Effects of Finite Length Registers on a Modified Directform Realization of a High Order H(z) Transfer Function

1984

Angel Vanrell
University of Central Florida

Find similar works at: https://stars.library.ucf.edu/rtd

University of Central Florida Libraries http://library.ucf.edu

Part of the Engineering Commons

STARS Citation

https://stars.library.ucf.edu/rtd/4707

This Masters Thesis (Open Access) is brought to you for free and open access by STARS. It has been accepted for inclusion in Retrospective Theses and Dissertations by an authorized administrator of STARS. For more information, please contact lee.dotson@ucf.edu.
EFFECTS OF FINITE LENGTH REGISTERS ON
A MODIFIED DIRECT-FORM REALIZATION OF
A HIGH ORDER H(z) TRANSFER FUNCTION

BY

ANGEL VANRELL, III
B.S.E.E., University of Florida, 1981

RESEARCH REPORT

Submitted in partial fulfillment of requirements
for the degree of Master of Science in Engineering
in the Graduate Studies Program of the
College of Engineering
University of Central Florida
Orlando, Florida

Summer Term
1984
ACKNOWLEDGEMENTS

I wish to thank the management of Martin Marietta's Engineering Computing Center for supporting this work and Dr. Fred O. Simons for his advice during my graduate program.
# TABLE OF CONTENTS

LIST OF FIGURES ................................................................. vii

LIST OF TABLES ................................................................. viii

CHAPTER

I. INTRODUCTION ............................................................... 1
II. EFFECTS OF FINITE WORD LENGTH ...................................... 3
III. DEVELOPMENT OF REALIZATION SCHEMES ............................. 16
IV. EMULATION OF DIGITAL PROCESSES .................................... 23
V. TEST CASE: A FOURTH-ORDER LOW-PASS BUTTERWORTH FILTER .......... 30
VI. CONCLUSION ................................................................. 40

APPENDIX ............................................................................. 42

LIST OF REFERENCES ............................................................. 48
LIST OF FIGURES

1. Statistical Model of a Digital Multiplier ......................... 10
2. Direct, Cascade, and Parallel Realizations ....................... 13
3. Modified Direct-Form Architecture ................................. 14
4. Canonic Direct-Form Architecture ................................. 15
5. Roundoff Error Algorithm ........................................ 26
6. Single- and Mixed-Precision Arithmetic ............................ 38
7. Frequency Response to the Fourth-Order
   Low-Pass Butterworth Digital Filter ............................ 31
8. Mechanization of the Modified Direct-Form ....................... 34
9. Mechanization of the Canonic Direct-Form ......................... 35
10. Block Diagram of a MAC Device ................................. 43
LIST OF TABLES

1. Feedforward Coefficients ........................................ 32
2. Feedback Coefficients of Canonic Direct-Form ................ 32
3. Feedback Coefficients of the Modified Direct-Form .......... 32
4. Scaling Parameters ................................................. 32
5. Statistics on Noise Sources ..................................... 37
6. Statistics on $e_{\text{tot}}$ Versus Word Length ............... 37
ABSTRACT

When a digital process is realized on a general-purpose computer or a special-purpose hardware, errors due to finite register length are introduced. These errors are due primarily to arithmetic roundoff, coefficient quantization, and scaling rules. This paper addresses the effects of finite word length on a direct-form implementation of a high order $H(z)$ transfer function.

The development and analysis of a modified direct-form realization suggested by Dr. Fred O. Simons, are carried out via FORTRAN emulation of a fourth-order low-pass Butterworth filter. The results are presented as a parametric tradeoff of signal-to-noise ratio at the filter output versus word length. Conclusions are drawn by comparing the modified direct-form with the canonic direct-form. The analysis presented here is intended to illustrate how a high order transfer function can be realized directly without decomposing into a group of low-order subfilters.
I. INTRODUCTION

The synthesis problem for fixed-point digital filters involves more than the specification of $H(z)$ transfer function. Given an external description such as $H(z)$, the problem is to determine a filter sub-structure which minimizes inaccuracies due to finite register length arithmetic. Knowing the numerator and denominator coefficients of the $H(z)$ transfer function, the direct-form realization is readily available.

The digital filter realization of a given transfer function

$$H(z) = \sum_{m=0}^{M} a_m z^{-m} \frac{1}{1 + \sum_{n=1}^{N} b_n z^{-n}}$$

(1.1)

is quite simple. Suppose the input is $x(k)$ with the $z$-transform $X(z)$ and the output is $y(k)$ with the $z$-transform $Y(z)$. Then, from equation 1.1 we can easily obtain

$$Y(z) = \sum_{m=0}^{M} a_m z^{-m} X(z) - \sum_{n=1}^{N} b_n z^{-n} Y(z).$$

(1.2)
From equation 1.2 and applying the inverse z-transform, we obtain the discrete time domain equation

\[ y(k) = \sum_{m=0}^{M} a_m x(k-m) - \sum_{n=1}^{N} b_n y(k-n) \]  

(1.3)

which is the difference (or delay) equation that realizes \( H(z) \) directly, and which also represents a computational algorithm.

