2^{m-i} \quad \text{we have}$$

$$T(2^{-k}X) = 2^{m+1} + 2^{-k}\left(-X_0 2^m + \sum_{i=1}^{m} X_i 2^{m-i} \right)$$

$$= 2^{m+1} - X_0 2^{m-k} + 2^{-k} \sum_{i=1}^{m} X_i 2^{m-i} \text{ . Since } X_0 = 1$$

$$2^{m+1} - X_0 2^{m-k} = \sum_{i=0}^{k} 2^{m-i} \quad \text{so}$$

$$T(2^{-k}X) = \sum_{i=0}^{k} 2^{m-i} + 2^{-k} \sum_{i=1}^{m} X_i 2^{m-i}$$

$$= \sum_{i=0}^{k-1} 2^{m-i} + 2^{m-k} + 2^{-k} \sum_{i=1}^{m} X_i 2^{m-i}$$

$$= \sum_{i=0}^{k-1} 2^{m-i} + 2^{-k}\left(2^m + \sum_{i=1}^{m} X_i 2^{m-i} \right)$$

$$= X_0 \sum_{i=1}^{k-1} 2^{m-i} + 2^{-k} \sum_{i=0}^{m} X_i 2^{m-i} = X_0 \sum_{i=0}^{k-1} 2^{m-i} + 2^{-k}T(X) \text{ .}$$

Proof of corollary to 4.5:

a) Necessity- Assume $T_{(m)}(Y)$ is defined where $Y = 2^k X$, i.e. $2^k X$ is in the domain of $T_{(m)}$. From theorem 4.5 $T_{(m)}(2^{-k}Y) = T_{(m)}(X) = X_0X_1\ldots X_m = Y_0^{(1)}Y_0^{(2)}\ldots Y_0^{(k)}Y_0Y_1\ldots Y_{m-k}$ where

$Y_0^{(1)} = Y_0^{(2)} = \ldots = Y_0^{(k)} = Y_0$ so $X_0 = X_1 = \ldots = X_k$.

Sufficiency- Assume $T_{(m)}(X) = X_0X_1\ldots.X_m$ where $X_0 = X_1 = \ldots = X_k$.

Construct $T_{(m)}(Y) = X_kX_{k+1}\ldots X_m 0^{(1)}0^{(2)}\ldots 0^{(k)}$.

From theorem 4.5 $T_{(m)}(2^{-k}Y) = X_k^{(1)}X_k^{(2)}\ldots X_k^{(k)}X_kX_{k+1}\ldots X_m$,

where $X_k^{(1)} = X_k^{(2)} = \ldots = X_k^{(k)} = X_k$, so

$T_{(m)}(2^{-k}Y) = T_{(m)}(X)$. Since $T_{(m)}(\ )$ has a unique inverse

$Y = 2^kX$ and $T_{(m)}(Y) = T_{(m)}(2^kX)$.

b) The required $T_{(m)}(2^kX)$ was constructed in the sufficiency proof of part a) .

Proof 4.6:

$$T_{(m+M)}(2^MX) = \begin{cases} 2^MX & \text{if } X \geq 0 \\ 2^{M+m+1} + X & \text{if } X < 0 \end{cases}$$

For $X \geq 0$, $2^MX = 2^MT_{(m)}(X)$.

For $X < 0$, $2^{M+m+1} + 2^MX = 2^M\ (\ 2^{m+1} + X\ ) = 2^M\ T_{(m)}(X)$.

Hence $T_{(m+M)}(2^MX) = 2^M\ T_{(m)}(X)$ so the result in lexicographic notation follows.

Proof 4.7:

Consider the following three mutually exclusive cases which cover all possibilities: a) $X_0 = 0$; b) $X_0 = 1$ and $X_{m-M+1}X_{m-M+2}\ldots X_m \neq 0$; c) $X_0 = 1$ and $X_{m-M+1}X_{m-M+2}\ldots X_m = 0$.

a) Since $X_0 = 0$, $X \geq 0$ and $T_{(m)}(X) = X$.

