THE UNIVERSITY OF CALGARY

A HIGH-QUALITY SINGLE CHIP PROGRAMMABLE DIGITAL FILTER

by

Brian D. Green

A THESIS

SUBMITTED TO THE FACULTY OF GRADUATE STUDIES

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE

DEGREE OF MASTER OF SCIENCE

DEPARTMENT OF ELECTRICAL ENGINEERING

CALGARY, ALBERTA

OCTOBER, 1987

ISBN 0-315-42450-8

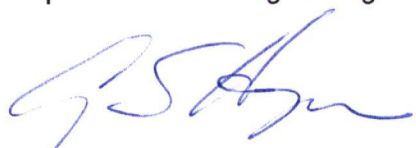
THE UNIVERSITY OF CALGARY

FACULTY OF GRADUATE STUDIES

The undersigned certify that they have read, and recommended to the Faculty of Graduate Studies for acceptance, a thesis entitled, "*A High-quality Single Chip Programmable Digital Filter*", submitted by Brian D. Green in partial fulfillment of the requirements for the degree of Master of Science.

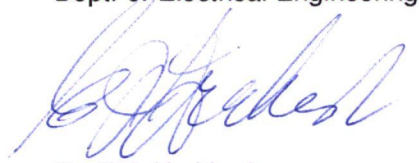
Supervisor - Dr. L.E. Turner
Dept. of Electrical Engineering

Dr. J.W. Haslett
Dept. of Electrical Engineering



Dr. G.S. Hope
Dept. of Electrical Engineering

Dr. M.R. Smith
Dept. of Electrical Engineering



Dr. E.J. Krakiwsky
Dept. of Surveying Engineering

Date: October 20, 1987.

Abstract

This thesis describes the development of a high-quality single chip programmable digital filter. Six candidate digital filter structures are examined in terms of sensitivity to multiplier coefficient quantization, susceptibility to overflow, and quantization noise generation over a broad range of bandpass transfer functions derived from a 6^{th}-order elliptic analog prototype filter.

Based on the non-ideal performance of the structures, the "LDI" structure is chosen for an actual filter implementation in CMOS gate array technology. The detailed design of the LDI filter is described, and measured results from the actual fabricated filter are compared with theoretically expected behavior. The LDI filter shows excellent agreement with theory, and attains a very high level of performance characterized by 16-bit input and output, freedom from signal overflow, quantization noise contributing less than one least significant bit to the output, sample rates as high as 180 kHz, programmable lowpass or bandpass filtering functions, and programmable passband ripple, passband edge frequencies, and stopband attenuation.

Acknowledgements

The author expresses his appreciation for the guidance, advice, and constructive criticism offered by Dr. L. E. Turner during the course of this research.

Also greatly appreciated are the Government of Alberta, the Natural Sciences and Engineering Research Council, and the Electrical Engineering Department, whose financial support respectively in the form of Ralph Steinhauer Award, Post Graduate Scholarship, and research and teaching assistantships allowed the author to undertake his studies.

For the actual production of working filter chips, and provision of facilities for the simulation and schematic entry of the design, the author is greatly indebted to the Alberta Microelectronic Centre and its staff.

The production of Figure 6.6 was made possible by the author's father D. F. Green, who generously provided instruction and equipment.

To Marie, for the hugs,

and Dave, for the jugs,

and all my other good friends who kept me going.

Table of Contents

List of Tables

List of Figures

List of Symbols

A	incident voltage wave (chapter 2) or filter state matrix (chapter 4) or number of adders (chapter 6)
$A_i(z)$	z-transformed recursion polynomial
a_i	constant coefficients
B	reflected voltage wave (chapter 2) or constraint bounding matrix (chapter 4)
$B_i(z)$	z-transformed recursion polynomial
b	input vector
b_i	constant coefficients
$b_{ij}(k)$	critical node i response sequence to node j input
C	capacitor
C	convex cone (chapter 4) or number of distinct coefficients (chapter 6)
C_i	prototype capacitance
C_i'	equivalent prototype capacitance
C_i^P	precompensated capacitance
$C0$–$C9$	multiplier coefficients
$C_i(z)$	z-transformed recursion polynomial
c	constant energy value
c_i	constant coefficients or bit values
D	number of subtracters
$D(z)$	z-transformed denominator polynomial

DF1	canonic direct form structure
DF2	cascaded direct form structure
d_i	constant coefficients or bit values
$E(x)$	quantization error function
$E(x,\Delta x)$	relaxed constraint quantization error function
e_i	i^{th} column of the identity matrix
$e_l(k)$	node l truncation error sequence
F	fractional coefficient bits
FA	full adder
$F[]$	non-linear quantization operation
f_l	lower cutoff frequency
f_u	upper cutoff frequency
f_{br}	bit rate clock frequency
G_i	port conductances
$H(z)$	discrete transfer function
HA	half adder
$h(k)$	unit sample response sequence
$h_i(k)$	critical node i unit sample response sequence
I	port current (chapter 2) or identity matrix (chapter 4) or integer coefficient bits (chapter 6)
$I(x)$	intermediate error function
$I'(z)$	transformed intermediate error function

j	complex operator
k	sequence index
k_E	constant of differentiation
k_i	lattice filter coefficients
L	inductor
L_i	prototype inductance
L_i'	equivalent prototype inductance
L_i^P	precompensated inductance
L_M	bounding energy value
$L(x)$	Liapunov function
LDI	lossless discrete integrator
LF1	two multiplier lattice structure
LF2	one multiplier lattice structure
LSB	least significant bit
l	sequence index
M	input sequence bound
MSB	most significant bit
$M1$–$M6$	adapter multiplier values
m	integer value
N	error sequence bound
$N(z)$	z-transformed numerator polynomial
n	integer value
n_i	constant coefficients

$o_e(k)$	output error component sequence
P1	one multiplier parallel adapter
P2	two multiplier parallel adapter
p_i	product bit values
Q	rotation matrix
R	resistor
R	port resistance (chapter 2) or number of register operations (chapter 6)
R_i	input resistance
R_o	output resistance
R_i	port resistances
R_{out}	wave filter output resistance
S	real symmetric matrix (chapter 4) or signal word length (chapter 6)
$S(z)$	z-transformed arbitrary sequence
S1	one multiplier series adapter
S2	two multiplier series adapter
s	continuous complex frequency variable
s_i	recursion variable
$s(k)$	arbitrary sequence
$s(k)_i$	critical node i signal sequence
T	sample period or matrix of right eigenvectors (chapter 4)
U	point of maximum energy deviation

$U(z)$	z-transformed input sequence
u_c	constant input
$u(k)$	input sequence
V	port voltage (chapter 2) or point of maximum energy deviation (chapter 4)
V_{in}	input voltage
V_{out}	output voltage
v_i	two multiplier lattice tap coefficients (chapter 2) or basis vectors (chapter 4)
$\hat{v}_i$	one multiplier lattice tap coefficients
$v_l(k)$	output response sequence to node l input
WDF	wave digital filter
X1-X6	LDI filter states
x_e	equilibrium state vector under constant input
x_i	i^{th} component of state vector x
$x(k)$	filter ideal state vector
$\hat{x}(k)$	filter non-ideal state vector
$Y(z)$	z-transformed output sequence
y	transformed state vector
y_i	i^{th} component of transformed state vector y
$y(k)$	output sequence
z	discrete complex frequency variable or transformed state vector (chapter 4)
z_0	normalized solution vector

α	constant multiplier or parameter
β	constant parameter
Δ	ideal passband ripple
Δe	energy function deviation
Δx	sign-magnitude quantization state vector error
δ	maximum sign-magnitude quantization error
Γ	limit cycle confinement region
γ	transformed state vector error
γ'	projection of γ onto C
Λ_A	diagonal matrix of eigenvalues of A
Λ_S	diagonal matrix of eigenvalues of S
λ_M	maximum modulus eigenvalue
Ω	discrete frequency variable
Ω_c	geometric center frequency
Ω_l	lower cutoff frequency (rads/sec.)
Ω_u	upper cutoff frequency (rads/sec.)
ω	continuous frequency variable
ϕ	rotation angle variable
ρ	maximum quantization error
θ	coordinate system rotation angle
μ_i	eigenvalues of S
μ_j	maximum eigenvalue of S

CHAPTER 1

INTRODUCTION

In its broadest sense, digital signal processing is the production of a set of numbers (the "output") through the application of a well-defined computational algorithm to another set of numbers (the "input") [1]. With the appearance of the digital computer and its subsequent rapid growth in performance, digital signal processing has played an increasingly prominent role across a broad range of applications such as music, communication, radar, sonar, speech, seismic, and medical signal processing [2]. The possibility of using integrated circuit technology to implement entire signal processing systems with digital components has been recognized since the mid-1960s [2]. In particular, the continued reduction in size and increase in density of integrated MOS processing elements has resulted in intensive efforts to develop compact and sophisticated MOS VLSI signal processing systems. While the requirement for dealing with analog signals in such systems is likely to remain, the trend is towards digital processing techniques [3], which are notable for guaranteed performance even in the presence of variable factors of both process and environment.

1.1. Digital Filters

One element of many signal processing tasks is filtering, which if performed in the digital realm is digital filtering. The most general definition of digital filtering is virtually indistinguishable from the definition of digital signal processing presented above. Although this definition certainly is accurate in the sense that it encompasses all possible meanings of the term "digital filter", it is far too ambiguous to be useful. In the context of this thesis, a much more restrictive definition of digital filtering is applied: a digital filter is a well-defined computational algorithm which is applied to a sequence of numbers (the "input") which represents

the instantaneous values of some time-varying signal at equally spaced intervals T in time and produces another sequence of numbers (the "output") which also represents the instantaneous values of a time-varying signal at intervals T. This definition is more in keeping with what is usually meant by the term digital filter, although the independent variable may sometimes be something other than time (e.g. space). A further restriction which is imposed here is that a digital filter ideally be linear and time invariant (sometimes called shift invariant) so that its operation can be described by the convolution summation

$$y(k) = \sum_{l=-\infty}^{l=\infty} u(l)h(k-l), \tag{1.1}$$

where $\{y(k)\}$, $\{u(k)\}$, and $\{h(k)\}$ respectively represent the output, input, and unit sample response sequences of the filter [2].

Convolution summations of the form (1.1) can be conveniently dealt with using a mathematical tool known as the z-transform [2]. For a sequence $\{s(k)\}$, its z-transform $S(z)$ is given by

$$S(z) = \sum_{k=-\infty}^{k=\infty} s(k)z^{-k}. \tag{1.2}$$

Two properties of the z-transform are of particular interest in the study of digital filters. First, under the z-transform, the convolution summation (1.1) is converted to the product

$$Y(z) = U(z)H(z), \tag{1.3}$$

so that the properties of an ideal digital filter are completely characterized by $H(z)$. Since $H(z)$ relates the z-transformed output of a digital filter to its z-transformed input, it is referred to as the transfer function of the digital filter. A second useful property of the z-transform is that by substituting $z = e^{j\omega T}$ in the z-transform of a sequence, the frequency spectrum of the corresponding time-sampled signal is produced. Using this property and the bilinear transformation [2] given by

$$s = \frac{2}{T}\frac{(z-1)}{(z+1)}, \tag{1.4}$$

the transfer function of an analog filter can be converted to a z-domain transfer function which retains all the attenuation characteristics of the analog filter. While the attenuation characteristics are retained, the bilinear transformation does however distort the frequency scale of the analog response, since it maps the entire $s=j\omega$ axis onto the unit circle in the z-domain according to the relation

$$\omega = \frac{2}{T}\tan(\frac{\Omega T}{2}), \tag{1.5}$$

where ω and Ω are the analog and discrete frequency variables respectively. Relation (1.5) can be used to map a set of digital filter cutoff frequencies into the analog domain, where a suitable analog prototype filter can be designed to meet these mapped cutoff frequencies. If the transfer function of this prototype is then converted to a discrete transfer function using the bilinear transformation, the discrete transfer function has the desired cutoff frequencies. Compensation for the frequency warping effect of (1.5) can be accomplished in this fashion for many important types of filter transfer functions including lowpass, bandpass, and highpass [2]. The discrete transfer functions which result from bilinear transformation of analog prototypes are typically rational polynomials of finite order. Such transfer functions represent digital filters which can be described by difference equations of the form

$$y(k) = \sum_{i=0}^{i=m} b_i u(k-i) - \sum_{j=1}^{j=m} a_j y(k-j), \tag{1.6}$$

where b_i and a_j are constant coefficients. Filters which implement difference equations of the form (1.6) can be be constructed using only the operations of addition, subtraction, multiplication, and delay (storage). The symbols used to represent these operations are shown in figure 1.1.

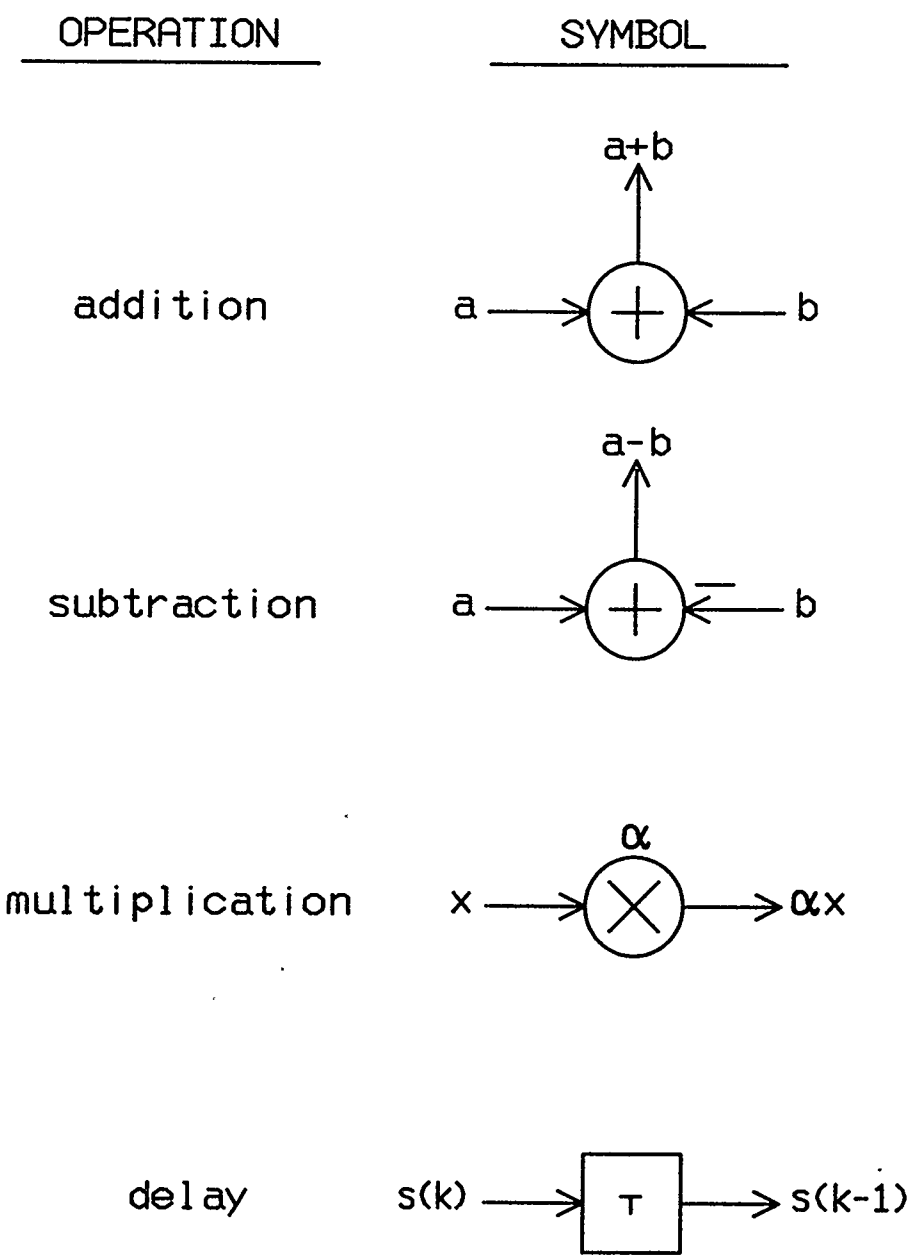


Figure 1.1 : Basic Digital Filter Operations

1.2. Research Goals

Recently, complete digital filters have been incorporated on single chips which perform certain fixed digital signal processing tasks [4, 5]. These filters generally implement fixed transfer functions, or have an extremely limited range of programmability. General purpose digital signal processing chips such as Texas Instruments' TMS320 and NEC's μPD77230 have also appeared. These devices allow a relatively broad range of digital signal processing tasks to be performed at the expense of some loss of speed and suitability to any particular processing task when compared to a fully customized special-purpose device. In order to investigate the design of a device which compromises between the fully specialized filter and the general purpose processor, the goal set for the research described in this thesis was to design a single chip digital filter capable of implementing a single class of transfer function with programmable cutoff frequencies, stopband attenuation, and passband ripple.

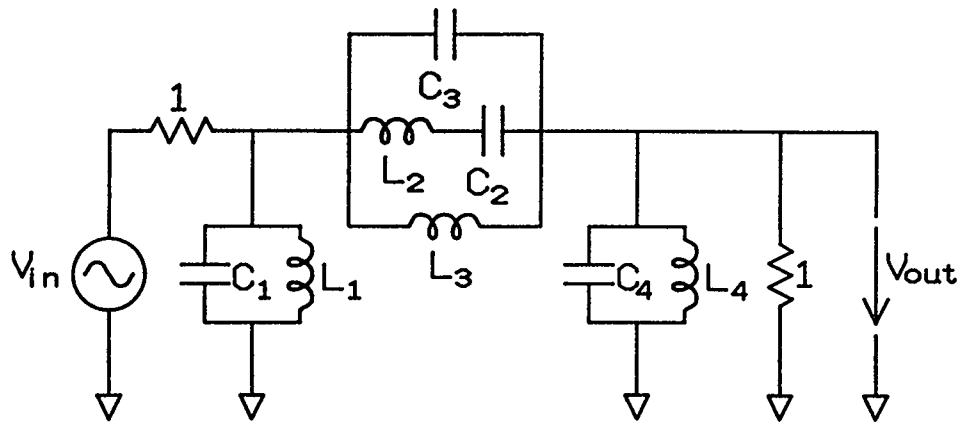
Research already underway had generated some familiarity with digital filters which implemented the bilinear transformed transfer function of the equivalent analog prototypes shown in figure 1.2, hence this was chosen as the prototype for the digital filter to be designed. The prototype filter is a sixth-order elliptic bandpass filter characterized by three zeros of attenuation and two equal peaks of attenuation in the passband, and transmission zeros in each of the stopbands. A typical bilinear transformed magnitude transfer function for this prototype is shown in figure 1.3. The bilinear transformed transfer function of the prototype is given by

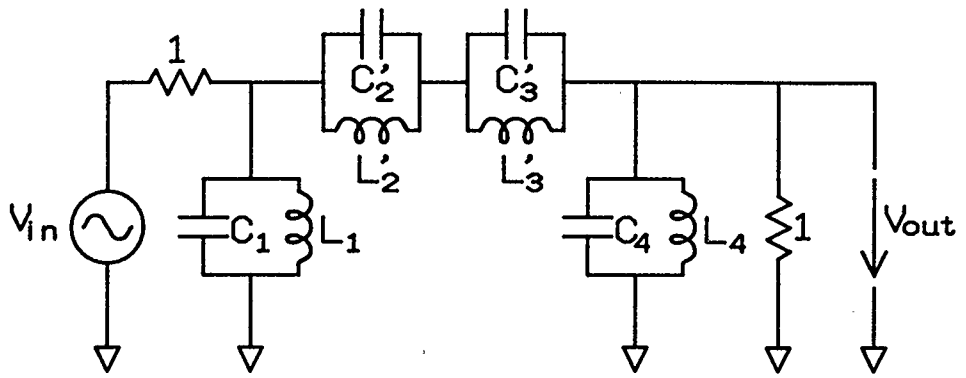
$$H(z) = \alpha\left\{\frac{z^6+a_5z^5+a_4z^4-a_4z^2-a_5z-1}{z^6+b_5z^5+b_4z^4+b_3z^3+b_2z^2+b_1z+b_0}\right\}, \tag{1.7}$$

which has the form

$$H(z) = \alpha\left\{\frac{(z^2-1)(z^2+c_1z+1)(z^2+c_0z+1)}{(z^2+d_5z+d_4)(z^2+d_3z+d_2)(z^2+d_1z+d_0)}\right\} \tag{1.8}$$

when the numerator and denominator are factored into their respective second-order terms.





$L_1 = L_4 \qquad C_1 = C_4$

Figure 1.2 : Equivalent Prototype Filters

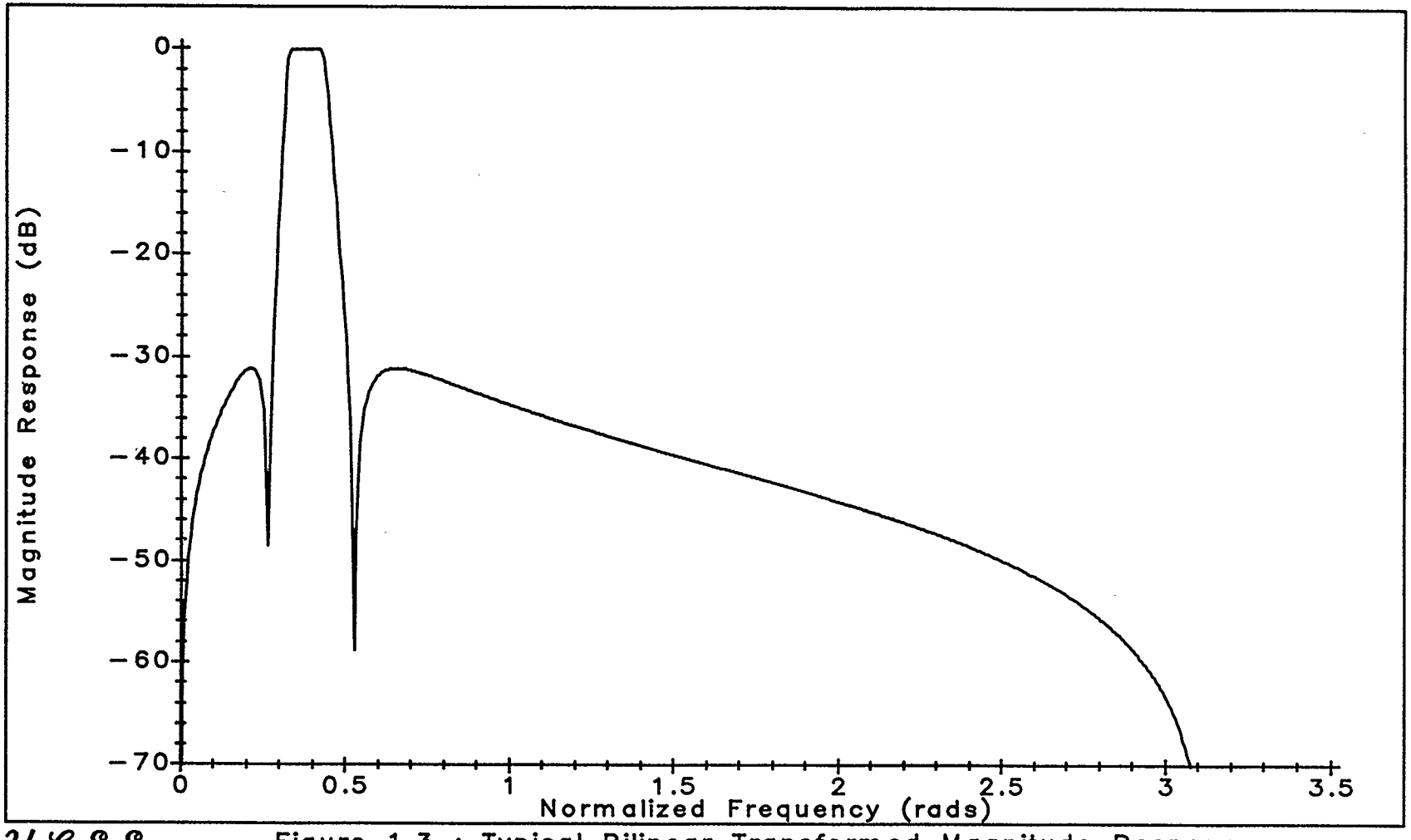


Figure 1.3 : Typical Bilinear Transformed Magnitude Response

Subject to the constraint of being able to fit a digital filter implementing transfer function (1.7) on a single chip, additional research goals were to accommodate 16-bit input and output data, maximize filter throughput, and minimize non-ideal effects.

1.3. Overview

There are many digital filter structures capable of implementing the transfer function (1.7). These structures will differ in hardware cost and non-ideal performance, so it is desirable to examine a range of structures for implementing the filter in order to take advantage of performance and/or cost benefits which may vary between structures. In chapter two, six different digital filter structures are presented which are subsequently assessed for their suitability with respect to the goals of the research. The non-ideal effects which are encountered in a practical digital filter implementation, and ways to reduce or eliminate them are discussed in chapters three and four. In chapter five, initial size and performance estimates are made for each of the candidate structures, and one structure is selected for detailed design. Facilities for fabrication of the selected structure in CMOS gate-array technology were made available by the Alberta Microelectronic Centre, hence the size estimates of chapter five are based on this technology. In chapter six, the detailed design of the selected filter structure is discussed, gate-array technology is briefly described, and measured results are presented for the fabricated digital filter. Performance indices based on the information gained during the detailed design and rough performance comparisons for alternative filter implementations are presented in chapter seven. The degree to which the research goals were attained is discussed, and recommendations for further research are presented.

CHAPTER 2

FILTER STRUCTURES

There are an unlimited number of filter structures which will implement a given transfer function. Various techniques exist which allow a specified transfer function or prototype filter to be reduced to a particular implementing structure in a systematic manner. In this chapter four such techniques are presented, giving rise to six structures capable of implementing transfer function (1.7). The intent of the descriptions presented here is to provide only a brief introduction to each technique and the resultant structures. For a detailed description of the different techniques, the reader should consult the references indicated in each of the following sections.

2.1. Direct Form Filters

Direct form structures [6] can be determined by inspection of the desired transfer function. Figure 2.1 shows the direct form implementation of (1.7). The multiplier coefficients for this structure are simply the coefficients of the powers of z in the desired transfer function. Direct form structures of the type shown in figure 2.1 have been referred to as "canonic form", but are referred to here as the DF1 structure for compactness. The factored transfer function (1.8) can be expressed as the product of three second-order transfer functions. If each of these second-order transfer functions is implemented by a canonic form structure, the second-order sections can be cascaded to give the direct form structure shown in figure 2.2. Structures of this type have been referred to as "cascade form" but will here be referred to as the DF2 structure.

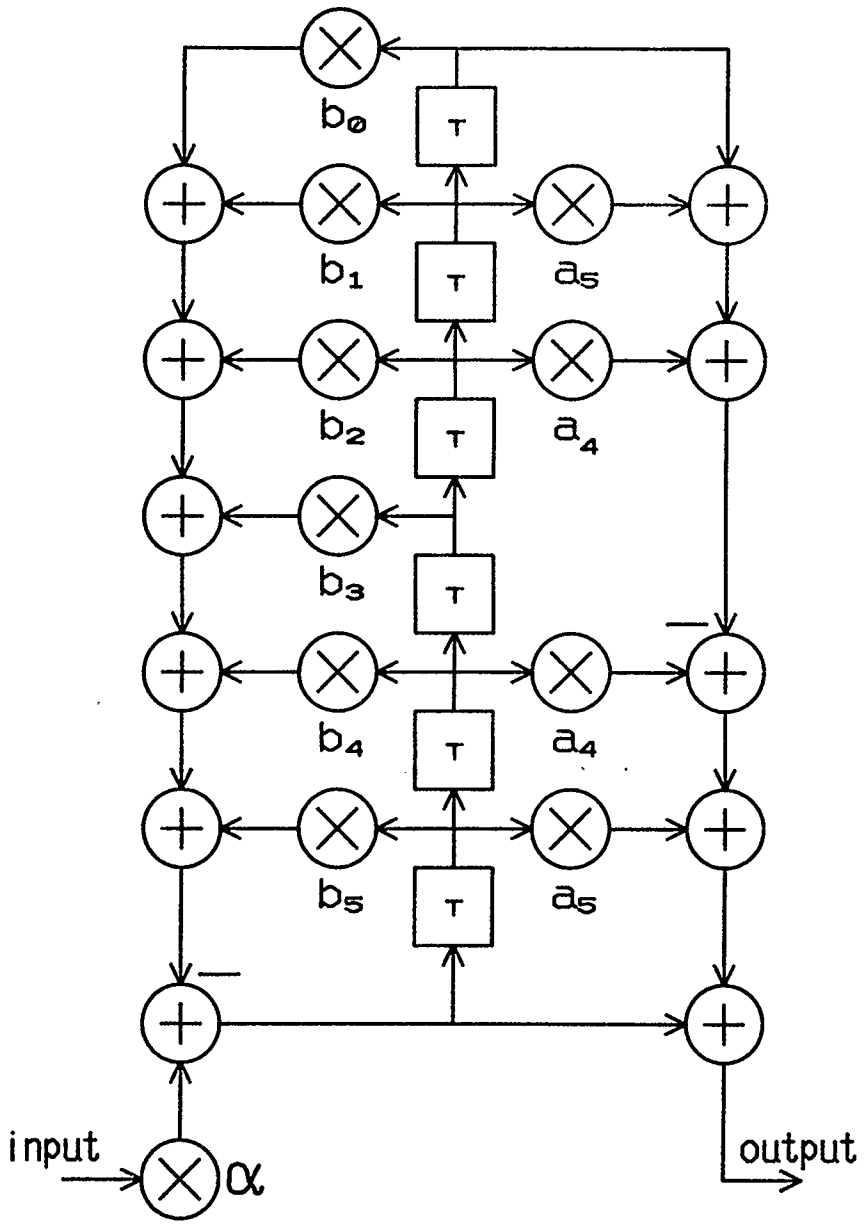


Figure 2.1 : Direct Form Type 1

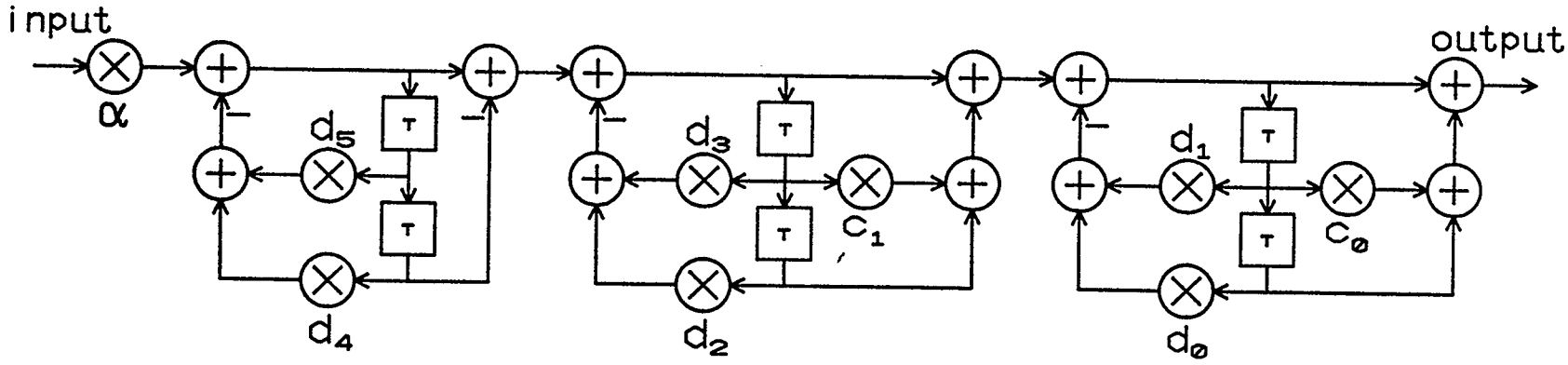


Figure 2.2 : Direct Form Type 2

2.2. Wave Digital Filters

The wave digital filter structure [7, 8, 9] is derived from an analog filter prototype by modelling wave propagation. The wave quantity may be voltage, current, or any linear combination of these, although it is often taken to be voltage. In the case of voltage waves, each prototype element can be considered as a one-port network [10] with a port resistance R and incident and reflected voltage waves A and B respectively given by

$$A = V + IR \tag{2.1}$$

$$B = V - IR, \tag{2.2}$$

where V and I are the port voltage and current respectively. The choice of R is arbitrary with the exception that two directly interconnected ports must have the same port resistance in order to maintain continuity of the wave quantities (equivalent to impedance matching). With appropriate choices for R, the wave implementation of the basic R, L, and C analog components can be very simple as shown in figure 2.3. The relationships between the wave quantities for series and parallel connections of elements can be determined by using the defining equations (2.1) and (2.2) and Kirchoff's voltage and current laws. Using these relationships, adapters can be constructed which allow series and parallel interconnections of the wave elements. Figures 2.4 through 2.7 show respectively the two-multiplier series adapter (S2), the one-multiplier series adapter (S1), the two-multiplier parallel adapter (P2) and the one-multiplier parallel adapter (P1). For the two multiplier adapters, the port resistances are independent and determine the values of the multipliers as follows:

$$M1 = \frac{-2R_1}{R_1+R_2+R_3} \tag{2.3}$$

$$M2 = \frac{-2R_2}{R_1+R_2+R_3} \tag{2.4}$$

$$M4 = \frac{G_1-(G_2+G_3)}{G_1+G_2+G_3} \tag{2.5}$$

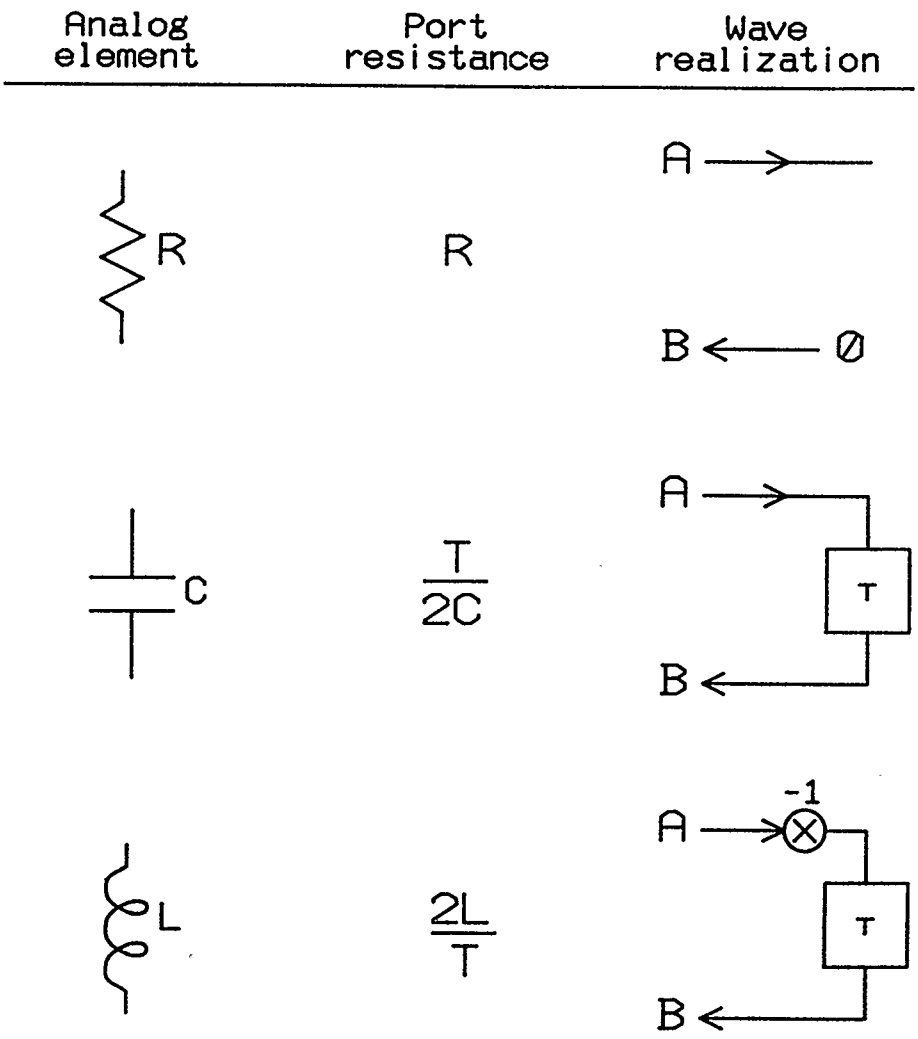
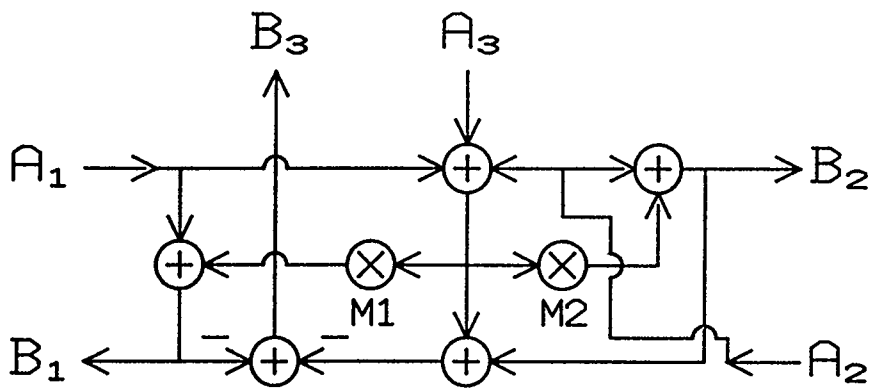
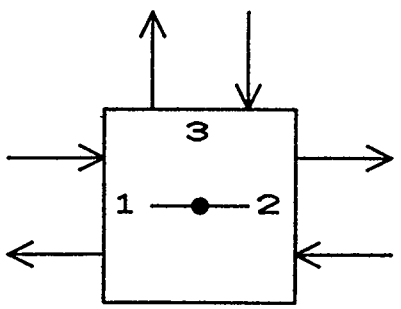