In the absence of a finite word length effects, synthesis of equation 1.1 is trivial. However, such a realization can produce inaccuracy which is orders of magnitude greater than other realizations. Therefore, the cardinal rule, described by Kaiser [1], is to use a design method that permits decomposition of the higher order filter into a group of low-order (first- or second-order) subfilters. This is a well documented fact. However, we intend to introduce a architecture with its corresponding algorithm which can match the performance of the decomposed architectures.
II. EFFECTS OF FINITE WORD LENGTH

A digital filter is represented by a linear shift-invariant discrete transfer function of the form

$$H(z) = \sum_{m=0}^{M} a_m z^{-m}$$

$$1 + \sum_{n=1}^{N} b_n z^{-n}$$

realized with finite accuracy in the representation of all data and parameter values. A digital filter can be simulated on a general-purpose computer or can be constructed with special-purpose hardware. Despite the many advantages provided by a digital filter, there is an inherent limitation on the accuracy of these filters, due to the fact that all digital architectures operate with a finite word length. Any digital transfer function can be implemented in a multitude of configurations. In a given digital filter architecture, the effect of finite word length results in three primary sources of error:

1. Input-quantization errors due to the analog-to-digital conversion of an analog signal,

2. coefficient-quantization due to the representation of the filter coefficients $a_m$ and $b_n$ by a finite number of bits, and
3. Arithmetic-quantization due to the accumulation of the roundoff errors committed at arithmetic operations

If the sequence of values driving the digital filter, denoted by $X(z)$, is derived by sampling analog signal $x(t)$, then the finite word length constraint requires that the analog-to-digital (A/D) conversion produces a finite number of values. This effect can be summarized by

$$x(n) = [x(t)]_q,$$

where $[ ]_q$ represents the A/D process with a quantization step size of $q$ on an analog signal $x(t)$ to produce a digital signal, $x(n)$.

The usual approach for treating the effect of input-quantization [2, 3] is to assume that the input to the actual filter is $x(n) + e(n)$, where $e(n)$ is the error due to the quantization of $x(t)$. Because $H(z)$ is linear, the superposition principle applies; thus, the output is the sum of two components, one due to $x(n)$ and the other due to $e(n)$. Typically, $e(n)$ is regarded as white noise with zero-mean and variance of $q^2/12$. The output due to $e(n)$ is zero-mean, wide sense, stationary sequence [4] with a power spectrum given by

$$(q^2/12) H(z) H(z^{-1}).$$

In this analysis, the other two sources of errors are ignored because their effect is minimum compared to the filter response $x(n)$. The mean square value of the error due to the input-quantization can be obtained by integrating the power spectrum density, equation 2.3 in the following manner:

$$(\pi j/2) \int (q^2/12) H(z) H(z^{-1}) \, dz/z.$$
Numerical techniques to evaluate equation 2.4 are readily available in the literature [5,6]. This paper will not address input-quantization errors further. The underlining consideration is that different architectures of the same digital filter will have the noise power spectrum at the output due to input-quantization.

When \( a_m \) and \( b_n \) in equation 2.1 cannot be specified exactly, \( a_m^* = \lfloor a_m \rfloor_q \) and \( b_n^* = \lfloor b_n \rfloor_q' \), coefficient-quantization errors are introduced into the filter synthesis process. Coefficient-quantization produces perturbations in the location of the poles and zeroes of the transfer function [7-9] which, in turn, lead to errors in the realization of the desired frequency response. Because filter characteristics can be computed to reflect changes in the filter coefficients due to quantization, coefficient quantization is deterministic in nature and straightforward to analyze. Nevertheless, an analysis of only the effects of coefficient quantization, i.e., neglecting arithmetic quantization noise, can be performed by statistical models to determine whether chosen word lengths are appropriate. According to the model originally conceived by Avenhaus [10] and later modified by Crochiere [11], the coefficient-quantization noise can be reduced by a factor of 2 for each additional bit added to the word length.

Initially, the data and parameters are represented with finite word lengths introducing quantization-noise. A result of further
processing will naturally lead to values requiring additional bits for their representation. When the resultants of multiplications and summations required to realize equation 2.1 are either rounded or truncated to a smaller number of bits, the error introduced is arithmetic-quantization noise, also known as roundoff noise. Roundoff noise can be treated as noise sources which give rise to output noise, reducing the overall signal-to-noise ratio at the output of the filter.

Inherent in the quantization process is a maximum value for the magnitude of the data, parameters, and arithmetic results, which is usually referred to as the dynamic range of the filter. Thus, the fact that the filter has a finite word length gives rise to yet another phenomenon called overflow. This occurs whenever the magnitude of a value exceeds the dynamic range of the filter, as determined by the number of bits available in the filter. Jackson [12] incorporated the effect of dynamic range constraint through use of amplitude scaling wherever applicable in the digital architecture. When a scaling rule is introduced, the probability of overflow is limited.

Three adjustable parameters determine the nature and extent of quantization errors:

1. Data and parameter word length,
2. Arithmetic-mode, and
3. Architecture.
Due to the limit on the word length available in mini-computers and the desired word length in special purpose hardware systems, quantization errors associated with finite coefficient word length become a critical aspect of the design. All data and parameters within the filter are quantized to a finite set of allowable values, with some error incurred as a result of quantization. Word length limits the dynamic range of the filter, and thus the realizable SNR. An erroneous selection of the data and parameter word lengths may lead to a filter that does not satisfy the original specifications of the filter. Since the cost and complexity of the realization of the filter depends on these word lengths, these should be kept to a minimum, but should be sufficient to fulfill the performance requirements.

