SIMULATION AND IMPLEMENTATION OF FIXED-POINT DIGITAL FILTER STRUCTURES

by

Daniel A. Bailey

Thesis submitted to the Faculty of the Virginia Polytechnic Institute and State University in partial fulfillment of the requirements for the degree of Masters of Science in Electrical Engineering

APPROVED

Dr. A.A. (Louis) Beex, Chairman

Dr. Jeffery H. Reed

Dr. Hugh F. Vanlandingham

September 26, 1995
Blacksburg, Virginia
SIMULATION AND IMPLEMENTATION OF FIXED-POINT
DIGITAL FILTER STRUCTURES

by

Daniel A. Bailey
Dr. A.A. (Louis) Beex, Chairman
Bradley Department of Electrical Engineering

(ABSTRACT)

The purpose of this research is to develop a fixed-point arithmetic model based on a common general purpose Digital Signal Processor (DSP). A detailed non-linear model is developed to emulate the convergent (un-biased) rounding process performed by the Motorola DSP56002 fixed-point DSP. This model is incorporated into several different filter structures and compared to the linear stochastic simulation and the actual hardware implementation. It turns out that the convergent rounding operation has an insignificant effect on the overall roundoff noise power. The Direct Form, Section Optimal and MA Lattice forms are studied. For these structures, Matlab routines are developed to automate the process of fixed-point scaling and DSP56002 code generation. Each structure's non-linear simulation is validated using two filter examples.

The scaling and simulation routines allow the filter designer to investigate the finite wordlength performance of various structures, scaling norms, overflow safety factors, and wordlengths to determine the best filter parameters prior to hardware implementation.
Acknowledgments

I would like to thank Dr. A.A. (Louis) Beex for his support of this project and for the direction of my graduate education; I could not have completed this work without his guidance. I would also like to thank Dr. Reed for introducing me to digital signal processing and communications, also for his guidance in my graduate education. I thank Dr. VanLandingham for his time and effort in reviewing this work. Most of all I thank my wife Linda for her sacrifices, support, and much needed encouragement.

My ultimate thanks is to my Creator, in the words of Abraham Lincoln "For without the help of that Divine Being I could not succeed, with His help I cannot fail."
Table of Contents

1. Introduction ............................................................................................................. 1
   1.1 Overview of Digital Filter Development Tools .............................................. 2
   1.2 Why Use a Fixed-Point DSP? ........................................................................ 3

2. Modeling of the Motorola DSP56002 Fixed-Point DSP ................................. 6
   2.1 Motorola DSP56002 Architecture ................................................................ 6
   2.2 Fixed-Point Data Representation ................................................................... 7
   2.3 Multiplier-Accumulator Architecture .............................................................. 9
      2.3.1 Overflow Saturation Mode ...................................................................... 9
      2.3.2 Convergent Rounding ............................................................................ 11
      2.3.3 Automatic Scaling Mode ......................................................................... 14
   2.4 Non-Linear Multiplier-Accumulator Model ..................................................... 15
   2.5 Linear-Stochastic Roundoff Noise Model ......................................................... 19

3. Filter Structure Simulation Description ............................................................... 22
   3.1 Matlab Filter Simulation .................................................................................. 22
   3.2 DSP56002 Code Generation ........................................................................... 25
   3.3 Matlab to DSP56002 Interface ......................................................................... 28

4. Implementation of Direct Form Filter Structures ............................................. 31
   4.1 High Order Direct Form II .............................................................................. 32
      4.1.1 Overflow Scaling .................................................................................... 34
      4.1.2 Structure Conversion ............................................................................... 37
      4.1.3 DSP56002 Implementation ..................................................................... 38
      4.1.4 Non-Linear Simulation Model ................................................................ 40
      4.1.5 Linear Stochastic Simulation Model ......................................................... 40
   4.2 Cascaded Second Order Direct Form II Structure ............................................. 43
      4.2.1 Overflow Scaling ..................................................................................... 47
<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>7.4</td>
<td>6th Order Butterworth Filter</td>
<td>106</td>
</tr>
<tr>
<td>7.5</td>
<td>12th Order FIR Filter</td>
<td>118</td>
</tr>
<tr>
<td>8.</td>
<td>Conclusions and Further Research</td>
<td>125</td>
</tr>
<tr>
<td></td>
<td>Bibliography</td>
<td>127</td>
</tr>
<tr>
<td></td>
<td>Vita</td>
<td>130</td>
</tr>
</tbody>
</table>
1.0 Introduction

The purpose of this research is to develop a fixed-point arithmetic model based on a common general purpose Digital Signal Processor (DSP). A detailed non-linear model is developed to emulate the convergent (un-biased) rounding process performed by the Motorola DSP56002 fixed-point DSP. Secondly, the implementation of various filter structures on the DSP56002 is studied to best utilize the architectural features of this processor. This research provides a digital filter simulator which is useful for the evaluation of finite wordlength errors in the following filter structures.

1) High Order Direct Form II
2) Cascaded 2nd Order Direct Form II Sections
3) Cascaded 2nd Order Direct Form I Sections
4) Parallel 2nd Order Direct Form II Sections
5) Cascaded State-Space Section Optimal
6) Direct Form FIR
7) MA Lattice

Linear stochastic simulations are also developed for each structure above. Both simulation models are validated by comparison to the actual filter implementation on the DSP56002.

Another contribution of this work is the development of Matlab routines for each structure which automates the process of coefficient computation and fixed-point scaling. These scaling and simulation routines will allow the filter designer to investigate the finite wordlength performance of various structures, scaling norms, overflow safety factors, and wordlengths to determine the best filter parameters prior to hardware implementation. The following additional objectives have been achieved in this work.
• Development of a Matlab to Motorola DSP56002 interface allowing filter algorithm execution and data exchange from within the Matlab environment.

• Coding of filter structure algorithms in DSP56002 assembly code and automatic linking of filter coefficients and filter code from within the simulation.

1.1 Overview of Digital Filter Development Tools

This work gives the filter designer the capability of investigating finite wordlength effects such as the magnitude response degradation due to coefficient quantization, the reduction in output SNR due to roundoff noise, and limit cycle conditions due to the non-linearity of rounding and saturation. These effects have been studied extensively and are discussed by Jackson [10], Oppenheim and Schafer [6], and Mullis and Roberts [5]. Their simulation examples use a linear-stochastic model (see Chapter 2) which ignores the effects of overflow errors. Also, each multiplication operation is treated as a source of roundoff error. Their simulation models do not consider common DSP architectural features such as a double length accumulator which retains the precision of a sum of products until the final result is moved from the accumulator. In this work a non-linear model is developed to emulate the DSP56002's double length accumulator, convergent rounding, and overflow saturation characteristics.

A major part of this research is the development of efficient routines for fixed-point scaling of each structure to prevent internal overflows and/or increase the dynamic range of the filter's internal signals, thereby increasing the output Signal-to-Quantization Noise Ratio (SQNR). Fixed-point filter scaling has been studied extensively by Jackson [12] for the Direct Form structures, and by Hwang [18] and Mullis and Roberts [15] for Optimal
State-Space structures. Only a few digital filter design packages on the market allow the user to specify scaling parameters and select between different filter structures. The filter design package DFDP3 developed by Atlanta Signal Processors Inc. [7] generates code only for the Cascaded Direct Form I structure with $L_{\infty}$ scaling. The design package Monarch developed by The Athena Group, Inc. offers a variety of structures and scaling norms.

1.2 Why Use a Fixed-Point DSP?

Why use a fixed-point processor when there are now general purpose floating-point DSPs on the market? This question flows from the common misconception that floating-point processors are superior to fixed-point processors. It might be thought that fixed-point DSPs will eventually be replaced by floating-point DSPs, making the operation of scaling obsolete. The choice between a fixed or floating point DSP is dependent upon requirements such as accuracy, dynamic range, speed, power consumption, and cost. This section presents several reasons why fixed-point DSPs will continue to be useful in meeting practical design requirements.

As for the issue of accuracy, floating-point arithmetic sacrifices resolution to increase dynamic range. Most floating-point processors are designed to the IEEE-754 standard for 32-bit floating point arithmetic. The 32-bit wordlength is divided into a 23-bit mantissa plus sign bit and a 7-bit exponent plus sign bit. The 24-bit mantissa represents fractional values between -1 and +1 and the 8-bit exponent contains the integer power of two scaling.
factor. Floating-point arithmetic is useful in applications where one must process signals with a wide dynamic range, such as in image processing.

Most real world signals are limited in dynamic range, such as in a speech or communication system. For example speech signals quantized to 16-bits have a dynamic range of 90 dB and may require an additional 20 dB of headroom resulting in a total dynamic range of 120 dB [20]. This is well within the 144 dB dynamic range of the 24-bit fixed-point Motorola DSP56002. It is for these applications that properly scaled fixed-point implementations can provide greater accuracy than a floating-point processor. Oppenheim and Weinstein [21] have shown that the noise-to-signal ratios are equal for fixed and floating point filter implementations which have an equal total wordlength. However, in their research the most conservative scaling norm ($l_1$) was used to exclude overflow in the fixed-point filter. By scaling the fixed-point structure with the less conservative $L_2$ scaling norm Van Veen and Baraniuk [22] have shown that for the same wordlength fixed-point arithmetic produces less roundoff noise distortion than floating-point arithmetic. Proper scaling of a fixed-point implementation leads to efficient use of the available wordlength.