Also $2^{-M}X \geq 0$ and $[2^{-M}X]\# \geq 0$ so

$T_{(m-M)}([2^{-M}X]\#) = [2^{-M}X]\# = [2^{-M}T_{(m)}(X)]\#$.

Note $2^{-M}T_{(m)}(X) < 2^{m+1-M}$ since $T_{(m)}(X) < 2^{m+1}$ so

$T_{(m-M)}([2^{-M}X]\#) = [\ [2^{-M}T_{(m)}(X)]\#\ ]^*$.

b) Since $X_0 = 1$, $X < 0$ and $T_{(m)}(X) = 2^{m+1} + X$.

Since $X_{m-M+1}X_{m-M+2}\cdots X_m \neq 0$, by part a) of theorem 4.5

$2^{-M}X$ is not an integer. There are two cases to consider.

Case 1: $-1 < 2^{-M}X < 0$

Here $[2^{-M}X]\# = 0$ so $T_{(m-M)}([2^{-M}X]\#) = [2^{-M}X]\# = 0$.

Also $2^{-M}T_{(m)}(X) = 2^{m+1-M} + 2^{-M}X$ so

$2^{m+1-M} > 2^{-M}T_{(m)}(X) > 2^{m+1-M} - 1$ and hence

$[2^{-M}T_{(m)}(X)]\# = 2^{m+1-M}-1$. Finally

$[\ [2^{-M}T_{(m)}(X)]\# + 1\ ]^* = [\ 2^{m+1-M} - 1 + 1\ ]^* = [2^{m+1-M}]^* = 0$,

which gives the proper result.

Case 2: $-\infty < 2^{-M}X < -1$

Here $[2^{-M}X]\# < 0$ so $T_{(m-M)}([2^{-M}X]\#) = 2^{m+1-M} + [2^{-M}X]\#$.

Since $2^{-M}X$ is not an integer we set

$2^{-M}X = -(I+F)$ where $I \geq 1$ is the integer part of $|2^{-M}X|$

and $1 > F > 0$ is the fractional part. Then

$2^{m+1-M} + [2^{-M}X]\# = 2^{m+1-M} + [-(I+F)]\# = 2^{m+1-M} - I$. Also

$[2^{-M}T_{(m)}(X)]\# = [2^{m+1-M} + 2^{-M}X]\# = [2^{m+1-M} - I - F]\# =$

$2^{m+1-M} - I - 1$ since $2^{m+1-M} - I$ is an integer and $1 > F > 0$.

Hence $T_{(m-M)}([2^{-M}X]\#) = [2^{-M}T_{(m)}(X)]\# + 1$. Since $I \geq 1$,

$[2^{-M}T_{(m)}(X)]\# \leq 2^{m+1-M} - 2$ so

$[\ [2^{-M}T_{(m)}(X)]\# + 1\ ]^* = [2^{-M}T_{(m)}(X)]\# + 1$ or

$T_{(m-M)}([2^{-M}X]\#) = [\ [2^{-M}T_{(m)}(X)]\# + 1\ ]^*$.

c) Since $X_0 = 1$, $X < 0$ and $T_{(m)}(X) = 2^{m+1} + X$.

Since $X_{m-M+1}X_{m-M+2}\cdots X_m = 0$, by part a) of theorem 4.5 $2^{-M}X$ is an integer. The only possibility is $-\infty < 2^{-M}X \leq -1$. This case may be treated by the argument of case 2 in part b) with the difference that $F = 0$ since $2^{-M}X$ is an integer. This gives $T_{(m-M)}([2^{-M}X]\#) = [2^{-M}T_{(m)}(X)]\#$ and the rest follows in a similar manner.

Proof 4.8:

Case 1: $0 \leq X \leq 2^m - 1$; $0 \leq Y \leq 2^m - 1$; $0 \leq X + Y \leq 2^m - 1$.