Figure 2.3 : Wave Filter Elements

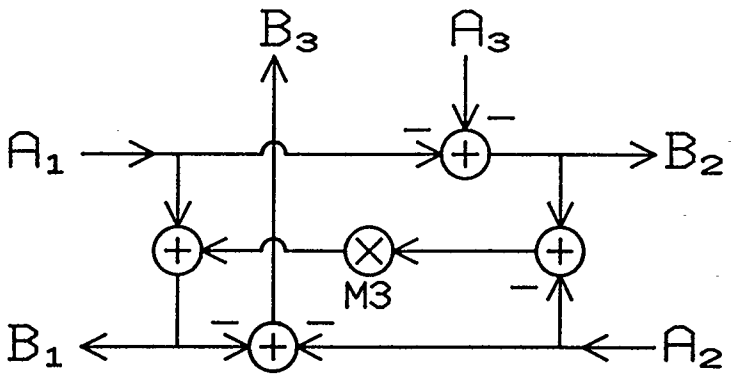


(a)

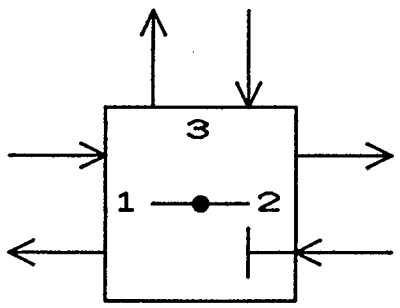


(b)

Figure 2.4 : (a) S2 Adapter
(b) S2 Symbol

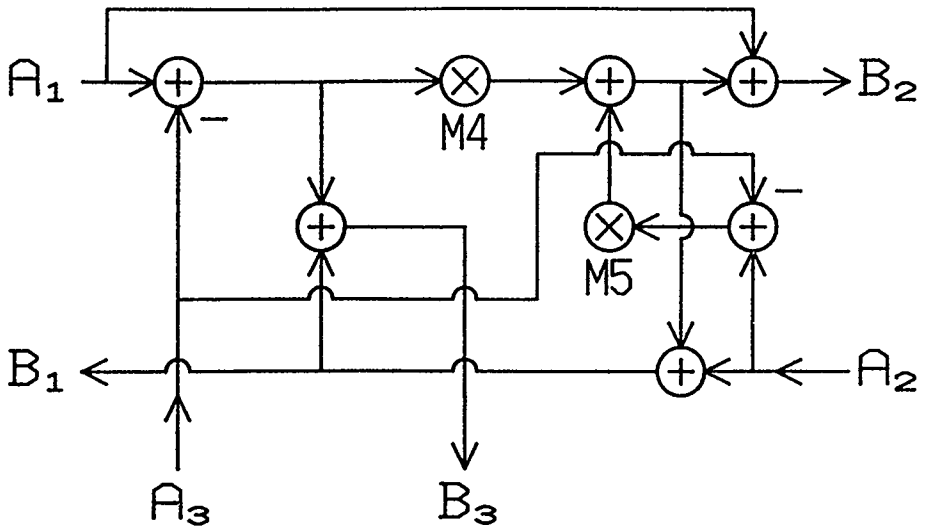


(a)

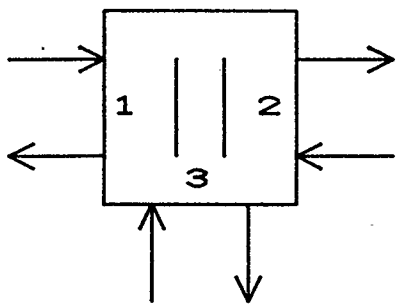


(b)

Figure 2.5 : (a) S1 Adapter
(b) S1 Symbol

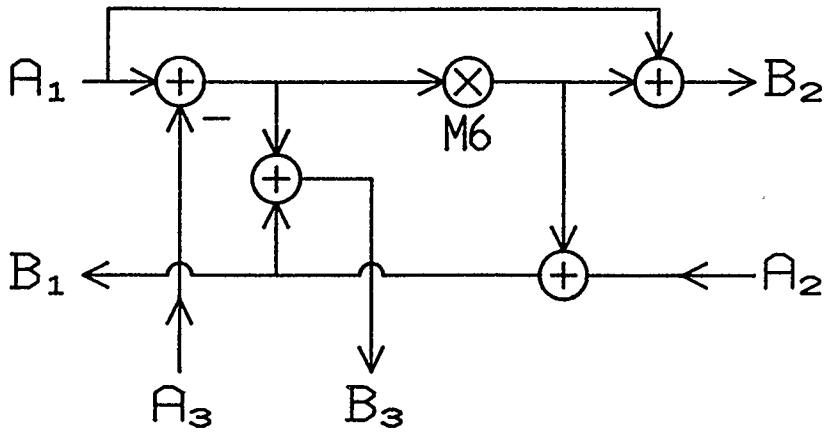


(a)

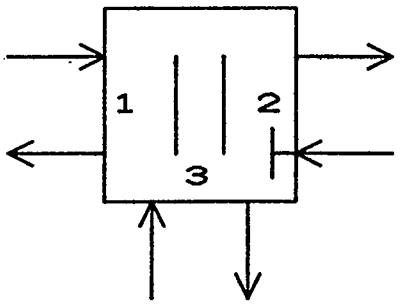


(b)

Figure 2.6 : (a) P2 Adapter
(b) P2 Symbol



(a)



(b)

Figure 2.7 : (a) P1 Adapter
(b) P1 Symbol

$$M5 = \frac{G_2-(G_1+G_3)}{G_1+G_2+G_3}, \tag{2.6}$$

where R_i and G_i are the resistance and conductance respectively of the i^{th} port. By forcing the port two resistance to be a function of the port one and three resistances, the one-multiplier adapters can be developed. For these adapters, the relationships between the multiplier values and port resistances or conductances are as follows:

$$R_2 = R_1+R_3 \tag{2.7}$$

$$M3 = \frac{R_1}{R_2} \tag{2.8}$$

$$G_2 = G_1+G_3 \tag{2.9}$$

$$M6 = \frac{-G_3}{G_2}. \tag{2.10}$$

Using the adapters and the wave elements shown in figure 2.3, the prototype filter of figure 1.2 can be realized as shown in figure 2.8. This implementation will be referred to as the WDF structure. The values of the adapter multipliers are determined from the analog component values using the equivalent port resistances of figure 2.3, equations (2.3) - (2.10), and the constraint that the port resistances of interconnected ports must be equal. In figure 2.8 the adapter multiplier coefficient symbols are shown in the upper corner(s) of the adapters, and the adapter interconnection ports are numbered 1 through 7. Shown in table 2.1 are the formulae for the adapter multiplier coefficients and interconnection port resistances (conductances) R_i (G_i) in terms of the prototype element values.

2.3. Lattice Digital Filters

Lattice digital filter structures [11], like direct form structures, implement arbitrary transfer functions. The two-multiplier form (each stage in the recursive section uses two multipliers) capable of implementing the transfer function (1.7) or (1.8) is shown in figure 2.9. This structure will be referred to as the LF1 structure. An alternative one-multiplier form is shown in figure 2.10. This structure will be referred to as the LF2 structure. The coefficients

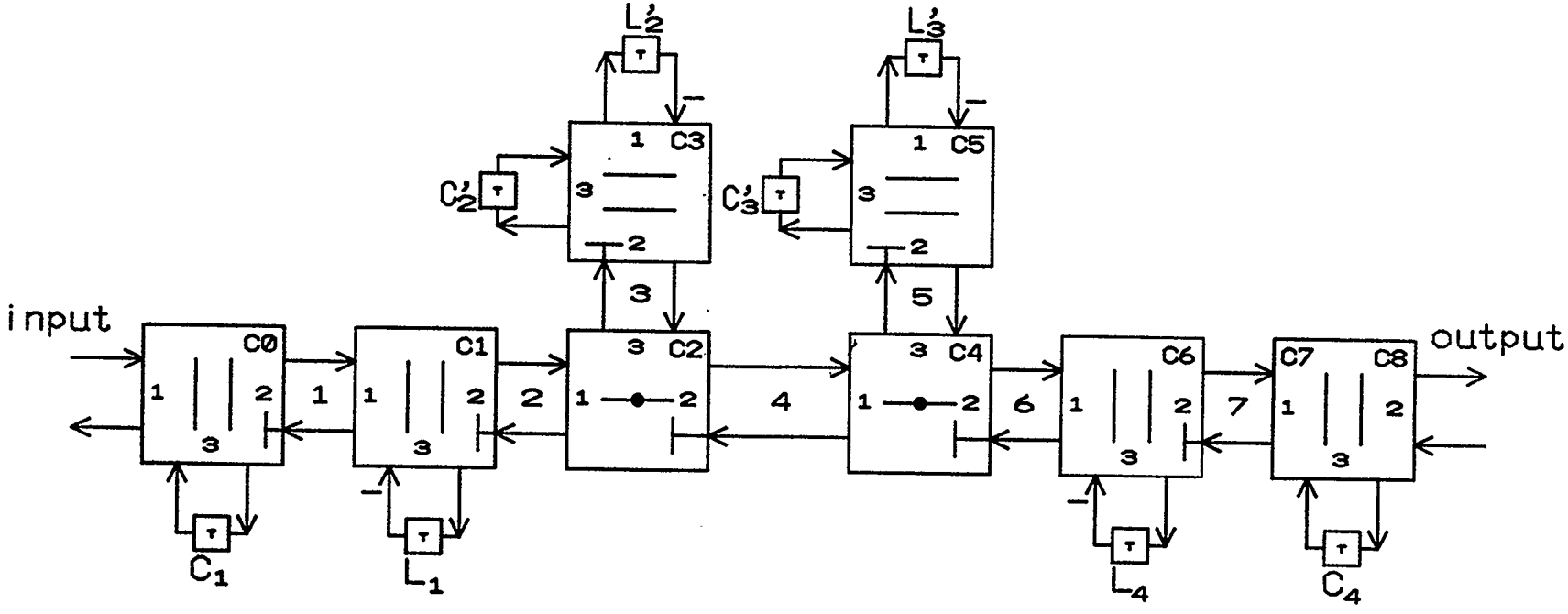


Figure 2.8 : Wave Digital Filter

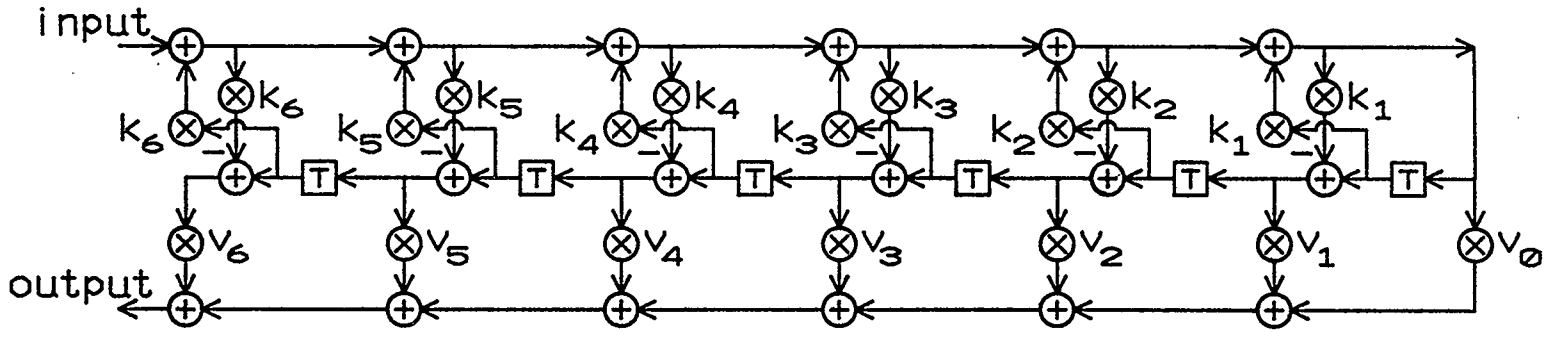


Figure 2.9 : Lattice Filter Type 1

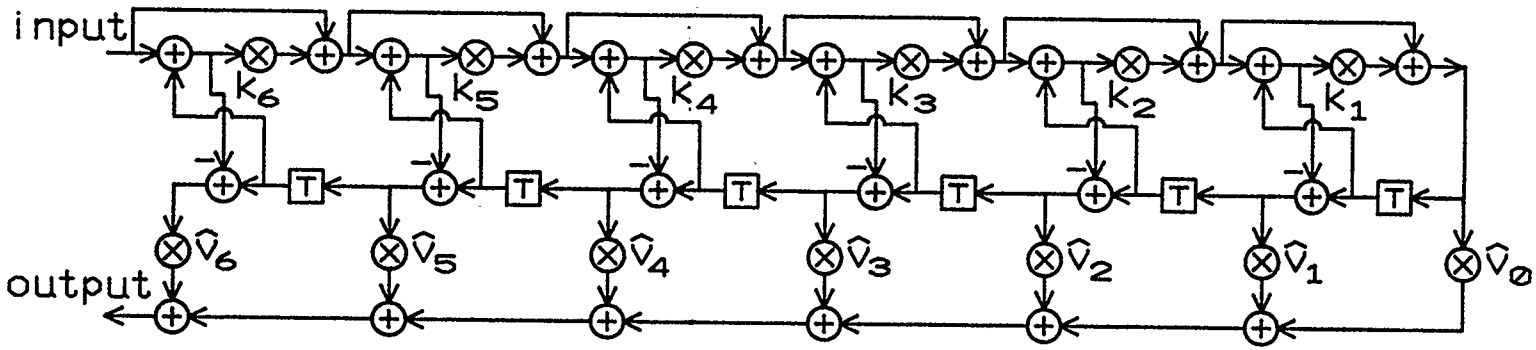


Figure 2.10 : Lattice Filter Type 2

$$C0 = \frac{-2C_1}{2C_1 + T} \qquad G_1 = \frac{1}{1 + C0}$$

$$C1 = \frac{-T}{T + 2G_1L_1} \qquad R_2 = \frac{1 + C1}{G_1}$$

$$C3 = \frac{-4C_2'L_2'}{T^2 + 4C_2'L_2'} \qquad G_3 = \frac{C2}{R_2(1 - C2)}$$

$$C2 = \frac{2C_2'R_2}{2C_2' - TC3} \qquad R_4 = \frac{R_2}{C2}$$

$$C5 = \frac{-4C_3'L_3'}{T^2 + 4C_3'L_3'} \qquad G_5 = \frac{C4}{R_4(1 - C4)}$$

$$C4 = \frac{2C_3'R_4}{2C_3' - TC5} \qquad G_6 = \frac{C4}{R_4}$$

$$C6 = \frac{-T}{T + 2G_6L_4} \qquad G_7 = \frac{G_6}{1 + C6}$$

$$C7 = \frac{G_7T - (T + 2C_4)}{G_7T + T + 2C_4}$$

$$C8 = \frac{T - (G_7T + 2C_4)}{T + G_7T + 2C_4}$$

Table 2.1 : WDF Multiplier Coefficients

k_i, v_i, and $\hat{v}_i$ can be determined from the desired transfer function using a straightforward recursion. In general, if $H(z)$ is the desired n^{th} order transfer function given by

$$H(z) = \frac{N(z)}{D(z)}, \tag{2.11}$$

where $N(z)$ and $D(z)$ are respectively the n^{th} order numerator and denominator polynomials

given by

$$N(z) = n_n z^n + n_{n-1} z^{n-1} + n_{n-2} z^{n-2} + \cdots + n_0 \tag{2.12}$$

$$D(z) = z^n + d_{n-1} z^{n-1} + d_{n-2} z^{n-2} + \cdots + d_0, \tag{2.13}$$

the following recursion defines the lattice filter coefficients:

$$A_i(z) = a_i z^i + a_{i-1} z^{i-1} + a_{i-2} z^{i-2} + \cdots + a_0 \tag{2.14a}$$

$$k_i = -a_0 \tag{2.14b}$$

$$B_i(z) = a_0 z^i + a_1 z^{i-1} + a_2 z^{i-2} + \cdots + a_i \tag{2.14c}$$

$$A_{i-1}(z) = \frac{A_i(z) + k_i B_i(z)}{z(1 - k_i^2)} \tag{2.14d}$$

$$C_i(z) = c_i z^i + c_{i-1} z^{i-1} + c_{i-2} z^{i-2} + \cdots + c_0 \tag{2.14e}$$

$$v_i = c_{n-i} \tag{2.14f}$$

$$s_i = \begin{cases} 1 & i=n \\ \prod_{j=i+1}^{n} (1 + k_j) & \text{otherwise} \end{cases} \tag{2.14g}$$

$$\hat{v}_i = \frac{v_i}{s_i} \tag{2.14h}$$

$$C_{i-1}(z) = C_i(z) - v_i B_i(z) z^{n-i}. \tag{2.14i}$$

In this recursion, i runs from n to 0, $A_n(z) = D(z)$, and $C_n(z) = N(z)$.

2.4. LDI Filters

The previous design techniques make use of the bilinear transformation either explicitly to obtain the transfer function (DF1, DF2, LF1, LF2) or implicitly in the development of the digital element realizations (WDF). It is possible to use the alternative transformation [12]

$$s = \frac{z^{1/2} - z^{-1/2}}{T} \tag{2.15}$$

to define the digital elements. Transformation (2.15) leads to a digital building block which is called a "lossless discrete integrator", and hence it is called the LDI transformation. The LDI

transformation does not directly lead to stable digital filters, and does not preserve the desirable mapping of the entire $s = j\omega$ axis onto the unit circle in the z-domain as does the bilinear transformation. It does however yield digital elements with a very simple form, and can avoid the delay-free loop problem [13] which is encountered when analog voltage-current signal flow graphs are converted directly to a digital structure using the bilinear transformation. Using a technique developed by Turner and Ramesh [14, 15], stable filters with exact bilinear transfer functions can be implemented using LDI digital elements. The technique relies on the relationship between impedances implemented with LDI elements and equivalent bilinear transformed impedances. This relationship is shown in figure 2.11. In this figure, the third column shows the bilinear transformed element which corresponds to the LDI implementation after impedance scaling by $(z+1)/2$. To digitally realize the transfer function of an analog prototype filter, the prototype is first manipulated so that the negative capacitors of figure 2.11 are implicitly present in parallel with each inductor and resistor of the prototype, but are cancelled out by additional parallel positive capacitors. This results in a prototype with precompensated element values, which when implemented with LDI immitances will yield the desired bilinear transfer function. The prototype filter is then converted into a voltage-current signal flow graph which is in turn manipulated by the introduction of controlled sources so that it contains only resistive and integrator branches. This signal flow graph is then implemented directly with the LDI elements of figure 2.11.

The LDI implementation of the prototype filter of figure 1.2 is shown in figure 2.12. The non-recursive input section implements a voltage source transformation which is required as a result of the precompensation procedure. Table 2.2 shows the formulae for the precompensated component values Z_i^P in terms of the original component values of figure 1.2, while table 2.3 shows the formulae for the multiplier coefficients of figure 2.12 in terms of the precompensated component values.

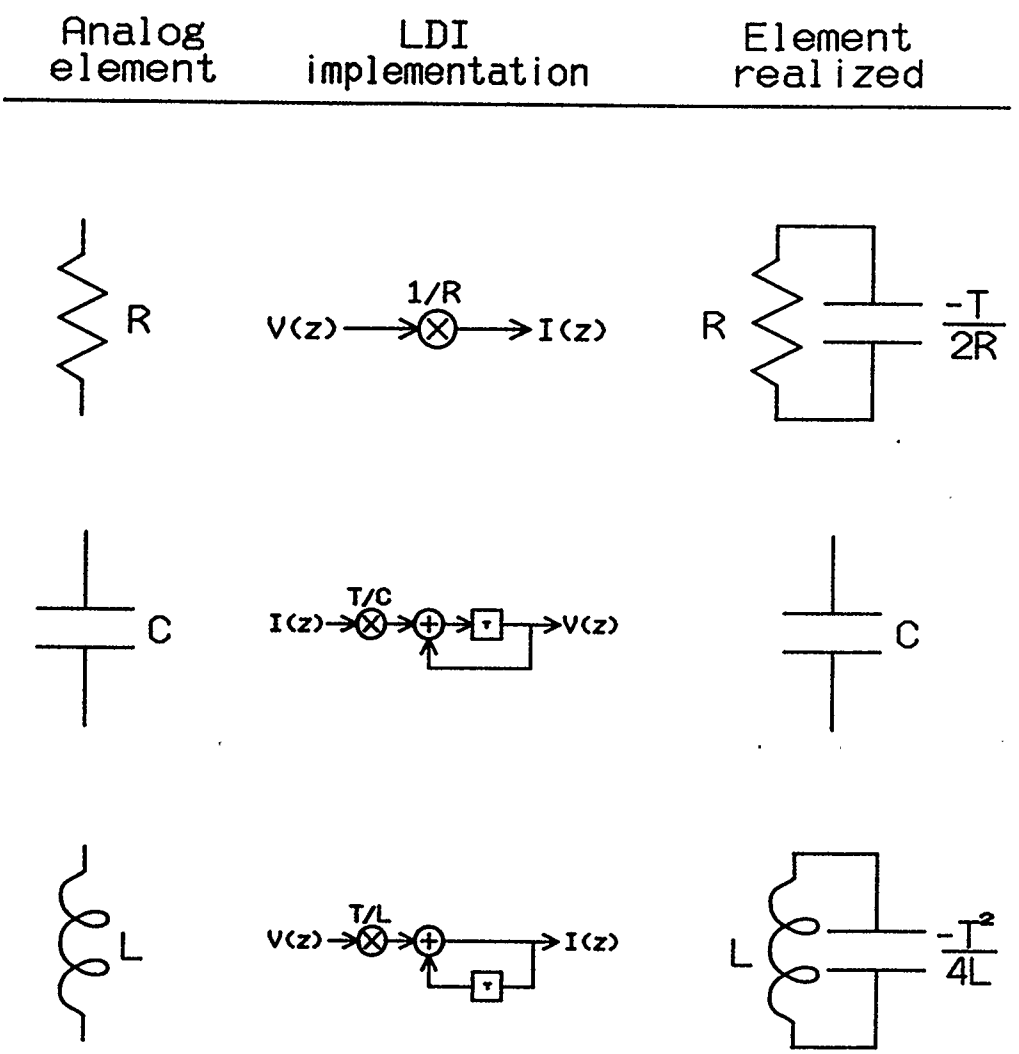


Figure 2.11 : LDI Filter Elements

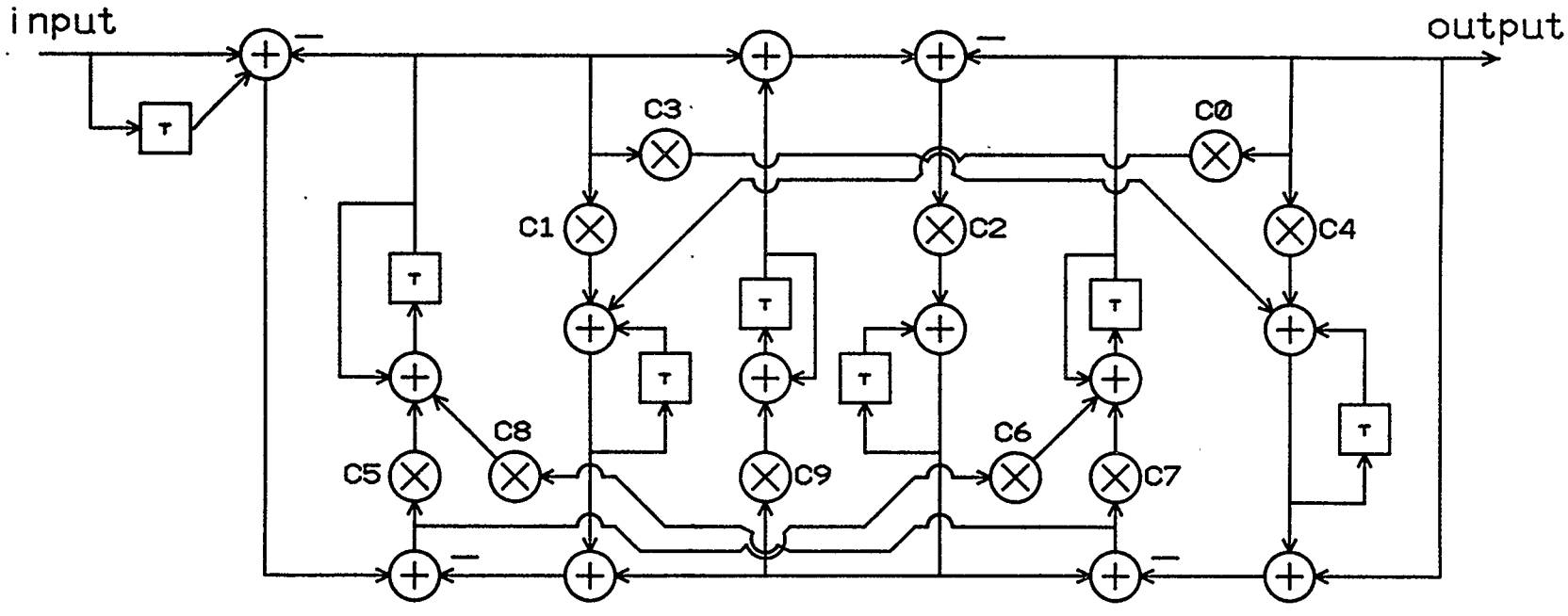


Figure 2.12 : LDI Filter

$$C_1^P = C_1 + \frac{T^2}{4L_1} + \frac{T}{2}$$

$$L_1^P = L_1$$

$$C_2^P = \frac{4L_2C_2^2}{4L_2C_2 + T^2}$$

$$L_2^P = L_2\left(1 + \frac{T^2}{4L_2C_2}\right)^2$$

$$C_3^P = C_3 + \frac{T^2}{4L_3} + \frac{T^2C_2}{4L_2C_2 + T^2}$$

$$L_3^P = L_3$$

Table 2.2 : LDI Precompensated Component Values

$$C0 = C3 = \frac{-T}{L_3^P}$$

$$C1 = C4 = \frac{T(L_1^P + L_3^P)}{L_1^P L_3^P}$$

$$C2 = \frac{T}{L_2^P}$$

$$C5 = C7 = \frac{T(C_1^P + C_3^P)}{C_1^{P2} + 2C_1^P C_3^P}$$

$$C6 = C8 = \frac{TC_3^P}{C_1^P(C_1^P + 2C_3^P)}$$

$$C9 = \frac{-T}{C_2^P}$$

Table 2.3 : LDI Multiplier Coefficients

CHAPTER 3

NON-IDEAL EFFECTS

Under ideal conditions each of the structures presented in the previous chapter is capable of exactly implementing the desired prototype transfer function (1.7). Unfortunately the condition of ideality is not met in practice due to the finite precision arithmetic which must be used, and non-ideal effects influence the capabilities and attractiveness (in terms of hardware or computational cost) of the structures. In this chapter the non-idealities which are encountered in the practical implementation of a digital filter are presented and their effects illustrated. Practical strategies for reduction of multiplier coefficient quantization effects and elimination of overflow effects are also presented.

3.1. Multiplier Coefficient Quantization

Each structure presented in chapter two makes use of multipliers which have coefficients derived either from a set of design equations such as (2.3) - (2.10) or directly from the desired transfer function. In general these coefficients will be non-integral and possibly even irrational. In a real digital filter, these multiplier coefficients must be represented with a finite number of digits with a fixed base (which will be assumed here to be two). As a result, the actual coefficient values implemented are selected from a finite set, and in general will not match the ideal (infinite precision) values exactly. Because floating-point arithmetic is more costly to implement than fixed point arithmetic (in terms of hardware and complexity), it is desirable to implement digital filters in fixed point arithmetic wherever possible. Multiplications are thus assumed to be fixed-point binary, and realizable coefficient values are fractions in which the denominator is an integral power of two. The result of using these quantized coefficients in an actual filter implementation is that the transfer function realized

does not exactly match the ideal transfer function.

The filter shown in figure 3.1 has been designed with coefficients which can be exactly implemented by binary fixed-point coefficients (this is not the case in general) having seven fractional bits. With the coefficients as shown, the filter poles are inside the unit circle in the z-domain, so the filter is stable [9]. To illustrate the effect of multiplier coefficient quantization, the magnitude transfer function of this filter is shown in figure 3.2 for coefficients with seven, six, and five fractional bits. In reducing the number of coefficient bits, two's complement truncation (described in detail in chapter four) was used. Note that the originally stable filter becomes unstable when five bit (fractional) coefficients are used. The transfer function of this filter is extremely sensitive to variations in the multiplier values. For a practical digital filter, this high sensitivity is undesirable because it requires that a large number of coefficient bits be used (increasing the hardware or computation cost) relative to a structure which has lower sensitivity.