The most commonly used arithmetic modes are fixed-point and floating-point. When two L-bit fixed-point numbers are added, their sum would still have L-bits, provided there is no overflow. Therefore, under this assumption, fixed-point addition introduces no errors; there is, however, a dynamic range limitation because overflow is possible. The product of two L-bit fixed-point numbers may have more than L-bits. After rounding or truncating the results to L-bits, roundoff noise is introduced into computations. On the other hand, floating-point is less affected by the dynamic range constraint, but roundoff errors are introduced during both additions and multiplications. Typically, the arithmetic-quantization noise in a floating-point filter is less than fixed-point filter with the same word length because of the automatic
scaling in a floating-point realization. Nevertheless, most special-purpose digital filters have been constructed with fixed point hardware because floating-point is more complex and costly to implement.

Even more importantly, floating-point hardware is slower than fixed-point hardware. Thus, floating-point arithmetic has a bandwidth constraint which is undesirable in real-time processing applications such as digital filtering.

There are three common ways of representing numbers in a fixed point configuration: sign-magnitude, one's complement, and two's complement. Once a word length is selected in any fixed-point representation, the set representable numbers is fixed. If the word length is L-bits (excluding the sign bit) and the numbers are normalized such that \( |v| < 1 \), the smallest number variation that can be represented is a 1 at the least significant register, which then corresponds to \( 2^{-L} \). Typically, the following two's complement representation is selected

\[
v = \begin{cases} 
  x, & v > 0 \\
  2 - x, & v < 0 
\end{cases}
\]

where

\[
x = \sum_{k=1}^{+00} v_k 2^{-k};
\]

because of finite register lengths, any number, \( v \), must be approximated by either rounding or truncating. Rounding can be effected by adding 1 at the position \( L + 1 \) and then truncating to \( L \)-bits. In truncating, those bits beyond the most significant \( L \)-bits are simply dropped.
Letting \( v^* \) be the machine representation of the result of either rounding or truncating, then

\[
x = \sum_{k=1}^{L} v_k 2^{-k} + \begin{cases} 
  v_{L+1} 2^{-L}, & \text{round} \\
  0, & \text{chop}.
\end{cases}
\]

A convenient way of analyzing the effect of quantization is to represent the error \( e = v^* - v \), statistically. For the case of two's complement fixed-point arithmetic, the error due to rounding and truncating is represented as a random variable [13] with a uniform probability density function, \( p(e) \), as shown in Figure 1.

Since the error introduced by truncating is more serious than that introduced by rounding, because of the bias shown in Figure 1, truncating is not desirable.

Digital filter architectures are composed of three building blocks: adders, constant multipliers and delays. Interconnection of these elements into a particular architecture is a critical step in filter syntheses. The architecture determines the spectrum of the output noise and along with the arithmetic mode, determines the word length required to satisfy the performance specifications.
Figure 1. Statistical Model of a Digital Multiplier
There can be significant difference between arithmetic-quantization noise of different implementations of the same transfer function, $H(z)$. This is a consequence of the fact that, for a polynomial whose roots are clustered, the sensitivity of the roots to changes in the polynomial coefficients increases as the order of the polynomial increases. Clearly, the roots can be more accurately controlled if the polynomial is factored into smaller sections. Direct-forms are vastly inferior to the parallel and cascade forms, Figure 2. Jenkins and Leon [14] reveal an interesting tradeoff between parallel and cascade forms. The parallel form exhibits better roundoff performance, while the cascade forms appear to have the better coefficient error figures. Their conclusion is that cascade forms appear to have better overall quantization error properties.

This paper presents a modified direct-form realization of $H(z)$ suggested by Dr. Fred O. Simons. The suggested architecture realizes a high-order transfer function with the following recursive relationship, as depicted in Figure 3,

$$y(k) = \sum_{m=0}^{M} a_m x(k-m) + \beta_0 y(k-1) + \sum_{n=1}^{N-1} \beta_n \Delta y(k-n)$$
\[
\begin{align*}
\beta_0 &= - \sum_{i=1}^{N} b_i, \\
\beta_n &= \sum_{i=n+1}^{N} b_i,
\end{align*}
\]

and \( \Delta y(k) = y(k) - y(k-1) \).

Such an implementation allows the use of computationally efficient algorithms developed by Dr. Simons which improves the real-time processing performance of the filter realization. The objective is to tradeoff word length requirements between the traditional direct form shown in Figure 4 and the suggested modified direct form.

Several analytical approaches apply statistical techniques to arrive at an optimal word length. Unfortunately, these techniques are not applicable here. The statistical models of quantization noise arrive at the noise power spectrum for both architectures. Thus, a FORTRAN emulation of a fourth-order low-pass Butterworth filter provides the statistics of the output error due to quantization noise versus word length.
DIRECT FORM:

\[ X(z) \rightarrow H(z) \rightarrow Y(z) \]

CASCADE FORM:

\[ H(z) = \prod_{i=1}^{N} C_i(z) \]

PARALLEL FORM:

\[ H(z) = \sum_{i=1}^{N} P_i(z) \]

Figure 2. Direct, Cascade, and Parallel Realizations
Figure 3. Modified Direct-Form Architecture
Figure 4. Canonic Direct-Form Architecture
III. DEVELOPMENT OF REALIZATION SCHEMES

Modified Direct-Form

Given a high order transfer function of the following form

\[ H(z) = \sum_{m=0}^{M} a_m z^{-m} \]

\[ 1 + \sum_{n=1}^{N} b_n z^{-n} \]

we can arrive at the architecture previously described in equation 2.4.