$T(X) = X$, $T(Y) = Y$, and $T(\ X + Y\ ) = X + Y$.

$[T(X) + T(Y)]^* = [X + Y]^* = X + Y = T(X+Y)$.

Case 2: $-2^m \leq X < 0$; $-2^m \leq Y < 0$; $-2^m \leq X + Y < 0$.

$T(X) = 2^{m+1} + X$, $T(Y) = 2^{m+1} + Y$, and $T(X+Y) = 2^{m+1} + (X+Y)$.

$[T(X) + T(Y)]^* = [2^{m+1} + X + 2^{m+1} + Y]^* = [2^{m+1} + (2^{m+1}+X+Y)]^*$

But $2^m \leq (2^{m+1} + X + Y) \leq 2^{m+1} - 1$ so

$[2^{m+1} + (2^{m+1} + X + Y)]^* = 2^{m+1} + (X+Y) = T(\ X + Y\ )$.

Case 3: $-2^m \leq X < 0$; $0 \leq Y \leq 2^m - 1$; $0 \leq X + Y \leq 2^m - 1$.

$T(X) = 2^{m+1} + X$, $T(Y) = Y$, and $T(X+Y) = X + Y$.

$[T(X) + T(Y)]^* = [2^{m+1} + X + Y]^*$

But $0 \leq X + Y \leq 2^m - 1$ so

$[2^{m+1} + X + Y]^* = X + Y = T(X + Y)$.

The case where the signs of X and Y are interchanged follows from the above by relabelling.

Case 4: $-2^m \leq X < 0$; $0 \leq Y \leq 2^m - 1$; $-2^m \leq X + Y < 0$.

$T(X) = 2^{m+1} + X$, $T(Y) = Y$, and $T(X+Y) = 2^{m+1} + (X+Y)$.

$[T(X) + T(Y)]^* = [2^{m+1} + X + Y]^*$

But $-2^m \leq X + Y \leq -1$ so

$[2^{m+1} + X + Y]^* = 2^{m+1} + (X+Y) = T(X + Y)$.

The case where the signs of X and Y are interchanged follows from the above by relabelling.

Proof 4.9:

T(X) and T(Y) are m+1 bit integers, i.e. $0 \leq T(X), T(Y) \leq 2^{m+1} - 1$. There are two possibilities: a) $0 \leq T(X) + T(Y) \leq 2^{m+1} - 1$; and b) $2^{m+1} \leq T(X) + T(Y) \leq 2^{m+2} - 2$.

a) $[T(X) + T(Y)]^* = T(X) + T(Y)$ in this case. Since no carry occurs $[T(X) + T(Y)]\not{c} = T(X) + T(Y)$. Hence

$[T(X) + T(Y)]^* = [T(X) + T(Y)]\not{c}$.

b) $[T(X) + T(Y)]^* = T(X) + T(Y) - 2^{m+1}$ in this case.

Note a single carry occurs and discarding a carry from the $m+1^{th}$ bit subtracts 2^{m+1} so

$[T(X) + T(Y)]\not{c} = T(X) + T(Y) - 2^{m+1}$. Hence

$[T(X) + T(Y)]^* = [T(X) + T(Y)]\not{c}$.

Proof 4.10:

a) If $X_0 \neq Y_0$ we have $-2^m \leq X < 0$ and $0 \leq Y \leq 2^m - 1$ or vice versa. Hence $-2^m \leq X + Y \leq 2^m - 1$ and an overflow cannot exist. If $X_0 = Y_0$ we have

Case 1: $0 \leq X, Y \leq 2^m - 1$

A. $0 \leq X + Y \leq 2^m - 1$

B. $2^m \leq X + Y \leq 2^{m+1} - 2$

Case 2: $-2^m \leq X, Y < 0$

A. $-2^m \leq X + Y < 0$

B. $-2^{m+1} \leq X + Y < -2^m$

Note case 1B and 2B are overflow conditions.

b) Necessity- If an overflow exists it is either case 1B or case 2B of a) .