The selection of a fixed-point versus a floating-point processor is dependent on the cost, speed, power consumption and time-to-market requirements of a particular application. Fixed-point processors will always have an advantage in the areas of cost, speed and power consumption. The additional circuitry required for floating-point arithmetic adds more complexity to the DSP's multiplier, resulting in higher cost, longer instruction cycle times, and more power consumption. This statement is supported by a
comparison of various fixed and floating-point DSPs [24]. For example, the Motorola fixed-point DSP56002 operating at 66 MHz performs 33 MIPS while consuming 450 mWatt maximum, and costs $41.06. Compare that to the Motorola floating-point DSP96002 operating at 40 MHz which performs 20 MIPS, consumes 4 Watt maximum, and costs $172.00. A recent survey predicts that in 1995 fixed-point processors will account for over half the sales in the DSP chip market [25].

The major weakness of fixed-point processors lies in the ever decreasing time-to-market requirements. The operation of scaling requires additional design effort and therefore can delay a product's time-to-market. This work will help alleviate this problem by providing a development tool for fixed-point filter scaling and analysis.
2.0 Modeling of the Motorola DSP56002 Fixed-Point DSP

The design of a practical fixed-point digital filter simulation must be based on current hardware architectures to be useful in the evaluation of a filter implementation. In this work the Motorola DSP56002 fixed-point 24-bit DSP was selected as the basis for the simulation models. Specifically, the operations of fixed-point finite wordlength addition and multiplication are modeled for the DSP56002 architecture. The DSP56002 architecture and word formats are summarized in Sections 2.1, 2.2 and 2.3, and then both non-linear and linear-stochastic Multiplier-Accumulator models are presented in Sections 2.4 and 2.5 respectively.

2.1 Motorola DSP56002 Architecture

The DSP architecture affects digital filter performance in terms of coefficient quantization, roundoff noise, limit cycles and processing speed. Below is a summary of the DSP56002 architectural features.

- Four 24-bit data registers, X0, X1, Y0, Y1
  - A parallel, single instruction cycle, multiply-accumulate instruction (MAC)
  - Two 56-bit accumulator registers, A and B composed of:
    8-bit Extension Registers, A2 and B2
    24-bit Upper Registers, A1 and B1
    24-bit Lower Registers, A0 and B0
- Three separately addressable memory spaces (Modified Harvard Architecture)
  - P, Program memory Space
  - X, Data memory space
  - Y, Data memory space
- Arithmetic Logic Unit (ALU) Accumulator shifter for automatic scaling (by factor of 2), rounding, automatic overflow detection and limiting.
The DSP56002 has other features such as parallel and serial communication ports and a programmable timer, all of which are discussed in the user manuals [1, 2]. This section concentrated on the computational aspects of the DSP56002.

Compared to a general purpose microprocessor, a DSP microprocessor has been optimized for numerical computations by using separate hardware computational units for the multiplier, accumulator, shifter, and data addressing operations. Data addressing is further optimized by the DSP’s Harvard Architecture, in which separate address and data busses allow parallel memory accesses to data and program memory in a single instruction cycle. The Motorola DSP56002 is designed with a Modified Harvard Architecture which has two separate Data Address Generators (DAGs) capable of simultaneously accessing two data memory spaces (X-data memory and Y-data memory) [3, 4]. The DAGs use indirect addressing with automatic pointer increment/decrement to implement circular buffers for a filter delay-line [1]. It is within the DSPs capability to perform a multiplication, addition to the accumulator, 1-bit shift, and two parallel memory transfers in one instruction cycle.

2.2 Fixed-Point Data Representation

The DSP56002 uses a fractional twos-complement data representation for all ALU and multiplier operations. The multiplier performs 24-bit by 24-bit word multiplications and adds the 48-bit long result to a 56-bit accumulator. Figure 2.1 compares the bit
weighting of words, long words, and accumulator word operands for fractional data representation. The DSP assumes that all values stored in memory and in registers are fractional. Data must be converted to a fractional number by fractional scaling before being stored in memory. The decimal points of each word are aligned between bits 0 and 1, with bit 0 being the sign bit.

![Diagram of word formats](image)

Figure 2.1 Motorola DSP word Formats and bit weighting for words, long words and accumulator words.

For words (24-bits) and long words (48-bits), the most positive number is $1-2^{-23}$ or $7FFFFFFF$ hexadecimal for words and $1-2^{-47}$ or $7FFFFFFFFF$ hexadecimal for long words. The most negative number that can be represented is -1, which is represented in 2's-complement format as the word $800000$ hexadecimal and as the long word $8000000000000000$ hexadecimal. This limitation also applies to all data stored in memory.
2.3 Multiplier-Accumulator Architecture

The DSP56002 contains a single 24-bit multiplier and two 56-bit accumulators (designated the A and B accumulators). The block diagram of the multiplier and a single accumulator is shown in Figure 2.2.

Two 24-bit words are passed to the multiplier through registers X and Y producing a 48-bit product which is stored in a 48-bit long word. This long word is then either stored to the accumulator or summed with the present accumulator contents. The long multiplier word retains the full precision of each multiplication result, thus multiplications do not generate finite wordlength errors. The operations of overflow saturation and rounding are discussed in Sections 2.3.1 and 2.3.2 below. The use of the accumulator shift register is discussed in Section 2.3.3 below.

2.3.1 Overflow Saturation Mode

As discussed in Section 2.2, the accumulator consists of a 48-bit fractional long word with an additional 8-bit overflow extension register. Therefore, at least $2^8$ intermediate summations may be performed without overflowing the accumulator's dynamic range of $\pm 256$. The saturation mode is enabled if the extension register overflows or if the contents of the accumulator are transferred to a 24-bit register while the extension register is non-zero. In this mode the value transferred to the 24-bit register by the ALU is either -1 or $1-2^{24}$ depending on the sign bit. Thus in an overflow condition a digital filter's output is
saturated (clipped) much like an analog active filter. The alternative to overflow saturation is 2's-complement overflow in which register values overflow from full scale positive to full scale negative and vice versa. It was shown by Mullis and Roberts [5, pp. 348-353] that 2's-complement overflow leads to limit cycle oscillations while saturation eliminates the limit cycle oscillations due to overflow; for second order direct form filter structures that is.

Figure 2.2 Typical DSP Multiplier-Accumulator Block Diagram.
2.3.2 Convergent Rounding

Assuming the number of intermediate summations is less than 256 and the final accumulator sum does not exceed unity, then rounding error represents the only loss of precision in a computation. Rounding is performed when the 48-bit fractional accumulator word is transferred to a 24-bit register or memory location. The most significant word (MSW) in the accumulator is defined to be bits 0 to 23 and the least significant word (LSW) in the accumulator is bits 24 to 47. Roundoff error is defined as the difference between the value of the 48-bit accumulator prior to rounding and the value of the MSW after rounding.

Consider the case of using conventional rounding to convert a 48-bit long word to a 24-bit word. If the magnitude of the LSW is greater than zero and less than $2^{24}$ (bit 24 equal zero), then the contents of the MSW is unchanged (i.e. the accumulator is rounded down) resulting in the following roundoff error.

$$e_{\text{down}} = \sum_{i=25}^{47} b_i 2^{-i}$$

where $b_i$ represents the binary value (1 or 0) of the $i$ th bit in the Long Word. If the magnitude of the LSW is greater than $2^{24}$, then $2^{23}$ is added to the MSW (i.e. the accumulator contents is rounded up) resulting in the following roundoff error.

$$e_{\text{up}} = -2^{-24} - \sum_{i=25}^{47} b_i 2^{-i} \Rightarrow e_{\text{up}} = -\sum_{i=25}^{47} \tilde{b}_i 2^{-i}$$
where \( \bar{b}_i \) is the 2's-complement of \( b_i \).

If the LSW equals \( 2^{-24} \), then by conventional rounding \( 2^{-23} \) is added to the MSW resulting in a rounding error of \( -2^{-24} \). The expected value (mean) of these rounding errors is given by the following.

\[
m_* = P_t e_{\text{down}} + P_g e_{\text{up}} - P_e 2^{-24} \tag{2.1}
\]

where \( P_t \) is the probability of the LSW being greater than zero and less than \( 2^{-24} \), \( P_g \) is the probability of the LSW being greater than \( 2^{-24} \), and \( P_e \) is the probability of the LSW being equal to \( 2^{-24} \). These probabilities are defined as follows.

\[
P_g = \frac{2^{-24} - 1}{2^{-24}}, \quad P_t = \frac{2^{-23} - 1}{2^{-24}}, \quad \text{and} \quad P_e = \frac{1}{2^{-24}}
\]

Substituting these probabilities into (2.1) the mean of the roundoff error for conventional rounding can be computed.

\[
m_* = \frac{2^{-23} - 1}{2^{-24}} \sum_{i=25}^{47} \bar{b}_i 2^{-i} - \frac{2^{-23} - 1}{2^{-24}} \sum_{i=25}^{47} \bar{b}_i 2^{-i} \frac{1}{2^{-24}} 2^{-24} = -\frac{1}{2^{-24}} 2^{-24} = 4 \times 10^{-15} \tag{2.2}
\]

An alternative rounding scheme called convergent rounding is used by the DSP56002 to reduce this bias. The rules for convergent rounding are the same as those for conventional rounding except when LSW is equal to \( 2^{-24} \). In convergent rounding when the LSW of the 48-bit register is equal to \( 2^{-24} \), then the MSW is rounded up only if bit 23 of the MSW is equal to 1. The following example in Figure 2.3 illustrates the operation of
convergent rounding as performed in the DSP56002 [1, pp. 3-15]. Two distinct cases are presented in which the Least Significant Word (LSW) is equal to 1/2 of the full scale value.

Accumulator bit 24 = 1 and bit 23 = 0, Do Not Round.