Equation 3.1 implies the following input-output relationship:

\[ \left[ \sum_{m=0}^{M} a_m z^{-m} \right] X(z) = \left[ 1 + \sum_{n=1}^{N} b_n z^{-n} \right] Y(z). \]

(3.2)

Taking the inverse z-transform of both sides of equation 3.2, the corresponding difference equation is available

\[ \sum_{m=0}^{M} a_m x(k-m) = y(k) + \sum_{n=1}^{N} b_n y(k-n). \]

(3.3)
Solving for $y(k)$,

$$y(k) = \sum_{m=0}^{M} a_m x(k-m) - \sum_{n=1}^{N} b_n y(k-n);$$

(3.4)

a recursive relationship is obtained which defines the current output as a function of past outputs and present and past inputs. Now, the objective is to modify the summation of weighted past output values in equation 3.4 into a sum of weighted difference operations.

Extracting from equation 3.4 the weighted sum of past output values, we must arrive at the following expression

$$\sum_{n=1}^{N} b_n y(k-n) = \sum_{n=0}^{N} \beta_n \Delta y(k-n),$$

(3.5)

where $\beta_n = f(b_1, b_2, \ldots, b_N)$

and $\Delta y(k) = y(k) - y(k-1)$.

(3.6)

Expanding the left-hand side of equation 3.5,

$$\sum_{n=1}^{N} b_n y(k-n) = b_1 y(k-1) + b_2 y(k-2) + \ldots$$

$$+ b_{N-2} y(k-N+2) + b_{N-1} y(k-N+1) + b_N y(k-N),$$

(3.7)

we can proceed to pair terms from $n=N$ to $n=1$ using difference operators. The first pair of terms, $n=N$ and $n=N-1$, can be combined by adding underlined redundant terms, as follows:
\[ \sum_{n=1}^{N} b_n y(k-n) = b_1 y(k-1) + b_2 y(k-2) + \ldots + b_{N-2} y(k-N+2) + [b_{N-1} + b_N] y(k-N+1) + b_N y(k-N+1) + y(k-N). \]

Grouping terms in equation 3.8 and applying equation 3.6,

\[ \sum_{n=1}^{N} b_n y(k-n) = b_1 y(k-1) + b_2 y(k-2) + \ldots + b_{N-2} y(k-N+2) + [b_{N-1} + b_N] y(k-N+1) + b_N \Delta y(k-N+1), \]

the first difference operator appears in the right hand side of equation 3.9. Repeating the above procedure, the next pair of terms, \(n=N-1\) and \(n=N-2\), in equation 3.9, form another difference expression by adding redundant terms to equation 3.

\[ \sum_{n=1}^{N} b_n y(k-n) = b_1 y(k-1) + b_2 y(k-2) + \ldots + b_{N-2} y(k-N+2) + [b_{N-1} + b_N] y(k-N+2)
\]

\[ - [b_{N-1} + b_N] y(k-N+2) + [b_{N-1} + b_N] y(k-N+1) - b_N \Delta y(k-N+1). \]

(3.10)
When we group terms and continue to apply equation 3.6,

$$
\sum_{n=1}^{N} b_n y(k-n) = b_1 y(k-1) + b_2 y(k-2) + ... \\
+ [b_{N-2} + b_{N-1} + b_N] y(k-N+2) \\
- [b_{N-1} + b_N] \Delta y(k-N+2) - b_N \Delta y(k-N+1).
$$

(3.11)

the trend becomes obvious:

$$
\sum_{n=1}^{N} b_n y(k-n) + \beta_0 y(k-) - \sum_{n=1}^{N-1} \beta_n \Delta y(k-n)
$$

where

$$
\beta_0 = \sum_{i=1}^{N} b_i \quad \text{and} \quad \beta_n = \sum_{i=n+1}^{N} b_i. \quad (3.12)
$$

At this point, an expression is obtained which satisfies the objective set forth in equation 3.5. To summarize the results, substitute equation 3.12 into 3.4.

$$
y(k) = \sum_{m=0}^{M} a_m x(k-m) + \beta_0 y(k-1) + \sum_{n=1}^{N-1} \beta_n \Delta y(k-n)
$$

where

$$
\beta_0 = \sum_{i=i}^{N} b_i, \quad \beta_n = -\sum_{i=n+1}^{N} b_i, \quad (3.13)
$$

and

$$
\Delta y(k) = y(k) - y(k-1).
$$
Canonic Direct-Form

A digital network is said to be "canonic" if the number of unit delays employed is equal to the order of the transfer function, \( H(z) \). The computational algorithms to realize such a canonic form are derived from

\[
H(z) = \frac{Y(z)}{X(z)} = \frac{\sum_{m=0}^{M} a_m z^{-m}}{1 + \sum_{n=1}^{N} b_n z^{-n}} \quad (3.14)
\]

in the following paragraphs.