Case 1B: $X_0 = Y_0 = 0$, $T(X) = X$, $T(Y) = Y$, and

$T = [T(X) + T(Y)]^* = [X+Y]^* = X + Y$ where

$2^m \leq X + Y \leq 2^{m+1} - 2$ so $T_0 = 1 \neq X_0 = Y_0$.

Case 2B: $X_0 = Y_0 = 1$, $T(X) = 2^{m+1} + X$, $T(Y) = 2^{m+1} + Y$, and

$T = [T(X) + T(Y)]^* = [2^{m+1} + X + 2^{m+1} + Y]^* =$

$2^{m+1} + (X+Y)$ where $0 \leq 2^{m+1} + (X+Y) < 2^m$, so

$T_0 = 0 \neq X_0 = Y_0$.

Sufficiency- If $T_0 \neq X_0 = Y_0$ we cannot have any of cases 1 through 4 in proof 4.8 since $T_0 = X_0 = Y_0$ there. The only other possible cases are 1B and 2B in part a) of this proof which are overflow conditions.

Proof 4.11:

Let $T = [\, T(X) + T(Y) \,]^*$.

Necessity- If an overflow exists $T_0 \neq X_0 = Y_0$ by theorem 4.10. There are two possibilities for $X_0 = Y_0$:

a) If $X_0 = Y_0 = 0$ then $C = 0$ and $T_0 = 0$ unless $V = 1$. Hence $V = 1$.

b) If $X_0 = Y_0 = 1$ then $C = 1$ and $T_0 = 1$ unless $V = 0$. Hence $V = 0$.

In a) and b) $C \oplus V = 0$.

Sufficiency- Assume $C \oplus V = 1$. There are two possibilities:

a) If $C = 0$ and $V = 1$ we cannot have $X_0 = 1$ and/or $Y_0 = 1$ since this implies $C = 1$. Hence $X_0 = Y_0 = 0$, so $T_0 = 1$ since $V = 1$.

b) If $C = 1$ and $V = 0$ we must have $X_0 = Y_0 = 1$, which also implies $T_0 = 0$ since $V = 0$.

In both a) and b) $T_0 \neq X_0 = Y_0$ so an overflow exists by theorem 4.10.

Proof 4.12:

An overflow corresponds to either case 1B or 2B in part a) of proof 4.10. Scaling X and Y by 1/2 leaves $-2^m \leq X, Y \leq 2^m - 1$ and gives $2^{m-1} \leq (X/2) + (Y/2) \leq 2^m - 1$ for case 1B or $-2^m \leq (X/2) + (Y/2) \leq -2^{m-1}$ for case 2B , which eliminates the overflow.

Proof 4.13:

T(Z/2) exists since Z/2 is an integer and prescaling eliminates the overflow by theorem 4.12. An overflow corresponds to either case 1B or 2B in part a) of proof 4.10, which with theorem 4.11 gives:

1B. $X_0 = Y_0 = 0$, $T_0 = 1$, C = 0 and V = 1.

$T = [T(X) + T(Y)]^* = T(X) + T(Y)$ since C = 0 .

Also $T(X) + T(Y) = X + Y$ so $T = X + Y$. Note $T_m = 0$ since T/2 is an integer. Hence

$T(Z/2) = (X+Y)/2 = T/2 = 2^{-1}(T_0T_1T_2\ldots T_m) = 0T_0T_1T_2\ldots T_{m-1}$.

$\overline{T}_0 = 0$, so $T(Z/2) = \overline{T}_0T_0T_1T_2\ldots T_{m-1}$.

2B. $X_0 = Y_0 = 1$, $T_0 = 0$, C = 1 and V = 0.

$T = [T(X) + T(Y)]^* = T(X) + T(Y) - 2^{m+1}$ since C = 1 .

Also $T(X) + T(Y) = 2^{m+1} + X + 2^{m+1} + Y$ so

$T = 2^{m+1} + X + 2^{m+1} + Y - 2^{m+1} = 2^{m+1} + X + Y$.