One approach to designing a digital filter with low coefficient sensitivity is to use a doubly-terminated analog LC filter as the prototype. These filters have well-known low sensitivity properties [16, 17] with respect to component values. The low sensitivity is a result of the fact that at points of zero insertion loss in the passband (where maximum power is transferred to the output resistance) the filter attenuation is completely insensitive to differential changes in the L and C component values. The prototype of figure 1.2 has three such points of zero sensitivity. In order for a digital filter derived from such a prototype to share this first order zero in sensitivity, differential changes in the multiplier values must correspond to differential changes in the prototype element values. This can be shown to be the case for the WDF and LDI structures by reversing the design procedure to yield expressions for the analog prototype component values in terms of the multiplier coefficients. These expressions are shown in table 3.1 and table 3.2 respectively. In table 3.1, G_i and R_i are as defined in table 2.1. Note that for the WDF structure the output resistance R_{out}

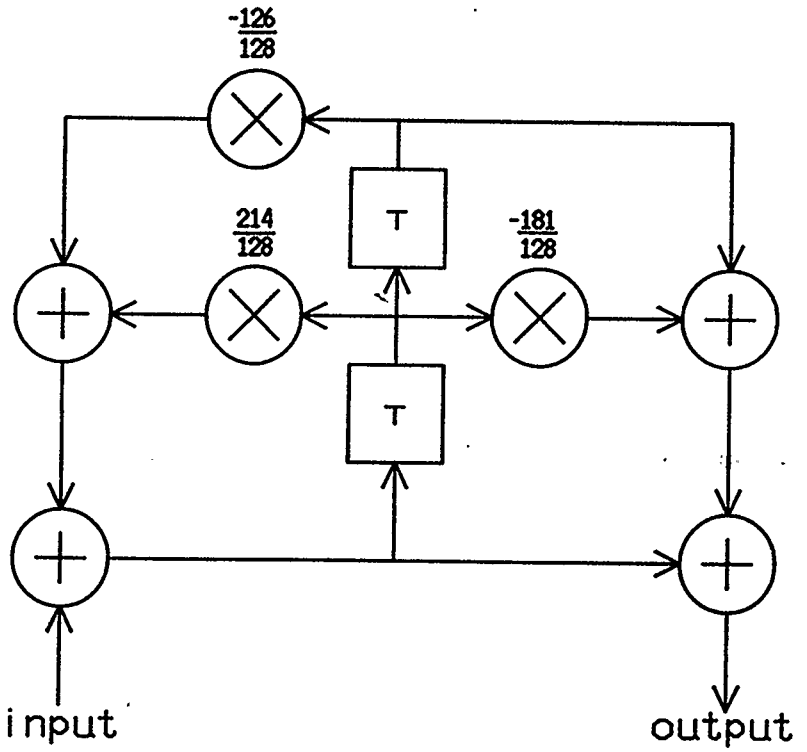


Figure 3.1 : Second Order Direct Form

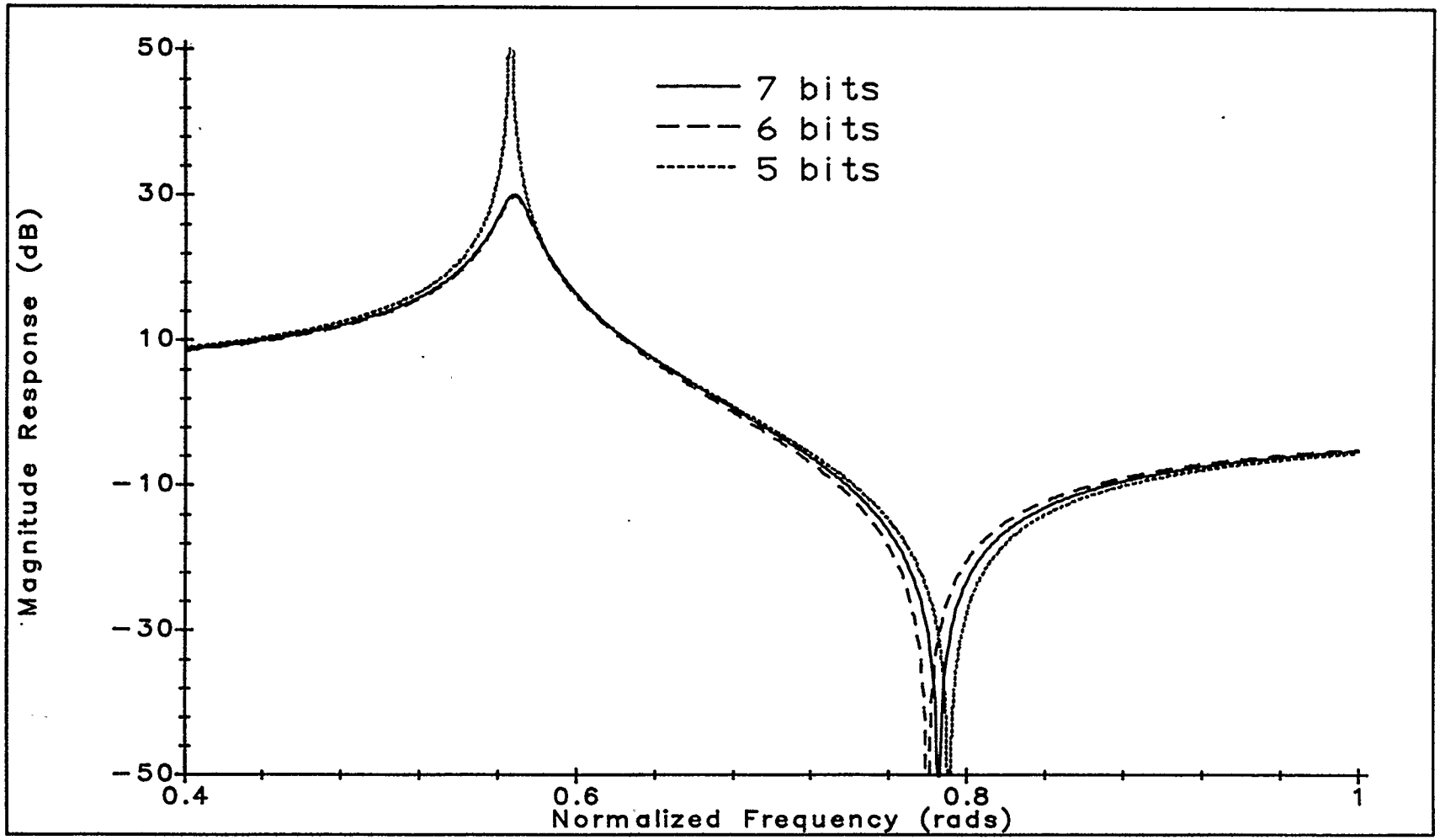


Figure 3.2 : Effect of Multiplier Coefficient Quantization

becomes a function of the coefficient values, and for the LDI filter it is assumed that equal infinite precision coefficients remain equal when quantized. Inspection of the expressions for the prototype element values reveals that they are differentiable with respect to all multiplier coefficients provided that

$$C_1 = \frac{-TC0}{2(C0+1)} \qquad L_1 = \frac{-T(1+C1)}{2G_1C1}$$

$$C_2' = \frac{-G_3TC3}{2} \qquad L_2' = \frac{T}{2G_3(1+C3)}$$

$$C_3' = \frac{-G_5TC5}{2} \qquad L_3' = \frac{T}{2G_5(1+C5)}$$

$$C_4 = \frac{-TG_7(C7+C8)}{2(1+C7)} \qquad L_4 = \frac{-T(1+C6)}{2G_6C6}$$

$$R_{out} = \frac{1+C7}{G_7(1+C8)}$$

Table 3.1 : Element Values From WDF Coefficients

$$C_1 = C_4 = T\left\{\frac{1}{C5+C6} - \frac{C1+C3}{4} - \frac{1}{2}\right\}$$

$$L_1 = L_4 = \frac{T}{C1+C3}$$

$$C_2 = \frac{-4T}{C9(4+C9C2)}$$

$$L_2 = \frac{T(4+C9C2)^2}{4C2}$$

$$C_3 = T\left\{\frac{C6}{C5^2-C6^2} + \frac{C3}{4} - \frac{C2}{4+C9C2}\right\}$$

$$L_3 = \frac{-T}{C3}$$

Table 3.2 : Element Values From LDI Coefficients

$$Ci \neq -1, \qquad i = 0, 1, 3, 5, 6, 7, 8 \tag{3.1a}$$

$$Ci \neq 1, \qquad i = 2, 4 \tag{3.1b}$$

$$C1 \times C6 \neq 0 \tag{3.1c}$$

for the WDF structure and

$$|C5| \neq |C6| \tag{3.2a}$$

$$C2 \times C9 \neq -4 \tag{3.2b}$$

$$C1 \neq -C3 \tag{3.2c}$$

$$C2 \times C3 \times C9 \neq 0 \tag{3.2d}$$

for the LDI structure. Violation of any of these conditions represents a singular point at which the topology of the analog prototype changes. For prototype filters with finite non-zero prototype element values conditions (3.1) and (3.2) are satisfied and the prototype element values are differentiable functions of the multiplier coefficients in some neighborhood of the ideal coefficient values. Consequently, a differential change of the multiplier coefficients corresponds to a differential change in the prototype element values, and the low sensitivity property of the prototype filter is retained by the WDF and LDI structures.

If a similar analysis is attempted for the direct form structures, the expression of the prototype element values as functions of the direct form multiplier coefficients requires the simultaneous solution of a set of non-linear equations. No solution of this problem has been attempted due to its complexity, however it is possible to gain some insight by examining the much simpler prototype of figure 3.3. For this circuit, the transfer function $H(s)$ is given by

$$H(s) = \frac{a_2 s^2 + a_0}{s^2 + b_1 s + b_0}, \tag{3.3}$$

where the coefficients a_2, a_0, b_1, and b_0 are related to the prototype element values as follows:

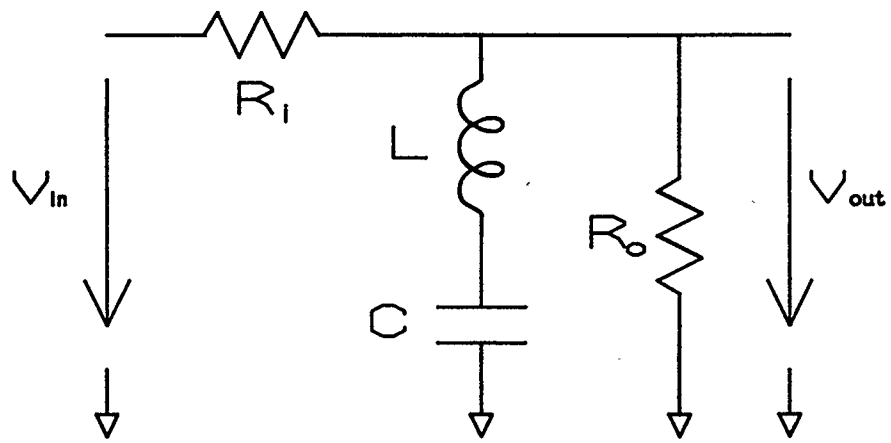


Figure 3.3 : Simple LCR Prototype

$$a_2 = \frac{R_o}{R_o + R_i} \tag{3.4a}$$

$$a_0 = \frac{a_2}{LC} \tag{3.4b}$$

$$b_1 = \frac{R_o a_2}{L} \tag{3.4c}$$

$$b_0 = \frac{1}{LC} . \tag{3.4d}$$

From these expressions it can be seen that

$$a_0 = b_0 a_2 \tag{3.5}$$

must hold in order for the transfer function (3.3) to represent the circuit of figure 3.3. Using the bilinear transformation the equivalent digital transfer function $H(z)$ is found to be

$$H(z) = \alpha \left\{ \frac{z^2 + c_1 z + 1}{z^2 + d_1 z + d_0} \right\}, \tag{3.6}$$

where α, c_1 , b_1, and b_0 are determined by a_2, a_0, b_1, b_0 , and the sample period T. By lengthy manipulation of the non-linear relationships between the digital and analog transfer function coefficients, it is possible to show that (3.5) implies

$$c_1 = \frac{2d_1}{1 + d_0} . \tag{3.7}$$

Since c_1, d_1, and d_0 are the ideal values of three of the multiplier coefficients in a direct form implementation of the circuit of figure 3.3, a differential change in these coefficients must obey (3.7) in order to correspond to a differential change in the analog element values. It is quite possible for a differential change of c_1, d_1, and d_0 to result in the violation of (3.7), which means that *a differential change in the multiplier coefficients of the direct form implementation may correspond to a change in the topology or element types of the prototype.* Although this property has been shown only for the circuit of figure 3.3, since there was nothing particularly unique about this circuit it is likely that the coefficients of any transfer function derived using the bilinear transformation from an analog LCR prototype will need to

obey non-linear constraints such as (3.7), and a direct form filter which implements such a transfer function will share this property. Since the argument for the low sensitivity of the analog prototype requires that neither the type of elements nor the topology varies, this implies that the DF1 and DF2 structures are unlikely to preserve the low sensitivity property of the analog prototype. The lattice filter coefficients are derived from the direct form coefficients, hence low sensitivity is also not expected in the LF1 and LF2 structures.

Simulations of the different digital structures support the conclusion that the WDF and LDI structures are less sensitive to coefficient variations than the direct form and lattice structures, and other work [18] has determined by different methods that this is the case for the WDF structure. Figure 3.4 shows typical passband magnitude transfer functions for the WDF, LDI, DF2, and LF2 structures with coefficients quantized to 14 fractional bits. As would be expected from the preceding analysis the LDI and WDF structure transfer functions depart from the ideal transfer function less than the DF2 and LF2 transfer functions. On the scale of figure 3.4 the ideal transfer function is virtually indistinguishable from the LDI transfer function. Note that the LDI structure retains three points of true zero attenuation. This is expected as a result of the fact that a change in the coefficient values does not correspond to a change in the termination resistances which alone determine the peak gain. The most apparent deviation in the LDI transfer function is a departure from the equi-ripple passband attenuation which is characteristic of the prototype of figure 1.2. In the low end of the passband, the attenuation slightly exceeds the ideal attenuation of 0.011 dB, while in the high end of the passband, the maximum attenuation is slightly less than the ideal value. The WDF structure departs from the ideal response with a frequency-independent attenuation of approximately 0.0025 dB, while the equi-ripple characteristic is essentially undisturbed. This is consistent with the change in effective output resistance which can accompany a change in coefficient values of the WDF structure. For both the direct form and lattice structures, the three local minima of attenuation are unequal, and there is a large maximum deviation from the ideal transfer function. This change is similar to what would happen to the ideal

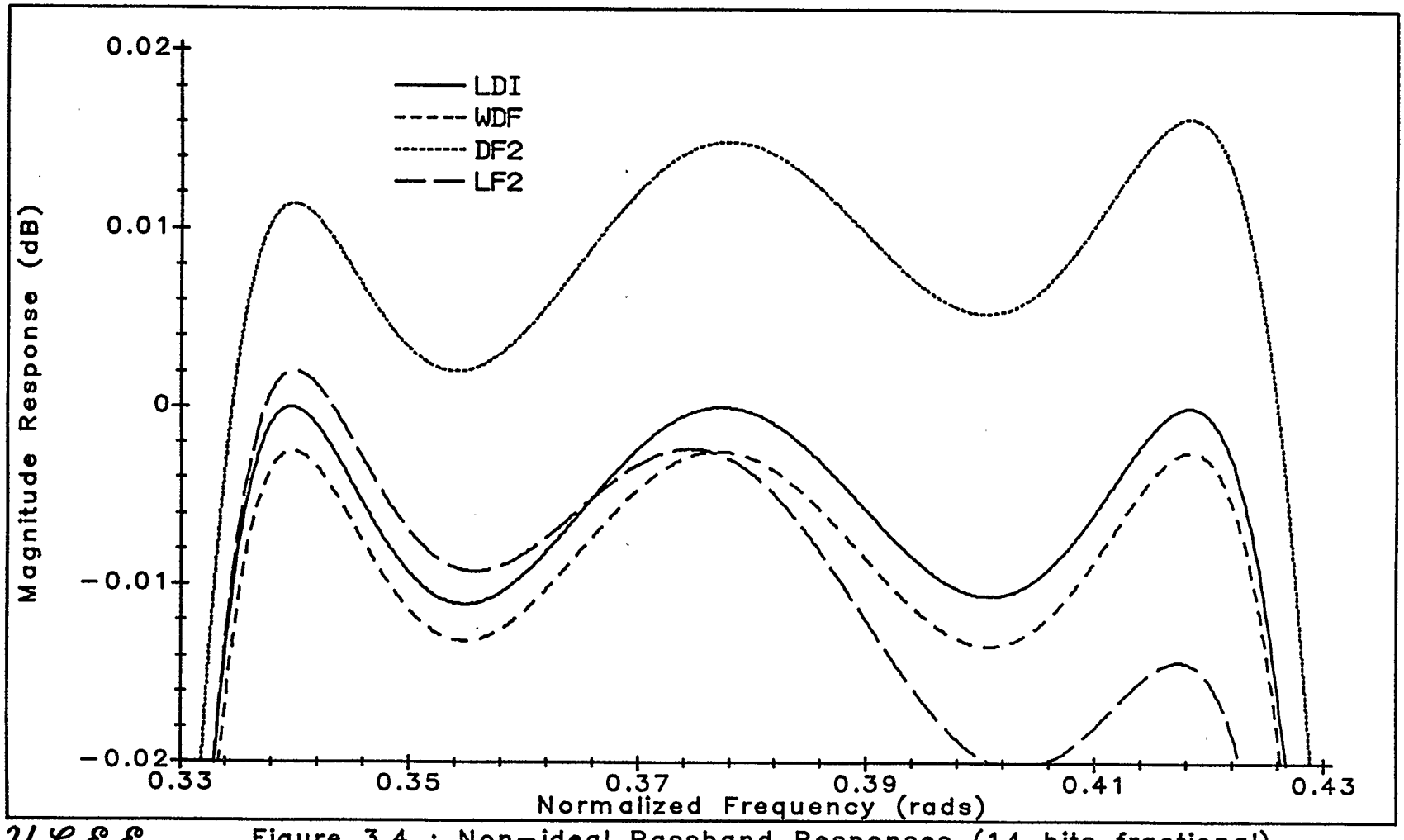


Figure 3.4 : Non−ideal Passband Responses (14 bits fractional)

transfer function if the output resistance was changed and dissipative elements (e.g. resistors) were added to the originally lossless LC section. The appearance of dissipation represents something other than a simple change of prototype element values, so this behavior supports the conjecture that differential changes in the direct form and lattice coefficient values can correspond to changes in the element types or topology of the prototype. For the DF1 and LF1 structures with 14-bit coefficients, the passband responses deviate from the ideal so much that they cannot be plotted on the same scale as in figure 3.4.

In practice, regardless of the structure chosen, the multiplier coefficients must be determined so that the realized transfer function meets some desired specification. Some researchers have successfully used optimization techniques to arrive at coefficient values which result in an acceptable transfer function with a minimum total number of coefficient bits [4]. While this is a useful technique for a filter which is to have fixed coefficients, it is less useful for a programmable filter in which the exact coefficient values are not known. In addition, these optimization techniques are time-consuming and complicate the filter design process. An alternative strategy is simply to determine by simulation the minimum number of coefficient bits required to implement a transfer function to within a desired accuracy. In either case, it is necessary to establish what amount of deviation from the ideal transfer function is acceptable. Orchard has pointed out [17] that it is less difficult to compensate for a frequency independent deviation (such as in figure 3.4 for the WDF structure) than it is to compensate for a frequency dependent deviation such as increased passband ripple because in the former case, only a single compensating gain stage is required. In addition, a passband attenuation which is less than the ideal attenuation will usually still meet the required filter specification. With this in mind, a useful criterion for acceptability of a non-ideal transfer function is Crochiere's "relative passband error" [19] given by

$$\text{relative error } (\%) = \begin{cases} \dfrac{(A_{max}-A_{min})-\Delta}{\Delta}\times 100, & A_{max}-A_{min} > \Delta \\ 0, & \text{otherwise} \end{cases} . \qquad (3.8)$$

Here, A_{max} and A_{min} are respectively the maximum and minimum passband attenuations (in dB) of the non-ideal transfer function and Δ is the difference between the maximum and minimum passband attenuations (in dB) of the ideal transfer function. Using this criterion, if the ideal transfer function has the maximum acceptable passband ripple, the required number of coefficient bits is the minimum number which yields a relative passband error of zero.

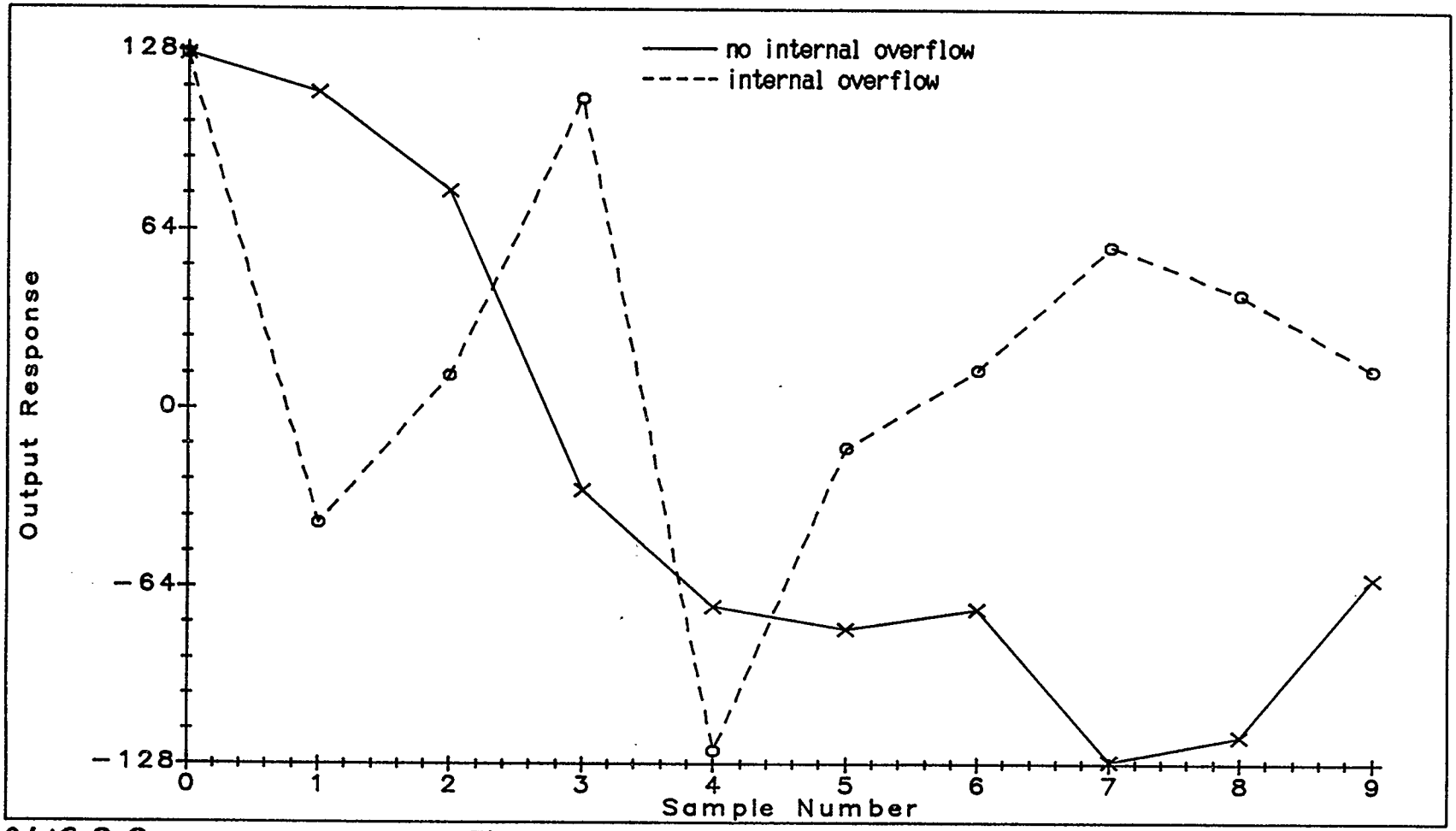
3.2. Overflow

Just as the multiplier coefficients of a real digital filter must be represented in a finite number of bits, so must the signal values. When a signal value occurs which is too large (in magnitude) to be represented in the number of bits available, a condition of overflow arises. Since overflow results in the loss of information in the most significant end of the signal, it can lead to large amplitude errors (with respect to the signal values encountered in the corresponding ideal filter). To illustrate the effect of overflow, figure 3.5 shows two responses of the filter in figure 3.1 to the driving sequence $u(k)$ given by:

$$u(k) = \{127,80,0,-127,-127,-40,60,40,20,0\} \quad k = 0,1,2,\cdots,9. \qquad (3.9)$$

The solid response is for the filter implemented with 20-bit (integer) signal values where no signal overflow occurs. The dashed response is for the filter implemented with 8-bit (integer) signal values. In this case even though the ideal output values are all representable in 8-bits, internal overflow leads to large deviations from the ideal output.

Due to the large magnitude of the errors introduced by overflow, it must be prevented during the normal operation of an actual digital filter. In order to eliminate overflow errors, careful consideration must be given to their origins. Since floating-point systems are



UCEE

Figure 3.5 : Effect of Internal Overflow

prohibitive in terms of the amount of hardware required for their implementation (for the purposes of this research), it is assumed here that all signals are represented in a fixed-point binary format. In a filter implemented using delays, two-input adders, and fixed-point multipliers, overflow can occur in two ways. Firstly, the addition of two signal values can result in a sum which requires one more bit to represent than the largest (magnitude) operand. If the largest operand already requires the full available word length for its expression, this can lead to overflow in the sum if no extra bits are available for its expression. Secondly, the multiplication operation can always be considered equivalent to multiplying an m-bit integer signal value with an n-bit integer coefficient to form an $m+n$-bit product with an implicit binary point at some bit position. If the signal operand requires the full available word length for its expression, then the multiplication can lead to overflow if extra bits are not made available for the resulting product. One way to avoid overflow is thus to provide the potentially necessary extra bits at the output of each addition or multiplication operation. While this is possible for a non-recursive filter, it is not possible for a recursive structure. In the latter case, there exists some path from the output of an adder or multiplier back to its input. Attempting to accommodate all possible bit growth in such a loop would require an infinite number of signal bits. Instead, at some point in the loop, the increased number of bits must be truncated down to the number of bits in the input to the "first" operation in the loop (of course the choice of which operation is the "first" is arbitrary). At the point of truncation, overflow is again possible.

In a digital filter, the result of an operation which results in overflow may be passed, possibly through a series of delays and/or truncations, to the input of an adder, to the input of a multiplier, or to the filter output. In a properly designed filter enough signal word bits are present at the output to represent the signal levels expected under normal operating conditions, thus if no interior overflow occurs, overflow at the output is not a problem. The effect of overflow in the interior of a digital filter depends strongly on the type of arithmetic used in the filter. Figure 3.6 compares the addition of six values in a three bit sign-and-magnitude

Sign-and-Magnitude

data		result		interpretation
0 01		1		1
+ 0 10	= 0 11	3		+2
+ 0 11	= 0 10	2	(overflow)	+3
+ 0 10	= 0 00	0	(overflow)	+2
+ 1 11	= 1 11	-3		-3
+ 1 10	= 1 01	-1	(overflow)	-2
= 1 01				= -1

Two's Complement

data		result		interpretation
001		1		1
+ 010	= 011	3		+2
+ 011	= 110	-2	(overflow)	+3
+ 010	= 000	0		+2
+ 101	= 101	-3		-3
+ 110	= 011	3	(overflow)	-2
= 011				= 3

Figure 3.6 : Example Additions

arithmetic system with the same addition in a three bit two's complement [20] arithmetic system. Note that truncation to three bits results in intermediate overflows in both systems, but that only the two's complement system determines the correct result, namely:

$$1 + 2 + 3 + 2 - 3 - 2 = 3, \tag{3.10}$$

which is representable without overflow in both arithmetic systems. It is a property of two's complement arithmetic that the addition of values which would ideally yield a result representable without overflow will yield the correct result irrespective of any intermediate overflows [21]. The only requirement for this to hold is that the result must be truncated to the smallest number of bits in which any addend is represented. Consequently, overflow at the input to an adder is not a problem in a two's complement system since it can tolerate an arbitrary number of intermediate addition overflows, but it can lead to errors in a sign-and-magnitude system.

In both two's complement and sign-and-magnitude systems, overflow at the input to a multiplier will result in large magnitude errors as shown in figure 3.7. In this figure asterisks represent bits lost due to the assumed overflow of the *m*-bit input data, and the *n*-bit multiplier coefficient is assumed to be non-integral with an implied decimal point between the second and third bits (for definiteness). The information lost due to overflow in the input data propagates into the most significant *n* bits of the result of the multiplication. In a sign-and-magnitude system, this error can never be tolerated, since any subsequent operation which relies on the result of the multiplication (including the output operation) may be in error. In a two's complement system, the only way that this error can be tolerated is in the unlikely event that the result is used as one input to a sequence of additions in which the sum is both free from overflow and truncated below the propagated error from the multiplication. For all multipliers which have a signal path from their output back to their input this condition will not be met because the result of the multiplication will somewhere have to be truncated to exactly the same format as the input. This results in inclusion of overflow error in the trun-

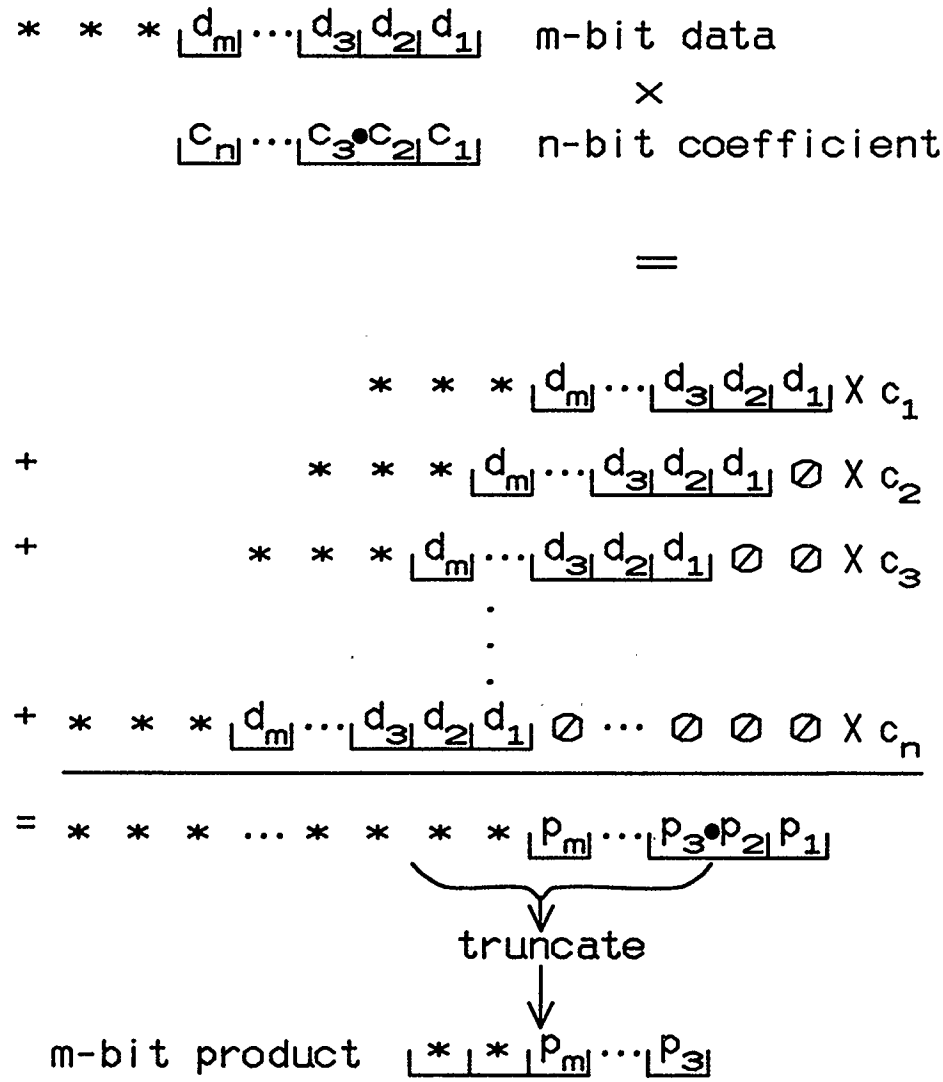


Figure 3.7 : Overflow Propagation

cated product whenever the multiplier coefficient is non-integral as shown in the last line of figure 3.7. For all practical purposes, overflow must therefore be prevented from occurring at the inputs to multipliers for both two's complement and sign-and-magnitude systems.

Having identified the points in a digital structure for which overflow must be avoided (the output and multiplier inputs for two's complement and sign-and-magnitude systems, and adder inputs for sign-and-magnitude systems), a method is required for establishing a system word length which will prevent the overflow. Such a method can be developed by considering the operation of a filter when overflow is not encountered. When overflow is avoided at the critical points, the operations of addition and multiplication proceed without error and the only deviation from ideal operation is the errors introduced in the least significant end of the signal by truncation operations. These errors are bounded however (the bounding value depending on the type of truncation used), so the operation of the digital structure without overflow can be modelled by an ideal linear digital filter which matches the actual structure with the truncation operations replaced by bounded error inputs. If there are n truncation operations, and m critical nodes at which overflow must be prevented, the signals $s(k)_i$ at each of these nodes can be expressed by the convolution summations:

$$s(k)_i = \sum_{j=0}^{k} h_i(k-j)u(j) + \sum_{l=1}^{n}\sum_{r=0}^{k} b_{il}(k-r)e_l(r) \qquad i = 1,2,3,\cdots,m \tag{3.11}$$

where $u(k)$ is the input sequence, $h_i(k)$ is the response sequence at critical node i to a unit sample input, $b_{il}(k)$ is the response sequence at critical node i to a unit sample input at truncation node l, and $e_l(k)$ is the error input sequence at truncation node l. If the input sequence is bounded by M and the n error sequences are bounded by N, then using (3.11) the maximum potential signal values at the critical nodes can be determined as:

$$|s(k)_i| \le M\sum_{j=0}^{\infty}|h_i(j)| + N\sum_{l=1}^{n}\sum_{r=0}^{\infty}|b_{il}(r)| \qquad i = 1,2,3,\cdots,m \tag{3.12}$$

for any value of k. For stable digital filters, the unit sample responses $h_i(k)$ and $b_{il}(k)$ will be

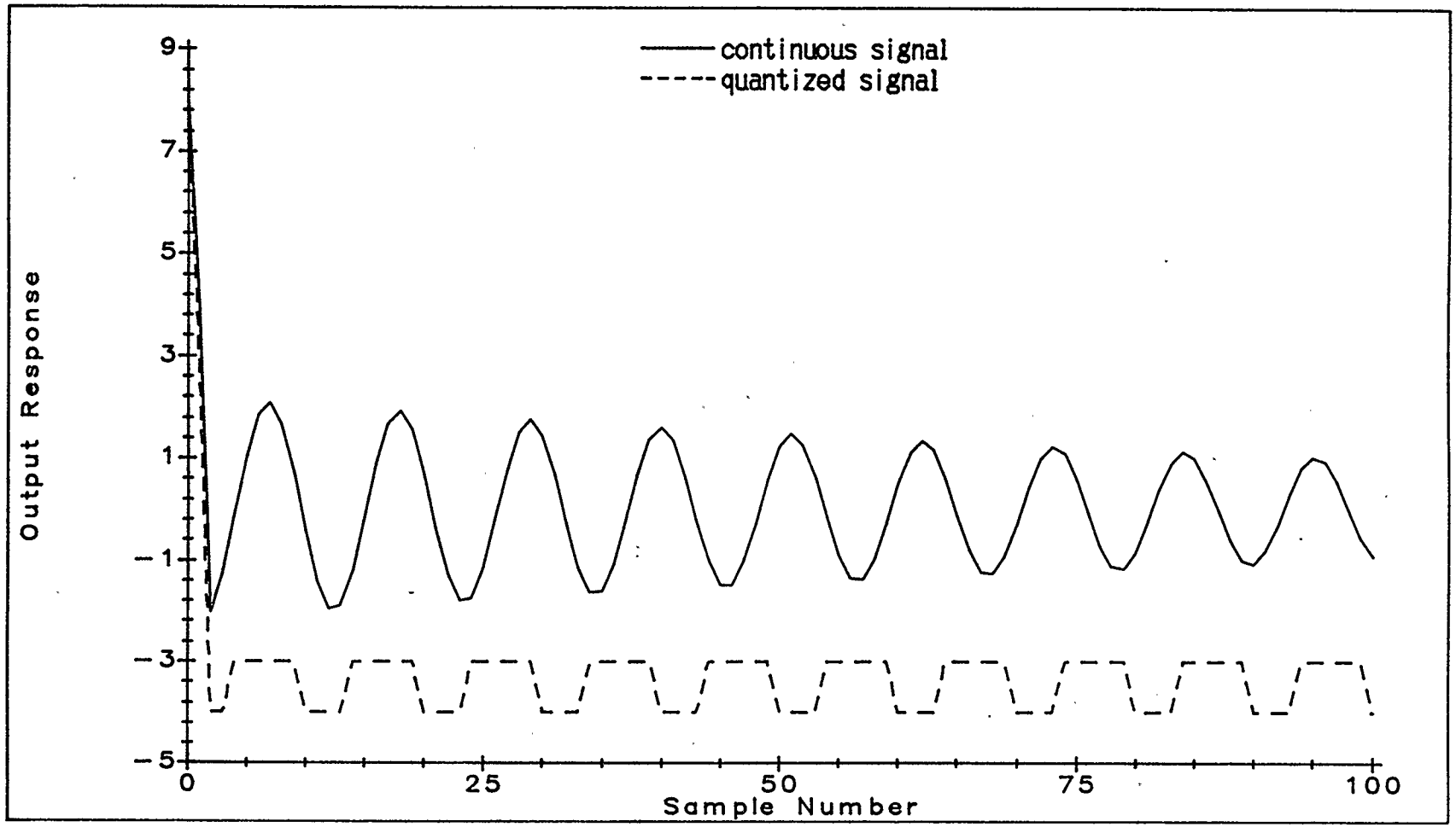
absolutely convergent, and signal bounds for the critical nodes can be practically determined by computing the sums in (3.12) to some number of terms where the incremental change becomes acceptably small. If the actual digital structure is built with enough bits at each critical node to accommodate the largest possible signal values thus determined, overflow errors will be prevented.

3.3. Signal Quantization

As discussed in section 3.2, once overflow is prevented at the critical nodes a digital filter behaves as an ideal linear filter with multiple error inputs. The errors are the result of truncating the signal values within the filter. This truncation essentially quantizes the internal signal resolution so that the smallest representable signal change is some finite (as opposed to infinitesimal) value. Although this signal granularity does not give rise to gross distortion as does overflow, it can lead to low-level effects which are troublesome. Figure 3.8 shows two output responses of the filter of figure 3.1 to an input sequence $u(k)$ given by

$$u(k) = \left\{-8,6,0,0,0, \cdots ,0\right\} \quad k = 0,1,2, \cdots ,100. \tag{3.13}$$

The solid response results from unquantized internal signals, while the dashed response results from the internal signals being quantized to integers. The truncation in this case was applied immediately following each multiplication operation, and was of the two's complement type. Instead of asymptotically approaching zero as the unquantized response does, the quantized response becomes a stable periodic oscillation with a constant offset. This type of oscillation is known as a zero-input limit cycle. The error sources which model signal quantization are sometimes crudely treated as independent noise sources, and their effects are summarized by some noise power spectral density at the filter output [22]. This approach does not adequately describe effects such as limit cycles, which result from correlation between the signal and the error sequences. Substantial efforts have been devoted to determining bounds on the amplitude of limit cycle oscillations which can occur in digital



UCEE

Figure 3.8 : Effect of Signal Quantization

filters; such bounds and practical approaches to reducing the effects of signal quantization are the subject of the next chapter.