An intermediate variable, \( W(z) \), can be introduced by partitioning equation 3.14

\[
H(z) = \begin{bmatrix} Y(z) \\ W(z) \end{bmatrix} = \begin{bmatrix} W(z) \\ X(z) \end{bmatrix} = \frac{N(z)}{D(z)} \quad (3.15)
\]

The first term,

\[
N(z) = \frac{Y(z)}{W(z)} = \sum_{m=0}^{M} a_m z^{-m}, \quad (3.16)
\]
leads to

$$Y(z) = \sum_{m=0}^{M} a_m z^{-m} W(z);$$ \hspace{1cm} (3.17)

or equivalently,

$$y(k) = \sum_{m=0}^{M} a_m w(k-m)$$ \hspace{1cm} (3.18)

in the time domain. Similarly, the second term,

$$\frac{1}{D(z)} = \frac{W(z)}{X(z)} = \frac{1}{N} + \sum_{n=1}^{N} b_n z^{-n}$$ \hspace{1cm} (3.19)

leads to

$$W(z) = X(z) - \sum_{n=1}^{N} b_n z^{-n} W(z);$$ \hspace{1cm} (3.20)

or equivalently,

$$w(k) = x(k) - \sum_{n=1}^{N} b_n w(k-n).$$ \hspace{1cm} (3.21)
Therefore, in place of equation 3.4, we have the following computational algorithms:

\[
\begin{align*}
    w(k) &= x(k) - \sum_{n=1}^{N} b_n w(k-n) \\
    y(k) &= \sum_{m=0}^{M} a_m w(k-m).
\end{align*}
\] (3.22, 3.23)

Equation 3.22-3.23 represent the digital architecture depicted in Figure 4.
IV. EMULATION OF DIGITAL PROCESSES

In two's complement fixed-point arithmetic, we assume that a quantity $y$ is approximated by $y^*$ as follows:

$$
y^* \begin{cases} 
x, & y^* > 0 \\
1.5^\left\lfloor \frac{y}{2} \right\rfloor & y^* < 0
\end{cases}$$

where $x = \sum_{i=1}^{-F} y_{i+F} \cdot 2^i$ (4.1)

and $y_N, y_{N-1}, y_{N-2}, ..., y_1$ are binary digits taking values 0, 1. The sign bit, $y_N$, denotes a positive value if equal to zero and negative value if equal to one.

Considering the previous definition, the numerical system can be defined with the following scaling rule:

$$S_y I_y \cdot F_y$$

and $S_y = \text{sign bit}$,

$I_y = \text{number of integer bits}$,

$F_y = \text{number of fractional bits}$,

and $N = I_y + F_y + 1$ = the total number of bits per binary word (refer to Figure 5).
Using this notation, we can examine how to emulate the only two operations required to realize a digital filter: summation and multiplication.

Given $y$, with its corresponding finite register representation $y^*$, the objective is to emulate the effect of quantization in a numerical system specified by $S_y I_y F_y$. The quantization process is accomplished by either truncating or rounding. The truncation, or chopping effect, can be emulated by the following expression:

$$y^* = 2^{-F_y} \cdot \text{INT} \left( 2^{F_y} y \right) \quad (4.3)$$

where $\text{INT} \left( \right)$ is a function that truncates the fractional part of a real number in FORTRAN and BASIC. Rounding is accomplished by adding the weight of the most significant bit of the discarded parts of the fraction. In order to emulate this effect, the following rule is applicable:

$$y_R = y_T + 2^{-F_y} \text{, if } y - y_T < 2^{-(F_y + 1)}$$
$$= y_T \text{, otherwise;} \quad (4.4)$$

where $y$ = the exact quantity,

$y^*$ = truncated $N$ bits,

$y$ = $y$ rounded to $N$ bits,

and $F_y$ = number of fractional bits $y^*$. 

When an N-bit number is added to another N-bit number, the two numbers must be described by the same rule. Thus,

$$y^* = a^* + x^*,$$  \hspace{1cm} (4.5)

where \( I = I_a = I_x \) and \( F = F_a = F_x \) in the scaling rules of \( a^* \) and \( x^* \). Assuming that no overflow occurs, the result of the addition, \( y^* \), can be represented accurately with N-bits. Overflow can occur whenever the two numbers, \( a^* \) and \( x^* \), have the same sign. In an emulation of a digital filter, overflows must be monitored. The dynamic range of the result can be expressed as

$$2^{N_y - 1} 2^{-F_y} \geq y \geq -2^{N_y} 2^{-F_y}$$  \hspace{1cm} (4.6)

The results of the addition must be constrained to the limits defined in equation 4.6. Observe that the positive full-scale value in two's complement numerical representation.