Accumulator Word, Before Rounding:

\[
\begin{array}{c}
xx........xx : xxx...... (MSW) ........ 010:100........ (LSW) ........ 000 \\
0 ........................................ 23 24........................................ 47
\end{array}
\]

Accumulator Word, After Rounding:

\[
\begin{array}{c}
xx........xx : xxx...... (MSW) ........ 010:000........ (LSW) ........ 000 \\
0 ........................................ 23 24........................................ 47
\end{array}
\]

Accumulator bit 24 = 1 and bit 23 = 1, Round Up.

Accumulator Word, Before Rounding:

\[
\begin{array}{c}
xx........xx : xxx...... (MSW) ........ 011:100........ (LSW) ........ 000 \\
0 ........................................ 23 24........................................ 47
\end{array}
\]

Accumulator Word, After Rounding:

\[
\begin{array}{c}
xx........xx : xxx...... (MSW) ........ 110:000........ (LSW) ........ 000 \\
0 ........................................ 23 24........................................ 47
\end{array}
\]

Figure 2.3 Example of the Motorola DSP56002's Convergent Rounding Operation.

The probability of bit 23 being 1 is equal to 1/2, and an LSW of value $2^{-24}$ results in an error of $-2^{-24}$ when rounded up and $+2^{-24}$ when rounded down. The mean roundoff error can be calculated for convergent rounding as was done for conventional rounding in (2.2).

Modeling of the Motorola DSP56002 Fixed-Point DSP 13
\[ m_* = \frac{2^{23}}{2^{24}} - \frac{1}{2^{24}} \sum_{i=25}^{47} b_i 2^{-i} - \frac{1}{2^{24}} \sum_{i=25}^{47} b^*_i 2^{-i} - \frac{1}{2^{24}} \left( \frac{1}{2} 2^{-24} - \frac{1}{2} 2^{-24} \right) = 0 \]  

(2.3)

Therefore, the bias due to convergent rounding will statistically converge to zero for long data sequences. Convergent rounding is performed in the same instruction cycle as a MAC command. The emulation of convergent rounding will be discussed in Section 2.4.

2.3.3 Automatic Scaling Mode

In many cases it will be necessary to multiply values which are not fractional. For example, second order filter coefficient magnitudes can be as large as 2.0. The automatic scaling mode makes it possible to efficiently implement these coefficients by automatically scaling the contents of the accumulator by 2 when the result is transferred from the accumulator to a 24-bit register. In this mode all filter coefficients must be scaled by 1/2 before being stored in memory, i.e. what is stored is half the actual value. When these coefficients are used for a filter operation the final accumulator contents needs to be scaled up by a factor of 2 using an automatic 1 bit left shift of the accumulator contents.

The accumulator contents can be shifted either left or right (2x or 1/2x scaling). This mode will be used when implementing second order filter sections. When a right shift is performed the least significant bit in the accumulator is lost. When using the scaling mode to perform a right shift the accumulator contents is first rounded at bit 24 (not bit 23) then the right shift is performed resulting in no loss of precision.
2.4 Non-Linear Multiplier-Accumulator Model

The operations of multiplication and addition in any fixed-point DSP processor introduce three types of finite wordlength effects, roundoff error, overflow error, and limit cycle oscillations. Roundoff error results from rounding the \(2w\)-bit accumulator contents to \(w\)-bits upon transfer to memory. Overflow errors occur when the final accumulator result exceeds the DSPs dynamic range capability of \(-1\) to \(1-2^{-w}\). Limit cycle oscillations can be a by-product of either one or both the rounding and overflow errors.

Since these errors are non-linear it is difficult to accurately predict a filter's output roundoff noise. A non-linear model of the DSP56002 multiplier-accumulator operation has been created so that these rounding and overflow errors may be investigated in a Matlab simulation of the filter implementation.

The flow chart in Figure 2.4 illustrates the multiplier-accumulator non-linear Matlab model \texttt{xquant.m}. The Multiplier-Accumulator model has been generalized to a variable register wordlength of \(w\)-bits, while preserving the DSP56002's double length multiplier resultant of \(2w\)-bits. The operations of convergent rounding and overflow saturation are modeled in \texttt{xquant.m}. The 8-bit extension register has not been modeled, and therefore it is assumed that computations do not exceed the 8-bit extension range of \(\pm256\). The m-file \texttt{xquant.m} was adapted from C. S. Burrus et. al. [27] and modified to include convergent rounding.

Referring to Figure 2.4, the input argument \(s\) to the function \texttt{xquant.m} is first quantized to \(2w\)-bits, and then multiplied by \(2^{w-1}\) yielding the decimal representation \(X_{1}\).
The operation $\text{fix()}$ is used to determine the integer portion of $X_1$, and the operation $X_1 - \text{fix}(X_1)$ results in the fractional portion of $X_1$. The fractional portion of $X_1$ is then tested for equality to 0.5, which corresponds to an LSW equal to $2^{24}$. If it is not equal, conventional rounding is performed on $X_1$. If the equality is true, then convergent rounding is performed. With convergent rounding the condition for rounding up is that bit 23 of the MSW must be equal to one, which corresponds to the integer portion of $X_1$ being an odd number.

Figure 2.4 Flow Graph of Rounding and Overflow Non-Linear Simulation Routine.
Convergent rounding is performed by the operation \( \text{fix}(X1) + \text{rem}(\text{fix}(X1), 2) \). The function \( \text{rem}(\text{fix}(X1), 2) \) returns a 1 if the integer portion of \( X1 \) is odd, and therefore \( X1 \) is truncated and then incremented by 1. If the integer portion of \( X1 \) is even then \( \text{rem}(\text{fix}(X1), 2) \) returns zero, and then \( X1 \) is replaced by its integer value. After rounding \( X1 \) is tested for overflow, and then converted to a \( w \)-bit fractional value.

Figure 2.5 illustrates the use of this model in a single pole IIR filter. Note that rounding and overflow only occur after the summation and are denoted by the operators \( \mathbf{R}[\cdot] \) and \( \mathbf{S}[\cdot] \) respectively. This model assumes that the input and multiplier values are quantized to \( w \)-bits. The operator \( \mathbf{Q}[\cdot] \) represents quantization to 8-bits, while the input \( x(n) \) and the coefficients \( a \) and \( b \) are converted to 8-bit values. This model has been validated by comparison to the DSP56002 output for various filter implementations (see Chapter 7).

![Diagram of Single Pole IIR Filter](image)

Figure 2.5 Example of the application of the Multiplier-Accumulator Non-Linear Model
It was stated in Section 2.3.2 that conventional rounding results in a bias due to the value of 1/2 always being rounded up and that convergent rounding reduces this bias by rounding 1/2 up if the MSW is odd and down if it is even. A test was performed to verify this statement using the single pole filter in Figure 2.5 above. The following parameters were used: \( a = -0.737, b=0.5 \) and a wordlength \( w \) of 8-bits. The input \( x(n) \) is a 20480 point pseudorandom sequence uniformly distributed between \( \pm 0.5 \) with zero mean. The filter output was generated by a Matlab simulation using the above convergent rounding algorithm and conventional rounding respectively. Table 2.1 lists the resulting mean and variance of the roundoff error for both convergent and conventional rounding. In this case convergent rounding reduced the mean roundoff error by an order of magnitude, with an increase in variance of less than a 0.4%. Total noise power differs by less than 0.01dB.

<table>
<thead>
<tr>
<th>Rounding Error</th>
<th>Mean</th>
<th>Variance</th>
<th>Power (dB)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Conventional Rounding</td>
<td>1.0779e-004</td>
<td>1.1012e-005</td>
<td>-49.577</td>
</tr>
<tr>
<td>Convergent Rounding</td>
<td>4.5234e-005</td>
<td>1.1054e-005</td>
<td>-49.564</td>
</tr>
</tbody>
</table>

Statistically, even and odd values are equally likely to occur and therefore the bias converges to zero. Since the probability of a convergent rounding operation occurring is equal to \( 2^w \), its effect is dependent on the wordlength. In most cases convergent rounding operations are rare and could be ignored if the observation time is less than \( 2^w \).
2.5 Linear-Stochastic Roundoff Noise Model

This section reviews the classical stochastic model for roundoff error. The linear-stochastic model is used to represent the nonlinear roundoff uncertainties produced by finite-precision arithmetic. Oppenheim and Schafer [6, pp. 351-359] outline the stochastic quantization model, where the rounding non-linearity $R[.]_\omega$ at the summation output is replaced by an additive noise source equivalent to the roundoff error at that node (see Figure 2.6). The roundoff error noise is defined as follows

\[ e(n) = R[s(n)]_\omega - s(n) \]  \hspace{1cm} (2.4)

where $s(n)$ represents the accumulator summation output.

![Figure 2.6 Linear-Stochastic Roundoff Noise model.](image)

The following classical assumptions are used to model the quantization errors as a linear stochastic process.

1. Each roundoff noise source $e(n)$ is a wide-sense stationary white-noise process.
2. Each noise source has a finite uniform distribution of amplitudes over one quantization interval.

3. Each roundoff noise source is uncorrelated with its corresponding signal input, all other roundoff noise sources, and the input to the system.

Rounding errors are assumed to have zero mean and a uniform distribution between plus and minus one-half of the finite word length step size $q = 2^{-(w-1)}$, as shown in the probability distribution function in Figure 2.7. For a $w$-bit register word length the roundoff error is uniformly distributed in the range $-\frac{2^{-w-1}}{2} < e(n) \leq \frac{2^{-w-1}}{2}$.