CHAPTER 4

LIMIT CYCLES AND NOISE

Signal quantization in a digital filter leads to small magnitude output deviations (with respect to a continuous signal filter) which deteriorate the quality of the filtering by introducing digital "noise". It is desirable to have techniques for bounding the magnitude of this digital noise so that signal levels can be set above it. One particularly troublesome type of "noise" is a limit cycle, in which the output of a filter oscillates indefinitely after the input signal has gone to zero or some constant level. In this chapter techniques for bounding limit cycles and noise are presented. For the analysis presented in this chapter, it is useful to use the state representation of a digital filter. The state representation reduces the single m^{th}-order difference equation (1.6) which describes an ideal filter to a set of m coupled first-order equations. Under zero input conditions the state representation allows the ideal filter operation to be described by the single compact matrix expression

$$x(k+1) = Ax(k), \tag{4.1}$$

where $x(k)$ is the (m x 1) state vector describing the system at sampling instant k, and A is the (m x m) constant state matrix describing the system under zero input conditions.

4.1. A General Limit Cycle Bound

Considerable work has been done with the aim of determining reasonable bounds for the amplitude of limit cycles ([23] - [27]), but many of the proposed bounds are difficult to apply to high-order filters ([23], [25]), or apply only to certain filter structures ([25], [26]). Here, a method is presented for computing limit cycle bounds which is applicable to any filter whose zero input operation can be expressed by the non-linear difference equation

$$\hat{x}(k+1) = F[A\hat{x}(k)], \tag{4.2}$$

where the ^ indicates the fact that non-idealities are present. $F[\bullet]$ is a non-linear quantization operation, which reflects the effects of internal signal quantization. Although the derivation of the bound is developed for systems under zero input conditions, the results are identical for a system operated under constant input conditions when a simple translation of the state coordinates is introduced. The general bound developed in this section establishes the framework for tighter bounds presented in sections 4.1 and 4.2, although these bounds are applicable only under zero input conditions to systems operated under a particular quantization scheme.

It is assumed that the infinite precision autonomous discrete filter

$$x(k+1) = Ax(k) \tag{4.3}$$

is stable (i.e. the eigenvalues of A all have moduli less than one), and that a linearly independent set of right eigenvectors of A exists (a requirement which is usually met for practical filters). A Liapunov function [28] $L(x)$ for the infinite precision system can then be defined as:

$$L(x) = \|T^{-1}x\|, \tag{4.4}$$

where $\|\bullet\|$ denotes the Euclidean norm, and T is the matrix of normalized right eigenvectors of A. Clearly $L(x)$ is positive definite, and it is a Liapunov function as follows:

$$L(x(k+1)) = L(Ax(k)) = \|T^{-1}ATT^{-1}x(k)\| \tag{4.5a}$$

$$= \|\Lambda_A T^{-1}x(k)\| \tag{4.5b}$$

$$\leq \|\Lambda_A\| \, \|T^{-1}x(k)\| = |\lambda_M| \, L(x(k)) \tag{4.5c}$$

$$< L(x(k)), \tag{4.5d}$$

where Λ_A is the diagonal matrix of eigenvalues of A, and λ_M is the eigenvalue of A with maximum modulus. Under infinite precision operation, Liapunov stability theory thus assures asymptotic stability to the origin under zero input conditions and to the equilibrium state

under constant input (as expected for a stable linear system). The same is not true for the finite precision non-linear system. However if the relations

$$L(\hat{x}(k+1)) < L(\hat{x}(k)) \text{ , for all } \hat{x}(k) \notin \Gamma \tag{4.6a}$$

$$\hat{x}(k+1) \in \Gamma, \text{ for all } \hat{x}(k) \in \Gamma \tag{4.6b}$$

hold, where Γ is some region in the state-space, this is sufficient to ensure asymptotic stability to the region Γ under zero input conditions, and any zero input limit cycles must therefore lie within Γ. Alternatively stated, (4.6a) forces trajectories which start outside Γ into the region Γ, and (4.6b) insists that once a trajectory has entered Γ, it is confined there.

In order to determine the region Γ, it is assumed that $F[x]$ has the property

$$L(F[x]) \leq L(x) + \Delta e \tag{4.7}$$

for some constant Δe. Then

$$L(\hat{x}(k+1)) = L(F[A\hat{x}(k)]) \leq L(A\hat{x}(k)) + \Delta e \tag{4.8a}$$

$$\leq |\lambda_M| \, L(\hat{x}(k)) + \Delta e \tag{4.8b}$$

where (4.5) has been utilized. If the constraint

$$|\lambda_M| \, L(\hat{x}(k)) + \Delta e < L(\hat{x}(k)) \tag{4.9}$$

is imposed so that (4.6a) is satisfied, it is found that

$$L(\hat{x}(k)) > \frac{\Delta e}{1-|\lambda_M|} \tag{4.10}$$

and any limit cycles must therefore be confined to the region bounded by

$$L(x) = \frac{\Delta e}{1-|\lambda_M|} \tag{4.11}$$

since (4.8) ensures that (4.6b) is also satisfied by this choice of boundary. It remains to be shown only that (4.7) is valid for some types of quantization. In fact, this relation holds for any non-linearity which maps its argument into some bounded region enclosing the argument. In particular, for any non-linearity $F[\bullet]$ with the property that

$$|| F[x] - x || \leq \rho \tag{4.12}$$

where ρ is some constant, (4.7) holds. This can be seen most clearly by expressing $L(x)$ in a new coordinate system. Note that

$$L^2(x) = x^T T^{-1^*} T^{-1} x = x^T S x \tag{4.13}$$

where $S \equiv T^{-1^*} T^{-1}$ is a real symmetric matrix and the superscript * denotes complex conjugate transpose. Thus S can be decomposed as follows [29]:

$$S = Q \Lambda_S Q^T \tag{4.14}$$

where Q is an orthogonal norm-preserving real matrix (i.e. $||Qx|| = ||x||$, for all x and $Q^T Q = I$) of the normalized eigenvectors of S and Λ_S is the diagonal matrix of (real) eigenvalues of S. A new coordinate system is defined as

$$y = Q^T x \tag{4.15}$$

so that

$$L^2(x) = x^T S x = x^T Q \Lambda_S Q^T x = y^T \Lambda_S y \tag{4.16a}$$

$$= \mu_1 y_1^2 + \mu_2 y_2^2 + \cdots + \mu_n y_n^2 \tag{4.16b}$$

where $\mu_1, \mu_2, \cdots, \mu_n$ are the eigenvalues of S. The level surfaces of $L(x)$ are thus n-dimensional hyper-ellipses, whose axes lie along the y coordinate axes, i.e. Q is a rotation matrix which rotates the x system into alignment with the axes of $L(x)$. The maximum rate of increase of $L(x)$ is thus equal to a constant $\sqrt{\mu_j}$ in a direction along the y_j axis corresponding to the largest eigenvalue μ_j of S. If property (4.12) holds, a suitable choice for Δe is thus given by

$$\Delta e = \sqrt{\mu_j} \rho. \tag{4.17}$$

The property (4.12) is valid for three commonly used quantizations:

i) rounding quantization

$$F[x]_i = round(x_i) \tag{4.18}$$

$$\rho = \frac{\sqrt{n}}{2} \tag{4.19}$$

ii) two's complement quantization

$$F[x]_i = floor(x_i) \tag{4.20}$$

$$\rho = \sqrt{n} \tag{4.21}$$

iii) sign-magnitude quantization

$$F[x]_i = sign(x_i)floor(|x_i|) \tag{4.22}$$

$$\rho = \sqrt{n} \tag{4.23}$$

where n is the system order and the *floor*(•) and *sign*(•) functions are defined as

$$floor(x_i) = greatest\ integer \leq x_i \tag{4.24a}$$

$$sign(x_i) = \begin{cases} 1, & x_i \geq 0 \\ -1, & x_i < 0 \end{cases} \tag{4.24b}$$

It is important to note that these quantizations refer only to the nature of the non-linearity involved, and do not imply any particular arithmetic system. (The two's complement and sign-magnitude quantizations described above are so named because they are the non-linearities resulting respectively from the truncation of the fractional portion of a value represented in binary two's complement or sign and magnitude form.)

The region described by (4.11) implies absolute bounds on the individual states x_i as can be seen by introducing yet another coordinate transformation

$$z = \sqrt{\Lambda_S}y = \sqrt{\Lambda_S}Q^T x \tag{4.25}$$

where $\sqrt{\Lambda_S}$ represents the matrix resulting from taking the positive square root of each element of Λ_S. In this coordinate system

$$L(x) = \sqrt{x^T Q\sqrt{\Lambda_S}^T\sqrt{\Lambda_S}Q^T x} = ||z|| \tag{4.26}$$

and the absolute bounds on the individual states can be determined by solving for $\max(x_i)$

subject to

$$||z|| = \frac{\Delta e}{1-|\lambda_M|}. \tag{4.27}$$

However, since x_i is simply the dot product of z with row i of $Q\sqrt{\Lambda_S}^{-1}$ (i.e. $x_i = [Q\sqrt{\Lambda_S}^{-1}z]_i$), the solution to this maximization problem can be determined by inspection as setting z equal to $\frac{\Delta e}{1-|\lambda_M|}$ times the normalized row i of $Q\sqrt{\Lambda_S}^{-1}$. The absolute bounds on the states x_i are then given by

$$|x_i| \le \frac{\Delta e\sqrt{e_i^T Q\Lambda_S^{-1}Q^T e_i}}{1-|\lambda_M|} \tag{4.28}$$

where e_i is the i^{th} column of the identity matrix, and the quantity under the square-root is simply the square of the norm of row i of $Q\sqrt{\Lambda_S}^{-1}$. Table 4.1 summarizes the computations required to determine the absolute amplitude bounds on the states, and table 4.2 shows values of Δe for typical quantizations.

If the system is operated under constant input u_c so that it is described by

$$x(k+1) = Ax(k) + bu_c \tag{4.29}$$

then a simple coordinate translation given by

$$y = x - x_e \tag{4.30a}$$

$$x_e = Ax_e + bu_c \tag{4.30b}$$

will transform the system to

$$y(k+1) = Ay(k) \tag{4.31}$$

which is exactly the same form as (4.3). The results developed for systems operating under zero input conditions are therefore applicable to the constant input case by simply considering the origin to be the new equilibrium point x_e.

$$\hat{x}(k+1) = F[A\hat{x}(k)]$$

$$\Lambda_A = T^{-1}AT \qquad |\lambda_M| = \max_i(|\Lambda_{A\,ii}|)$$

$$S = T^{-1^*}T^{-1}$$

$$\Lambda_S = Q^TSQ \qquad \mu_M = \max_i(|\Lambda_{S\,ii}|)$$

$$L(x) = ||T^{-1}x|| \qquad L(F[x]) \le L(x)+\Delta e$$

$$|x_i| \le \frac{\Delta e\sqrt{e_i^T Q\Lambda_S^{-1}Q^T e_i}}{1-|\lambda_M|}$$

Table 4.1 : Summary of bound computations.

QUANTIZATION	Δe
rounding	$\frac{\sqrt{\mu_M n}}{2}$
two's complement	$\sqrt{\mu_M n}$
sign-magnitude	$\sqrt{\mu_M n}$
n = system order	

Table 4.2 : Δe for different quantizations.

4.2. Second Order Sections

For second order filters operating under sign-magnitude quantization, the correlation of quantization with signal sign can be used to reduce the bound (4.28) with $\Delta e = \sqrt{\mu_M n}$ by reducing the value substituted for Δe. Figure 4.1 shows a constant energy ellipse $L(x) = c$, the state (x) coordinate axes, and the y coordinate axes (as defined by (4.15)) which are aligned with the ellipse. It can be shown (see Appendix) that the maximum value of $L(\hat{x})$ attainable through sign-magnitude quantization of x to $\hat{x}$ occurs when $\hat{x}$ is either point U or V, and that Δe in (4.7) is therefore bounded by

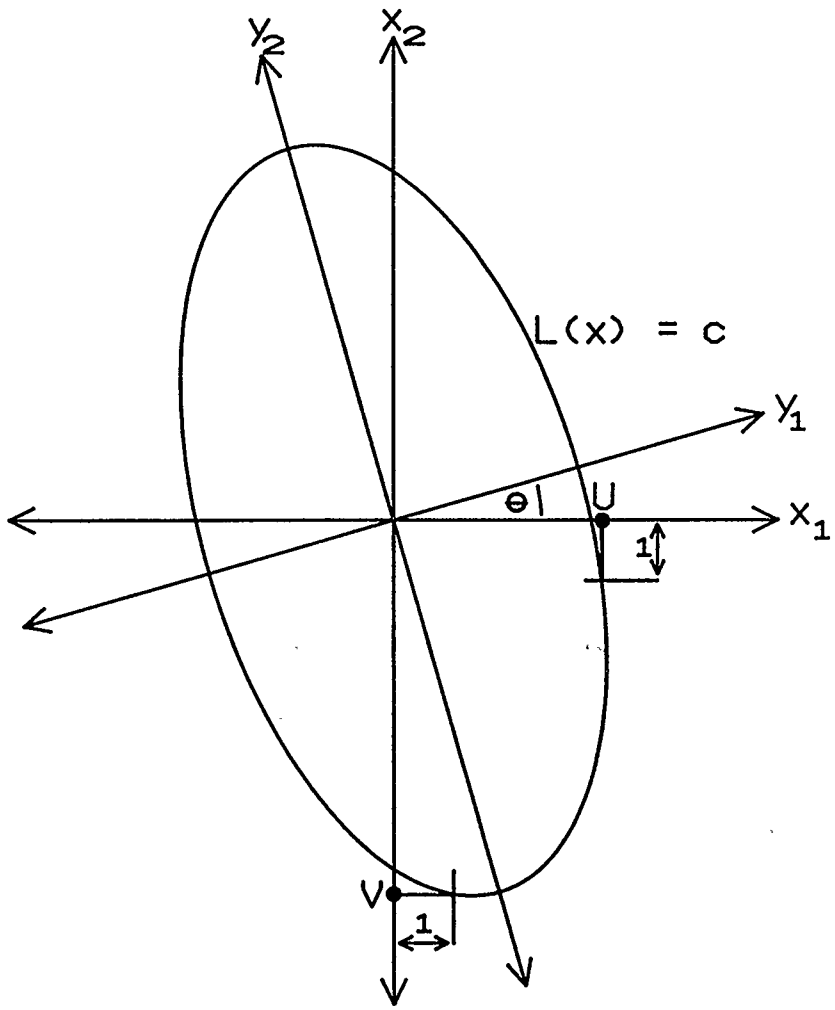


Figure 4.1 : Geometry of L(x)

$$\Delta e < \max_{\{\phi = \theta,\ 90-\theta\}} \left[\frac{\delta\sqrt{\mu_1\mu_2}}{[\mu_2\cos^2(\phi) + \mu_1\sin^2(\phi)]^{1/2}} \right] \tag{4.32}$$

where

$$\delta = \frac{1 - \sqrt{\mu_2/\mu_1}}{\cot(\phi) + \sqrt{\mu_2/\mu_1}\tan(\phi)} \ , \tag{4.33}$$

$\mu_1 \geq \mu_2$ are the eigenvalues of S, and $\theta = \cos^{-1}(Q_{11})$ (with Q and S as defined by (4.13) and (4.14)). Thus for the second order system under sign-magnitude quantization, the bound (4.28) is replaced by

$$|x_i| \leq \frac{\delta\sqrt{\mu_1\mu_2 e_i^T Q \Lambda_S^{-1} Q^T e_i}}{(1 - |\lambda_M|)[\mu_2\cos^2(\phi) + \mu_1\sin^2(\phi)]^{1/2}} \tag{4.34}$$

where δ and ϕ are as defined by (4.32) and (4.33). It is worth noting that as μ_2/μ_1 approaches unity, or as θ approaches 0° or 90°, δ approaches zero, and the region in which limit cycles may exist becomes vanishingly small. In the limiting cases, the Liapunov function $L(x)$ as defined by (4.4) simply becomes the Euclidean norm (or a weighted form of it) of the states, so that effectively $L(x) = ||x||$. Under sign-magnitude quantization, $||\hat{x}|| \leq ||x||$ so the infinite precision Liapunov function is also a Liapunov function for the non-linear system in these limits. Filters with normal state matrices [30] (i.e. $A^TA = AA^T$) such as the second order coupled form satisfy both of these limits, and so are free of zero input limit cycles under sign magnitude quantization.

4.3. Higher Order Sections

Substantial work has been done in the exact design of high-order low sensitivity digital filters both as all-pole ladder structures [31]-[33] and elliptic filters such as the LDI and WDF structures designed directly from doubly terminated analog LC ladder prototypes. These filters are not always possible to consider as simply a collection of second-order sections, and consequently limit cycle bounds for filters of order higher than two are of considerable

interest. The substantial bound reduction achieved for second order sections suggests a similar technique might be successfully applied for higher order sections operated under sign-magnitude quantization. Unfortunately, analytic expressions analogous to (4.32) and (4.33) have not been determined due to the difficulty of applying geometric interpretations in higher order spaces. Nonetheless, the basic idea of using the correlation of quantization with signal sign to reduce the value substituted for Δe in (4.7) remains valid for sections of arbitrary order. In this section a computational bound based on this concept is presented.

For the computational procedure the problem is formulated in terms of $L^2(x)$ rather than $L(x)$ so equations (4.7) - (4.11) are no longer valid. However a similar set of equations can be derived by assuming that $F[x]$ has the property

$$L^2[F[x]] \leq L^2(x) + \alpha L(x) + \beta \tag{4.35}$$

for some positive constants α and β. Then

$$L^2[\hat{x}(k+1)] \leq L^2[x(k+1)] + \alpha L[x(k+1)] + \beta \ , \tag{4.36}$$

and if the relation

$$L^2[x(k+1)] + \alpha L[x(k+1)] + \beta < L^2[\hat{x}(k)] \tag{4.37}$$

can be satisfied, this implies

$$L^2[\hat{x}(k+1)] < L^2[\hat{x}(k)] \tag{4.38}$$

and similarly to (4.6), zero input limit cycles are again confined to some region. Noting that (4.5) implies

$$\frac{L^2[x(k+1)]}{|\lambda_M|^2} \leq L^2[\hat{x}(k)] \ , \tag{4.39}$$

relation (4.37) will hold provided

$$(|\lambda_M|^2 - 1)L^2[\hat{x}(k)] + \alpha|\lambda_M|L[\hat{x}(k)] + \beta < 0 \tag{4.40}$$

is valid. Since $|\lambda_M|^2 - 1 < 0$ by the stability assumption, (4.40) will hold for all $L(\hat{x}(k))$ greater

than the largest solution to

$$(|\lambda_M|^2 - 1)L^2[\hat{x}(k)] + \alpha|\lambda_M|L[\hat{x}(k)] + \beta = 0 \tag{4.41}$$

which will be denoted as L_M. Zero input limit cycles must then be confined to the region bounded by

$$L(x) = L_M. \tag{4.42}$$

The problem is thus reduced to one of finding constants α and β such that (4.35) is satisfied.

For filters operated under sign-magnitude quantization, the problem of determining α and β can be approached by attempting to maximize the quantization error function

$$E(x) = (x + \Delta x)^T S(x + \Delta x) - x^T S x \tag{4.43}$$

subject to the constraints

$$x_i \Delta x_i \leq 0 \tag{4.44}$$

$$L^2(x) = x^T S x = c^2 \tag{4.45}$$

$$|\Delta x_i| < 1 \tag{4.46}$$

$$|\Delta x_i| < |x_i| \tag{4.47}$$

where c is an arbitrary positive constant, and Δx is the error vector defined by

$$\Delta x = F[x] - x. \tag{4.48}$$

The numerous inequality constraints and the non-linear nature of (4.44) make this problem very difficult to approach from either an analytic or computational standpoint, so a solution to the problem is attempted with constraint (4.47) removed and Δx considered to be independent of x (essentially removing constraint (4.48)). The removal of constraints does not decrease the maximum of (4.43), so using the solution of the problem with fewer constraints to determine α and β merely adds some pessimism to the limit cycle bounds.

Even with the constraints removed, an analytic solution of the maximization problem does not seem likely. However, certain features of the problem can be used to reduce the number of possible solutions to a finite set over which $E(x)$ (which, under the removal of the constraints as discussed above, actually becomes $E(x,\Delta x)$) can be computed. The absolute maximum can then be determined by comparing the values at each of the potential solution points. The main simplifying property of the problem is that Δx must lie entirely on the constraint boundary implied by (4.44) and (4.46). This can be seen by differentiating $E(x,\Delta x)$ with respect to the elements of Δx:

$$\frac{\partial E}{\partial \Delta x_i} = k_E + 2S_{ii}\Delta x_i \tag{4.49a}$$

$$\frac{\partial^2 E}{\partial \Delta x_i^2} = 2S_{ii} > 0 \tag{4.49b}$$

where S is as defined by (4.13) and has positive diagonal entries, and k_E is a constant. Since (4.49) implies that any local extrema in $E(x,\Delta x)$ due to Δx are minima, the maximum value of $E(x,\Delta x)$ must therefore lie on a constraint in Δx. Potential solutions to the problem then have the property that Δx_i is one of 0, 1, or -1, and the maximum can be determined by maximizing $E(x,\Delta x)$ in x for each candidate Δx, then comparing the resulting values of $E(x,\Delta x)$. The problem is thus reduced at each stage to one of maximizing the intermediate function (which is the only variable term of $E(x,\Delta x)$ when Δx is considered constant)

$$I(x) = 2\Delta x^T S x \tag{4.50}$$

subject to (4.45) and

$$\pm x_i \leq 0 \tag{4.51}$$

with the $\pm$ determined by the signs of the constants Δx_i.

If the foregoing problem is re-stated in the z coordinate system as defined by (4.25), the constraint (4.45) becomes simply

$$L(x) = ||z|| = c \tag{4.52}$$

and $l(x)$ is given by

$$l(x) = l'(z) = \gamma^T z \tag{4.53}$$

where γ is given by

$$\gamma = 2\sqrt{\Lambda_S} Q^T \Delta x. \tag{4.54}$$

Constraint (4.51) becomes

$$[-Bz]_i \geq 0 \tag{4.55}$$

where

$$B = \pm Q\sqrt{\Lambda_S}^{-1} \tag{4.56}$$

with the $\pm$ indicating that the rows of $Q\sqrt{\Lambda_S}^{-1}$ are multiplied through by the sign of Δx_i, or set to zero if $\Delta x_i = 0$.

Examining (4.53), it is evident that maximizing $l'(z)$ is a matter of minimizing the angle between the constant vector γ and the fixed length vector z, subject to constraint (4.55). The direction of the solution vector will be independent of the arbitrary constant c, so that if we denote by z_0 the solution with $c = 1$, the solution for any c is given by $z = cz_0$. Clearly then, the maximum value of $l'(z)$ is proportional to c through the proportionality factor $||\gamma^T z_0||$, and for fixed Δx

$$L^2(F[x]) - L^2(x) \leq E(x,\Delta x) \leq ||\gamma^T z_0|| c + \Delta x^T S \Delta x. \tag{4.57}$$

Comparing (4.57) with (4.35), a suitable choice for α and β is

$$\alpha = ||\gamma^T z_0|| \tag{4.58a}$$

$$\beta = \Delta x^T S \Delta x \tag{4.58b}$$

where γ is given by (4.54). The procedure for determining the region in which limit cycles must be confined has thus been reduced to one of computing z_0 for every candidate Δx,

then using (4.58) to determine the maximum solution of (4.41). The maximum of these solutions over all candidate Δx becomes the L_M of (4.42) and determines the region in which limit cycles must lie. The only remaining problem is the determination of z_0 given Δx.

Interpreted geometrically, constraint (4.55) requires that z lie in or on a convex cone C bounded by the hyper-planes passing through the origin which are described by the rows of $-B$. Equations (4.54) and (4.56) imply $[-B\gamma]_i \leq 0$ so that γ must lie outside C for any non-zero Δx (since γ cannot satisfy (4.55)). Minimizing the angle between z and γ (in order to maximize $l'(z)$) thus constrains the solution z_0 to lie on the surface of C, hence z_0 will simply be the normalized projection of γ onto a hyper-plane bounding C or the intersection of two or more such hyper-planes. Figure 4.2 shows an example of this geometry in three dimensions. Since removal of constraints in a maximization problem will not reduce the maximum, z_0 can be determined by projecting γ first onto each hyper-plane bounding C, then projecting onto the intersection of each distinct pair of hyper-planes bounding C, then the intersection of each distinct triple of hyper-planes and so on until a result which satisfies (4.55) is obtained. If the maximum of $l'(z)$ is positive (i.e. the angle between γ and C is less than 90° at some point), this procedure will produce the solution. If the maximum of $l'(z)$ is negative, the above procedure will continue until γ is being projected onto a zero space (i.e. the intersection of n distinct hyper-planes where n is the system size), yielding the zero vector as the solution. In this case, the actual solution must lie along an extremal ray of C [34] which is described by the intersection of n–1 distinct hyper-planes bounding C. Since there are only n such extremal rays, the solution which maximizes $l'(z)$ can be determined by comparing the values obtained along each ray.

To actually compute the projection of γ (denoted by γ') onto the intersection of a subset of the hyper-planes bounding C, it is just necessary to note that each row of $-B$ is normal to one of the bounding hyper-planes, hence the intersection of m hyper-planes bounding C is the orthogonal complement of the vector space V spanned by the corresponding m

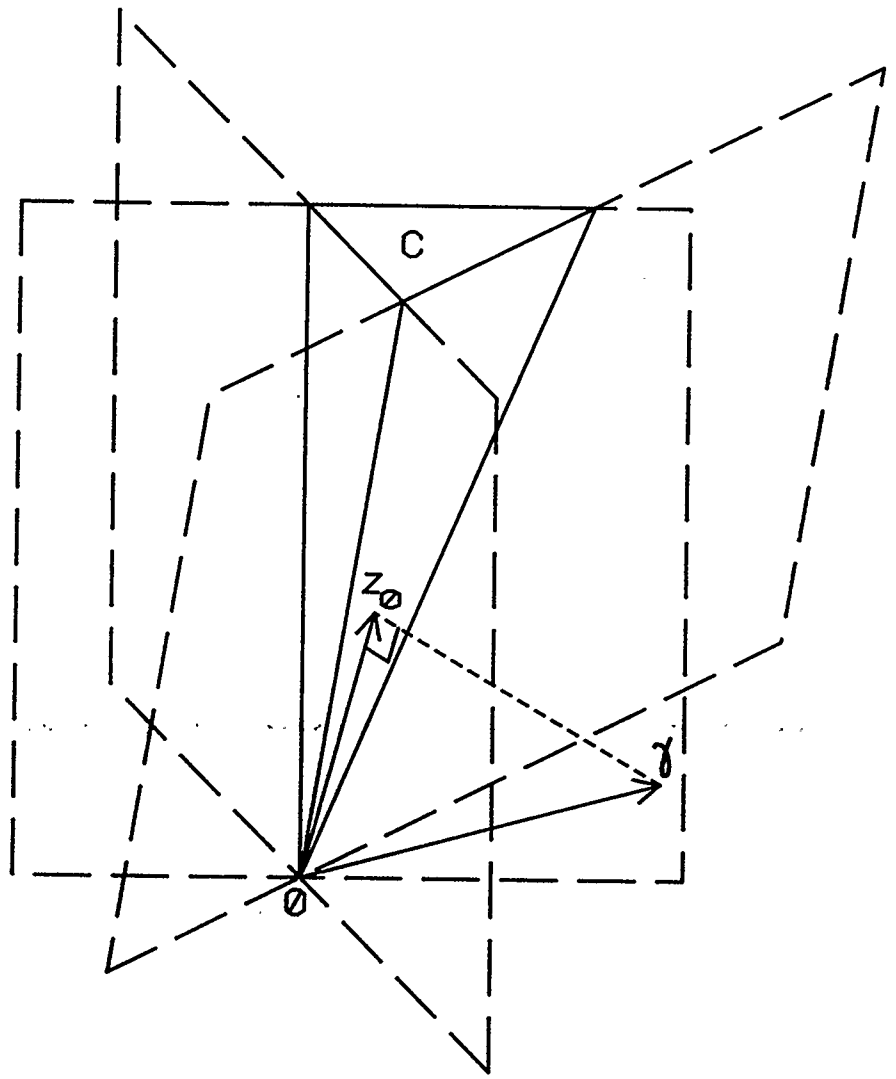


Figure 4.2 : Projection of γ Onto Cone C

rows of $-B$. These rows may be reduced by a standard procedure such as Gram-Schmidt orthogonalization to yield an orthonormal basis $\{v_1, v_2, \cdots, v_m\}$ of V, at which point γ' can be computed as

$$\gamma' = \gamma - \sum_{i=1}^{m} \frac{v_i^T \gamma}{\|\gamma\|} v_i. \tag{4.59}$$

The preceding discussion completely describes a computational procedure for determining the value L_M which defines the region in which zero input limit cycles must lie for systems operated under sign-magnitude quantization. By comparison to (4.11) and (4.28), (4.42) determines absolute amplitude bounds on the individual states as given by

$$|x_i| \leq L_M \sqrt{e_i^T Q \Lambda_S^{-1} Q^T e_i} \tag{4.60}$$

where e_i is again the i^{th} column of the identity matrix.

4.4. A General Noise Bound

While the limit cycle bounds presented in the preceding sections provide useful and in some cases quite tight bounds on the amplitude of limit cycles, they suffer from two shortcomings. Firstly, the bounds apply only to limit cycles under prescribed input conditions and hence do not provide any information about how far the filter output can deviate from that of an ideal continuous filter under arbitrary input conditions. Secondly, they require that truncation occur only at the input to the delay (state) elements. Depending on the filter structure, this condition can be difficult to achieve in practice. In addition, the bounds which rely on sign-magnitude truncation characteristics suffer from the disadvantage that for particular filter architectures (discussed in detail in chapter five) it is quite difficult to implement sign-magnitude truncation.

A useful bound which overcomes the foregoing limitations can be determined by considering equation (3.11). If the output of the filter is considered to be a critical node, (3.11) shows that the component $o_e(k)$ of the output due to the truncation operations is given by:

$$o_e(k) = \sum_{l=1}^{n}\sum_{j=0}^{k} v_l(k-j)\, e_l(j) \tag{4.61}$$

where $v_l(k)$ is the response sequence at the output to a unit sample input at truncation node l, and $e_l(k)$ is again the error input sequence at truncation node l. Using the bounded nature of the error sequences, an absolute bound on the noise component of the output is found to be:

$$|o_e(k)| \le N\sum_{l=1}^{n}\sum_{j=0}^{\infty} |v_l(j)| \tag{4.62}$$

for any value of k, where N is the bound on the error sequences. The bound (4.62) applies to a digital filter under arbitrary input conditions, requiring only that the filter be free of internal overflow so that (3.11) remains valid. While (4.62) is more pessimistic than other noise bounds which are typically applied [24], it has the advantage of being an absolute bound, and is in fact less pessimistic in some cases than the bound described in section 4.3, which is itself less pessimistic than previously reported bounds applicable to filters of order higher than two [35]. To illustrate this point, table 4.3 compares the bound (4.62) with the bound described in section 4.3 when applied to the 6^{th}-order LDI bandpass structure considered in [35].

State	Bound	
	expression (4.62)	section 4.3
x1	22	29
x2	14	27
x3	20	30
x4	29	32
x5	22	29
x6	14	27

Table 4.3 : LDI Bound Comparison

Just as the overflow bound (3.12) can be practically approximated for stable filters, so can the noise bound (4.62). Once the output noise level is bounded, an actual digital filter implementation can guarantee that any required level of noise will not be exceeded simply by scaling the input signal up by a factor α and the output signal down by the reciprocal factor $1/\alpha$. The overall transfer function is not changed by these scaling operations, but the output noise is reduced by the factor $1/\alpha$. When the input is scaled up it is of course necessary to increase the internal signal word length correspondingly at the critical nodes in order to ensure that overflow is prevented.

CHAPTER 5

SIZE ESTIMATES AND CHOICE OF STRUCTURE

In chapters three and four, techniques were described which allow non-ideal effects to be eliminated (in the case of overflow) or reduced to acceptable levels (coefficient and signal quantization). Before these techniques can be applied however, a particular filter structure must be chosen. In selecting a structure from those presented in chapter two, there is no single best choice since each structure has different non-ideal performance which depends on the desired transfer function. Since the goal of this research was to produce a filter with a programmable transfer function, no single transfer function could reasonably be chosen as a basis for the comparison of non-ideal effects within the structures. Instead, a set of transfer functions was determined which represented a range over which it was felt that a practical programmable filter should operate (this range is of course subjective). In order to compare the non-ideal performance of the structures, it was required that each be capable of implementing all transfer functions of the representative set with zero relative passband error, freedom from overflow with 16-bit input, and output noise less than one least-significant bit. In order to meet the foregoing requirements, each structure requires a certain minimum number of coefficient and signal bits. In this chapter these minimum requirements are used to make initial estimates of the hardware cost of each structure. Based on these estimates and other qualitative factors a structure is chosen for the filter implementation.