When an L-bit number is multiplied with an M-bit number, the resultant product can be specified by, at most, \( L + M \).
Choose word length such that $N \leq 2$

* Set $Y_{\text{max}}$

* If $\text{ABS}(y) > Y_{\text{max}}$, $y = Y_{\text{max}}$

* If $\text{BIT} > F_{Y}$, $Y_{r} = Y_{t} + 1$

* $Y_{t} = \text{INT}(\text{TEMP})$

* $\text{TEMP} = y \times 2^{F_{Y}}$

* Final result assuming an $N$-bit word with

\[
\begin{align*}
Y_{t}, R & = 2^{+F_{Y}} \times Y_{r}, R \\
2N_{y} & = -F_{Y} \\
(2N_{y} - 1) 2^{-F_{Y}} & > y \\
N_{y} & = \left(\frac{y}{\text{ABS}(y)}\right)
\end{align*}
\]
In this case,
\[ y^* = a^* x^*, \quad (4.7) \]
where the resultant scaling rule with the double precision accuracy is
\[ S_y S_y (I_a + I_x) . (F_a + F_x) \quad (4.8) \]
given the scaling rule of the factors \( S_a I_a . F_a \) and \( S_x I_x . F_x \). Observe the result has two sign bits; their value conforms to the normal algebraic rules pertaining to positive and negative factors. The roundoff error is introduced into the computations when the result must be stored in an \( N \)-bit word if \( N < L+M \). This single-precision computation can be modeled by applying equation 4.3.

In digital signal processing applications, the sum of \( K \) products
\[
y^* = \sum_{i=1}^{K} a_i x^* \quad (4.9)
\]
is encountered frequently (see Appendix). These can be computed in single-precision, i.e., by summing quantized results, or in mixed precision, by quantizing only the final result, Figure 6.
Figure 6. Single- and Mixed-Precision Arithmetic
By quantizing the result only once, the arithmetic roundoff error is decreased by a factor $K$; and hence the digital process is more accurate by using mixed-precision arithmetic.

An emulation of any digital process can be carried out with a general purpose computer using equations 4.3, 4.4, 4.5, and 4.8. These provide adequate models for truncation, rounding, dynamic range constraints and summation of products.
V. TEST CASE: A FOURTH-ORDER LOW-PASS BUTTERWORTH FILTER

The quantization errors in a fourth-order low-pass Butterworth filter was analyzed using emulation of both direct-form schemes discussed thus far, canonic and modified. A continuous, frequency normalized filter \( w_c = 1 \) was transformed by the standard z-transform to produce an impulse-invariant design with a sampling frequency of 5.0Hz [14]. This sampling rate was sufficient to prevent significant aliasing distortion, since the Nyquist rate was about 0.318 Hz and the filter was about 120 db down at the sampling frequency. The frequency response of this example is plotted in Figure 7.

The filter feedforward and feedback coefficients are shown in Tables 1 to 3. Both architectures were internally scaled in an attempt to reduce quantization errors. Values for scaling factors \( K_i \), where \( i = 0, 1, 2, 3 \), were chosen to maximize the signal amplitudes within the architecture while preventing overflow (see Table 4). In Table 1 to 4, the ideal parameter is listed in the first column with the scaled ideal value in the second column and quantized value in the third. Because rounding has the same effect as adding an extra bit, all of the filter parameters were rounded during the quantization process to 16 bits.
Figure 7. Frequency Response of the Fourth-Order Low-Pass Butterworth Digital Filter
### Table 1
**Feedforward Coefficients**

<table>
<thead>
<tr>
<th>$m$</th>
<th>$a_m$</th>
<th>$a_m/K_0$</th>
<th>$\bar{a}_m = [a_m/K_0]_R$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
</tr>
<tr>
<td>1</td>
<td>1.11679976933780</td>
<td>0.18960004334517</td>
<td>0.18960571289063</td>
</tr>
<tr>
<td>2</td>
<td>4.09288249892140</td>
<td>0.66439283512713</td>
<td>0.66439819333938</td>
</tr>
<tr>
<td>3</td>
<td>0.899432659757342</td>
<td>0.14600710352771</td>
<td>0.14599609375000</td>
</tr>
</tbody>
</table>

### Table 2
**Feedback Coefficients of CANONIC DIRECT-FORM**

<table>
<thead>
<tr>
<th>$n$</th>
<th>$b_n$</th>
<th>$b_n/K_1$</th>
<th>$\bar{b}_n = [b_n/K_1]_R$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>-3.46881512365680</td>
<td>-0.4348318905710</td>
<td>-0.4248497070313</td>
</tr>
<tr>
<td>2</td>
<td>2.56797554873600</td>
<td>0.37099694359200</td>
<td>0.5709338671875</td>
</tr>
<tr>
<td>3</td>
<td>-2.68089022859080</td>
<td>-0.33511127857385</td>
<td>-0.33508300781250</td>
</tr>
<tr>
<td>4</td>
<td>0.59261867788150</td>
<td>0.07412023347352</td>
<td>0.07412719726563</td>
</tr>
</tbody>
</table>

### Table 3
**Feedback Coefficients of THE MODIFIED DIRECT-FORM**

<table>
<thead>
<tr>
<th>$n$</th>
<th>$\beta_n$</th>
<th>$\beta_n/K_2$</th>
<th>$\bar{\beta}_n = [\beta_n/K_2]_R$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0.99876793572345</td>
<td>0.24969198393086</td>
<td>0.24969482421875</td>
</tr>
<tr>
<td>1</td>
<td>2.48007187187933</td>
<td>0.6200117968334</td>
<td>0.62002536476563</td>
</tr>
<tr>
<td>2</td>
<td>-2.08792368080265</td>
<td>-0.32198209020066</td>
<td>-0.52197265625000</td>
</tr>
<tr>
<td>3</td>
<td>0.592961867788150</td>
<td>0.14825439453125</td>
<td>0.14824046694704</td>
</tr>
</tbody>
</table>