![Figure 2.7: Quantization Model Distribution.](image)

The statistics for each noise source are characterized by the following mean and variance.

$$m_e = 0$$  \hspace{1cm} (2.5)

$$\sigma_e^2 = \frac{2^{-2(w-1)}}{12}$$  \hspace{1cm} (2.6)
This model has been simplified to take advantage of the computational ease and speed of a linear simulation by neglecting the overflow saturation operation $S[ ]_1$. Therefore, this model assumes that each filter structure has been scaled such that overflow does not occur at all or such that the probability of overflow is negligibly small. At each summation output node an independent noise source $e_i(n)$ with the mean (2.5) and variance (2.6) is added to the signal. Any filter structure can then be simulated as a multi-input single-output linear system with input $x(n)$ and stochastic inputs $e_i(n)$. 

Modeling of the Motorola DSP56002 Fixed-Point DSP
3.0 Filter Structure Simulation Description

The major contribution of this work is the development of a set of digital filter simulations for use in evaluating the performance of various structures using fixed-point arithmetic with variable wordlength. These simulations were written in Matlab and incorporate the fixed-point DSP56002 non-linear and linear stochastic models developed in Sections 2.4 and 2.5. A second contribution has been the development of an interface between the Matlab environment and the Motorola DSP56002 Application Development System (ADS). This interface allows the direct evaluation of a DSP56002 filter implementation using filter coefficients and input data generated in Matlab.

3.1 Matlab Filter Simulation

The Matlab simulation is composed of several m-files which exchange variables through the Matlab workspace. A general block diagram of the filter simulator is shown in Figure 3.1, which also lists the filter and simulation variables used and created by each m-file. A simulation m-file has been written for each filter structure. In Figure 3.1 the simulation m-file is named \textit{\{type\}sim.m} where \textit{\{type\}} represents a set of characters which identify the structure. Figure 3.1 also shows four supporting m-files, each of which will be discussed in this section with the exception of the DSP56002-Matlab Interface which will be discussed in Section 3.2 below.
Figure 1 Filter Simulator General Block Diagram.
The supporting m-files `filtread.m`, and `gensig.m` perform the following functions.

`filtread.m`

This m-file reads the filter design data file created by the Atlanta Signal Processors Inc.'s Digital Filter Design Package 3.0 [7].

`gensig.m`

A signal generator which creates a sinusoid or pseudorandom input test signal $x$. The user may specify the number of samples in the sequence and the input's quantization resolution.

Each structure simulation `{type}sim.m` performs the following operations.

1) Compute the filter structure coefficients and scale for overflow.
2) Quantize the coefficients to $w$-bits.
3) Generate DSP56002 data file, assemble and link DSP56002 filter code.
4) Run linear stochastic simulation.
5) Run non-linear simulation.
6) Run DSP56002 filter on Motorola development system.

For each filter structure separate m-files were written to compute the structure's scaled coefficients, run the stochastic simulation and run the non-linear simulation. The filter simulation prompts the user to enter structure parameters for overflow scaling and quantization wordlength. The details of coefficient computation and overflow scaling are discussed in Chapter 4 for Direct Form Structures, Chapter 5 for the State-Space Section Optimal structure and Chapter 6 for the Lattice structures.
Using the simulations each filter structure can be compared in terms of the frequency response error due to coefficient quantization, output roundoff noise power $P_e$, and roundoff noise power spectral density. A plot of the frequency response error is made by simply subtracting the frequency responses of the quantized and unquantized filters and plotting on a linear scale. The filter's output roundoff error $e_n$ for the non-linear simulation, and $e_l$ of the linear stochastic simulation, is calculated as the difference between the response of the $w$-bit fixed-point structure model and the same structure implemented with Matlab's double precision floating-point precision. The resulting roundoff noise power is computed by (3.1) from the roundoff noise error $e$.

$$P_e = \sigma_e^2 + (m_e)^2$$  \hspace{2cm} (3.1)

The filter simulations assume that the filter coefficients and input signal $x_i$ exist in the Matlab workspace before running the simulator. For the IIR structures the coefficients must exist as a second order system matrix named sos [8]. For FIR structures the coefficients must be stored in the vector Num.

3.2 DSP56002 Code Generation

A simple DSP56002 filter code generator has been built into each structure simulation m-file. Given the quantized filter coefficients, desired fixed-point wordlength and scaling integers the filter simulator creates a DSP56002 assembly file name.ssm where name is user specified. This assembly file can be assembled, linked into a loadable object file name.cld, and executed on the Motorola DSP56002 ADS; all from within the Matlab
environment. The assembly file created by the Matlab simulators have the format of Listing 3.1 below.

```assembly
; ***** DIRECT FORM I CASCADE *****
; Target Processor: DSP56002
; DATE: 7/29/1995     TIME: 12:27

page 132,66,0,10       ; Define lists file page size
opt cex,mex            ; expand macros in list file
maclib 'cgen56\'        ; path for macro files

; Data and Coefficients
coef equ $0            ; starting address for coefficient data storage
state equ $0            ; starting address for filter state storage
nsec equ $3             ; number of filter sections
nbits equ $18           ; fixed-point wordlength
NB equ $0               ; power of 2 scaling gain

; org x:state
S1 ds nsec             ; filter state storage
S2 ds nsec             ;

; org y:coef           ; {Filter Coefficients}
dc $1039               ; 0.5*bq[1][0]= 0.00049507617950
dc $2072               ; 0.5*bq[1][1]= 0.00099015235901

; include 'cgen56\filtmain.asm'       ; main filter routine
; include 'cgen56\uirlf\iird1cb.asm'  ; filter structure macro
end
```

Listing 3.1: Sample assembly code generated by filter simulation file d1csim.m for the implementation of Cascaded 2nd order Direct Form I sections.

As seen in Listing 3.1, the scaled filter coefficients, together with other parameters, are saved to an assembly file. The filtmain.asm and filter sub-routine iird1cb.asm assembly
files are then included into this file. The assembly file `filtmain.asm` stores an input data file to the data buffer in the DSP56002's X-data memory, calls the filter sub-routine to process the data and saves the filtered output data to a data file. These operations are detailed in Section 3.3 below.

Each filter structure implementation has been written as an assembly sub-routine, such as `iird1cb.asm` for the Cascaded Direct Form I structure. These sub-routines (also called macros) must be in the format of Listing 3.2 to be "included" into the `name.asm` file as shown in Listing 3.1.

```assembly
; iird1cb.asm
;
; Input in $x0, Output to $x0

SECTION iird1cb
XREF coef,nsec,nbits
XDEF FILT_INIT,FILT_START

org P:$.6a
FILT_INIT
  { Initialization of memory and registers }
  rts

FILT_START
  { Filter Routine }
  rts
  nop
ENDSEC

Listing 3.2: Format for DSP5602 filter macro assembly code.

In Listing 3.2, the filter implementation is divided into an initialization section `FILT_INIT` where registers and data buffers are loaded or cleared, and the filter
algorithm section FILT_START. All filter implementations are written with the
coefficients residing in the Y-data memory space and filter states stored to the X-data
memory space.

3.3 Matlab to DSP56002 Interface

The Motorola Application Development System (ADS) is a tool for the development
and implementation of real time signal processing algorithms and software. The ADS
configuration, pictured in Figure 2.1, consists of a Personal Computer (PC) with interface
board, the On Chip Emulation (OnCE) command converter board and the DSP56002
Application Development Module (ADM). The OnCE port is a direct serial interface to
the DSP56002's internal core registers, through which the ADS control software can up-
load user programs to the ADS, up/down-load data files, display and alter program or data
registers, and debug programs. The command converter board performs command
translation between the OnCE port and the PC software. Development software, running
on the PC, supervises the transfer of DSP56002 registers, memory and status data
between the PC and the DSP56002 OnCE port. The DSP56002 OnCE port also allows
the ADS software to issue single step and break point debug control commands. The
DSP56002 software can be executed on the ADM board under the control of the OnCE
converter board or the AMD can operate as a stand alone DSP system.
The Matlab program \texttt{dsp56i.m} and the DSP56002 program \texttt{filtmain.asm} were created to automate the transfer of data between the Matlab environment and the DSP56002 ADS. The program \texttt{dsp56i.m} takes the name of a loadable DSP56002 object file and a data vector such as \texttt{xi}, and creates a DSP56002 ADS command file \texttt{dsp56.cmd} and ascii data file \texttt{dspin.dat}. The command file \texttt{dsp56.cmd} instructs the ADS to reset the development system, load the object code into program memory, open the data file \texttt{dspin.dat} for input, open an output file \texttt{dspout.dat}, store the length of \texttt{xi} to X-data memory location 1FF hexadecimal, to then execute the \texttt{force ru} command which executes the filter program described in Section 3.2 above. A sample command file is shown below in Listing 3.3. The ADS command file can be automatically executed when starting the ADS software from within Matlab using the command \texttt{!ads56002 dsp56.cmd}.

The assembly routine \texttt{filtmain.asm} clears system registers, and then loads the input data file \texttt{dspin.data} into a 4K x 24-bit buffer in X-data memory starting at address 200
hexadecimal. To perform this data transfer the following registers must be initialized by
fitmain.asm: X0 is loaded with the number of values in dspin.dat, R0 must contain the
starting address of the data block (200 hex), and R1 must contain the value of 1 to
indicate that the data is to be stored in the X-data memory. The processor is then placed
in debug mode by the debug instruction at program address FIN. The data transfer is
then performed through the OnCE port to the DSP56002 ADS memory. After having
filled the input buffer, fitmain.asm passes each data sample to the filter sub-routine and
stores the filter output samples back to the same data buffer. The output samples are then
transferred to the output file dspout.data as described above. A detailed description of
the Motorola ADS input and output operations for the DSP56002 ADS is available from
the manufacturer [9].