5.1. Representative Transfer Functions

In determining the representative transfer functions, three parameters were considered to be of interest: relative bandwidth, center frequency, and passband ripple. The set of

representative transfer functions is composed of the 27 possible combinations in which these parameters range over three values deemed to be "high", "medium", and "low". For each representative transfer function the minimum stopband attenuation is 30 dB. Relative bandwidth is defined as

$$\text{relative bandwidth } (\%) = \frac{(\Omega_u - \Omega_l)}{\Omega_c} \times 100 \tag{5.1}$$

where Ω_u, Ω_l, and Ω_c are respectively the upper, lower, and geometric center frequencies of the passband as given by

$$\Omega_u = f_u \times 2\pi \tag{5.2}$$

$$\Omega_l = f_l \times 2\pi \tag{5.3}$$

$$\Omega_c = \sqrt{\Omega_u \Omega_l}\,. \tag{5.4}$$

Here, f_u and f_l are respectively the upper and lower passband edge frequencies, where the attenuation just exceeds the maximum passband attenuation. The "low", "medium", and "high" values of the parameters for the representative transfer functions are shown in table 5.1. For a sample frequency of 40 kHz the low, medium and high center frequencies are 600 Hz, 2400 Hz, and 10000 Hz respectively. These frequencies were chosen so that using the high value for relative bandwidth the audio frequency range is roughly partitioned into

Parameter	Low	Medium	High
relative bandwidth	6%	24%	96%
$\Omega_c T$	0.03π rads	0.12π rads	0.5π rads
passband ripple	0.011 dB	0.098 dB	1.25 dB

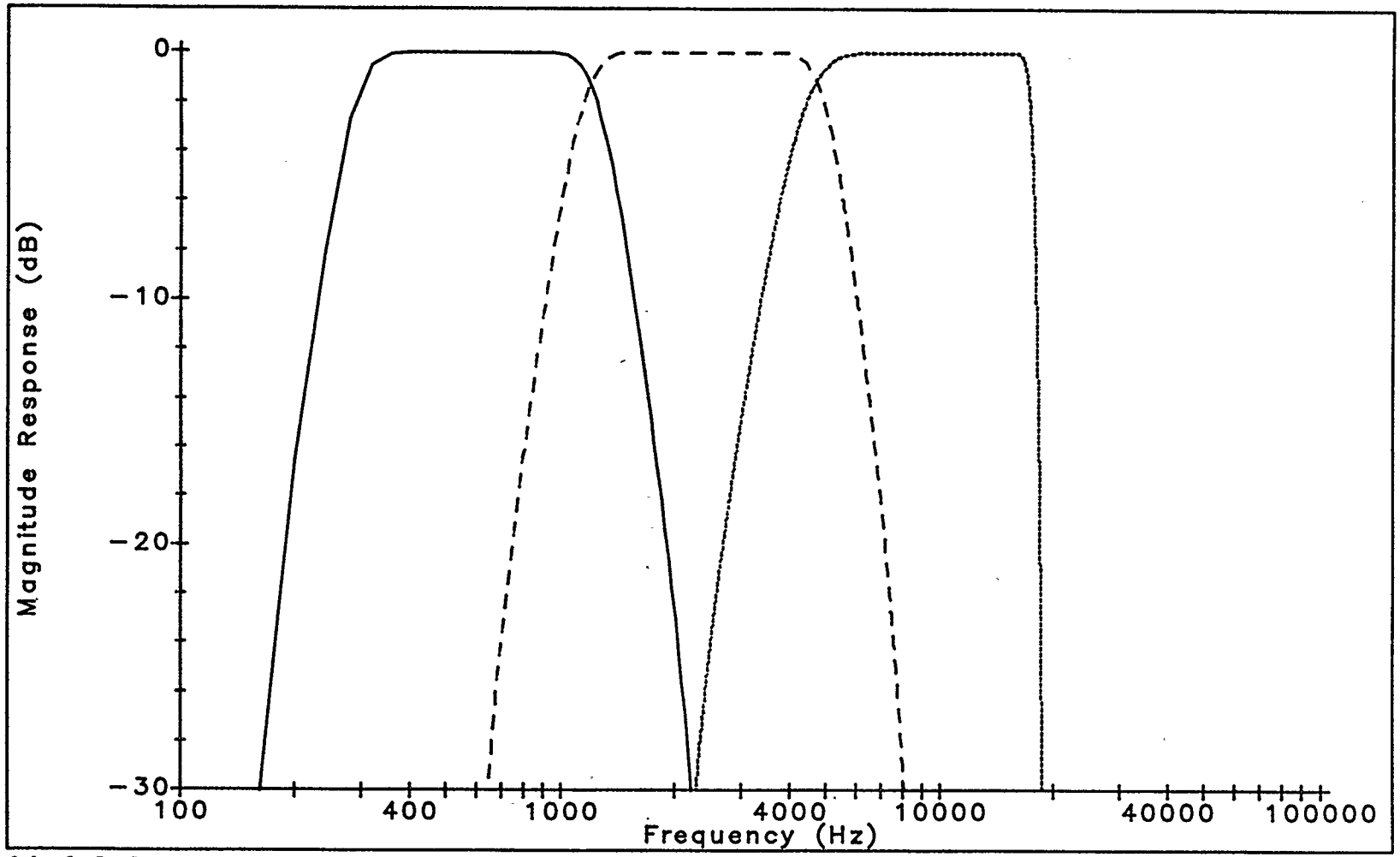
Table 5.1 : Representative Parameter Values

logarithmic thirds as shown in figure 5.1. The low, medium and high values for relative bandwidth were determined by requiring that the medium-to-low and high-to-medium ratios be approximately four as for the center frequency parameter, as well as by insisting that the low value for relative bandwidth be on the order of a few percent, which is the typical limit for conventional LC filters [36]. The ratio of medium-to-low and high-to-medium for the passband ripple parameter was selected to be roughly ten to give a relatively large range of ripple values. The actual ripple values were chosen to correspond to those used in tables of normalized lowpass filter component values in [36]. The component values of the prototype of figure 1.2 which correspond to each representative transfer function were therefore determined simply by using a lowpass to bandpass transformation [37] on a normalized lowpass prototype with component values determined from tables. The prototype element values thus determined were used in conjunction with the expressions shown in tables 2.1, 2.2 and 2.3 to determine the multiplier values for the corresponding WDF and LDI structures. In order to determine the z-domain transfer function coefficients required to compute the multiplier values for the DF1, DF2, LF1, and LF2 structures, the LDI filters were analyzed with a digital filter simulation program [38].

5.2. Restrictions Imposed by Coefficient Quantization and Overflow

The lowpass to bandpass transformation used to generate the prototype element values and the LDI filter analysis were performed using double-precision floating point arithmetic on a VAX-11/750. The factoring of the representative transfer functions to determine the DF2 coefficients and the implementation of recursion (2.14) to determine the lattice coefficients were also performed in this manner. Since the double-precision format of the VAX allows a 56-bit mantissa (corresponding to a resolution of roughly one part in 10^{17}), the multiplier values determined were considered to be essentially ideal. Using the filter simulation program, each structure was analyzed over each of the representative transfer functions to determine the smallest number of fractional multiplier coefficient bits required to imple-

University of Calgary, Department of Electrical Engineering



UCEE

Figure 5.1 : Low, Medium, and High Center Frequency Filters

ment the complete set of transfer functions with zero relative passband error as defined by (3.8). For each transfer function the ideal coefficient values were quantized using rounding truncation uniformly on all multipliers, starting with 30 fractional bits and proceeding downward one bit at a time. The total number of coefficient bits required for each structure is the minimum number of fractional bits plus however many coefficient bits are required for the integer portion of the coefficient values, plus one sign bit. For all except the WDF and LDI structures, it was found that 30 fractional bits were insufficient to achieve zero relative passband error for some transfer functions. While this can be explained in part by the sensitivity properties discussed in chapter three, it was noted that in some cases the simulated filters failed to meet the zero relative passband error criterion even when simulated in full double-precision floating point arithmetic (this was especially prevalent for the low center frequency, low relative bandwidth transfer functions).

Close examination of the simulation results revealed that in some cases the simulator was not determining the transfer function of the simulated filter as accurately as might be expected from the arithmetic precision. As a specific example, the low center frequency, low relative bandwidth, high passband ripple transfer function as determined from simulation of the LDI structure met the zero relative passband error criterion, but the DF1 structure derived from it did not, even with full precision simulation. The transfer function coefficients determined by simulation of the DF1 structure differed from those in the LDI transfer function in the ninth decimal place, indicating an accuracy of only one part in 10^9 (roughly 30 binary bits). A direct calculation using the DF1 multiplier coefficients yielded a transfer function which was accurate to at least 13 decimal places in all coefficients. No attempt was made to investigate the source of this inaccuracy in the simulator, as this would be outside the scope of this investigation due to the complexity of the program. A possible explanation however is that the high sensitivity of the direct form and lattice structures reappears in the form of numerical ill-conditioning in the simulator. This explanation is consistent with the results that the simulations of both the LDI and WDF structures always achieved the zero

relative passband error criterion. In addition, the transfer functions for which the inaccuracies appeared had closely clustered poles, which is a condition that typically results in problems which are difficult to deal with numerically [39]. In any case, it was felt that the less accurate simulator results should not be used to eliminate the direct form and lattice filters from consideration. For comparisons involving the required number of coefficient bits it is therefore assumed that the direct form and lattice structures require some number of fractional coefficient bits greater than 30 to meet the zero relative passband error criterion.

By simulating each structure with the multiplier coefficients quantized to the minimum number of bits (or 30 fractional bits for the direct form and lattice structures), a lower bound can be placed on the required signal word length. This can be seen by recalling that in chapter three it was shown that overflow must be prevented at the inputs of multipliers for systems implemented in either two's complement or sign-and-magnitude arithmetic. Thus, regardless of the arithmetic chosen for an actual filter implementation, a minimum set of critical nodes with regard to expressions (3.11) and (3.12) is the set consisting of just the inputs to each multiplier. Without assuming anything about signal truncation within the structure, the first term of expression (3.12) can be evaluated over the minimum set of critical nodes to determine a lower bound on the signal word length required to prevent overflow.

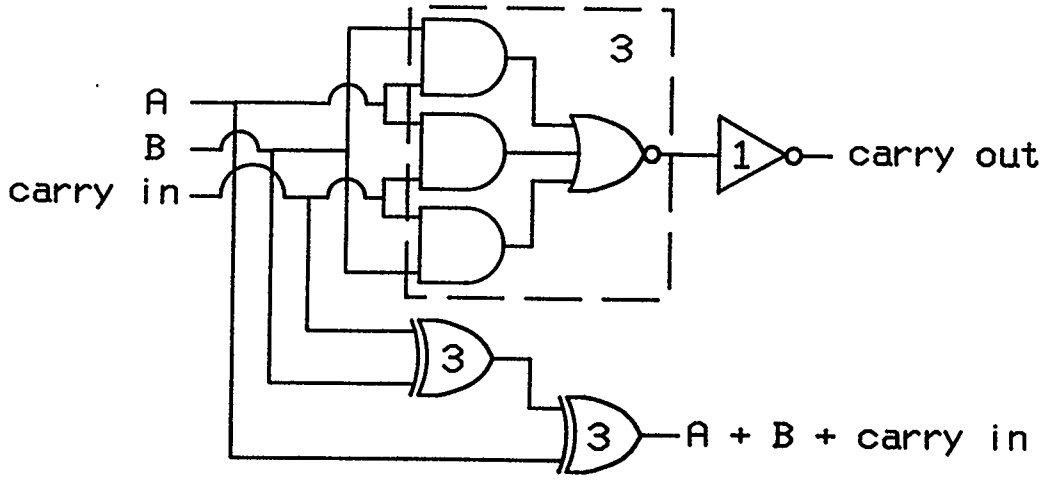
Once the number of coefficient bits and minimum number of signal bits are known, a lower bound on the hardware cost for implementing each structure can be determined by adding the hardware cost of one multiplier (assuming that one multiplier can be multiplexed to perform all the required multiplications) to the hardware costs of coefficient storage and signal storage (delay elements). By examining typical strategies for performing parallel multiplication [40] it can be seen that in order to multiply an m-bit data value by an n-bit coefficient, the minimum hardware required is mn two-input AND gates to form each partial product (the products of the data value with each of the individual coefficient bits) and $(m-1)(n-1)-1$ full adders and n half adders to accumulate the partial products. The minimum

hardware cost of a parallel multiplier is thus given by:

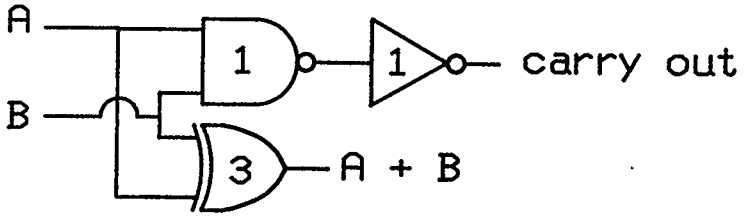
$$\text{hardware cost} = mn\text{AND} + (mn-m-n)\text{FA} + n\text{HA} \tag{5.5}$$

where AND, FA, and HA respectively represent the hardware cost of a two-input AND gate, a full adder, and a half adder. Note that the choice of which multiplication operand is the data and which is the coefficient is arbitrary, so that the smaller of the two should be chosen as the coefficient to minimize the hardware cost as shown by (5.5). The schematics of full and half adders implemented with gate array elements are shown in figure 5.2. In this figure, the numbers indicate the number of "gate equivalents" (roughly the area required for four transistors) required for each logic operation. A full adder requires 10 gate equivalents, and a half adder requires 5. Two-input AND gates are formed using a two-input NAND gate followed by an inverter, each of which requires 1 gate equivalent. Each bit of coefficient or signal storage will require at least one bit of memory, which requires 4 gate equivalents. Table 5.2 shows the minimum coefficient and signal bit requirements for each structure and the corresponding lower bounds of the hardware cost (in gate equivalents). It is important to note that the figures in table 5.2 assume that a single multiplier can be multiplexed over all the required multiplications, and no allowance has been included for timing/control, steering, interface, or test circuitry. Inevitably in the detailed design of a circuit, unforeseen factors are encountered which impose additional hardware requirements. All of these additional requirements indicate that the bounds presented in table 5.2 are likely to be very optimistic.

Using the fabrication facilities made available by the Alberta Microelectronic Centre, the largest die size available for the filter implementation has a capacity of roughly 10000 gate equivalents. In order to ensure routability however, the maximum array area usage recommended is 80% [41], so that in effect the maximum number of gate equivalents available for the filter implementation is roughly 8000. From table 5.2 it is clear that none of the structures can be implemented in 8000 gates or less if parallel multipliers are used and all the desired performance criteria are met. Since floating point multipliers are even more



(a)



(b)

Figure 5.2 : (a) Full Adder
(b) Half Adder

Hardware Requirements		WDF	LDI	DF1	DF2	LF1	LF2
Coefficients	minimum bits	25	29	>36	>32	>31	>40
	number	9	10	11	9	13	13
	gate equiv.	900	1160	>1584	>1152	>1612	>2080
Signal	minimum bits	23	23	45	28	51	22
	delays	8	7	6	6	6	6
	gate equiv.	736	644	1080	672	1224	528
Multiplier (gate equiv.)		6535	7599	>18810	>10292	>18307	>10050
Total (gate equiv.)		8171	9403	>21474	>12116	>21143	>12658

Table 5.2 : Hardware Requirement Lower Bounds

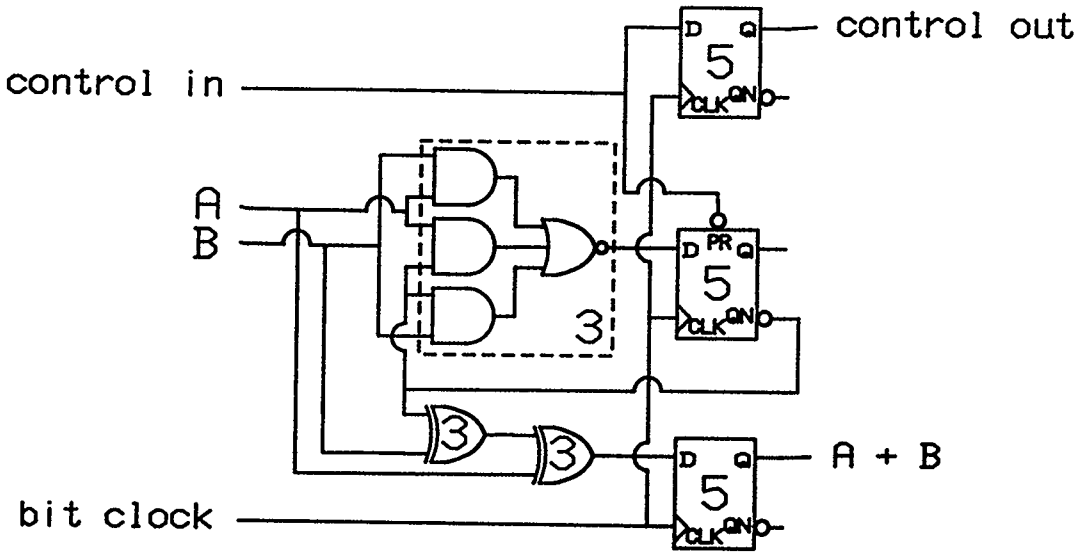
costly in terms of hardware than parallel multipliers, the possibility of using floating point arithmetic in the filter implementation is also precluded. In order to successfully implement a complete filter within the available hardware and without compromising on performance, it is clear that filter elements much smaller than the parallel multiplier must be used.

5.3. Bit-serial Architecture

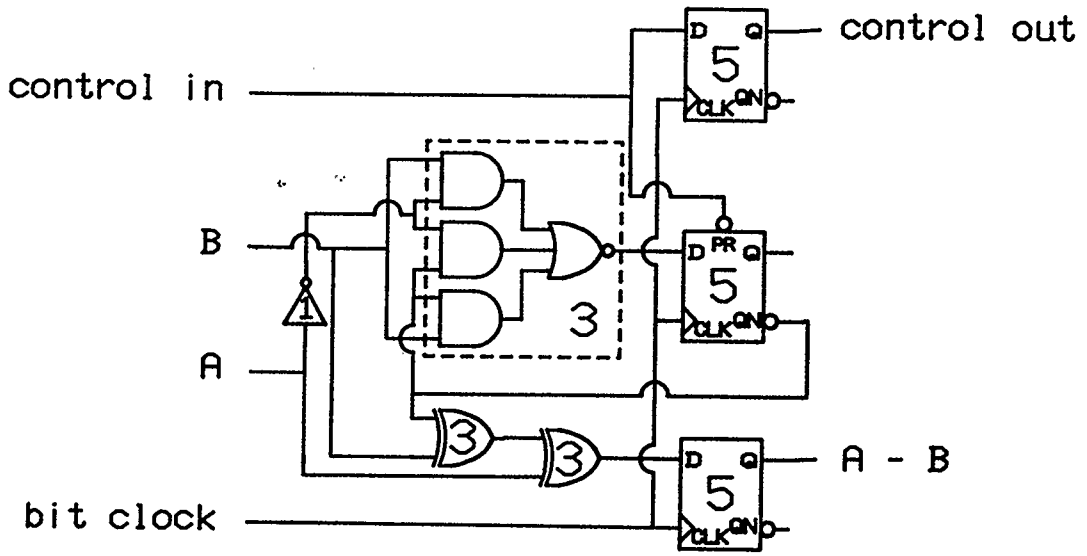
An alternative to the large parallel multiplier is a bit-serial multiplier, which has the advantage of requiring much less hardware for given data and coefficient lengths than the corresponding parallel multiplier. The general principle of bit-serial architectures is that arithmetic operations are performed bit-by-bit on the operands to produce the result bit-by-bit [42]. Since a single wire can carry a signal (or operand) of arbitrary length, bit-serial hardware systems tend to be an improvement over parallel systems both in terms of the routing complexity (or area required for routing) and the hardware requirements. Because the results of bit-serial operations must be generated one bit at a time, it is important to choose an arithmetic in which the result of operations on the current bit do not influence previous bits in the result. Two's complement arithmetic meets this requirement provided that

operands and results are propagated from the least significant bit (LSB) to the most significant bit (MSB). Sign-and-magnitude arithmetic does not meet this requirement as can be understood by considering the addition of two values. If the values are propagated from LSB to MSB, the entire result remains undetermined until the last bits (which determine the sign of the result) are presented. If instead the values are propagated from MSB to LSB, the entire result remains undetermined until the last bits are presented and the effect of carrys can be determined. Short of performing a serial-to-parallel conversion of the operands and a parallel-to-serial conversion of the result (which eliminates the hardware savings), there is no effective way of implementing sign-and-magnitude arithmetic in a bit-serial architecture, hence the subsequent discussions will consider only two's complement arithmetic.

The basic digital filtering operations of addition (or subtraction) and delay have a very simple form in a bit-serial architecture, while the operation of multiplication is somewhat more complex (as is also the case for parallel architectures). A bit-serial adder can be made by connecting the carry out of the full adder shown in figure 5.2 back to its carry in through a unit (one bit) delay. A control bit applied simultaneously with the LSBs of the operands sets the carry feedback delay output to zero at the beginning of each sum. In practice, the adder will usually have a one-bit output delay to facilitate pipelining of operations, and the control signal will be propagated along with the result to simplify the distribution of control signals. Figure 5.3 shows implementations of pipelined bit-serial adders and subtracters using gate-array elements. As in figure 5.2, the numbers represent the number of gate equivalents required for each logic operation. A delay element in a digital filter can be considered to be simply a register or memory element. Consequently, in a bit-serial architecture the delay operation is a cascade of as many unit delays (one-bit memories) as required to accommodate the signal. Bit delays which are introduced for pipelining (as in the adder) are also one-bit memories, and effectively serve to distribute part of the filter delay operations to the arithmetic processing elements.



(a)



(b)

Figure 5.3 : (a) Serial Adder
(b) Serial Subtracter

In a bit-serial multiplier, the basic technique of forming and summing the partial products is employed [43]. In contrast to the parallel multiplier however, the partial products and their sum are formed in a serial fashion. Figure 5.4 shows the functional diagram of a basic four-bit two's complement serial multiplier. In this multiplier data and coefficients propagate through stages which form the successive partial products and accumulate them to form the final product. In order to achieve a high throughput, each adder has a one-bit delay so that the maximum bit clock rate is limited only by the time required to perform a single bit pipelined addition. A control signal is propagated along with the LSB of the data to reset the adder carry circuits and latch the coefficient bits. In this multiplier, all but the last stage replaces the LSB of the accumulated sum with a sign extension of the previous accumulated sum. In this way, only the most significant bits of the product are actually presented at the output, and the multiplier can operate continuously on *m*-bit data values to produce *m*-bit products which are the two's complement truncated version of the full product. This operation is illustrated in figure 5.5. Since the weighting of the MSB of a two's complement value is negative, the last stage of the multiplier must subtract the final partial product rather than add it as in the previous stages. It should also be noted that the data value must contain a sign repetition in order for the sign bit of the accumulated partial products to be correctly determined and extended at each stage.

There are two potential drawbacks to a multiplier of the type illustrated in figure 5.4. First, it can be seen that the latency (time between the appearance of the first data bit at the input and the appearance of the first product bit at the output) of the operation is twice the coefficient length times the bit clock period. As discussed above, the latency of a serial operation represents distribution of the filter delay operation. Consequently for long coefficients, the latency of the multiplication illustrated could conceivably be greater than the desired amount of filter delay. This would require that the filter delay be increased (or equivalently the data length be increased) with a corresponding decrease in the filter throughput. The second potential drawback of the illustrated multiplication method is that the

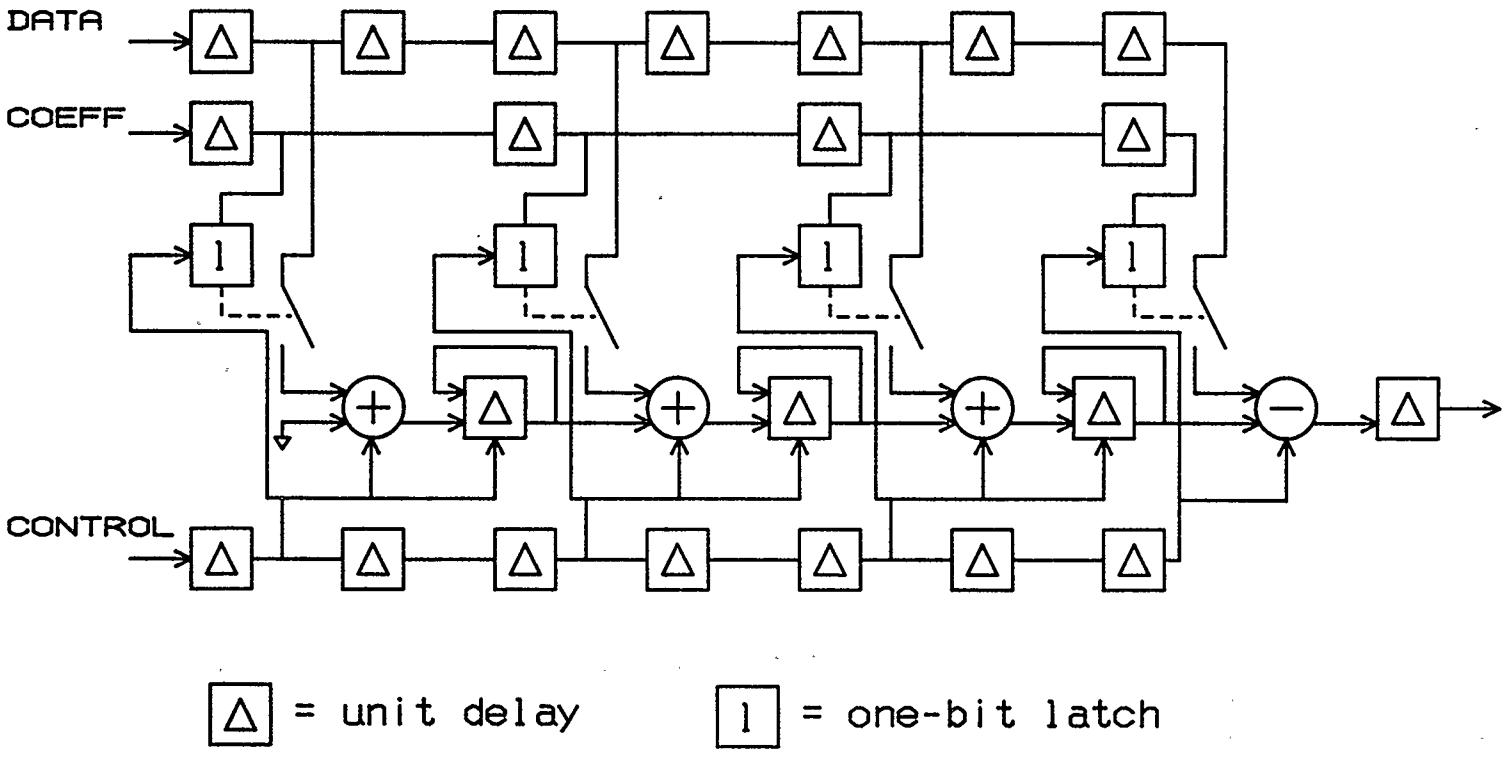


Figure 5.4 : Two's Complement Serial Multiplier

```
    00101      5
  x 1.011    x -5/8
  -------
    00.101
+   001.010
+  0000.000
- 00101.000
  ---------
= 11100.111  = -25/8
```

PARALLEL MULTIPLY

```
   00000101     00000101
+  00001010   = 00001111
+  00000000   = 00001111
-  00101000   = 11100111
```

partial products

accumulated partial products

sign extend bits

SERIAL MULTIPLY

```
stage 1     NNN00101     NN00010P
stage 2  +  NN00101P   = N00011PP
stage 3  +  N00000PP   = 00001PPP
stage 4  -  00101PPP   = 11100PPP
```

N = next product bit
P = previous product bit

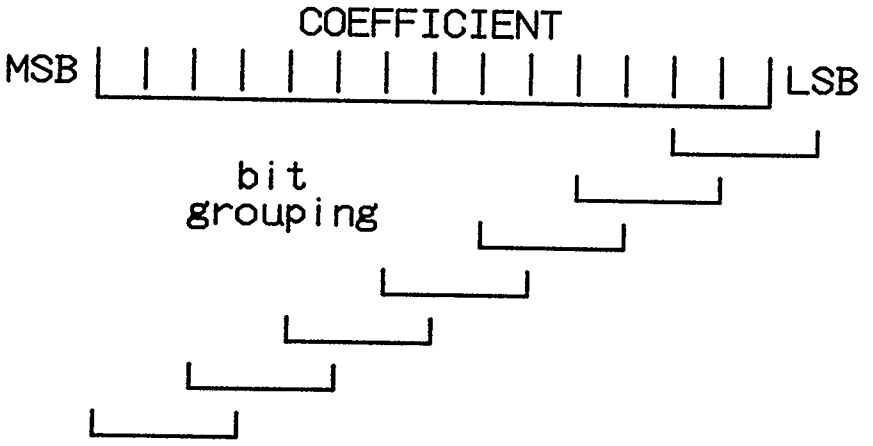
Figure 5.5 : Example Multiplication

effective range of coefficients lies between -1 and 1 since the most significant bits of the product are always produced. This drawback could be overcome by following the multiplier by a left shift operation, but this cannot recover the less significant bits of the product, and hence results in unacceptably large truncation errors.

The latency and coefficient range problems can be respectively reduced and eliminated by modifying the basic multiplier of figure 5.4. The first modification is to introduce circuitry which recodes the coefficient and allows each stage to add ±2, ±1, or 0 times the data into the accumulated partial products. This five-level recoding results in a multiplier which has only one stage for every two coefficient bits. Each stage has a latency of 3 bits, so the overall latency of the recoded multiplier is reduced in comparison to the basic multiplier. The method of recoding the coefficient is illustrated in figure 5.6. To allow coefficient values of magnitude greater than one, the circuitry is further modified so that one or more of the final stages can inject zero bits into the least significant end of its partial product (at the expense of neglecting an equal number of bits at the most significant end of the partial product), and begin computation of the accumulated partial products early. The number of zero injection stages required is the integer portion of half the number of integer coefficient bits. An added benefit of the modification to allow coefficients with magnitude greater than one is that the multiplier latency is reduced by the number of integer coefficient bits times the bit clock period. A modified multiplier was designed and verified using simulation software [44] in order to determine the hardware requirements for the stages as well as to confirm correct functionality.

5.4. Bit-serial Hardware Requirements

The bit-serial multiplier produces a result which is two's complement truncated to the same number of bits as the input data, so in a filter implemented with bit-serial multipliers the location and type of the truncation operations are fixed. In addition, the choice of a bit-serial architecture dictates that two's complement arithmetic be used, hence overflow need



Coefficient Bit Group	Partial Product
0 0 0	zero
0 0 1	data
0 1 0	data
0 1 1	2data
1 0 0	-2data
1 0 1	- data
1 1 0	- data
1 1 1	zero

Figure 5.6 : Coefficient Recoding

only be prevented at the input to the multiplications. This information along with the desired noise performance is sufficient to allow the computation of the bounds (3.12) and (4.62) for each structure over each representative transfer function. For each filter and for each representative transfer function, the bound (4.62) is computed based on truncation error sources located immediately following each multiplier, and bounded by 1 (the maximum error a two's complement truncation can cause). This determines an upper bound on the noise component of the filter output. This bound in turn determines the output scaling factor required to ensure that the output noise is less than one LSB, and this scaling factor in turn determines the bound on the reciprocally scaled input signal. With the input bound known, the overflow bound (3.12) can be computed to determine the required internal signal word length. Table 5.3 lists the worst-case noise bound and signal word length requirements which were determined in this manner for each structure. An input signal length of 16 bits was assumed in each case, and the required sign repetition bit is included in the signal word length requirement.

Once the coefficient and signal word length requirements for each structure are known, it is a relatively straightforward task to determine lower bounds for the amount of hardware required to implement each of the structures with a bit-serial architecture. Table 5.4 summarizes each structure's requirements in terms of the various parameters which

Structure	Noise Bound (bits)	Signal Word Length (bits)
LDI	12	36
WDF	14	37
DF1	20	66
DF2	14	43
LF1	5	56
LF2	19	42

Table 5.3 : Signal Word Length Requirements

Parameter	Symbol	Structure Requirements					
		LDI	WDF	DF1	DF2	LF1	LF2
Integer coefficient bits	*I*	6	0	5	1	0	10
Fractional coefficient bits	*F*	24	24	>30	>30	>30	>30
Number of distinct coefficients	*C*	6	9	9	9	13	13
Signal word length	*S*	36	37	66	43	56	42
Number of register (delay) operations	*R*	7	8	6	6	6	6
Number of adders	*A*	14	20	9	7	12	18
Number of subtracters	*D*	4	14	2	4	6	6

Table 5.4 : Structure Requirements Summary

influence hardware requirements. Each register (delay) operation requires S unit delays. Since the multiplier coefficients must be delivered serially to the multiplier, it is assumed that coefficient storage will also be in unit delays. Each distinct coefficient therefore requires $I+F+1$ unit delays (the 1 is for the sign bit), and the total number of unit delays required for signal and coefficient storage can be estimated by:

$$\text{storage unit delays} = RS + C(I+F+1). \tag{5.6}$$

Each unit delay requires at least 5 gate equivalents, so the minimum storage requirement is given by:

$$\text{storage requirement} = 5RS + 5C(I+F+1) \tag{5.7}$$

where it is understood that the unit of hardware requirement is the gate equivalent. Similarly, the requirement for adders and subtracters is given by:

$$\text{add/subtract requirement} = 20A + 21D \tag{5.8}$$

where the requirement for individual adders and subtracters has been determined from figure 5.3. Note that the output unit delays were not included in this determination as it is assumed that they will form part of the (already counted) delay registers. From the serial multiplier designed for verification, it was found that each fractional stage required 117 gate equivalents, and each integer stage required 122 gate equivalents. The multiplier hardware requirement can therefore be expressed by:

$$\text{multiplier requirement} = 122\,\text{floor}\left(\frac{I}{2}\right) + 117\left\{\frac{(F+I+1)}{2} - \text{floor}\left(\frac{I}{2}\right)\right\} \tag{5.9}$$

where it is assumed that $I+F+1$ is even. Since the recoded multiplier always incorporates an even number of coefficient bits, if this sum is odd it must be increased by one for the computation of hardware requirements. The sum of the storage, add/subtract and multiplier requirements provide a lower bound on the hardware required to implement each structure in a bit-serial architecture. Table 5.5 shows the lower bounds determined for each structure.

Structure	Requirement			
	Storage (gate equiv.)	Add/Subtract (gate equiv.)	Multiplier (gate equiv.)	Total (gate equiv.)
LDI	2220	364	1887	4471
WDF	2650	695	1521	4866
DF1	>3600	222	>2116	>5938
DF2	>2730	224	>1872	>4826
LF1	>3760	366	>1872	>5998
LF2	>3990	486	>2482	>6958

Table 5.5 : Serial Hardware Requirements

Recalling that roughly 8000 gate equivalents are available for the filter implementation, it is conceivable that any of the structures could be implemented based on the values in table 5.5. It is important to remember however that these figures are lower bounds only because they do not include hardware requirements for timing and control, testing, interface, multiplexing, or unforeseen circuitry. The hardware requirements for this additional circuitry cannot be known with any accuracy until a detailed design is attempted. The choice of which structure for which to attempt a detailed design is therefore determined more by qualitative issues than quantitative ones.

5.5. Choice of Structure for Implementation

It has already been seen that in some cases it is difficult to accurately determine the multiplier coefficients for the direct form and lattice filters. From the standpoint of the filter user, this makes these structures less attractive because the effort required to implement a desired transfer function is increased. Even in the cases where the coefficients can be determined without accuracy problems, the procedures for determining them require simulation, factorization, and a complex recursion as opposed to the relatively straightforward application of a few equations as for computation of the LDI and WDF structure coefficients. In addition, the longer signal word lengths required for the direct form and lattice filters result in potentially lower throughput for these structures than for the WDF and LDI structures. For

a filter which multiplexes a single serial multiplier over all the required multiplications, the potential maximum throughput is limited by the time required to perform all the multiplications, which is proportional to the product of the signal word length with the number of multiplication operations. Because of the greater system word lengths, this product is larger for the direct form and lattice structures than it is for the LDI and WDF structures, and the potential maximum throughput rates are therefore lower for the direct form and lattice structures than for the WDF and LDI structures as shown in table 5.6. Finally, for the DF1, LF1, and LF2 structures the relatively small margin between the lower bound on hardware requirements and the maximum available hardware may be insufficient to accommodate the required timing, interface, testing, and multiplexing circuitry. All of these factors favor the

Structure	Throughput Limit
LDI	$\frac{f_{br}}{360}$
WDF	$\frac{f_{br}}{333}$
DF1	$\frac{f_{br}}{726}$
DF2	$\frac{f_{br}}{387}$
LF1	$\frac{f_{br}}{1064}$
LF2	$\frac{f_{br}}{546}$
f_{br} = bit-rate clock frequency	

Table 5.6 : Maximum Potential Throughputs

LDI and WDF structures over the direct form and lattice structures, which were therefore eliminated as candidates for the filter implementation.