### Table 4
**Scaling Parameters**

<table>
<thead>
<tr>
<th>$i$</th>
<th>$K_i$</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>6.1603392783262</td>
<td>$\sum_{m=0}^{M} a_m$ mod 2</td>
</tr>
<tr>
<td>1</td>
<td>8.0</td>
<td>$\max(b_n)$ mod 2</td>
</tr>
<tr>
<td>2</td>
<td>4.0</td>
<td>$\max(\beta_n)$ mod 2</td>
</tr>
<tr>
<td>3</td>
<td>1.232064276594980-3</td>
<td>$1 + \sum_{n=1}^{N} b_n$</td>
</tr>
</tbody>
</table>
In the mechanization of both computational algorithms, word lengths and scaling rules are shown in Figures 8 and 9. The assumption is that a 16X16 parallel multiplier accumulator (see Appendix) is used to carry out the hardware realization.

Using the techniques presented in Chapter IV, an emulation of the architectures was carried out. The statistics of the output noise, where $Y_{true}$ is the ideal response and $Y_{filter}$ is the actual filter response, were computed for a step input,

$$x(k) = \begin{cases} 
0.5, & k \geq 0 \\
0.0, & k < 0
\end{cases}$$

(5.2)

All internal states were initialized to zero. The mean, $m_e$, and the variance, $\sigma_e^2$, were computed with the following equations:

$$m_e = \frac{1}{N_{samp}} \sum_{k=0}^{N_{samp}} e(k)$$

(5.3)

$$\sigma_e^2 = \frac{1}{N_{samp}} \sum_{k=0}^{N_{samp}} [e(k) - m_e]^2$$

(5.4)
Figure 8. Mechanization of the Modified-Direct Form
Figure 9. Mechanization of the Canonic Direct-Form
where \( N_{\text{samp}} \) was selected empirically as 100 samples. After 100 samples of the output, the output of the ideal filter settles out within \( 2^{-9} \) of the actual filter. Observe that the D/A of the filter output was assumed to be 10 bits (including sign bit). In digital signal processing of analog signals, the quantization error is commonly viewed as additive noise signal. The ratio of the signal-to-noise power (SNR) is a useful measure of the relative strength of the signal and the noise. A common rule of thumb, derived from the statistical models discussed in Chapter II, is that the signal-to-noise ratio increased approximately 6 db with each additional bit. Thus,

\[
\text{SNR} = 6L
\]

where \( L \) is equal to the word length sign bit. For the fourth-order low-pass filter, a SNR of 54 db is adequate.

The results of the emulation are summarized in Tables 5 and 6. In Table 5, the statistics of the following noise sources are shown:

\begin{itemize}
  \item[a)] \( e_{\text{tot}} \), output noise as defined by equation 4.1;
  \item[b)] \( e_R \), noise due only to roundoff of computations;
  \item[c)] \( e_{b}\beta + e_R \), noise due to roundoff and quantization of feedback coefficients; and
  \item[d)] \( e_a + e_R \), noise due to roundoff and quantization of feedforward coefficients.
\end{itemize}
### TABLE 5

STATISTICS ON NOISE SOURCES

<table>
<thead>
<tr>
<th>Noise Source</th>
<th>Modified-Direct Form</th>
<th>Canonic-Direct Form</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>( m_e )</td>
<td>( \sigma_e^2 )</td>
</tr>
<tr>
<td>( e_{\text{tot}} )</td>
<td>0.0502</td>
<td>0.0191</td>
</tr>
<tr>
<td>( e_R )</td>
<td>0.0499</td>
<td>0.0191</td>
</tr>
<tr>
<td>( e_b + e_R )</td>
<td>0.0569</td>
<td>0.0188</td>
</tr>
<tr>
<td>( e_a + e_R )</td>
<td>0.0569</td>
<td>0.0188</td>
</tr>
</tbody>
</table>

### TABLE 6

STATISTICS ON \( e_{\text{tot}} \) VERSUS WORD LENGTH

<table>
<thead>
<tr>
<th>Word Length</th>
<th>Modified-Direct Form</th>
<th>Canonic-Direct Form</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>( m_e )</td>
<td>( \sigma_e^2 )</td>
</tr>
<tr>
<td>18</td>
<td>0.0293</td>
<td>0.0197</td>
</tr>
<tr>
<td>17</td>
<td>0.0356</td>
<td>0.0195</td>
</tr>
<tr>
<td>16</td>
<td>0.0502</td>
<td>0.0190</td>
</tr>
<tr>
<td>15</td>
<td>0.0907</td>
<td>0.0189</td>
</tr>
<tr>
<td>14</td>
<td>0.0635</td>
<td>0.0241</td>
</tr>
</tbody>
</table>
In Table 6, the statistics of \( e_{\text{tot}} \) were gathered with perturbations of the word length. Because of the architectures used parallel multiplier-accumulator, as the single-precision word length, \( L \) was varied the mixed-precision word length of the accumulator was also varied accordingly (see Appendix). Also included in Tables 5 and 6 is \( m^2 + \sigma^2 \), the total noise power. Using \( m_e, \sigma_e^2 \), and \( m_e^2 + \sigma_e^2 \) as figures of merit, a comparison of the direct forms is possible.