```
force r                       ; Reset processor.
load test                    ; Load object module.
output off                   ; Close existing output
input off                    ; and input files.
change x:$1ff $400           ; Store length of input data.
input P:FIN indsp.dat -rh    ; Open input data file.
output P:FOUT outdsp.dat -rh ; Open output file
force ru                     ; Reset and run program.
wait 10                      ; Wait for processor to finish.
quit                         ; Exit ADS program.
```

Listing 3.3: Sample ADS command file.
4.0 Implementation of Direct Form Filter Structures

The Direct Form structures are the most simplistic IIR digital filter structures due to the direct relationship between the coefficients of the difference equation and the coefficients of the filter transfer function $H(z)$. In general a filter is composed of a gain $b_0$, zeros $z_i, i=1\ldots M$, and poles $p_i, i=1\ldots N$. As shown in (4.1) the zeros and gain combine to form an $M$th order numerator polynomial and the poles combine to form the $N$th order denominator polynomial. It is assumed that the transfer function is rational.

$$H(z) = \frac{b_0(z - z_1)(z - z_2)\ldots(z - z_M)}{(z - p_1)(z - p_2)\ldots(z - p_N)} = \frac{b_0 + b_1z^{-1} + b_2z^{-2} + \ldots + b_Mz^{-M}}{1 + a_1z^{-1} + a_2z^{-2} + \ldots + a_Nz^{-N}}$$  (4.1)

The filter transfer function in (4.1) is the basis for the implementation of the following filter structures.

i. High Order Direct Form II.

ii. Cascade of 2nd Order Direct Form II Sections.

iii. Parallel 2nd Order Direct Form II Sections.

iv. Cascade of 2nd Order Direct Form I Sections.

The following sections discuss how the coefficients for these structures are obtained, the scaling of each structure to prevent overflow, the implementation of each structure on the Motorola DSP56002, and both the non-linear and linear stochastic quantization error models.
4.1 High Order Direct Form II

The High Order Direct Form II structure is composed of a single section, therefore all poles (zeros) are coupled through the system transfer function numerator (denominator). The corresponding canonical difference equations shown below follow directly from (4.1).

\[ y(n) = b_0 s(n) + b_1 s(n-1) + b_2 s(n-2) + \ldots + b_M s(n-M) \]  \hspace{1cm} (4.2a)

\[ s(n) = x(n) - a_1 s(n-1) - a_2 s(n-2) - \ldots - a_N s(n-N) \]  \hspace{1cm} (4.2b)

As discussed in Section 2.2, fixed-point DSP hardware such as the Motorola DSP56002 is limited to signed fractional values in the range of -1 to 1-2^-W. In general, a fixed-point processor's dynamic range is from -1 to 1-2^-W, where W represents the processor's register wordlength. This dynamic range limitation introduces non-linearity into (4.2a, b). The signal flow graph in Figure 4.1 illustrates the fixed-point implementation of the High Order Direct Form II structure. The operators \( R[\cdot]_w \) and \( S[\cdot]_1 \) represent the non-linear operations of Convergent Rounding and Saturation performed by the Arithmetic-Logic Unit (ALU) of the Motorola fixed point DSP56002 DSP. These non-linear operations will be modeled as in Sections 2.2 and 2.3.

As discussed in Section 2.1, rounding and saturation occur at the end of a summation operation when the contents of the accumulator is rounded to W-bits and transferred to a register or memory location. Computations which exceed or overflow the dynamic range are saturated by the \( S[\cdot]_1 \) operator to the maximum value, resulting in overflow errors. These errors can be controlled by introducing the overflow scaling gain \( B_0^* \) at the filter.
input, resulting in a scaled state vector $s_i^*$ at the output of the summation operation. The numerator coefficients $b_i$ must also be scaled to restore the overall filter gain, and these scaled coefficients are denoted as $b_i^*$. Coefficients or transfer functions modified by overflow scaling are denoted by an asterisk.

![Diagram of High Order Direct Form II Signal Flow Graph](image)

**Figure 4.1: High Order Direct Form II Signal Flow Graph.**

For filter orders equal to or greater than two (2), the magnitude of the Direct Form structure numerator and denominator coefficients is not constrained to unity. Therefore, the filter coefficients must also be scaled to a fractional value for representation in DSP memory. This scaling will be referred to as fractional scaling and it is implemented by dividing the numerator and denominator coefficients by powers of 2 as follows, $a_i/2^{na} \quad i=1...N$ and $b_i^*/2^{nb} \quad i=0...M$. Where nb and na are the minimum powers of 2 necessary to
scale the maximum $b_i^*$ and $a_i$ into the allowed dynamic range. Fractional scaling is compensated by shifting the accumulator contents to the left $nb$-bits after computing (4.2a) and $na$-bits left after computing (4.2b).

Incorporating the operations of fractional scaling and overflow scaling into (4.2a, b) results in the following.

\[
y(n) = S \left[ R \left[ 2^{nb} \left\{ \frac{b_0^*}{2^{nb}} s^*(n) + \frac{b_1^*}{2^{2nb}} s^*(n-1) + \frac{b_2^*}{2^{3nb}} s^*(n-2) + \cdots + \frac{b_M^*}{2^{Mnb}} s^*(n-M) \right\} \right] \right]
\]

(4.3a)

\[
s^*(n) = S \left[ R \left[ 2^{na} \left\{ \frac{B_0^*}{2^{-na}} x(n) - \frac{a_1}{2^{na}} s^*(n-1) - \frac{a_2}{2^{2na}} s^*(n-2) - \cdots - \frac{a_N}{2^{na}} s^*(n-N) \right\} \right] \right]
\]

(4.3b)

Equations (4.3a, b) represent the fixed-point implementation of the High Order Direct Form II structure. The following section discusses the calculation of the overflow scaling gain $B_0^*$ and the scaled coefficients $b_i^*$.

### 4.1.1 Overflow Scaling

The High Order Direct Form II structure is scaled according to the method described by Jackson [10, pp. 309-322]. Referring to Figure 4.1, the scaled transfer function from the input $x(n)$, to the state node $s^*(n)$, is defined as $F^*(z)$, where the asterisk represents the scaled transfer function.
\[ F^*(z) = \frac{S^*(z)}{X(z)} = \frac{B^*_0}{1 + a_1 z^{-1} + a_2 z^{-2} + \ldots + a_N z^{-N}} \quad (4.4) \]

It is desired that the norm of the response from the input \( x(n) \), to the state node \( s^*(n) \), be bounded by unity to prevent internal overflows. This scaling requirement is expressed by \( \| F^* \|_p \leq 1 \), which is interpreted as the \( p \)-norm of the scaled branch node transfer function being constrained to be less-than or equal to unity. This is accomplished by defining the scaling gain \( B^*_0 \) as follows.

\[ B^*_0 = \frac{1}{\| F^* \|_p} \quad (4.5) \]

The unscaled branch node transfer function \( F(z) \) is defined as,

\[ F(z) = \frac{1}{1 + a_1 z^{-1} + a_2 z^{-2} + \ldots + a_N z^{-N}} \quad (4.6) \]

Given the filter's Direct Form II state-space description [\( A, B, C, d \)], (4.6) can be conveniently calculated as follows.

\[ F(z) = B \left( z I - A \right)^{-1} \quad (4.7) \]

The scaling gain \( B^*_0 \) will be calculated from (4.5) using one of the \( L_p \)-norms described by Jackson [10]. The \( L_p \)-norm of \( F(z) \) is given by the following equation.
\[ L_p : \left\| F_i(\omega) \right\|_p = \left[ \frac{1}{\omega_s} \int_0^{\omega_s} \left| F_i(\omega) \right|^p d\omega \right]^{1/p} \]  

Specifically, the \( L_1, L_2, \) or \( L_\infty \) norms are the most useful norms. For each \( p \)-norm, (4.8) must be approximated using the following procedure obtained from Hall [11].

1) Calculate the impulse response associated with \( F(z) \) in Matlab using

\texttt{dimpulse.m} for \( N \) points (\( N \) is user-specified).

2) Compute the \( N \) point Discrete Fourier Transform (DFT) of the above impulse response sequence using Matlab's Fast Fourier Transform (FFT).

3) Approximate the \( L_1, L_2, \) or \( L_\infty \) norms from \( F_N(k \cdot 2\pi/N) \), samples of the \( N \)-point Discrete Fourier Transform (DFT), using one of the following equations.

\[ L_1 : \left\| F(\omega) \right\|_1 \approx \frac{1}{N} \sum_{k=0}^{N-1} \left| F_N \left( k \cdot \frac{2\pi}{N} \right) \right| \]  

\[ L_2 : \left\| F(\omega) \right\|_2 \approx \left[ \frac{1}{N} \sum_{k=0}^{N-1} \left| F_N \left( k \cdot \frac{2\pi}{N} \right) \right|^2 \right]^{1/2} \]  

\[ L_\infty : \left\| F(\omega) \right\|_\infty \approx \max_{0 \leq k \leq N-1} \left| F_N \left( k \cdot \frac{2\pi}{N} \right) \right| \]  

We now have \( \left\| F^* \right\|_p \), and therefore \( B_0^* \) in (4.5), so the scaled numerator coefficients can be calculated as follows.

Implementation of Direct Form Filter Structures 36
\[ b_k^* = \frac{b_k}{B_0^*} \] (4.10)

4.1.2 Structure Conversion