The comparisons and estimates described to this point indicate very little difference between the LDI and WDF structures. While the LDI structure has the advantage that it preserves true attenuation zeros in the passband even with coefficient deviations, the WDF structure has the advantage that the required coefficients are always less than one in magnitude. The WDF structure has a higher potential throughput than the LDI structure, but the LDI structure has a smaller lower bound on hardware requirements. Ultimately, the LDI structure was chosen for implementation because an elegant scheme for multiplexing the multiplication operation became apparent from the regularity of the structure. No similar regularity was observed in the WDF structure, so the only method determined to multiplex the multiplier in it is to adopt a general architecture in which the multiplication operands are fetched from a set of addressable registers and the resulting product is stored back to some addressed register. This type of multiplexing would likely require a substantial overhead in terms of both hardware and throughput because of the required storage and recall of intermediate results. Although there may in fact be some elegant way to multiplex the multiplier in the WDF structure, the absence of any significant performance superiority favors the LDI structure with its known multiplexing scheme for the filter implementation.

CHAPTER 6

THE LDI IMPLEMENTATION

As discussed in the previous chapter, the LDI structure was ultimately selected for the filter implementation because a practical way of multiplexing the multiplication operation was determined. In this chapter, this multiplexing scheme is described along with various details of the actual implementation. The test setup for the fabricated filter is briefly described, and measured results are compared to those determined by simulation.

6.1. Multiplexing the LDI Structure

Figure 6.1 shows the signal flow graph for the LDI structure. In this figure the states in the recursive section are identified as X1 through X6, and the multiplication operations are numbered according to the order in which they are multiplexed. Referring to the figure, it can be seen that the output of every multiplication operation feeds directly into a digital "integrator" loop of the form associated with either state X1 or X2. Further inspection of the figure reveals that the state values can be updated in a particularly elegant manner: the next values for states X2, X4, and X6 can be computed from the current values of X1, X3, and X5, and the next values for states X1, X3, and X5 can then be computed from the *updated* X2, X4, and X6 values (as well as the value presented from the non-recursive input stage). This is equivalent to converting all the integrator loops to the type associated with state X1 (in which the delay element is in the forward path), and clocking the upward and downward directed branches alternately. With the integrator loops made identical in this fashion, the filter operations can be decomposed as shown in figure 6.2. In this figure the non-multiplexed addition operations are introduced at the outputs of a set of identical integrator loops (X1 through X6) so that the input values for the multiplication operations are

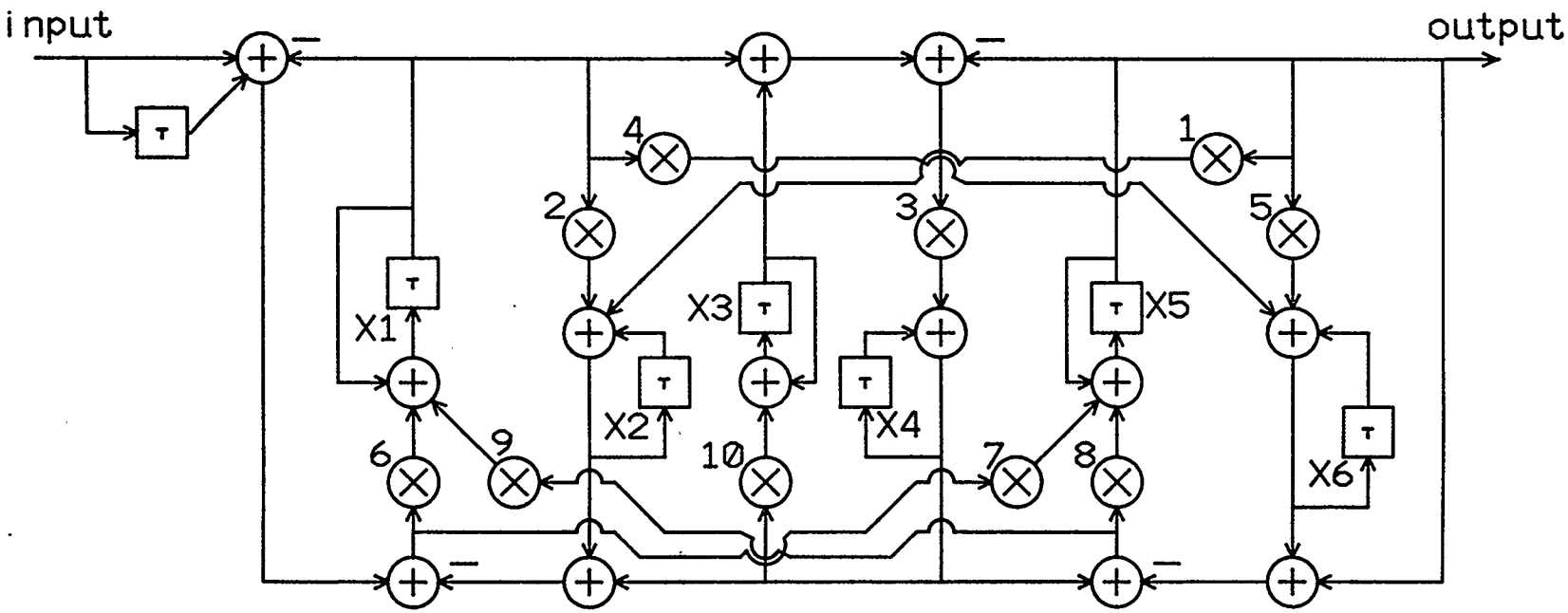
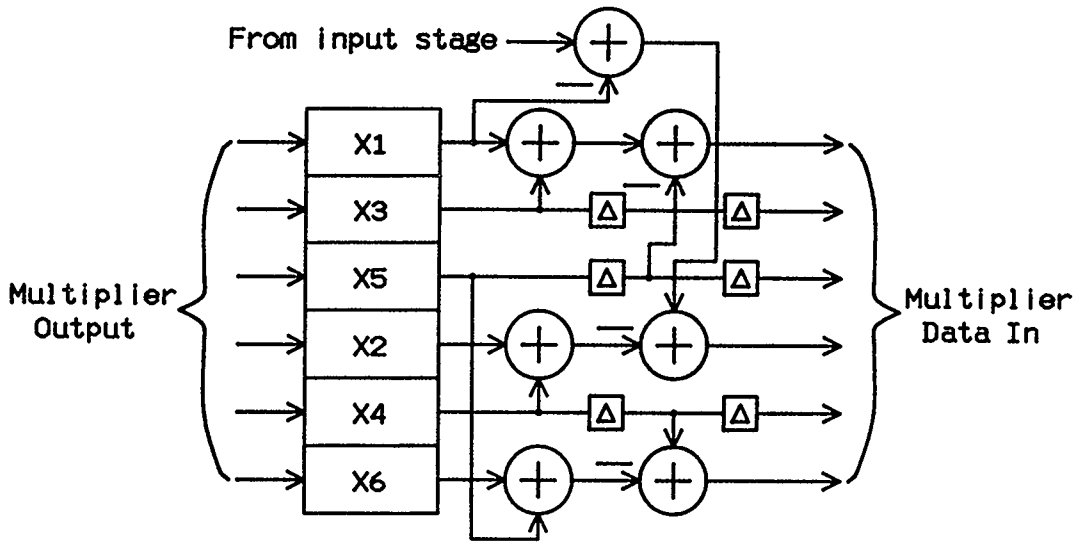
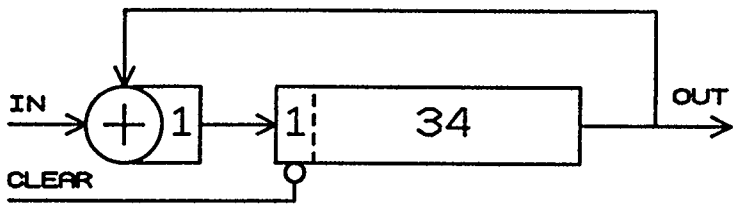


Figure 6.1 : LDI States and Multiplex Order



(a)



(b)

Figure 6.2 : (a) Non-multiplexed Operations
(b) Integrator Loop

continuously generated as shown in part (a) of the figure. Part (b) shows the detail of the integrator loops, with the numbers representing the distribution of the 36-bit system word length through the various components of the loop.

When the filter operations are partitioned in the manner described above, a simple multiplexing strategy is to select in turn each multiplier input value and route the products to the appropriate integrator loop, effecting the alternate clocking by the order of operations. In order for this multiplexing technique to be possible, the order of the operations must be chosen so that the products fed into the integrator loops do not alter any state values which are still required for generating multiplication input values. The latency of the serial multiplier and the integrator loops are useful in this regard because they provide some delay between when a multiplication input is used and when the effect of the product appears at the output of the integrator loops. Depending on the order of the multiplications however, this latency also sometimes requires idle cycles to be introduced during which the multiplier is not computing a useful result and the results of earlier operations are propagating to the outputs of the integrator loops. The numbers adjacent to the multiplication operations in figure 6.1 indicate one order in which the operations can be performed with no idle time. Although this has not been proven, it is suspected that the order of operations shown in figure 6.1 is one of only two possible orders which do not require idle time, yet still implement the correct filtering algorithm.

In order for this sequence of operations to implement the filter correctly, the multiplication operation must have a latency equal to twice the system word length minus two so that the products summed into the "down" integrators X2, X4, and X6 have propagated through the integrator loops in time to be used in the computations for the "up" loops X1, X3, and X5. As a specific example, multiplication operation 3 updates state X4, the value of which is used in multiplication operation 6 three system word length times later, after the result of multiplication 3 has propagated through the multiplier, the X4 integrator loop, and two addi-

tion operations. In order to achieve the required latency, the recoded multiplier was padded with an appropriate number of unit delays. The functional block diagram for the multiplexed LDI structure is shown in figure 6.3. In this figure, the multiplexing is represented by rotary switches controlled by modulo ten counters. Note that although only six distinct coefficients are required to implement the filter corresponding to the prototype of figure 1.2, it was decided to allow all the coefficients to be distinct in case this could provide more flexibility in the types of transfer functions which could be implemented.

6.2. Testability and Interface

The bulk of the circuitry for the filter implementation is required to perform the basic filtering algorithm. Substantial circuitry requirements however remain for interface and testing circuitry. It is important to incorporate testability into any design intended for integration so that devices can be rejected by automatic testing prior to packaging if they are faulty. Since the package is often the most expensive part of an integrated device, it is desirable to detect as many types of faults as possible with reasonable amounts of test circuitry and if possible using only a small number of test vectors. For the LDI filter implementation testability is incorporated by including multiplexers to inject test signals, and test points on various data paths as shown in figure 6.3. In test mode the input data pins are used as test signal inputs, and the output data pins are connected to the test points. This arrangement allows the multiplier, coefficient storage/selection circuitry, state circuitry (including the non-multiplexed operations), and timing/control circuitry to be independently tested. Because of the bit-serial nature of the implementation, it is possible to get a very high degree of fault coverage with a small amount of test circuitry and a relatively small number of test vectors [45].

As well as test circuitry, interface circuitry is required in order to allow loading of the coefficients, generation of input and output data timing signals, and external control of the filter. In addition, because the boundary between interface and other circuitry is not clear-

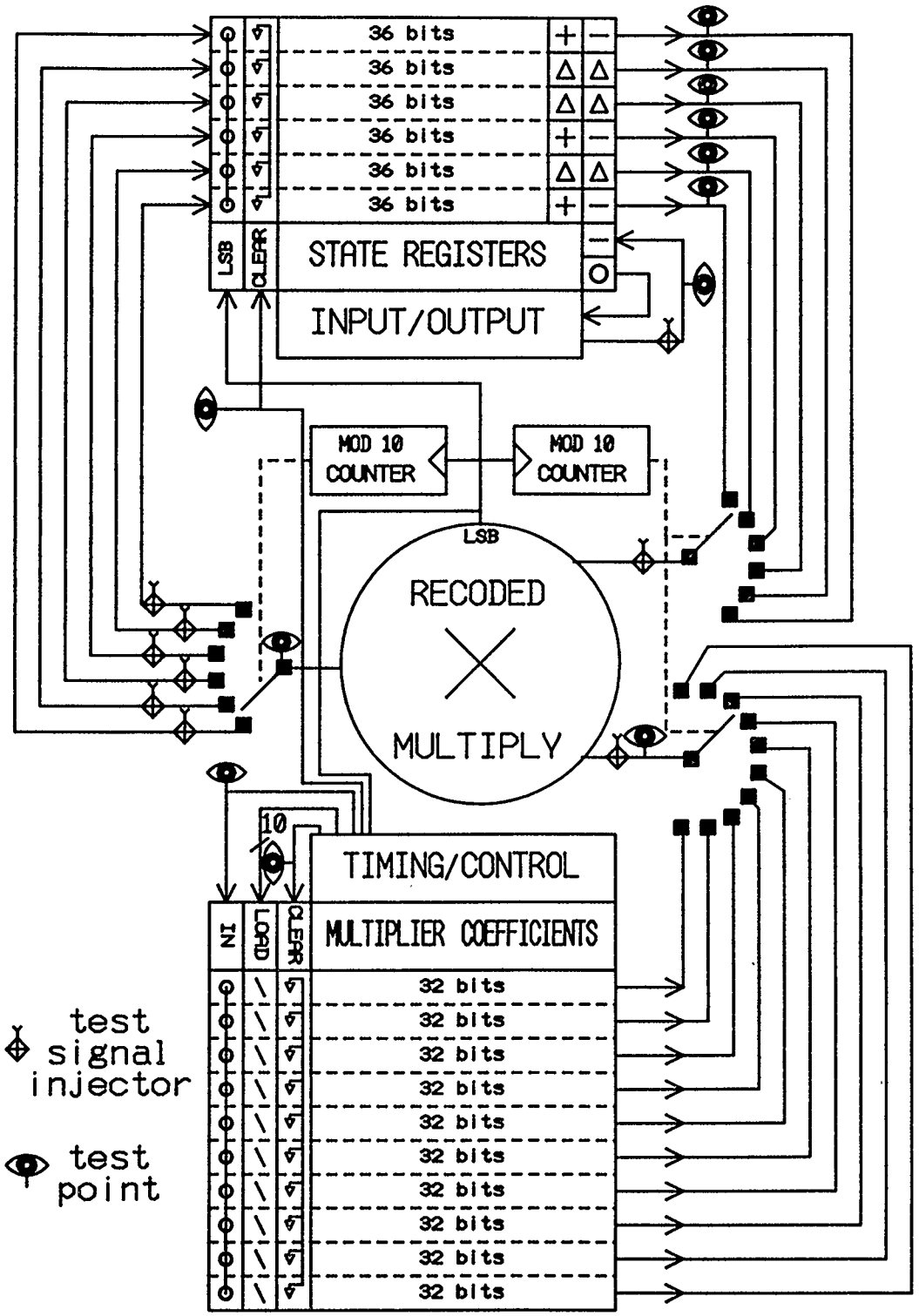


Figure 6.3 : Functional Block Diagram

cut, the term interface circuitry here includes the circuitry required to implement the non-recursive input section of the filter as well as the timing circuitry for the filter. It was desired to keep the filter interface requirements as simple as possible for ease of use. Basic control of the filter including loading of coefficients is accomplished through 8 register data lines and 3 control lines as shown in figure 6.4. A register address counter allows sequential access to the coefficient address register, the control register, and the four coefficient byte registers which are loaded to form a single multiplier coefficient. The three control lines RAR, RAI, and RLD respectively reset the register address to zero, increment the register address (or wrap it back to zero), and cause the value on the register data lines to be loaded into the addressed register.

The most significant four bits of the value loaded into the control register cause the actions indicated in figure 6.4 to be performed if the corresponding bit is set, while the least significant four bits of the control value specify a shift value which determines the position of the 16-bit input and output values within the 36-bit internal word. If the specified shift value is x, the LSB of the input/output data is aligned with bit $x+4$ in the internal word (where the internal bit numbers range from 0 to 35), so that the maximum shift value of 15 places the input/output data as high as possible in the internal word while still leaving room for the sign duplication bit required for proper multiplication. Increasing the shift value reduces the amount of quantization noise at the output while increasing the likelihood of internal overflow. The reset operation causes the filter to reset the internal timing circuitry and clear all the states and coefficients. The states and coefficients can be independently cleared with the clear states and clear coefficients bits in the control register. Setting the coefficient write bit in the control register causes the coefficient value in registers 0 through 3 to be loaded into the filter coefficient whose number is in register 4. The coefficient write operation is synchronized so that the filter coefficients are not altered in the middle of a multiplication operation. (Note that the control register bits do not have to be explicitly reset - the actions are performed only once for each load of the control register).

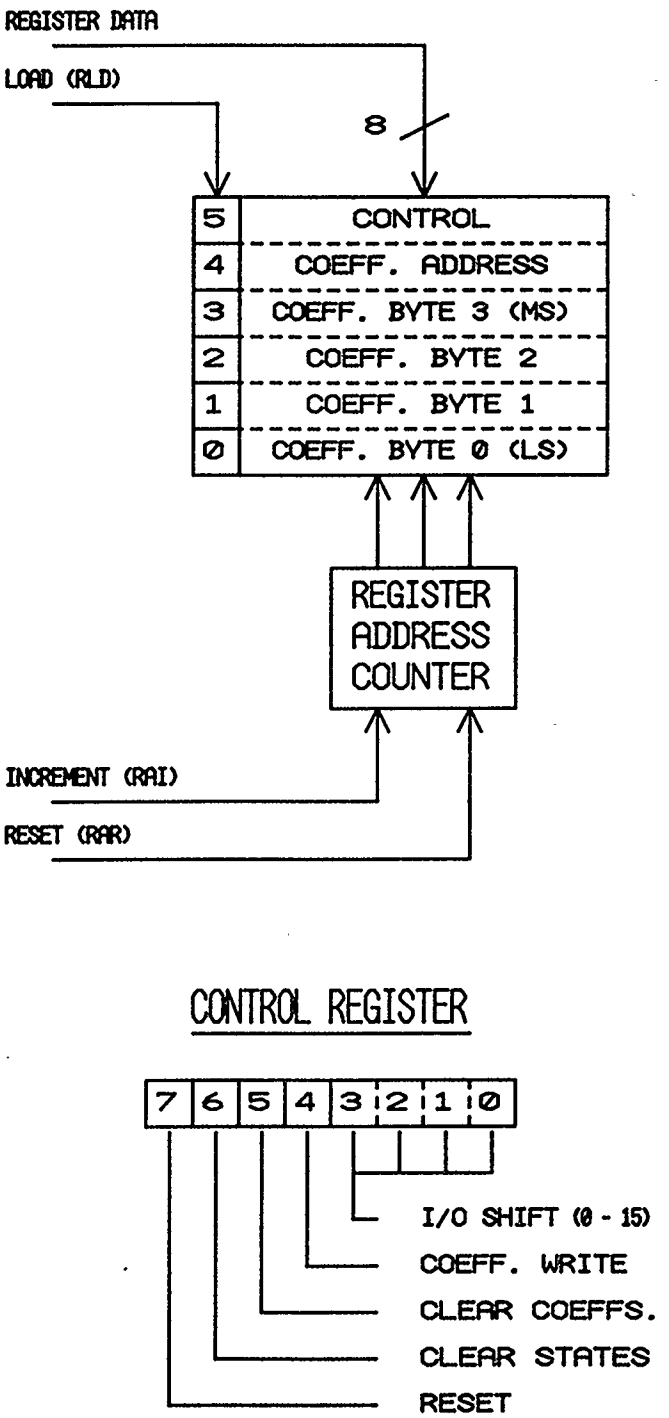


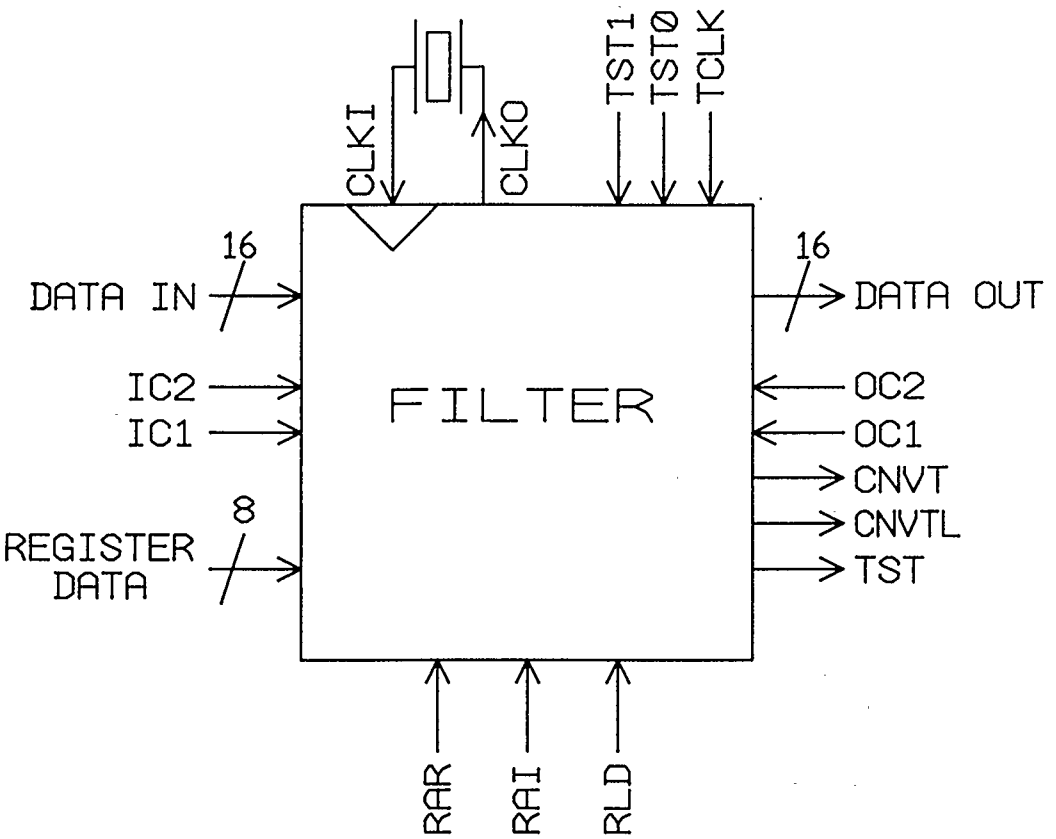
Figure 6.4 : Programming Model

The complete filter interface is shown in figure 6.5. Input and output data is in parallel format, and synchronized with the convert signals CNVT and CNVTL (active high and active low respectively) so that input data is loaded when the convert signal is active, and output data is valid at that time. Input pins IC1, IC2, OC1, and OC2 allow the filter input and output data formats to be independently selected from the set shown in figure 6.5. Input pins TST0 and TST1 select the filter test modes. The TST output is used for input voltage threshold characterization, and the two clock connections CLKI and CLKO allow clock generation using a single crystal. Alternatively, the CLKI input can be driven directly with a clock signal if desired. Finally, the TCLK input is used for high-speed testing as discussed in the following section.

The final filter implementation including all circuitry required for testability and interface required 7816 gate equivalents, and occupies an area of approximately 1 cm^2. Figure 6.6 shows a photograph of a complete fabricated filter die, and an enlarged view of a small region on the die. The filter circuit is produced by depositing two layers of metal interconnections on a predefined arrangement of transistors. The transistors are arranged on the die in 31 columns separated by wiring channels. The wiring channels can be seen as the dark vertical bands in figure 6.6 (a). Gate level modules are formed by interconnections of transistors within the transistor columns, while gate interconnections to form the final circuit are placed in the wiring channels. The small section of the die shown in figure 6.6 (b) has been configured to form two unit delays, which are formed by the blocks of interconnection wiring (white in the figure) over the pink transistor gates on either side of the wiring channel (the vertical grey band). The schematic diagram of a unit delay is shown in figure 6.7.

6.3. Implementation Details

The detailed design of the filter was undertaken with the aid of a gate-level simulator tailored to the particular gate array series in which the filter was to be implemented [44]. The primary use made of this simulator was to verify the gross functionality of the filter design



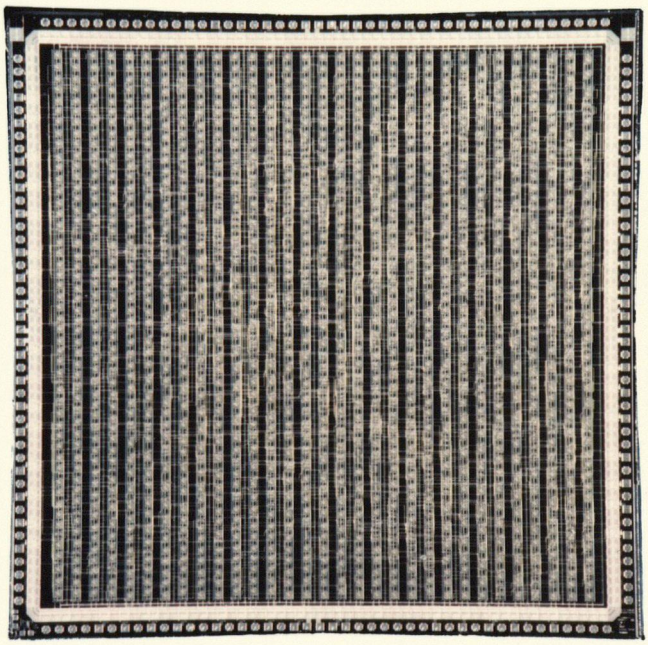
INPUT/OUTPUT CODE

IC2/OC2 IC1/OC1	INTERPRETATION
∅∅	two's complement
∅1	complementary offset binary
1∅	offset binary
11	complementary two's complement

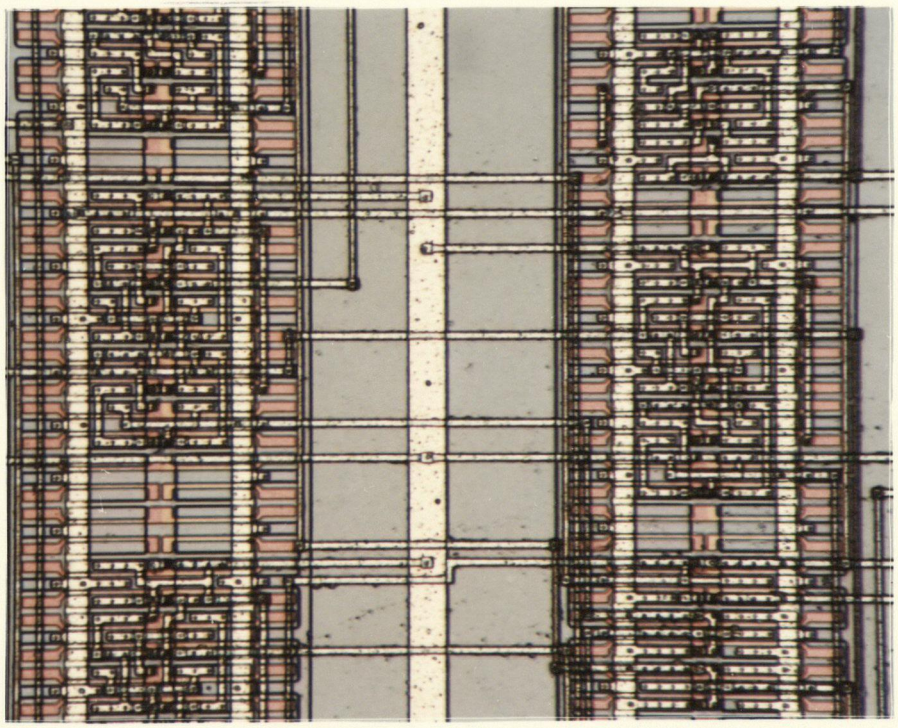
TEST MODES

TST1 TST0	INTERPRETATION
∅∅	normal
∅1	timing/multiply
1∅	states
11	not used

Figure 6.5 : Filter Interface



(a)



(b)

Figure 6.6 : (a) Filter Chip
(b) Two Unit Delays

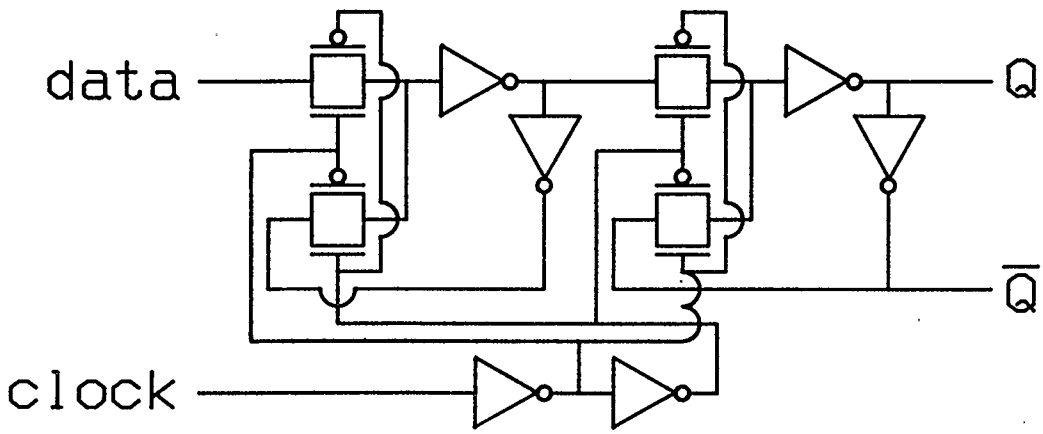


Figure 6.7 : Unit Delay Schematic

and to indicate potential timing violations for the individual gates in the design. It was claimed by the producers of the simulator that a device which simulated successfully could be fabricated to perform to at least the level simulated. With the aid of the simulator the detailed design of the filter was enhanced to the point that the bit-rate clock frequency could be as high as 40 MHz. Attaining this high clock rate required subtle modifications to the initial design in order to prevent gate timing violations. The serial adder circuit shown in figure 6.8 is one example of such a modification. (Note that figures 6.8, 6.9, and 6.10 are reproduced directly from the development software and show the exact form in which the circuit is specified.) Comparing this final form of the serial adder with the original form presented in figure 5.3 it can be seen that the circuit has been modified to include a gate in the carry propagation path (U7) to clear the carry bit rather than using the asynchronous clear as in figure 5.3, and that an inverting buffer (U6) has been added to the carry feedback path. The modification to the carry clear circuitry is required due to the difficulty of generating a clear carry signal which is shorter than one period of the bit-rate clock. The clear carry signal must be derived from the timing circuitry which generates the bit-rate clock, and hence typically lags the leading edge of the clock by a few nanoseconds. When the clear carry signal is the same length as one period of the bit-rate clock, but applied asynchronously as in figure 5.3 and skewed late as described, it will cause the carry bit to be incorrectly cleared for one additional bit period. This is because the skewed clear carry signal will overlap the rising edge of the bit-rate clock for the second bit in the sum. This problem might be overcome by appropriately delaying the bit-rate clock distributed to the serial adders, but this leaves the potential for a similar problem during computation of the most-significant bit of the sum. In any case, such a solution depends critically on the delays in individual gates, and will be extremely sensitive to process variations.

A much better solution is the one adopted in the circuit of figure 6.8, in which the carry bit is cleared prior to the carry storage element, so that an overlap of the clear carry signal beyond the bit-rate clock edge can be tolerated. By adding a gate in the carry propagation

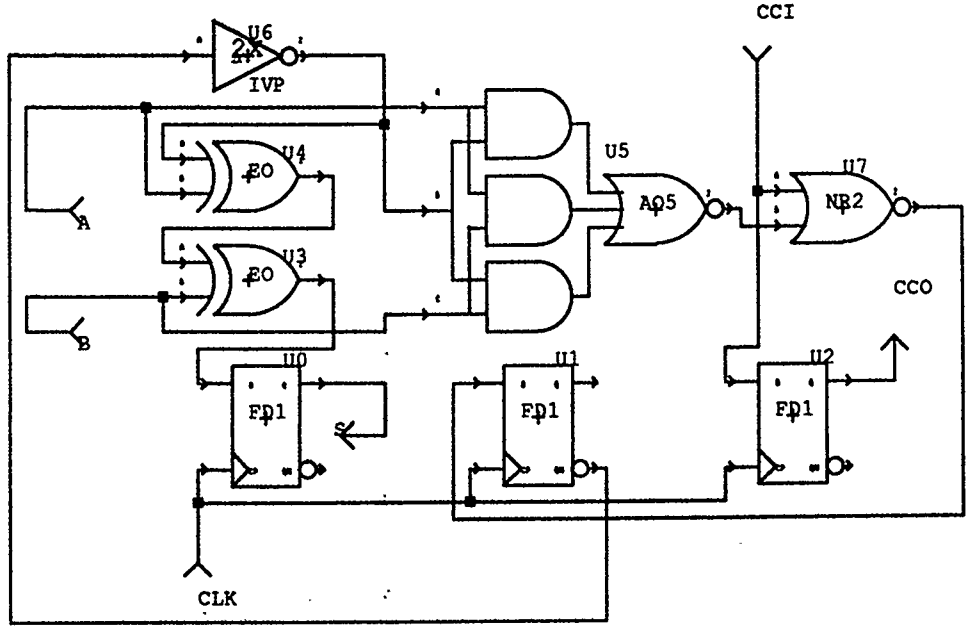


Figure 6.8 : Final Serial Adder

path however, the propagation delay around the loop was increased enough that the desired 40 MHz clock rate could not be achieved. One factor contributing to this situation is that the carry storage element (U1) has a fairly low output drive capability, and the exclusive OR input to which it feeds (U4) is a high load input. This problem is solved by introducing the inverting buffer (U6), which has a high drive output, but a low load input. Although the inclusion of this buffer actually adds one gate to the carry propagation path, the change in loadings and drive capabilities results in a net reduction of the propagation time for the carry signal so that the 40 MHz bit rate is attainable.

A similar problem arose in the add/subtract modules of the serial multiplier. The final circuit for this module is shown in figure 6.9. In this case, in order to allow sign extension, the sum/difference storage element (U4) has a two-input multiplexer for its input. This additional input circuitry increases the required setup time for the element to the point where two cascaded exclusive OR gates cannot be used to form the sum/difference. Instead, a three input exclusive OR gate is used (U0), but this gate has such a high input loading that an additional stage of buffering is required in the carry path. This buffering is included in U1, which is actually the cascade of two inverting buffers, the second with more output drive than the first. In addition, to select whether the sum or difference will be formed, the exclusive OR gate U3 is added to the basic adder form. This gate has a high enough input loading that fast enough propagation of the sum/difference from the previous stage through the carry generation circuitry requires that the stages generate the sum/difference output at two points. The inverting buffer U7 allows the output sum/difference to be generated faster than it could be if only the non-inverting output of U4 was used. The reasons for this are twofold: by including U7, the total load that the non-inverting output of U4 drives is reduced which improves the speed of this output, while the inverting output which drives U7 changes before the non-inverting output by the amount of delay introduced by one inversion internal to U4. In this way, the time critical but relatively light load carry path of the subsequent add/subtract stage is driven by the output of U7 slightly faster than the heavier load but less

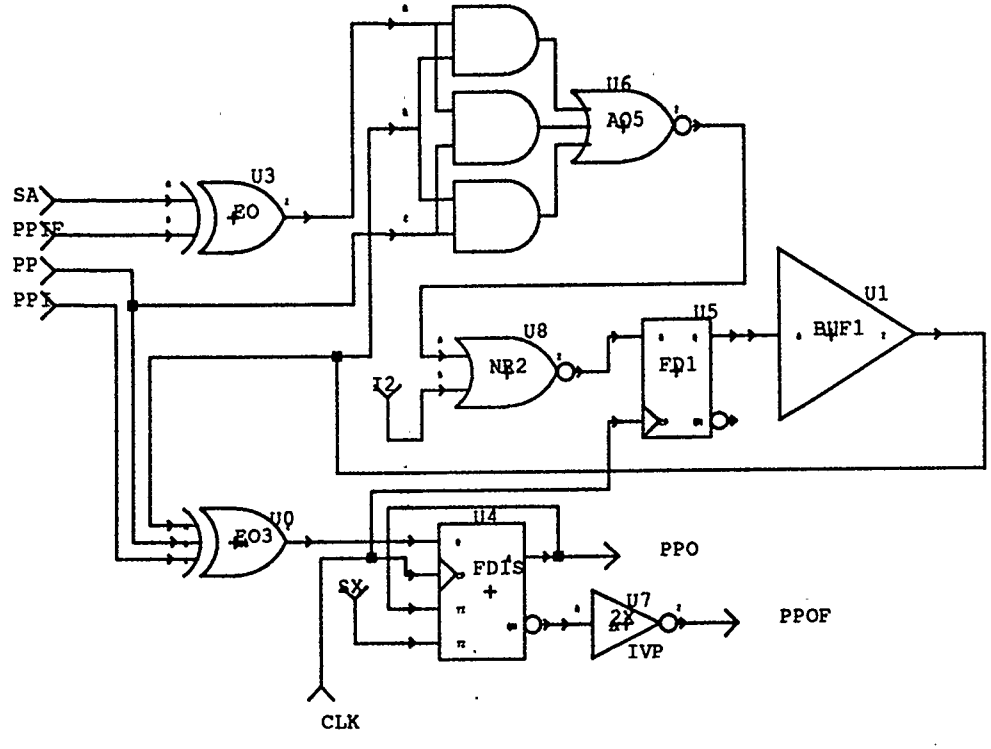


Figure 6.9 : Add/Subtract Module

time critical sum/difference forming circuitry driven by the non-inverting output of U4, and the 40 MHz bit rate is attainable.