Note $T_m = 0$ since T/2 is an integer. Hence

$T(Z/2) = 2^{m+1} + (X+Y)/2 = 2^m + T/2 = 2^m + 2^{-1}(T_0T_1T_2..T_m)$

$= 2^m + T_0T_1T_2\ldots T_{m-1}$. Since $\overline{T}_0 = 1$ and inserting a 1 in the missing high order bit of $T_0T_1T_2\ldots T_{m-1}$ corresponds to adding 2^m we have $T(Z/2) = \overline{T}_0T_0T_1T_2\ldots T_{m-1}$.

Proof 4.14:

Note T(Z) = T by definition of an inverse function.

a) Positive overflow- This corresponds to case 1B in part a) of proof 4.10. Here

$T = [\ X + Y\ ]^* = X + Y$ but since $2^m \leq X + Y \leq 2^{m+1} - 2$

Z must be negative, i.e. $T(Z) = 2^{m+1} + Z = X + Y$. Hence

$Z = X + Y - 2^{m+1}$.

b) Negative overflow- This corresponds to case 2B in part a) of proof

4.10. Here

$T = [2^{m+1} + X + 2^{m+1} + Y]^* = 2^{m+1} + X + Y$ but since

$0 \leq 2^{m+1} + X + Y < 2^m$ Z must be positive, i.e.

$T(Z) = Z = 2^{m+1} + X + Y$.

Proof 4.15:

By theorem 4.14 we have for each T_k, $k = 2,3,\ldots,n$, $T^{-1}(T_k) = T^{-1}(T_{k-1}) + X_k + q_k 2^{m+1}$ where we define q_k as -1, 0, or 1 corresponding to a positive overflow, no overflow, or a negative overflow, respectively, during the k^{th} step. Proceding by induction in reverse order from $T^{-1}(T_n)$ we have

$$T^{-1}(T_n) = (X_1 + X_2 + \ldots + X_n) + (\sum_{k=1}^{n} q_k) 2^{m+1} \text{ , but since}$$

$$\sum_{k=1}^{n} q_k = N - P \text{ , } T^{-1}(T_n) = S + (N - P) \, 2^{m+1} \text{ .}$$

Proof of corollary 1 to 4.15:

From theorem 4.15 if $N = P$ then $T^{-1}(T_n) = S + (N-P)2^{m+1} = S$.

If $N \neq P$ then $N - P \neq 0$ and $T^{-1}(T_n) \neq S$.

Proof of corollary 2 to 4.15:

From theorem 4.15 $T^{-1}(T_n) = S + (N-P)2^{m+1}$ or

$S = T^{-1}(T_n) - (N-P)2^{m+1}$ where $-2^m \leq T^{-1}(T_n) \leq 2^m - 1$.

If $N - P \neq 0$, since N and P are integers, either $S \geq 2^m$ or $S < -2^m$, which contradicts the hypothesis. Hence

$N - P = 0$ and $T^{-1}(T_n) = S$.

REFERENCES

[1] J. N. Franklin, Matrix Theory, Prenctice-Hall, N. J., 1968.

[2] R. V. Churchill, Introduction to Complex Variables and Applications, McGraw-Hill, N. Y., 1948.

[3] G. Doetsch, Introduction to the Theory and Application of the Laplace Transformation, Springer-Verlag, N. Y., 1974.

[4] J. R. Ragazinni and G. F. Franklin, Sampled Data Control Systems, McGraw-Hill, N. Y., 1958.

[5] A. V. Oppenheim and R. W. Schafer, Digital Signal Processing, Prentice-Hall, N. J., 1975.

[6] B. Gold and C. M. Rader, Digital Processing of Signals, McGraw-Hill, N. Y., 1969.

[7] K. Ogata, State Space Analysis of Control Systems, Prentice-Hall, N. J., 1967.