The Matlab m-file `sos2df2.m` forms the high order numerator and denominator polynomials, and performs overflow scaling and fractional scaling. The flowchart in Figure 4.2 illustrates the order in which these operations are performed and indicates the equations used. The original filter description must exist in Matlab's second order system matrix format under the variable name `sos`. For details on the Second Order Section matrix format see the Matlab Signal Processing Toolbox [8]. From the `sos` matrix the High Order Direct Form II numerator and denominator polynomials of (4.1) are formed.

Referring to Figure 4.2, the m-file `snorm.m` approximates the \( L_1, L_2 \) or \( L_\infty \) norms as indicated in (4.9a, b, c) respectively and returns the norm of the state. The input scaling gain \( B_0^* \) is calculated from (4.5) and stored as `B0s`. The scaled numerator coefficients \( b_i^* \) are calculated from (4.10), then fractionally scaled to less than unity by \( 1/2^{nb} \), and stored to the Matlab workspace as vector `Nums`. The denominator coefficients \( a_i \) are fractionally scaled to less than unity by \( 1/2^{na} \) and stored as vector `Dens`.

The Matlab m-file `df2hisim.m` is the main simulation user interface for the High Order Direct Form II Structure. This routine quantizes the coefficients in `Nums`, `Dens`, and `B0s` to \( w \)-bits and stores them as `Numsq`, `Densq`, and `B0sq` respectively. The user may also generate a text file with all filter coefficients in a format which can be directly included
into a Motorola DSP56002 assembly file for filter testing. This routine also allows the user to run the non-linear and linear stochastic simulations discussed in Sections 4.1.5 and 4.1.6 below.

Figure 4.2: Flowchart for sos2df2.m and snorm.m.

4.1.3 DSP56002 Implementation

The Motorola DSP56002 assembly code in file iird2b.asm implements difference equations (4.3a) and (4.3b). The data structures for this code are pictured in Figure 4.3. The coefficients $B^*$, $a_j/2^n$, and $b_j/2^n$ are stored in a circular buffer located in Y data memory and indexed by the data address pointer register R4. The filter states, $s^*(n)$ to $s^*(n-N)$, are stored in a circular buffer in X-data memory and indexed by the data address
pointer register R1. This program was adapted from Motorola's IIR/FIR applications notes [14].

The filter code in iird2b.asm is a combination of two subroutines. The first, starting at address reference FILT_INIT, initializes memory address registers and clears the delay-line. This section is called once at the beginning of the main filter routine. The second subroutine, starting at address reference FILT_START, implements the difference equations (4.3a, b). This filter routine multiplies the input value \( x(n) \), assumed to be stored in register \( x0 \), by the gain \( B^*_0 \) and then left and right shifts by \( nB \) and \( na \) bits respectively. The state node \( s^*_n(n) \), in (4.3b) is then calculated, and its value rounded and stored to the delay line in X-memory. Equation (4.3a) is calculated from the present and past states stored in the delay line, and the resulting accumulator contents is then shifted by \( nb \)-bits, rounded to \( w \)-bits and stored to register \( X0 \). This routine requires a total of \( 4*N+25 \) instruction cycles, where \( N \) represents the filter order.

![Data Structures for iird2b.asm](image)

Figure 4.3: Data Structures for iird2b.asm, High Order Direct Form II DSP56002 implementation.

Implementation of Direct Form Filter Structures 39
4.1.4 Non-Linear Simulation Model

The Matlab m-file d2_nlm.m was written to simulate the Direct Form II High Order Structure by emulating the convergent rounding and saturation performed by the Motorola DSP56002. The routine returns the non-linear filtered output signal \( y_n \), and filter states \( S_n \) given the input vector \( x_i \), fractionally scaled and quantized coefficients in \( Numsq \) and \( Densq \), quantized scaling gain \( B0sq \), and integers \( na \) and \( ub \). This simulation implements the difference equations (4.3a, b) using the convergent rounding and saturation models developed in Section 2.2. These models include the accumulator dynamic range constraint represented by the limiting operator \( S[\cdot]_1 \), and convergent rounding operation \( R[\cdot]_w \).

4.1.5 Linear-Stochastic Simulation Model

The Linear Stochastic Model described by L.B. Jackson [10] is shown below in Figure 4.4. As described in Section 2.3, each non-linear quantization and rounding operator \( R[\cdot]_w \) has been replaced by a uniform uncorrelated noise source denoted by \( e_1(n) \) and \( e_2(n) \). The saturation non-linearity \( S[\cdot]_1 \) was not modeled, estimates of roundoff noise are obtained through a linear simulation.

Referring to Figure 4.4, \( G_1^*(z) \) and \( G_2^*(z) \) represent the roundoff noise transfer functions from the noise sources \( e_1(n) \) and \( e_2(n) \), respectively, to the filter output \( y(n) \) and are defined below.
\[ G_1^*(z) = H^*(z) = \frac{2^{nb} \left\{ \frac{b_0^*}{2^{nb}} + \frac{b_1^*}{2^{nb}} z^{-1} + \frac{b_2^*}{2^{nb}} z^{-2} + \ldots + \frac{b_M^*}{2^{nb}} z^{-M} \right\}}{2^{na} \left\{ \frac{1}{2^{na}} + \frac{a_1}{2^{na}} z^{-1} + \frac{a_2}{2^{na}} z^{-2} + \ldots + \frac{a_N}{2^{na}} z^{-N} \right\}} \] (4.11a)

\[ G_2^*(z) = 1 \] (4.11b)

Equations (4.12a), (4.12b) and (4.12c) below represent the state-space model for the structure in Figure 4.4. These state equations are developed in the control canonical form.

Figure 4.4: High Order Direct Form II linear-stochastic model.
Input Equation: \[ u(n) = B_0^* \cdot x(n) \cdot \frac{1}{2^{na}} \]  

(4.12a)

State Equation:

\[
\begin{bmatrix}
    s_1^* (n+1) \\
    \vdots \\
    s_{N-1}^* (n+1) \\
    s_N^* (n+1)
\end{bmatrix} =
\begin{bmatrix}
    0 & 1 & \cdots & 0 \\
    \vdots & \ddots & \ddots & \vdots \\
    0 & \cdots & 0 & 1 \\
    -a_N & \cdots & -a_2 & -a_1
\end{bmatrix}
\begin{bmatrix}
    s_1^* (n) \\
    \vdots \\
    s_{N-1}^* (n) \\
    s_N^* (n)
\end{bmatrix} +
\begin{bmatrix}
    0 & 0 & 0 \\
    \vdots & \ddots & \ddots & \vdots \\
    0 & \cdots & 1 & \vdots \\
    1 & 1 & 0
\end{bmatrix}
\begin{bmatrix}
    2^{na} u(n) \\
    e_1 (n) \\
    \vdots \\
    e_2 (n)
\end{bmatrix}
\]

(4.12b)

Output Equation:

\[
y(n) = 2^{nb} \left( \frac{b_0^*}{2^{nb}} - 2^{na} \frac{a_N}{2^{na}} \frac{b_0^*}{2^{nb}} \right) \cdots 2^{nb} \left( \frac{b_1^*}{2^{nb}} - 2^{na} \frac{a_2}{2^{na}} \frac{b_0^*}{2^{nb}} \right) 2^{nb} \left( \frac{b_1^*}{2^{nb}} - 2^{na} \frac{a_1}{2^{na}} \frac{b_0^*}{2^{nb}} \right) \begin{bmatrix}
    s_1^* (n) \\
    \vdots \\
    s_{N-1}^* (n) \\
    s_N^* (n)
\end{bmatrix} +
\begin{bmatrix}
    2^{nb} \frac{b_0^*}{2^{nb}} & 2^{nb} \frac{b_0^*}{2^{nb}} & 1
\end{bmatrix}
\begin{bmatrix}
    2^{na} u(n) \\
    e_1 (n) \\
    e_2 (n)
\end{bmatrix}
\]

(4.12c)

Note that the \( b_i^* \) coefficients are quantized to \( w \)-nb bits and the \( a_i^* \) coefficients are quantized to \( w \)-na bits.

The Matlab m-files sosd2ss.m and Isimd2.m were written to form and simulate the state-space model of equations (4.12a, b and c). Routine Isimd2hi.m returns the results from the linear-stochastic model in the form of the filter output signal \( y_l \), and the filter states \( Sl \), from a given input vector, \( x_l \), scaled and quantized coefficients in vectors Numsq and Densq, scaling gain \( B0sq \), and the fractional scaling constants \( na \) and \( nb \).
4.2 Cascaded Second Order Direct Form II Structure

The Cascaded 2nd Order Direct Form II Structure is obtained by factoring (4.1) into complex conjugate pole and zero pairs. These conjugate pole and zero pairs are combined to form a product of second order Direct Form II sub-filters (sections) as follows. Real valued poles and zeros are also combined into second order sections.

\[ H(z) = B_0 \prod_{i=1}^{L} H_i(z) \quad (4.13a) \]

\[ H_i(z) = \frac{b_{0i} + b_{1i}z^{-1} + b_{2i}z^{-2}}{1 + a_{1i}z^{-1} + a_{2i}z^{-2}} \quad (4.13b) \]

In (4.13a) the product's upper limit L, is the integer part of \((N+1)/2\) where N equals the filter order. Each first or second order section (4.13b) is realized as a Direct Form II structure.

Oppenheim and Schafer [6, pp. 345-344] showed that this structure exhibits less coefficient quantization sensitivity than the High Order Direct Form structure because the poles and zeros are decoupled across sections. The following guidelines for pole-zero pairing and section ordering were developed by L.B. Jackson [10, pp.322-323].

I. Pole pairs closest to the unit circle should be paired with the nearest zeros.

Thus the high gain of the poles is mitigated by the adjacent zeros. Pairing is continued with the next pole pair closest to the unit circle, etc.