Because the final filter design successfully simulated at bit-rate clock speeds as high as 40 MHz, it was expected that the fabricated filter would perform at that speed. Unfortunately however, the automated tester used for testing the fabricated chips is not capable of applying a continuous clock at frequencies above 10 MHz. The tester does however have the capability to apply single pulses as narrow as 8 ns, and this capability was used to perform limited automatic testing of the circuit speed. During the design and simulation of the filter, it became apparent that the most time critical module is the add/subtract stage shown in figure 6.9. This module was therefore embedded in the test circuit shown in figure 6.10, which was included in the final filter implementation as a separate circuit section.

In this circuit the single clock input (CLK) is buffered through a series of inverters so that the input values (on the lines entering from the left) are clocked in on the rising edge of CLK, while the output values (on the lines exiting to the right) are latched on the falling edge of CLK. The input circuitry is configured in exactly the same form as the circuitry which drives the critical add/subtract modules in the multiplier. Using this configuration test patterns can be set up on the input to the test circuit, a narrow pulse applied to the CLK input, and the output values examined to verify correct computation. By setting the pulse width to be the smallest clock period which does not violate any gate timings in simulation, the high speed performance of the test circuit can be evaluated by the automatic tester. The CLK input to the test circuit is connected to the TCLK input of figure 6.5, and the inputs/outputs can be applied/examined when the filter is in test mode.

Various other modifications were required to ensure that the 40 MHz bit rate could be attained, including the pipelining of some originally non-pipelined operations. Perhaps the greatest difficulty associated with the high clock rate however was the distribution of the bit-rate clock to all of the clocked gates in the filter. In order to minimize clock skew, the clock

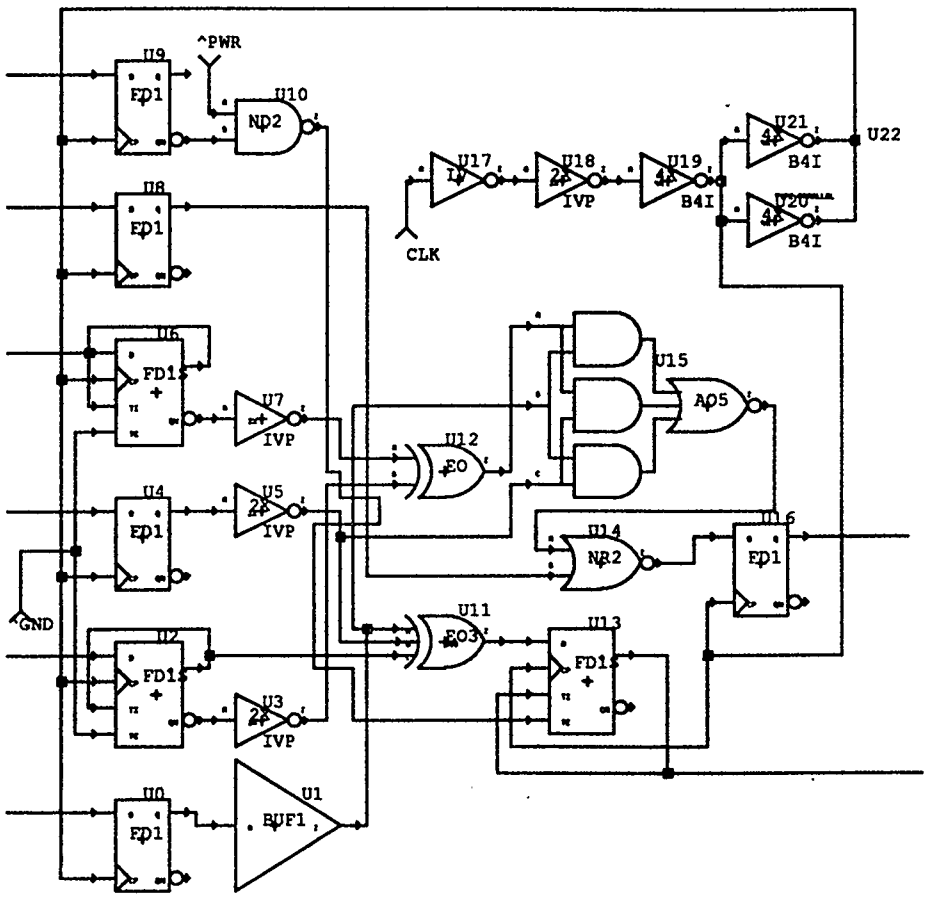


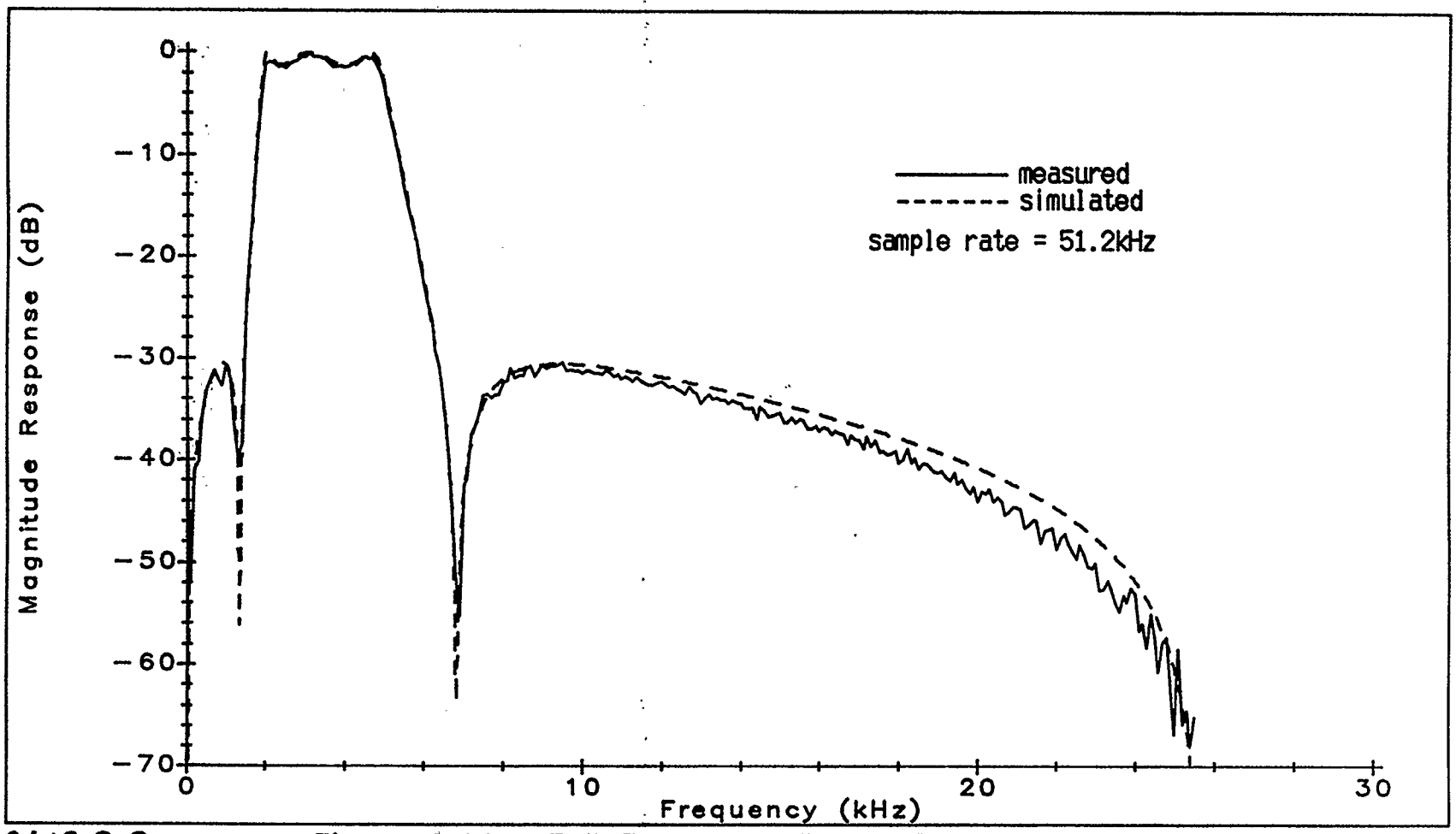
Figure 6.10 : High Speed Test Circuit

distribution lines (which can be seen in figure 6.6) were laid out manually in wide metal in every second wiring channel. The vertical clock distribution lines were connected horizontally at the top and bottom of the die (also in wide metal), and these horizontal busses were connected to 14 high drive clock buffers distributed across the top and bottom of the array in the input/output region (the bright band encircling the array). In figure 6.6 (b) a section of a vertical clock distribution line can be seen in the center of the figure. Because the clock lines were manually placed, each cell which required a clock connection was manually connected to the clock distribution lines before the remainder of the circuitry could be automatically routed. This manual placement and connection was a very time consuming process.

6.4. Measured Results

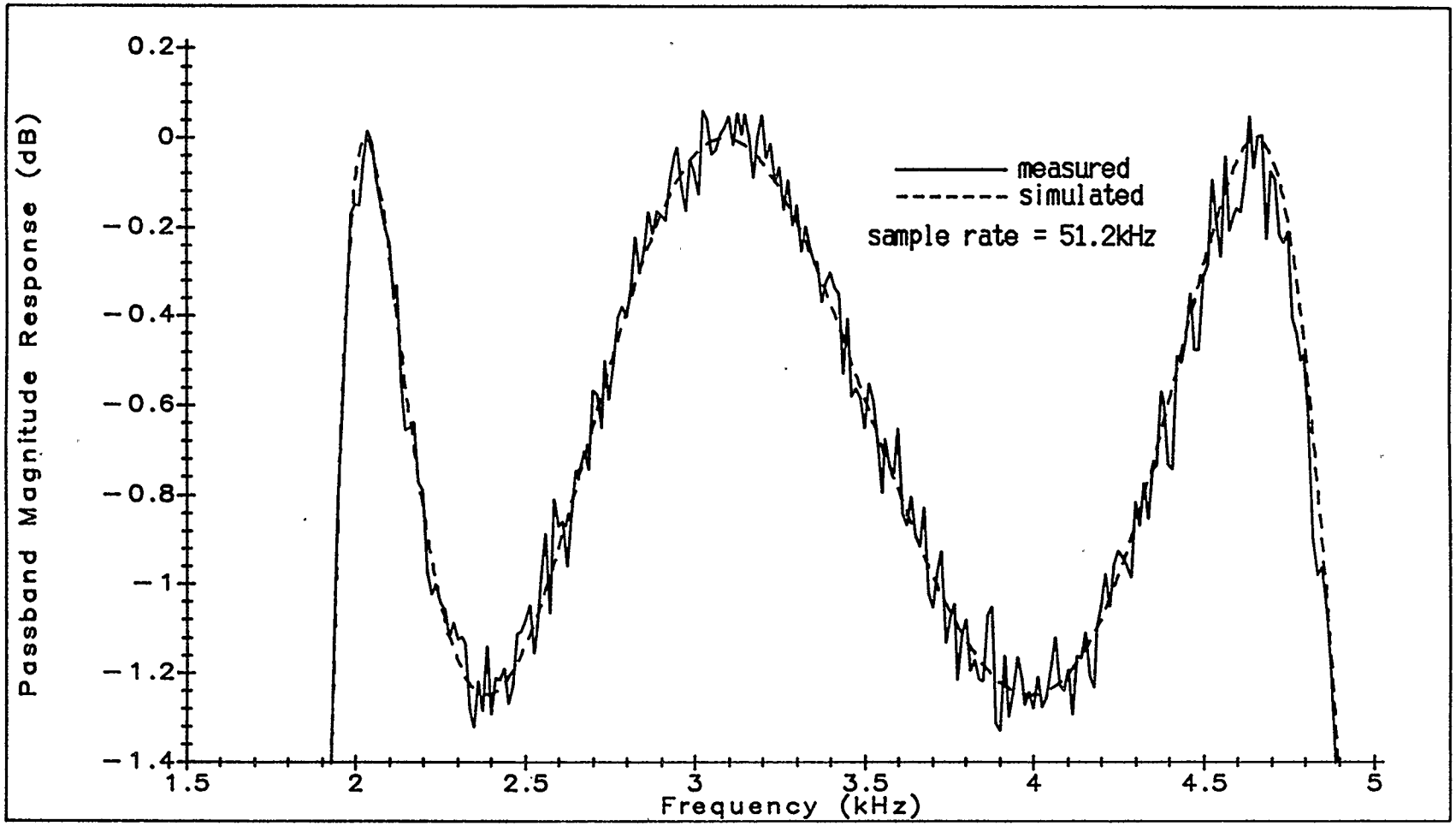
A simple facility for testing the packaged filters was built which included a sample/hold, analog-to-digital converter, and digital-to-analog converter, and allowed the filter to be programmed and exercised under the control of a Motorola 6809 microprocessor. A software package was developed for the test facility which allowed the user to individually program the filter coefficients, exercise the filter control functions, select the input/output formats, and sequence through sets of filter coefficients. Using a Hewlett-Packard Structural Dynamics Analyzer (SDA) the transfer function of the filter under test was measured.

Comparison to simulated results showed excellent agreement as can be seen from the example in figures 6.11 and 6.12. In figure 6.11, the "droop" of the measured response at high frequencies is due to the $\sin(x)/x$ roll-off introduced by the sample-and-hold operation. This can be seen from figure 6.13, in which the simulated results have been adjusted to include this roll-off. Figures 6.14 and 6.15 respectively show the power spectral density as measured by the SDA over the full frequency range and in the passband for the example filter function of figure 6.11. In both cases the filter was excited with a maximum amplitude sinusoidal signal within the passband, and in both cases the signal-to-noise ratio is at least as great as the 72 dB which can be expected from the 12-bit converters used. Unfortunately



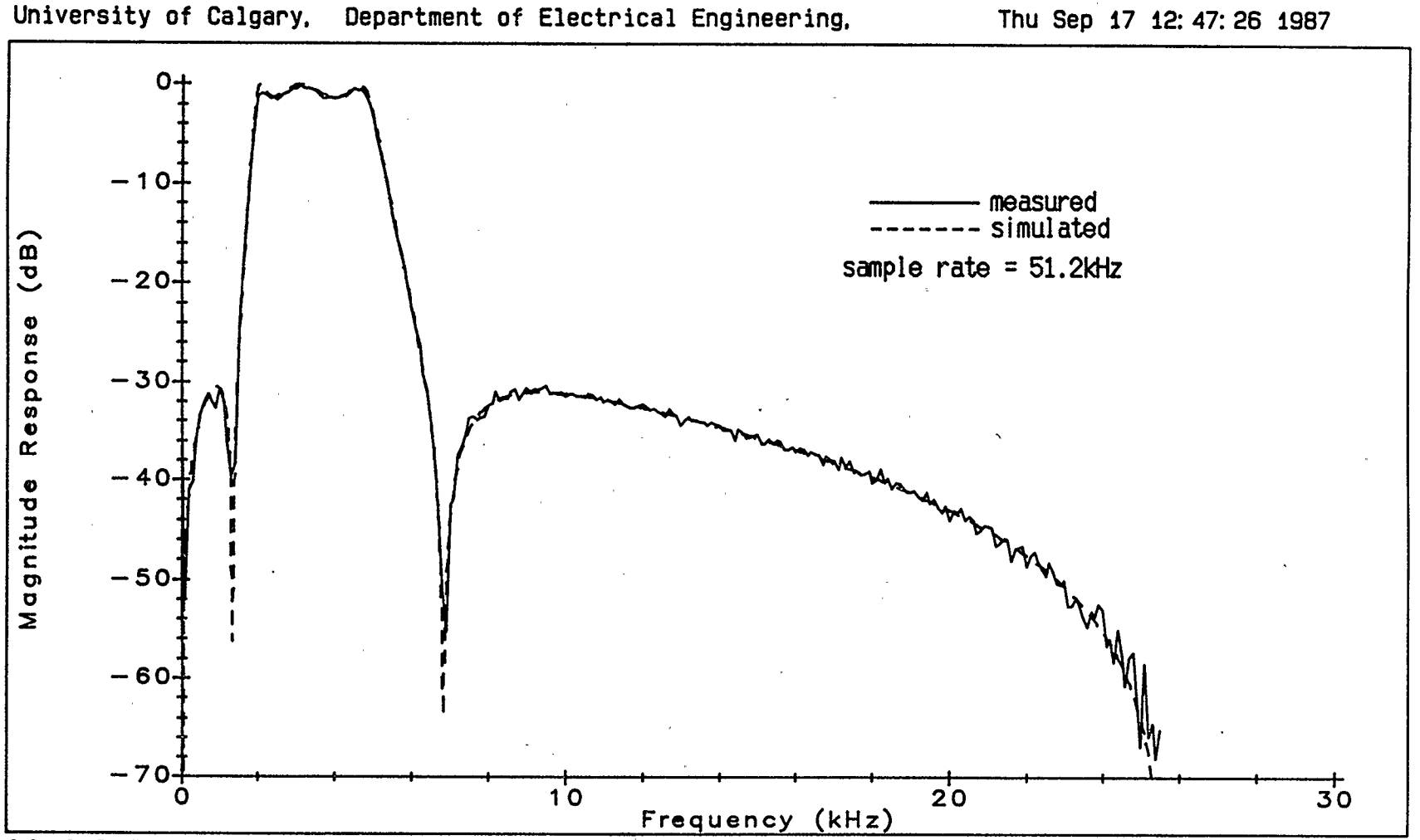
UCEE

Figure 6.11 : Full Frequency Range Response Comparison



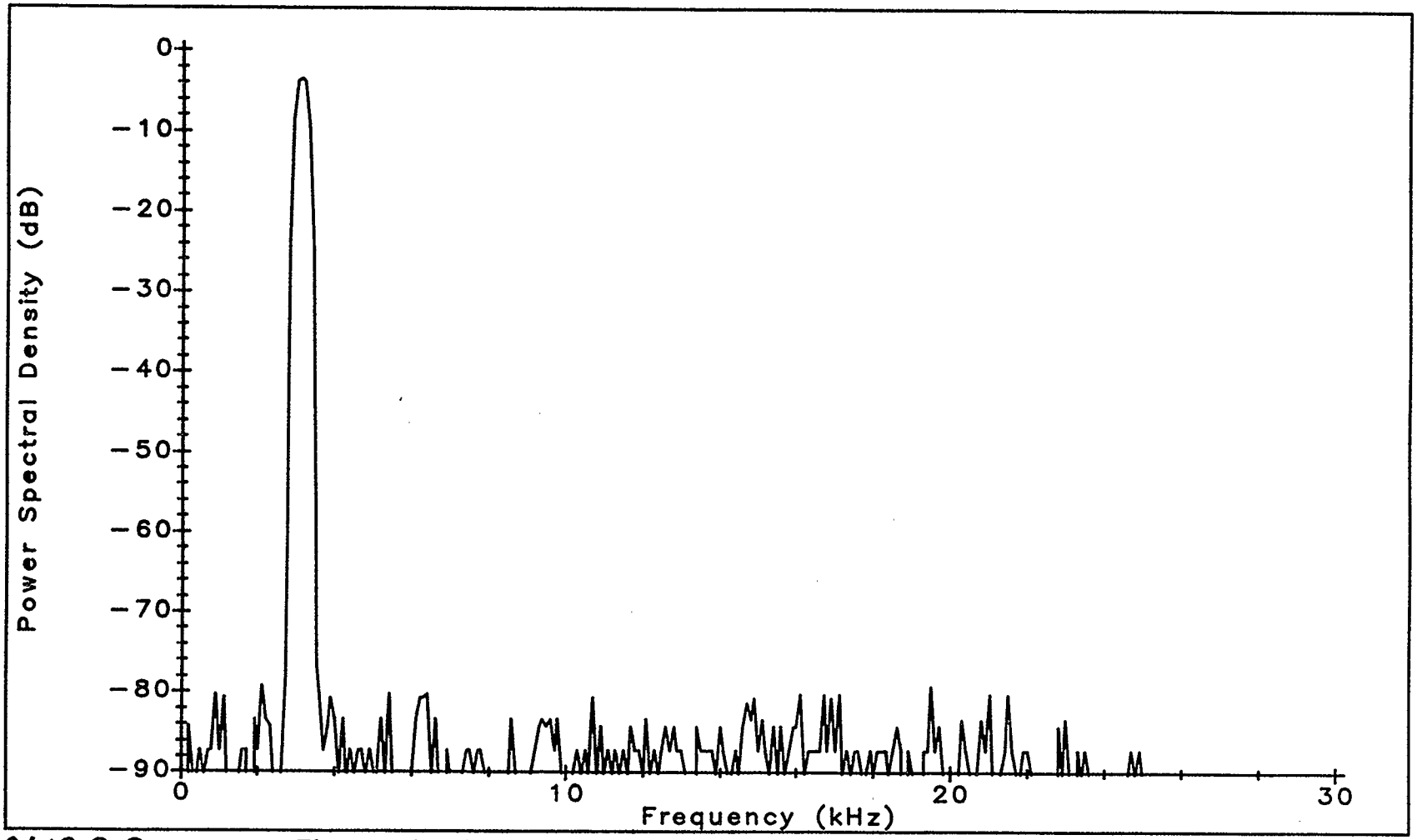
UCEE

Figure 6.12 : Passband Response Comparison



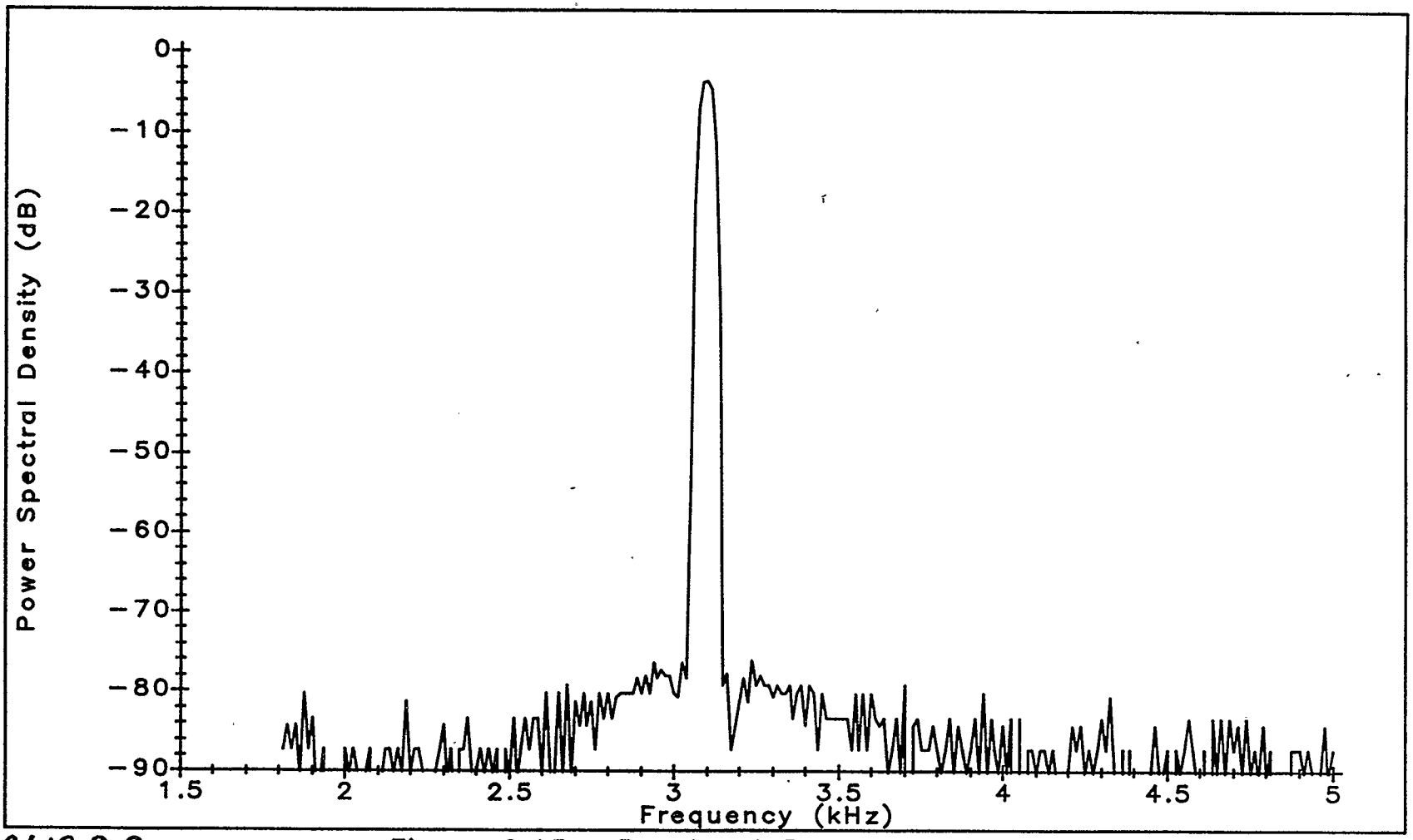
UCEE

Figure 6.13 : sin(x)/x Corrected Response Comparison



UCEE

Figure 6.14 : Full Frequency Range Power Spectral Density



UCEE

Figure 6.15 : Passband Power Spectral Density

no suitable 16-bit converters were available for testing the filter. Of the 19 filters which passed the automated testing, only 3 failed to meet the 40 MHz design goal (one of these may have been damaged by abusive input clocking), and 12 operated at bit-rate clock frequencies greater than 60 MHz.

CHAPTER 7

CONCLUSION

By the production of a functional single-chip programmable digital filter, the major goal of the research described in this thesis was achieved. During the development of the filter, various concepts and possibilities for alternative designs were recognized as candidates for further study. Both the degree of fulfillment of secondary research goals and topics for further study are discussed in this chapter.

7.1. Achievement of Research Goals

In addition to simply accommodating a digital filter on a single chip, stated goals of the research were to allow 16-bit input and output data, maximize filter throughput, and minimize non-ideal effects. Capability for 16-bit input and output was incorporated into the basic filter design once it became apparent that this would not preclude the possibility of successfully integrating the filter on a single chip. Efforts were made to maximize the throughput by pipelining the entire design and by making subtle circuit enhancements (chapter six) which allowed successful simulation at internal bit-clock rates as high as 40 MHz. These efforts were clearly very effective with the result that some of the units fabricated operated at internal clock rates as high as 65 MHz. The resulting 16-bit sample rate in excess of 180 kHz is more than adequate for high-quality audio range filtering maintaining ten or more samples per cycle at band edges up to 18 kHz. In fact it is difficult to find true 16-bit analog-to-digital converters which are capable of operating at frequencies as high as 180 kHz.

By compromising between a completely dedicated design implementing only one fixed transfer function and a general purpose digital signal processor as the TMS320 [46], the programmable digital filter gains a substantial speed advantage over the TMS320, while still

retaining the capability of implementing multiple transfer functions. Although the TMS320 does not have sufficient internal word length to completely duplicate the operation of the filter, it can be programmed to implement the sixth order elliptic bandpass filter with somewhat degraded non-ideal performance (compared to the filter developed in this research). When programmed in this fashion, the TMS320 is capable of a maximum sample rate of 75 kHz - roughly one half the speed attainable with the single-chip filter. In terms of programming flexibility, the range of bandpass functions which the programmable filter can implement is limited only by the non-ideal effects of multiplier coefficient quantization, signal overflow, and signal quantization.

The goal of minimizing non-ideal effects in the single-chip filter was achieved through three features of the design. Firstly, 32-bit multiplier coefficients were incorporated into the design allowing sufficient accuracy to implement all of the representative transfer functions with zero relative error. Secondly, the design incorporates a 36-bit internal word length so that internal overflow is avoided. Thirdly, programmable placement of the 16-bit input/output within the 36-bit internal word length assures that noise resulting from internal signal quantization contributes less than one LSB of noise at the filter output. Although the level of freedom from non-ideal effects described above cannot be guaranteed for all transfer functions which could possibly be implemented with the filter, a broad enough range of transfer functions was included in the representative set that non-ideal effects should not be significant for practical applications of the filter.

In addition to the explicitly targeted features described above, other positive aspects of the final filter implementation are comprehensive testability and a simple interface, as well as a capability to implement both lowpass and bandpass type transfer functions through the appropriate choice of multiplier coefficients. As mentioned in chapter six, the bit-serial nature of the filter architecture allowed simple and effective test circuitry to be introduced. This enabled the filter to be rapidly and completely tested by an automatic tester. The filter digital

data interface accommodates common converter codes with no additional hardware, and the filter is easily controlled using a pair of 8-bit parallel unidirectional interfaces. Also for ease of interface, the filter clock can be provided either by directly driving the clock input, or by simply connecting a suitable crystal between the two filter pins CLKI and CLKO. Finally, by retaining the capability to independently set all ten multiplier coefficients (where the analog prototype of figure 1.2 dictates that only six unique coefficients are required), additional filtering functions may also be available [47].

The choice of which structure to use for the filter implementation was made largely on qualitative grounds as discussed in chapter 5. Using detailed information from the actual implementation, it is possible to make more accurate estimates of the hardware cost of each of the candidate structures. This was done by determining the hardware overhead for control/interface, steering, and testing circuitry in the actual implementation, and adding this overhead to the hardware requirements of table 5.6. Since the multiplexing scheme for the LDI structure is very straightforward and the minimum interface requirements for each structure would be comparable, the figures determined in this manner are reasonable lower bounds on the true hardware requirements for each structure. Using these figures and the maximum potential throughput values shown in table 5.7, a figure of merit was computed for each structure. This figure of merit is proportional to the maximum potential throughput divided by the hardware requirement (in gate equivalents) and is shown for each structure in figure 7.1. From the figure it can be seen that the WDF structure has the highest figure of merit, followed closely by the LDI structure. The remaining structures have substantially lower figures of merit. Since these figures of merit represent upper bounds for all but the LDI structure (which is an *actual* value), they support the choice of either the LDI or WDF structures for the implementation. However a simple multiplexing scheme has not been determined for the WDF structure, hence it is quite possible that any multiplexing scheme would require multiplier idle time, in which case the figure of merit for the WDF structure could easily be reduced below that of the LDI structure. Whether or not this is the case falls

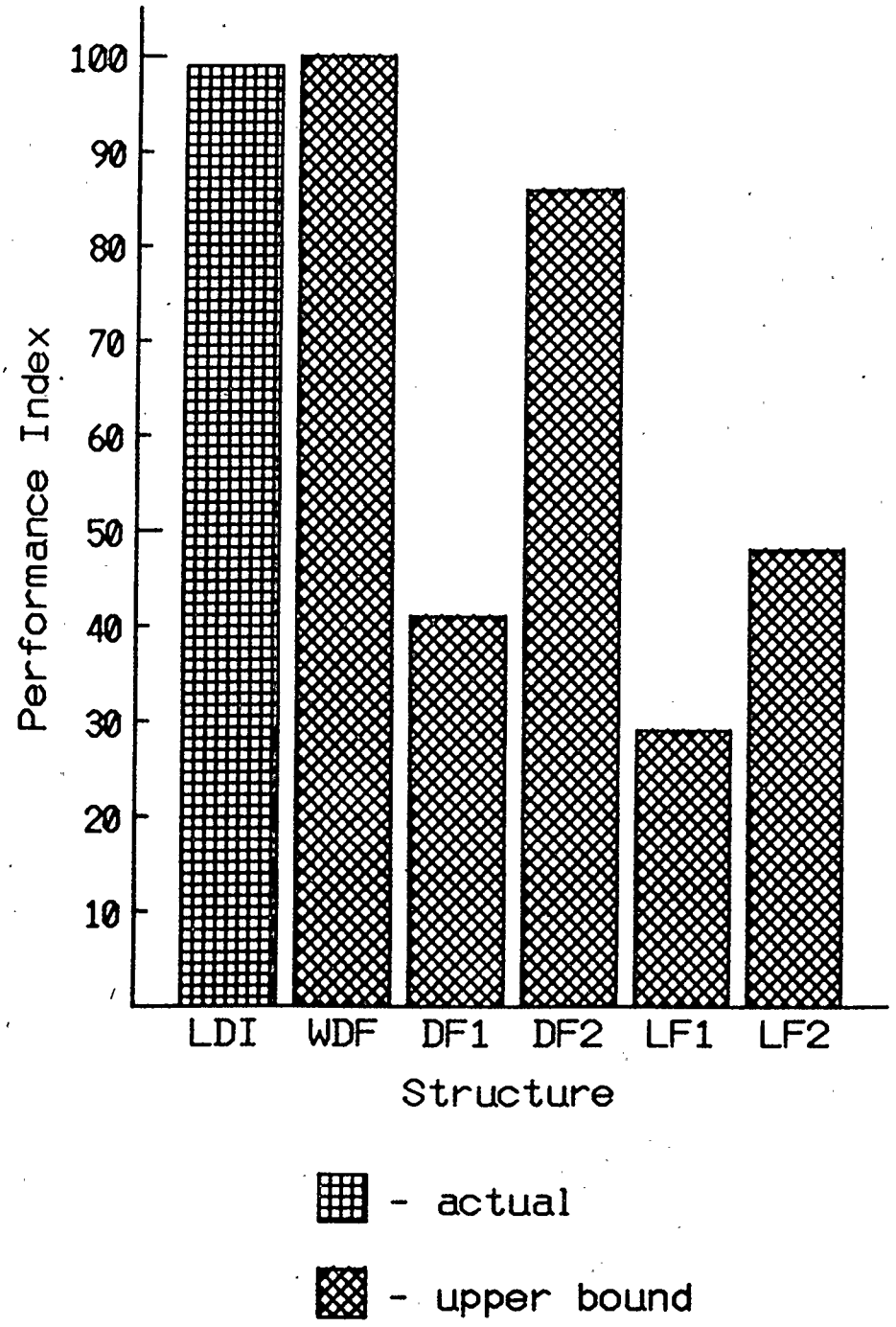


Figure 7.1 : Structure Performance Indices

under the more general algorithm partitioning problem discussed in the following section. In any case, it is important to note that the LDI structure is favored by a number of factors:

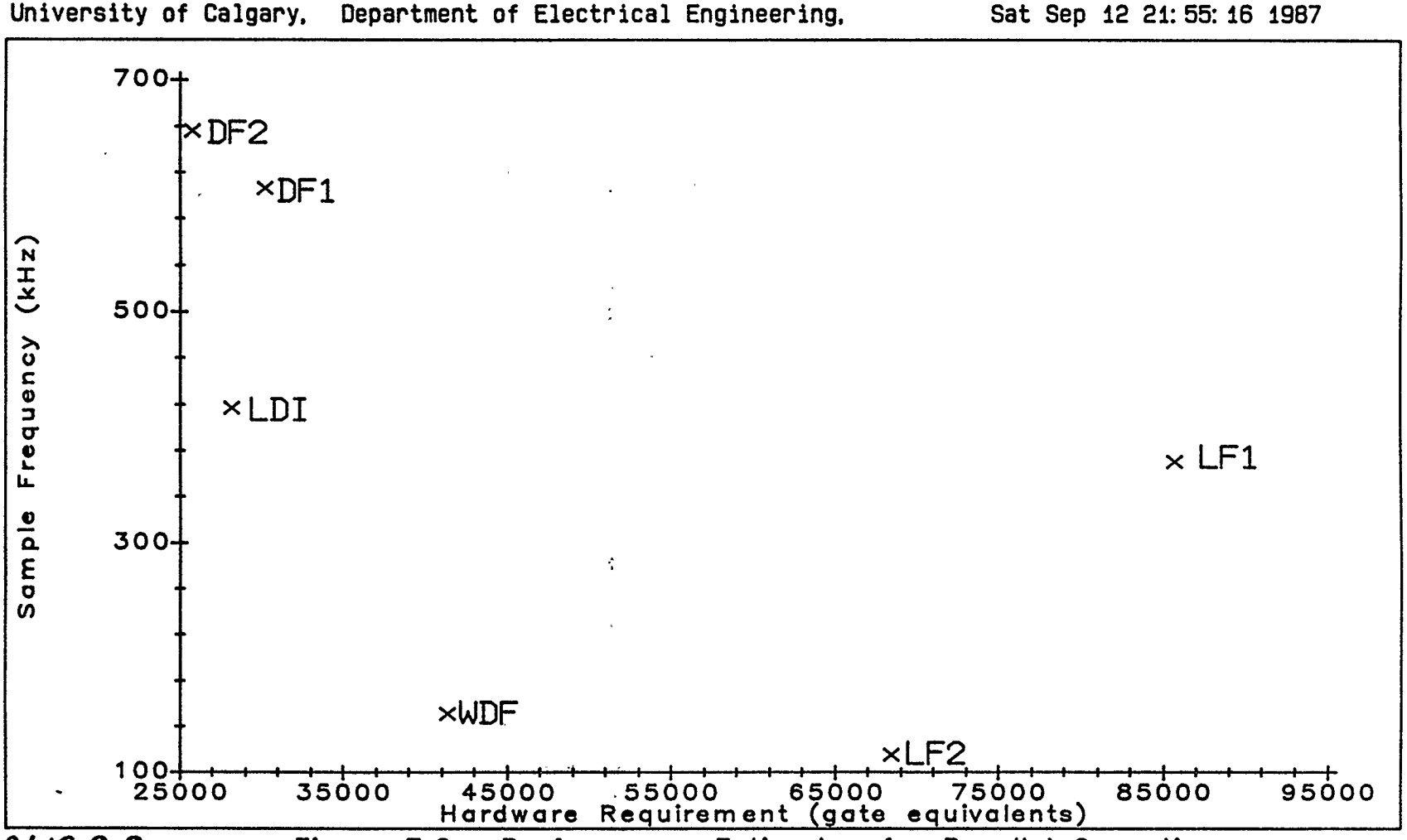
- The LDI structure has the smallest lower bound hardware requirement of all the structures, hence is the best candidate for implementation within the fixed hardware capacity imposed by the implementation technology, based on the fact that many hardware requirements will inevitably remain unrecognized until the final detailed design of a filter is complete.
- A known compact and efficient multiplexing scheme exists for the LDI structure based on a clear regularity and modularity of the structure.
- The LDI structure has the highest potential throughput of all but the WDF structures.
- The LDI structure does not exhibit the numerical difficulties that the DF1, DF2, LF1, and LF2 structures do with respect to computation of multiplier coefficients for some of the representative transfer functions.