From Table 5, several qualitative and quantitative conclusions can be drawn. The canonic direct-form is much more sensitive to quantization noise than roundoff noise. On the other hand, the suggested modified direct form exhibits the opposite trend; it is more susceptible to roundoff noise. Another obvious result is a 50\% reduction in the noise level in the modified direct-form versus the canonic direct-form.

From Table 6, an interesting trend is presented. The canonic direct-form is more sensitive to perturbations of the word length. By adding two additional bits, a reduction of 35\% in the noise power is apparent comparable to the suggested direct-form baseline.

Although the results appear to indicate that the overall performance of the modified direct-form surpasses that of the canonic form, there are several drawbacks inherent to the suggested direct-form. For instance, more double-precision computations are required to implement the suggested direct-form and twice as many delays (memory
locations) are also required in the modified direct-form realization. Like in any other synthesis and design problem, a tradeoff is introduced with new alternatives.
VI. CONCLUSION

As the capabilities of digital components continue to grow while their cost drops, digital signal processing has become more attractive than analog circuitry for many applications. In the filtering area of signal processing, digital filters are replacing their analog counterparts because digital filters do not drift with temperature or voltage, require less maintenance or calibration, and have almost no limit on the possible signal-to-noise ratio assuming adequate word length. Moreover, one fast digital filter can be timeshared between many independent inputs, giving it even more useful channel-bandwidth capability. This circuit can also be precisely simulated, and easily, exactly repeated. The advantages and applications are many.

Despite the many advantages provided by digital filtering techniques, there is an inherent limitation on the performance of these filters, due to the effects of finite word length. There is an infinite variety of network implementations that realize a given transfer function when the parameters are represented with infinite accuracy. It is to be expected that some of these architectures will be less sensitive than others to quantization noise, i.e., the transfer function of the resulting realization approach to the desired function.

For realizing the transfer function as a computer program or in hardware, a digital architecture must be chosen such that quantization
effects are minimal. There are many other considerations such as implied hardware or software complexity and difficulty in determining parameters of the filters. Unfortunately, no systematic approach has been developed to determine an optimal realization in terms of the typical constraints: number of multipliers, word length, and memory allocation.

In this paper, two direct-form realizations were examined. Instead of a detailed mathematical analysis of the parameter sensitivity, a more direct approach was taken. By using an emulation of both architectures, a comparison of performance was possible. The suggested modified direct-form did prove to be less sensitive to quantization noise. The empirical results show a 50% reduction of the noise level in the modified direct-form versus the canonic direct-form. In spite of its inherent drawbacks, this architecture merits further study.
APPENDIX: THE PARALLEL MULTIPLIER-ACCUMULATOR

An element of critical importance in the hardware realization of a digital filter is the multiplier, which is required to produce the product of a filter parameter and a digital signal. The multiplication operation requires complex circuitry and is the principal controlling factor in the processing rate and the power consumption of the filter. Due in part to its importance in digital filters, considerable attention has been devoted to research and development of both multiplication algorithms and multiplier circuits.

A series of monolithic parallel multiplier-accumulators developed by TRW are capable of performing 8X8-, 12X12-, or 16X16- bit multiplication and the accumulation of product terms with the contents of an output register. Mac-8, -12, and -16 are informal ways of referring to the TDC1008J, TDC1009J, and TDC1010J and their handling of 8-, 12-, and 16- bit word [15]. These are high performance, TTL compatible devices fabricated in a bipolar VLSI technology which can be employed as the central building block of a digital filter. A functional diagram of such a device is shown in Figure 10, showing the pair of clocked input registers, followed by the multiplier, accumulator, and clocked output registers and the three-state output buffers [16].
ACC  Accumulate Control
SUB  Add/Subtract Select
RND  Rounding vs. Truncation Select
TC   Number Representation Select
TSX  Select XTP (Extended Product)
TSM  Select MSP (Most Significant Product)
TSL  Select LSP (Least Significant Product)
N    MAC-N, where N = 8, 12, or 16

Figure 10. Block Diagram of a MAC Device
The three most critical characteristics of a multiplier are speed of multiplication, power dissipation, and word length. These devices are very fast, with a multiply-accumulate time of 70 nanoseconds for Mac-8, 95 nanoseconds of Mac-12, and 115 nanoseconds for Mac-16. Power consumption from a single 5 volt source is moderate (1.2 watts for Mac-8, 2.5 watts for Mac-12, and 3.5 watts for Mac-16) [16].

Other features of secondary importance are expandability, data representation, and rounding. Expandability refers to building of large multiplier arrays for several multipliers. The TRW is non-expandable; but pipelining is an acceptable solution to obtain high speed multiplication of words greater than 16-bits in length [17]. Data representation is either unsigned or signed (two's complement); this is a selectable function in the TRW series.

Rounding is useful when a single-length product is desired. This is accomplished by raising the rounding pin to logic one, which propagates a carry into the least significant bit (of the single length product) if the most significant bit of the lower half is one, see Figure 10.

Devices like the TRW MAC series are being used to upgrade the real-time capability of the microprocessor systems. While the microprocessor is left to perform the basic tasks of memory management and control, the MAC does the "number crunching" operations at a data speed order of a magnitude higher than possible in the microprocessor.
LIST OF REFERENCES