II. For a narrow band input signal the sections are ordered according to their "peakedness", which Jackson defines as follows.

\[ \rho_i = \frac{\|H_i(z)\|_\infty}{\|H_i(z)\|_2} \]  

(4.14)

Ordering occurs from the section with the lowest to the highest \( \rho \). This ordering scheme reduces the amount of quantization noise power passed to the output by placing the narrower band filters last.

Each 2nd order section is implemented using the following difference equations.

\[ y_i(n) = b_{0i} s_i(n) + b_{1i} s_i(n - 1) + b_{2i} s_i(n - 2) \]  

(4.15a)

\[ s_i(n) = u_i(n) - a_{1i} s_i(n - 1) - a_{2i} s_i(n - 2) \]  

(4.15b)

As with the High Order Direct Form II structure, the coefficients of this structure are not constrained to be fractional; however, for a stable second order section, the coefficients are less than or equal to 2.0 in magnitude. The numerator coefficients of the \( L \)th section are an exception in that these coefficients may exceed 2.0 due to the overall filter gain requirement. In this case fractional scaling by \( 1/2^{nB} \) is required for the \( L \)th section's numerator coefficients \( b_i^* \). The integer \( nB \) represents the power of 2 necessary to scale the \( b_i^* \) coefficients to less than unity magnitude. The modified Cascaded 2nd Order Direct Form II structure as shown in Figure 4.5 can be implemented on any fixed-point
Figure 4.5: Cascaded 2nd Order Direct Form II Structure Signal Flow Graph.
DSP using fractional numeric representation. The following difference equations describe this structure.

\[ u_1(n) = \frac{B_0^*}{2} x(n) \]  

(4.16a)

\[ s_{i}^*(n) = S \left[ R \left[ 2 \left( u_i(n) + \frac{a_{2i}}{2} s_i^*(n-1) + \frac{a_{2i}}{2} s_i^*(n-2) \right) \right] \right] \]  

(4.16b)

\[ y_i(n) = \left\{ \frac{b_{2i}^*}{2} s_i^*(n) + \frac{b_{2i}^*}{2} s_i^*(n-1) + \frac{b_{2i}^*}{2} s_i^*(n-2) \right\} \]  

(4.16c)

\[ y(n) = S \left[ R \left[ 2^{n_0-1} y_L(n) \right] \right] \]  

(4.16d)

As discussed in Section 2.3.3, the Motorola DSP56002 employs an automatic scaling mode for the implementation of second order filter sections. In this mode each filter coefficient is multiplied by 1/2 to ensure that it is less than unity magnitude. To compensate for this scaling the accumulator contents is automatically shifted left by one bit (x2) prior to transferring the accumulator contents to registers or memory. This 2x scaling occurs with no additional processing overhead and automatically handles rounding and saturation. The automatic 2x is illustrated in Figure 4.5 prior to rounding and saturation.
4.2.1 Overflow Scaling:

Each cascaded Direct Form II section is scaled according to the methods described by L.B. Jackson [10]. Referring to Figure 4.5, the transfer function from the input \( x(n) \), to the summation node (or state vector) \( s_i^*(n) \), is defined as \( F_i^*(z) \) where the asterisk represents the scaled transfer function.

\[
F_i^*(z) = \frac{B_0^*}{1 + a_1 z^{-1} + a_2 z^{-2}} \prod_{k=1}^{i-1} H_k^*(z) \tag{4.17}
\]

As with the Direct Form High Order structure, scaling is performed such that the \( p \)-norm of \( F_i^*(z) \) is bounded by unity to prevent internal overflows in the \( i \)th second order section. For each cascaded section the following transfer function must be scaled to prevent overflow.

\[
F_i(z) = \frac{1}{1 + a_1 z^{-1} + a_2 z^{-2}} \prod_{k=1}^{i-1} H_k(z), \quad \text{where } i = 1, \ldots, L \tag{4.18}
\]

An overflow scaling gain is computed for each section as follows.

\[
K_i = \frac{1}{\|F_i\|_p}, \quad \text{where } i = 1, \ldots, L \tag{4.19}
\]

The \( p \)-norm of \( F_i \) is approximated using (4.9a, b, or c) as discussed in Section 4.1.2. The scaling gain \( B_0^* \) and scaled filter coefficients are calculated as follows.
\[ B_0^* = K_1 \]  \hspace{1cm} (4.20)

\[ b_{ki}^* = \frac{K_{i+1}}{K_i} b_{ki}, \quad i = 1, ..., L \text{ and } K_{i+1} = B_0 \]  \hspace{1cm} (4.21)

4.2.2 Structure Conversion

The Matlab m-file sos2df2c.m performs the operations of overflow scaling and coefficient magnitude scaling on each filter section. Figure 4.6 illustrates the order in which each of these operations is performed and indicates the equations used. The original filter description must exist in Matlab's second order system matrix format under the variable name sos. For details on the Second Order Section matrix format see the Matlab Signal Processing Toolbox [8].

The m-file snorm.m calculates the \( L_1, L_2 \) and \( L_\infty \) norm approximations, which were defined in (4.9a, b, and c) respectively, and returns the norms of the state for each section. The input scaling gain \( B_0^* \) is defined as in (4.20); this value is divided by 2 and stored as \( B_0s \). For each section the scaled numerator coefficients are calculated from (4.21). Each scaled set of second order section coefficients is divided by 2 and stored to the variable sos2s. If the numerator coefficients of the last section (Lth) are greater than unity, then they are fractionally scaled by \( 2^{-nB} \) and the integer nB is stored in the workspace.

The Matlab m-file df2csim.m is the users interface to the filter simulation. The Cascaded Direct Form II coefficients B0s and sos2s are quantized to \( w \)-bits and stored as
**B0sq** and **sos2sq**. The user may then save these coefficients to a text file which can be included into a Motorola DSP56002 assembly program. This routine also allows the user to run the non-linear and linear stochastic simulations discussed in Sections 4.2.5 and 4.2.6 below.

![Flowchart](image)

**Figure 4.6: Flowchart for sos2df2c.m and snorm.m.**

### 4.2.3 DSP56002 Implementation

Difference equations (4.16a, b, c and d) are implemented on the Motorola DSP56002 by the assembly code in **iird2cb.asm** DSP56002 Code. The data structures for this code are pictured in Figure 4.7. The coefficients are stored in a circular buffer in Y-memory...
and pointed to by R4. The state variables $s_i^*(n-1)$ and $s_i^*(n-2)$ are stored as a circular buffer in X-memory indexed by R1 and R2 respectively.

![Diagram](image)

**Figure 4.7: Data Structures for iird2cb.asm, Cascaded 2nd Order Direct Form II Structure DSP56002 implementation.**

This assembly program combines an initialization subroutine and the filter subroutine. The filter subroutine enables the 2x scaling mode, and then calculates the difference equations (4.16a, b, c and d) for each section. The input values are passed to the filter routine through register X0 and the filtered output data values is passed to the main
program through register X0. This routine requires a total of $5^*(N/2)+7$ instruction cycles, where $N$ represents the filter order.

4.2.4 Non-Linear Simulation Model

The Matlab file d2c_nl.m implements the non-linear simulation of (4.16a, b, c and d) using the convergent rounding and saturation emulation routine developed in Section 2.2. This routine will return the non-linear filtered output signal $y_n$, and filter states $S_n$ given the input vector $x_i$, fractionally scaled and quantized coefficients in sos2sq, scaling gain $B0sq$, constant $nB$, and desired wordlength $w$.

4.2.5 Linear-Stochastic Simulation Model

The Cascaded Second Order Direct Form II structure linear stochastic model is shown in Figure 4.8. The transfer functions $G_i^\ast(z)$ represent roundoff noise transfer functions from each noise source $\epsilon_i(n)$ to the filter output $y(n)$.

\[
G_i^\ast(z) = \prod_{k=1}^{L} H_i^\ast(z), \quad i = 1, \ldots, L
\]  
(4.22a)

\[
G_{L+1}^\ast(z) = 1
\]  
(4.22b)

The following linear state-space representation for the Cascaded Second Order Direct Form II structure is defined for the linear stochastic simulation.
Input Equation: \[ u_i(n) = \frac{B^*_0}{2} \cdot x(n) \] (4.23a)

Section Equation:

\[
\begin{bmatrix}
    s^*_1(n+1) \\
    s^*_2(n+1)
\end{bmatrix} =
\begin{bmatrix}
    0 & 1 \\
    -a_{ii} & -a_{ii}
\end{bmatrix}
\begin{bmatrix}
    s^*_1(n) \\
    s^*_2(n)
\end{bmatrix} +
\begin{bmatrix}
    0 & 0 \\
    1 & 1
\end{bmatrix}
\begin{bmatrix}
    2u_i(n) \\
    e_i(n)
\end{bmatrix}
\] (4.23b)

\[
\gamma_i(n) = \left( b^*_2 - a_{ii} \cdot b^*_o \right) \left( b^*_u - a_{ii} \cdot b^*_o \right) \begin{bmatrix}
    s^*_1(n) \\
    s^*_2(n)
\end{bmatrix} + \begin{bmatrix}
    b^*_o \\
    b^*_o
\end{bmatrix} \begin{bmatrix}
    2u_i(n) \\
    e_i(n)
\end{bmatrix}
\] (4.23c)

Output Equation: \[ y(n) = y_L + e_{L+1}(n) \] (4.23d)

Note that coefficients \( a_i, b_i^*, \) and \( B_0^* \) are quantized to \( w-1 \) bits due the fractional scaling by 2.