7.2. Subjects for Further Study

The single greatest difficulty revealed by the research described in this thesis is that of determining which of a multitude of possible implementations for a filter is "best". It has become clear to the researchers that there is no way to make this determination in general - it depends too much on the particular requirements of a given application. For example, although the TMS320 is slower and has worse non-ideal performance than the filter developed in this research, it would clearly be superior for an application in which speed was not critical and bandpass filtering was only a small part of the signal processing task. In addition, differences in the technology of implementation make it difficult to compare filters or processors on a performance per unit area basis. What is clear however is that for any particular filtering task, tradeoffs exist between the silicon area used and the way in which the filtering algorithm is partitioned (this in turn influences the potential speed of the imple-

mentation). At one extreme, all the arithmetic operations for a filter may be carried out in parallel, and at the other, a single arithmetic unit may be multiplexed over all the filter operations. Between these two extremes lie intermediate possibilities such as performing in parallel the operations required to update each filter state, but updating the states sequentially. The range of possibilities is potentially endless, and the performance achieved and hardware required for a particular partitioning of the filtering algorithm is highly dependent upon the structure chosen. Simply defining partitions of the algorithm is difficult to do in many instances. For example, the WDF structure considered in this research would be very competitive with the LDI filter developed if a simple and efficient way to multiplex the multiplication operations could be determined, yet repeated examination has revealed no such efficient scheme.

While the implementation technology made available for this research largely made the foregoing problems of no consequence (the size limitation essentially forced the method of partitioning the filter algorithm), different technologies and potential improvements in integration densities make the problem of how to partition filter algorithms an important one. Preliminary work has been performed on this problem directed towards eventually being able to efficiently reduce a filter specification directly to a functional circuit within the constraints of size and speed dictated by the particular application [38]. From the results of this work it was possible to produce estimates for the size and speed of the candidate structures if they were implemented in gate array technology with no multiplier multiplexing (i.e. all the filter operations occur in parallel). These estimates are shown in figure 7.2. For comparison, speed and size of the multiplexed structures are presented in the same format in figure 7.3. In both these figures, it is assumed that the technology of implementation is gate array, and that the maximum bit-clock rate is 40 MHz (the nominal design speed for the programmable filter). Further study in this area holds the promise of a more formalized and consistent method of determining and implementing suitable partitions of filtering algorithms.



UCEE Figure 7.2 : Performance Estimates for Parallel Operations

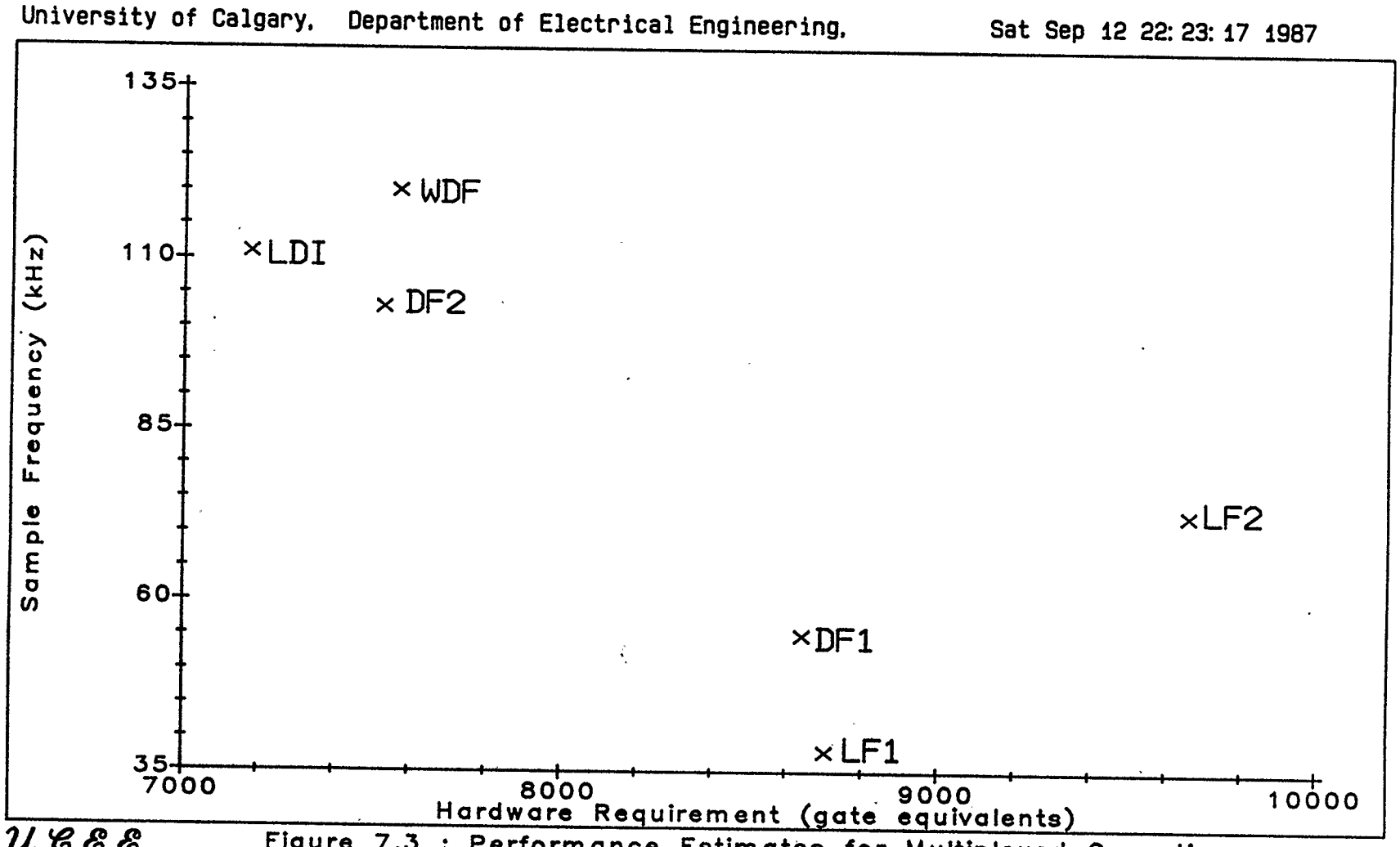


Figure 7.3 : Performance Estimates for Multiplexed Operations

In addition to the problem of partitioning filter algorithms, an outstanding problem is that of quantitatively comparing potential filter implementations both within and across algorithm partitions. As mentioned above, this is likely to be nearly impossible without at least keeping the technology of implementation as a constant. Even with this constraint, it is difficult to arrive at a figure of merit which adequately describes the various benefits and/or shortcomings of different potential implementations. There are many important factors which are inherently qualitative such as degree of programmability, ease of cascading or expansion, and regularity of structure (which is important from the design and simulation standpoint). In addition, important quantitative factors such as non-ideal performance seem to defy succinct description when the application for the filter or processor is not strictly defined (such as for the programmable filter developed in this research). Further study attempting to characterize and quantify the nature and structural dependence of non-ideal effects is warranted for this reason.

As pointed out in chapter 5, the final decision between candidate structures was made on the basis of the existence of an elegant multiplexing scheme. This scheme is largely suggested by the form of the LDI integrators. Preliminary research based on lattice analog prototypes (as distinct from lattice digital structures) indicates that there is a relationship between the LDI transformation and wave filter concepts. Initial investigation in this area has shown that the particular prototype filter used for this research can be converted into a lattice prototype which leads to an LDI filter implementation which has only six multiplication operations. This implementation requires the same number of coefficient and signal bits as the existing filter in order to implement all of the representative transfer functions with the same level of non-ideal effects, and also retains the elegant multiplexing scheme of its predecessor. Using this implementation, a speed improvement of 67% could immediately be realized along with a substantial reduction in hardware cost and inclusion of a simultaneous complementary (band-reject/highpass) filtering function. Because a large percentage of the elements required for bit serial filter implementations are registers (flip-flops), the

combination of compact implementations such as the one described above with full or semi-custom design in which the static registers of the gate-array technology could be replaced by smaller dynamic elements holds the potential for extremely compact high-performance digital filters.

References

[1] H.D. Helms and L.R. Rabiner (ed.), *Literature in Digital Signal Processing: Terminology and Permuted Title Index*, IEEE Press, New York, 1973.

[2] L. R. Rabiner and B. Gold, *Theory and Application of Digital Signal Processing*, Prentice-Hall, Inc., Englewood Cliffs, NJ, 1975.

[3] Y. Tsividis and P. Antognetti (ed.), *Design of MOS VLSI Circuits for Telecommunications*, Prentice-Hall, Inc., Englewood Cliffs, NJ, 1985.

[4] J. Van Ginderdeuren, H. De Man, B. De Loore, G. Van Den Audenaerde, "Application Specific Integrated Filters for HIFI Digital Audio Signal Processing", *1986 IEEE Int. Conf. Acoust., Speech, Signal Processing*, Tokyo, Japan, pp. 1537-1540.

[5] R. Jain, F. Catthoor, J. Vanhoof, B.J.S. De Loore, G. Goossens, N.F. Goncalvez, L.J.M. Claesen, J.K.J. Van Ginderdeurden, J. Vandewalle, H.J. De Man, "Custom Design of a VLSI PCM-FDM Transmultiplexer from System Specifications to Circuit Layout Using a Computer-Aided Design System", *IEEE J. Solid-State Circuits*, Vol. SC-21, No. 1, pp. 73-85, Feb. 1986.

[6] A. Peled, B. Liu, *Digital Signal Processing*, John Wiley & Sons, Inc., New York, 1976.

[7] A. Fettweis, "Digital Filter Structures Related to Classical Filter Networks", *Arch. Elektron. Uebertrag.*, vol. 25, pp. 79-89, 1971.

[8] A. Fettweis, "Digital Circuits and Systems", *IEEE Trans. Circuits Syst.*, Vol. CAS-31, No. 1, pp. 31-48, Jan. 1984.

[9] A. Antoniou, *Digital Filters: Analysis and Design*, McGraw-Hill, Inc. 1979.

[10] M.E. Van Valkenburg, *Network Analysis*, 3rd Ed., Prentice-Hall, Inc., Englewood Cliffs, NJ, 1974.

[11] A.H. Gray, Jr. and J.D. Markel, "Digital Lattice and Ladder Filter Synthesis", *IEEE Trans. Audio Electroacoust.*, Vol. AU-21, No. 6, pp. 491-500, Dec. 1973.

[12] L.T. Bruton, "Low Sensitivity Digital Ladder Filters", *IEEE Trans. Circuits Syst.*, Vol. CAS-22, No. 3, pp. 168-176, Mar. 1975.

[13] B.K. Ramesh, *A Method of Designing Digital LDI Ladder Filters Using the Bilinear Transformation*, Master's Thesis, University of Calgary, Calgary, Alberta, Canada, Dec. 1986.

[14] L.E. Turner and B.K. Ramesh, "Low Sensitivity Digital Ladder Filters with Elliptic Magnitude Response", *IEEE Trans. Circuits Syst.*, Vol. CAS-33, pp. 697-706, July 1986.

[15] B.K. Ramesh and L.E. Turner, "A Generalized Method of Designing Digital LDI Ladder Filters Using The Bilinear Transformation", *1986 IEEE Int. Symp. Circuits Syst.*, Vol 2., pp. 665-668, May 1986.

[16] G.C. Temes and H.J. Orchard, "First-Order Sensitivity and Worst Case Analysis of Doubly Terminated Reactance Two-Ports", *IEEE Trans. Circuit Theory*, Vol CT-20, No. 5, pp. 567-571, Sept. 1973.

[17] H.J. Orchard, "Loss Sensitivities in Singly and Doubly Terminated Filters", *IEEE Trans. Circuits Syst.*, Vol. CAS-26, No. 5, pp. 293-297, May 1979.

[18] A. Fettweis, "Pseudopassivity, Sensitivity, and Stability of Wave Digital Filters", *IEEE Trans. Circuit Theory*, Vol. CT-19, No. 6, pp. 668-673, Nov. 1972.

[19] R.E. Crochiere, "Digital Ladder Structures and Coefficient Sensitivity", *IEEE Trans. Audio Electroacoust.*, Vol. AU-20, No. 4, pp. 240-246, Oct. 1972.

[20] Y. Chu, *Digital Computer Design Fundamentals*, McGraw-Hill, Inc., 1962.

[21] L.B. Jackson, J.F. Kaiser, and H.S. McDonald, "An Approach to the Implementation of Digital FIlters", *IEEE Trans. Audio Electroacoust.*, Vol. AU-16, No. 3, pp. 413-421, Sept. 1968.

[22] A. Fettweis, "Roundoff Noise and Attenuation Sensitivity in Digital Filters with Fixed-Point Arithmetic", *IEEE Trans. Circuit Theory*, Vol. CT-20, No. 2, pp. 174-175, March 1973.

[23] I. W. Sandberg and J. F. Kaiser, "A Bound on Limit Cycles in Fixed-Point Implementations of Digital Filters", *IEEE Trans. Audio Electroacoust.*, vol. AU-20, pp. 110-112, June 1972.

[24] S. R. Parker and S. F. Hess, "Limit-Cycle Oscillations in Digital Filters", *IEEE Trans. Circuit Theory*, vol. CT-18, pp. 687-697, Nov. 1971.

[25] A. N. Willson, "Limit Cycles Due to Adder Overflow in Digital Filters", *IEEE Trans. Circuit Theory*, vol CT-19, pp. 342-346, July 1972.

[26] D. Mitra, "A Bound on Limit Cycles in Digital Filters which Exploits a Particular Structural Property of the Quantization", *IEEE Trans. Circuits Syst.*, vol CAS-24, pp. 581-589, Nov. 1977.

[27] S. Yakowitz and S. R. Parker, "Computation of Bounds for Digital Filter Quantization Errors", *IEEE Trans. Circuit Theory*, vol. CT-20, July 1973.

[28] T. Myint-U, *Ordinary Differential Equations.* New York: Elsevier North-Holland, Inc., 1978.

[29] P. Lancaster and M. Tismenetsky, *The Theory of Matrices, second edition.* Orlando: Academic Press, 1985.

[30] C.W. Barnes, "Roundoff Noise and Overflow in Normal Digital Filters", *IEEE Trans. Circuits Syst.*, Vol. CAS-26, pp. 154-159, Mar. 1979.

[31] D. A. Vaughanpope and L. T. Bruton, "Transfer Function Synthesis Using Generalized Doubly Terminated Two Pair Networks", *IEEE Trans. Circuits Syst.*, vol. CAS-24, pp. 79-88, Feb. 1977.

[32] E. S. K. Liu, L. E. Turner, and L. T. Bruton, "Exact Synthesis of LDI and LDD Ladder Filters", *IEEE Trans. Circuits Syst.*, vol. CAS-31, pp. 369-381, Apr. 1984.

[33] L. T. Bruton and D. A. Vaughanpope, "Synthesis of Digital Ladder Filters from LC Filters", *IEEE Trans. Circuits Syst.*, vol. CAS-23, pp. 395-402, June 1976.

[34] P. Binding and P. J. Browne, "Existence conditions for eigenvalue problems generated by compact multiparameter operators", *Proceedings of the Royal Society of Edinburgh,* 96A, pp. 261-274, Aug. 1984.

[35] B.D. Green and L.E. Turner, "New Limit Cycle Bounds for Digital Filters", accepted for publication in *IEEE Trans. Circuits Syst.*

[36] A.I. Zverev, *Handbook of Filter Synthesis*, John Wiley & Sons, New York, 1967.

[37] G.C. Temes and S.K. Mitra, *Modern Filter Theory and Design*, John Wiley & Sons, New York, 1973.

[38] L.E. Turner and P.B. Denyer, *The Analysis and Implementation of Digital Filters Using a Special Purpose CAD Tool*, Internal Report 52 DS 87 LE, June 1987

[39] R.L. Burden, J.D. Faires, A.C. Reynolds, *Numerical Analysis*, 2nd Ed., Prindle, Weber & Schmidt, Boston, Massachusetts, 1978.

[40] M. Hatamian and G.L. Cash, "A 70-MHz 8-bit × 8-bit Parallel Pipelined Multiplier in 2.5-*micro*m CMOS", *IEEE J. Solid-State Circuits*, Vol. SC-21, No. 4, pp. 505-513, Aug. 1986.

[41] LSI Logic, *Databook and Design Manual: HCMOS Macrocells and Macrofunctions*, LSI Logic Corp., Milpitas, CA, Oct. 1986.

[42] P. Denyer and D. Renshaw, *VLSI Signal Processing: A Bit Serial Approach*, Addison-Wesley, 1985.

[43] R.F. Lyon, "Two's Complement Pipeline Multipliers", *IEEE Trans. Communications*, Vol. COM-24, No. 4, pp. 418-425, April 1976.

[44] LSI Logic, *Design Manual for LDS Software Systems*, LSI Logic Corp., Milpitas, CA, 1986.

[45] A.F. Murray and P.B. Denyer, "Testability and Self-test in NMOS and CMOS VLSI Signal Processors", *IEE Proceedings*, Vol. 132, Pt. G, No. 3, pp. 93-104, June 1985.

[46] Texas Instruments, *TMS32010 User's Guide*, Texas Instruments publication SPRU001B, 1985

[47] B. Nowrouzian, L.E. Turner, R. Fong, L.S. Lee, "Design of Allpass Digital Ladder Networks Using Bilinear Transformation and LDI Integrators", *1987 IEEE Int. Symp. Circuits Syst.*, Philadelphia, PA, Vol. 2, pp. 649-655, May 1987.

Appendix

Determination of Δe for Second Order Sign-magnitude Sections

In two dimensions, a geometric interpretation of the form of $L(x)$ can be used to determine a bound on Δe for systems operated under sign-magnitude quantization. Figure A.1 shows an ellipse $L(x) = c$ on which an infinite precision point x_0 is assumed to lie, the state (x) coordinate axes, and the y coordinate axes (aligned with the ellipse). The boxes at points A, B, C, and D indicate the direction and maximum distance sign-magnitude quantization will move an infinite precision point x_0 in each of the four state quadrants. We wish to determine a bound on the maximum increase in $L(x)$ which can be induced by sign-magnitude quantization of x_0 to the finite precision point $\hat{x}_0$. Clearly, in the second and fourth state quadrants, $\hat{x}_0$ must lie inside or on the curve $L(x) = c$ so that these quadrants need not be considered. By symmetry, the maximum possible increase of $L(x)$ must be the same in quadrants one and three, hence only quadrant one will be considered here. If x_0 lies on the portion of the ellipse bounding the shaded region, any change in the x_2 component due to quantization moves the result towards the interior of the ellipse (smaller values of $L(x)$), and any change in the x_1 component due to quantization moves the result towards the exterior of the ellipse (larger values of $L(x)$). In this region, the maximum increase of $L(x)$ must therefore be bounded by the case where the change in the x_2 component is zero and the change in the x_1 component is one. With the bounding deviation determined in this region, the initial point x_0 which gives rise to the maximum increase in $L(x)$ with this deviation must be determined.

In the y coordinate system, the ellipses $L(x) = k$ for constant parameter k can be expressed parametrically as

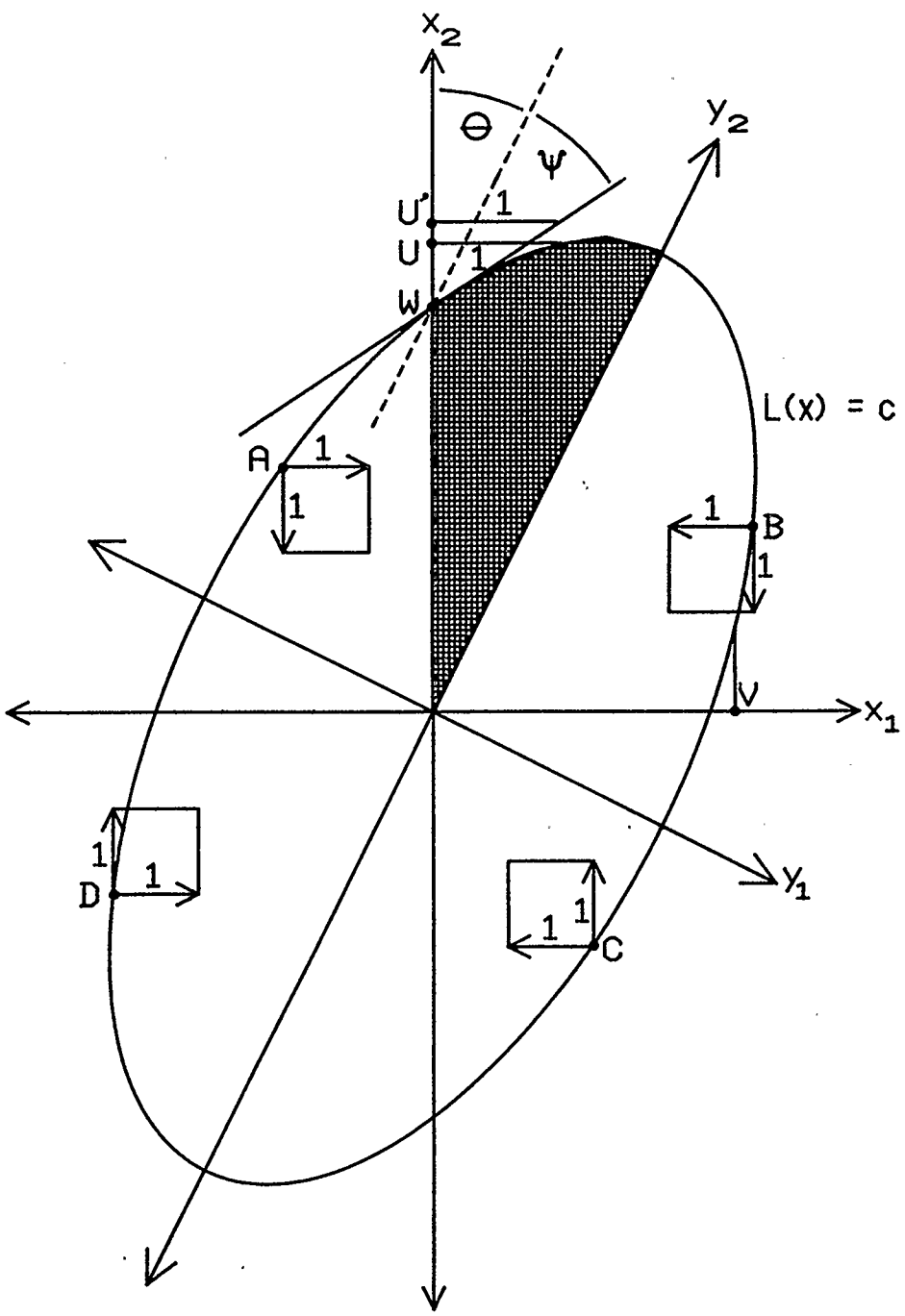


Figure A.1 : Extremes of Quantization

$$y_1 = \frac{k}{\sqrt{\mu_1}}\cos(\tau) \tag{A.1a}$$

$$y_2 = \frac{k}{\sqrt{\mu_2}}\sin(\tau) \tag{A.1b}$$

where τ is an angular parameter ranging from zero to 2π as the ellipse is traversed in the counter-clockwise direction from the y_1 axis. The slope of the ellipses in the y coordinate system can be determined by differentiating (A.1) to yield

$$\frac{dy_2}{dy_1} = -\sqrt{\mu_1/\mu_2}\cot(\tau) \tag{A.2}$$

which is independent of the arbitrary constant k and monotonically increasing with τ in the range zero to π. If $\hat{x}_0$ is the bounding point determined from x_0 as discussed above, then it represents a larger value of the angular parameter τ than the point x_0 does. Thus for any x_0 on the ellipse bounding the shaded region of figure A.1, the slope of the ellipse at x_0 is smaller than the slope of the ellipse $L(x) = L(\hat{x}_0)$ at $\hat{x}_0$. Motion of the point x_0 clockwise must thus decrease the value of $L(\hat{x}_0)$, and counter-clockwise motion of x_0 must increase the value of $L(\hat{x}_0)$. This situation is illustrated in figure A.2. The same argument can be applied when x_0 is assumed to lie on the ellipse in the unshaded region of the first quadrant, except here the maximum increase in $L(x)$ is bounded by the value obtained when the x_1 component remains unchanged and the x_2 component changes by one, and the value of $L(\hat{x}_0)$ increases as x_0 moves clockwise around the ellipse. Points U and V in figure A.1 represent the extremes of counter-clockwise or clockwise motion of x_0 in the first quadrant, and thus either $L(U)$ or $L(V)$ bounds the maximum value of $L(\hat{x}_0)$ when x_0 lies on the ellipse $L(x) = c$.

From figure A.1 it can be seen that $L(U) < L(U')$. The point U' can be determined by simple geometry if the angle ψ is known. Figure A.3 shows an expanded view of the geometry defining ψ and Δ in the following equations. Since ψ is simply the inverse tangent of the slope of the ellipse (in the y coordinate system) where it crosses the x_2 axis, (A.2) immediately gives

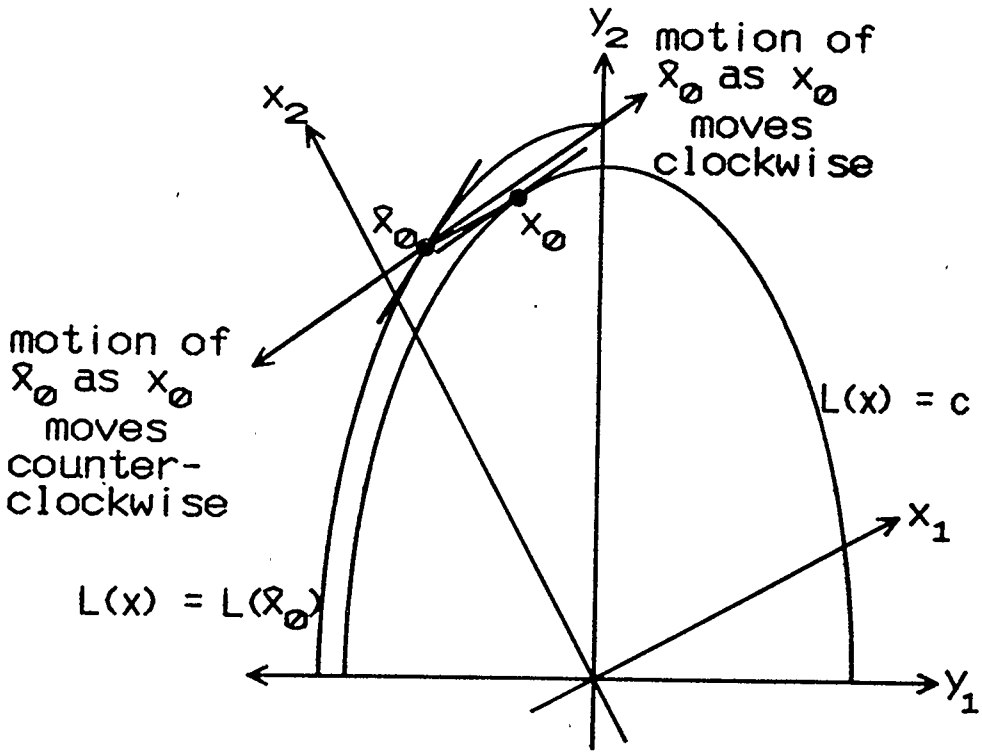


Figure A.2 : Motion of $\hat{x}_\emptyset$ with $x_\emptyset$

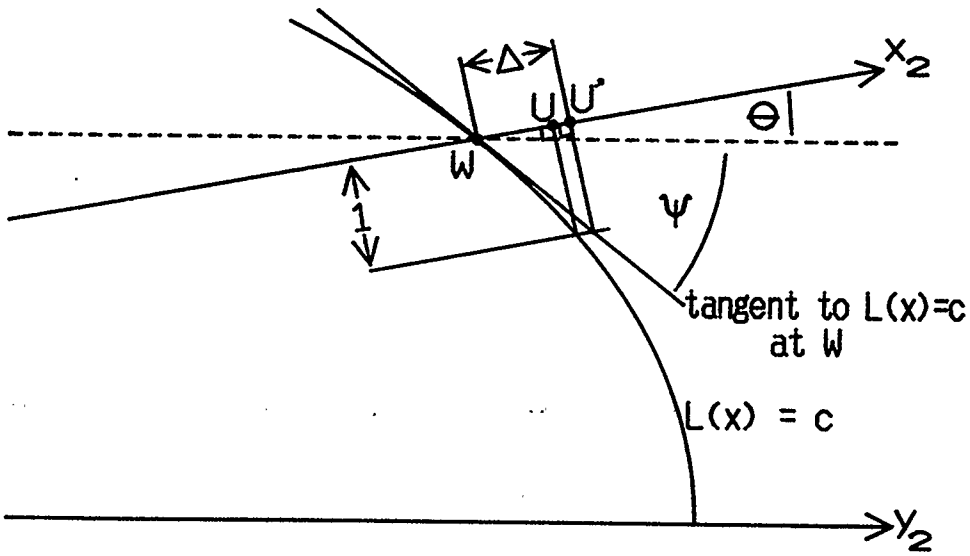


Figure A.3 : Geometry Defining Ψ

$$\psi = \tan^{-1}(-\sqrt{\mu_2/\mu_1}\tan(90+\theta)) = \tan^{-1}(\sqrt{\mu_2/\mu_1}\cot(\theta)) \tag{A.3}$$

from which the distance between points U' and W (denoted Δ) can be expressed as

$$\Delta = \frac{1}{\tan(\theta+\psi)} = \frac{1-\tan(\theta)\tan(\psi)}{\tan(\theta)+\tan(\psi)} \tag{A.4a}$$

$$= \frac{1-\sqrt{\mu_2/\mu_1}}{\tan(\theta)+\sqrt{\mu_2/\mu_1}\cot(\theta)} \, . \tag{A.4b}$$

However using (A.1) we find

$$\Delta^2 = \frac{[L(U')-L(W)]^2\cos^2(90+\theta)}{\mu_1} + \frac{[L(U')-L(W)]^2\sin^2(90+\theta)}{\mu_2} \tag{A.5a}$$

$$= [L(U')-L(W)]^2\left\{\frac{\sin^2(\theta)}{\mu_1} + \frac{\cos^2(\theta)}{\mu_2}\right\} \tag{A.5b}$$

which gives

$$L(U')-L(W) = \frac{\Delta\sqrt{\mu_1\mu_2}}{[\mu_1\cos^2(\theta)+\mu_2\sin^2(\theta)]^{1/2}} \, . \tag{A.6}$$

Exactly the same analysis can be performed at V, the only difference being that θ is replaced by $90-\theta$ so that

$$L(\hat{x}_0)-L(x_0) < \max_{\{\phi=\theta,\, 90-\theta\}} \left[\frac{\Delta\sqrt{\mu_1\mu_2}}{[\mu_1\cos^2(\phi) + \mu_2\sin^2(\phi)]^{1/2}}\right] \tag{A.7}$$

where

$$\Delta = \frac{1 - \sqrt{\mu_2/\mu_1}}{\tan(\phi) + \sqrt{\mu_2/\mu_1}\cot(\phi)} \tag{A.8}$$

which is the desired result ((A.7) and (A.8) are identical to (4.32) and (4.33) if ϕ is replaced by $90 - \phi$ and Δ is replaced by δ).