[8] J. E. Rubio, The Theory of Linear Systems, Academic Press, N. Y., 1971.

[9] T. Bickart, N. Balabanian, and S. Seshu, Electrical Network Theory, Wiley, N. Y., 1969.

[10] S. Seshu and M. B. Reed, Linear Graphs and Electrical Networks, Addison-Wesley, Mass., 1961.

[11] E. S. Kuh and R. A. Rohrer, "The State Variable Approach to Network Analysis", Proc. IEEE, vol. 53, pp. 672-686, July 1965.

[12] A. Fettweis, "Digital Filter Structures Related to Classical Filter Networks", Arch. Elek. Ubertragung., vol. 25, pp. 79-89, Feb. 1971.

[13] A. Sedlmeyer and A. Fettweis, "Digital Filters with True Ladder Configuration", Int. J. Circuit Theory and Applications, vol. 1, pp. 5-10, March 1973.

[14] A. Fettweis and K. Meerkötter, "On Adaptors for Wave Digital Filters", IEEE Trans. Acoustics, Speech, and Sig. Proc., vol. ASSP-23, pp. 516-524, Dec. 1975.

[15] A. Fettweis, "Realizability of Digital Filter Networks", Arch. Elek. Ubertragung., vol. 30, pp. 90-96, Feb. 1976.

[16] G. O. Martens and K. Meerkötter, "On N-Port Adaptors for Wave Digital Filters with Application to a Bridged - Tee Filter", Proc. 1976 IEEE Symp. Circuits and Systems, Munich, pp. 514-517.

[17] A. Fettweis, "Pseudopassivity, Sensitivity, and Stability of Wave Digital Filters", IEEE Trans. Circuit Theory, vol. CT-19, pp. 668-673, Nov. 1972.

[18] A. Fettweis and K. Meerkötter, "Suppression of Parasitic Oscillations in Wave Digital Filters", IEEE Trans. Circuits and Systems, vol. CAS-22, pp. 239-246, March 1975.

[19] T. A. Claasen, W. F. Mecklenbräuker, and J. B. Peek, "On the Stability of the Forced Response of Digital Filters with Overflow Nonlinearities", IEEE Trans. Circuits and Systems, vol. CAS-22, pp. 692-696, Aug. 1975.

[20] A. Fettweis, "On the Connection between Multiplier Word Length Limitation and Roundoff Noise in Digital Filters", IEEE Trans. Circuit Theory, vol. 19, pp.486-491, Sept. 1972.

[21] A. Fettweis, "Wave Digital Filters with Reduced Number of Delays", Int. J. Circuit Theory and Applications, vol. 2, pp. 319-330,March 1974.

[22] A. Fettweis, "Canonic Realization of Ladder Wave Digital Filters", Int. J. Circuit Theory and Applications, vol. 3, pp. 321-332,March 1975.

[23] A. Fettweis, H. Levin, and A. Sedlmeyer, "Wave Digital Lattice Filters", Int. J. Circuit Theory and Applications, vol. 2, pp. 203-211, March 1974.

[24] A. T. Ashley, "Minimal Wave Digital Filter Realizations: The n-Port Approach", Ph.D. Thesis, Dept. of Elec. Eng., University of Manitoba, May 1978.

[25] H. H. Le, "Wave Digital Adaptors for Brune, Darlington C and D, and Twin-T Sections", Ph.D. Thesis, Dept. of Elec. Eng., University of Manitoba, August 1977.

[26] G. W. Reitweisner, "Binary Arithmetic" in Advances in Computers, F. L. Alt ed., vol. 1, pp. 231-308, Academic Press, N. Y., 1960.

[27] O. L. MacSorley, "High Speed Arithmetic in Binary Computers", Proc. IRE, vol. 49, pp. 67-91, Jan. 1961.

[28] N. R. Scott, Electronic Computer Technology, McGraw-Hill, N. Y., 1970.

[29] H. A. Curtis, Design of Switching Circuits, Van Nostrand, N. Y., 1962.