The Matlab m-files sosd2css.m and lsimd2c.m were written to form and simulate the state-space model of (4.23a, b, c and d). Routine lsimd2c.m returns the results from the linear-stochastic model in the form of the output signal \( y_L \), and the filter states \( S_L \) from a given input vector \( x_i \), fractionally scaled coefficients in the sos2sq matrix, scaling gain \( B0sq \), constant \( nB \), and desired wordlength \( w \).
Figure 4.8: Linear Stochastic Model for Cascaded 2nd Order Direct Form II filter structure.
4.3 Parallel Second Order Direct Form II Structure

The Parallel 2nd Order Direct Form II Structure is formed by decomposing (4.1) into a summation of 2nd Order Direct II sections and a constant gain as follows.

\[ H(z) = K_p + \sum_{i=1}^{L} H_i(z) \]  

(4.24a)

\[ H_i(z) = \frac{b_{2i} + b_{1i}z^{-1}}{1 + a_{1i}z^{-1} + a_{2i}z^{-2}} \]  

(4.24b)

where \( L \) represents the number of 2nd order filter sections.

This structure is formed by first expanding (4.1) in a partial fraction expansion, and then each second order sub-filter (4.24b) is formed by combining the complex conjugate poles and their corresponding residues. As discussed in Section 4.2.1, the pairing of complex conjugate poles into second order sub-filters reduces a structure's sensitivity to coefficient quantization. Oppenheim and Schafer also indicate that the decomposition of this structure's zeros into first order numerator polynomials causes the zero locations to be more sensitive to coefficient quantization [6, pp. 305-306].

The implementation of this structure is similar to that of the Cascaded Second Order Direct Form II structure discussed in Section 4.2.1. Each filter coefficient must be fractionally scaled by 1/2 for fractional representation in a DSP. Overflow scaling is also required to reduce the probability of overflow at the output of each summation node. The
following difference equations (4.25a, b, c and d) describe the actual implementation of (4.24a, b) including the rounding $R$ and saturation $S$ operations.

\[
u_i(n) = \frac{B_{oi}^*}{2} x(n) \tag{4.25a}\]

\[
s_i(n) = S \left\lceil 2 \left\{ u_i(n) - \frac{a_{ui}}{2} s_i^*(n-1) - \frac{a_{2ui}}{2} s_i^*(n-2) \right\} \right\rceil \tag{4.25b}\]

\[
y_i(n) = \frac{b_{oi}^*}{2} s_i^*(n) + \frac{b_{iui}^*}{2} s_i^*(n-1) \tag{4.25c}\]

\[
y(n) = S \left\lceil 2 \left\{ \frac{K_p}{2} + \sum_{i=1}^{L} y_i(n) \right\} \right\rceil \tag{4.25d}\]

The Parallel 2nd Order Direct Form II structure is shown in Figure 4.9, a structure that can be implemented on any fixed-point DSP using fractional numeric representation.

### 4.3.1 Overflow Scaling

In the parallel implementation, each of the transfer functions $F_i(z)$, from input $x(n)$ to the summation output $s_i(n)$, is independent of any other. Therefore, a unique scaling gain $B_{oi}$ must be calculated from each state response transfer function $F_i(z)$. The scaling procedure for each sub-filter is the same as that used for the Direct Form High Order structure, as given in Section 4.1.2.
Figure 4.9: Parallel 2nd Order Direct Form II Structure Signal Flow Graph.
4.3.2 Structure Conversion

The Matlab m-file sos2df2p.m calculates the parallel section coefficients and performs overflow scaling. The flowchart in Figure 4.10 illustrates the order in which each of these procedures is performed. The original filter description must exist in Matlab's second order system matrix format under the variable name sos. For details on the Second Order Section matrix format see the Matlab Signal Processing Toolbox [8].

Figure 4.10: Flowchart for sos2df2p.m and snorm.m.
Each 2nd order Direct Form II sub-filter is combined in a matrix named sop. This matrix has the same format as the sos described above but must be interpreted as a set of parallel sections. The vector of scaling gains $B_0^*$ is saved to the workspace as the vector $B0s$. The numerator coefficients of each section are scaled for overflow, then all coefficients are fractionally scaled by 1/2 and saved to the matrix sop2s.

The Matlab m-file df2psim.m is the simulator user interface for the Parallel 2nd Order Section Direct Form II Structure. This routine quantizes the coefficients in sop2s and B0s to w-bits and stores them as sop2sq and B0sq respectively. The user may also generate a text file with all filter coefficients in a format which can be directly included into a Motorola DSP56002 assembly file for filter testing. This routine also allows the users to run the non-linear and linear stochastic simulations discussed in Sections 4.3.5 and 4.3.6 below.

4.3.3 DSP56002 Implementation

The Motorola DSP56002 assembly program for this structure is iird2pb.asm. The output of each sub-filter is calculated from difference equations (4.25a,b and c) and summed into the B accumulator together with the input scaled by the gain term $K_p$ according to (4.25d). The data structures for this code are pictured in Figure 4.11.

This assembly program is a combination of an initialization subroutine and a filter subroutine. The filter subroutine enables the 2x scaling mode and then calculates the difference equations (4.25b, c, and d) for each section and output equation (4.25a). After
processing the filter output is passed to the main program through register X0. The filter output execution of the filter the main program in register X0. This routine requires a total of 7*(N/2)+7 instruction cycles, where N represents the filter order.

![Diagram](image)

**Figure 4.11: Data Structures for iird2pb.asm, Parallel Direct Form II Sections DSP56002 implementation.**

### 4.3.4 Non-Linear Simulation Model

The Matlab file `d2par_nl.m` implements the non-linear simulation of (4.25a, b, c and d) using the convergent rounding and saturation emulation routine developed in Section 2.2. This routine will return the non-linear filtered output signal yn, and filter states Sn given
the input vector $\mathbf{x}_i$, fractionally scaled and quantized coefficients in matrix $\mathbf{sos2pq}$ matrix, parallel gain $K_{pq}$, scaling gain vector $\mathbf{B0sq}$, and desired wordlength $\mathbf{w}$.

### 4.3.5 Linear-Stochastic Simulation Model

The linear stochastic model for the Parallel Second Order Direct Form II structure is shown in Figure 4.12. The transfer functions $G_i^\star(z)$ represent the roundoff noise transfer function from the noise sources $e_i(n)$ to the filter output $y(n)$.

\[
G_i^\star(z) = H_i^\star(z), \quad i = 1, \ldots, L
\]  
\[
G_{L+1}^\star(z) = 1
\]

The following linear state-space representation for a second order Direct Form II structure is defined for the linear stochastic simulation.

**Input Equation:**

\[
u_i(n) = \frac{1}{2} B_{oi}^\star \cdot x(n) \cdot \]  
(4.27a)

**Section State Equation:**

\[
\begin{bmatrix}
    s_i^{\star}(n+1) \\
    s_{2i}^{\star}(n+1)
\end{bmatrix} =
\begin{bmatrix}
    0 & 1 \\
    -a_{2i} & -a_{ii}
\end{bmatrix}
\begin{bmatrix}
    s_i^{\star}(n) \\
    s_{2i}^{\star}(n)
\end{bmatrix} +
\begin{bmatrix}
    0 & 0 \\
    1 & 1
\end{bmatrix}
\begin{bmatrix}
    2u_i(n) \\
    e_i(n)
\end{bmatrix}
\]  
(4.27b)

**Section Output Equation:**

\[
y_i(n) = \left[ b_{2i}^\star - a_{2i} \cdot b_{oi}^\star \right] \left[ s_i^{\star}(n) \right] + \left[ b_{oi}^\star \right] \left[ 2u_i(n) \right] \]  
(4.27c)

Implementation of Direct Form Filter Structures 60
Figure 4.12: Linear stochastic model for parallel 2nd order Direct Form II filter structure.
Parallel Output Equation:

\[ y(n) = \left( K_p + \sum_{i=1}^{L} y_i(n) \right) + e_{L+1}(n) \]  \hspace{1cm} (4.27d)

All coefficients are quantized to \( w-1 \) bits due to the division by 2.

The Matlab m-files sosd2pss.m and lsimd2p.m were written to form and simulate the state-space model of equations (4.27a, b, c and d). Routine lsimd2p.m returns the linear-stochastic approximation of the filtered output signal \( y_l \), and filter states \( S_l \) given the input vector \( x_i \), fractionally scaled and quantized coefficients in matrix sos2pq matrix, parallel gain \( Kpq \), scaling gain vector \( B0sq \) and \( nB \).
4.4 Cascaded Second Order Direct Form I Structure

The Cascaded 2nd Order Direct Form I Structure is obtained from the Direct Form II signal flow graph, Figure 4.5, by transposing the direction of signal flow branches and by transposing summation nodes and branch nodes. This cascaded transpose structure has the same coefficients and transfer function as the Cascaded Direct Form II structure (4.13a, b). The transposed network leads to an alternate set of difference equations (4.27a, b, and c) for the cascade of second order sections.

\[
y_i(n) = \left[ \left[ 2 \left( \frac{b_{2i}^*}{2} x_i(n) + \frac{1}{2} \cdot s_{ii}^*(n-1) \right) \right] \right]_w
\]

(4.27a)

\[
s_{ii}^*(n) = \left[ \left[ 2 \left( \frac{b_{2i}^*}{2} x_i(n) + \frac{1}{2} \cdot s_{ii}^*(n-1) - \frac{a_{ii}}{2} y_i(n) \right) \right] \right]_w
\]

(4.27b)

\[
s_{ii}^2(n) = \left[ \left[ 2 \left( \frac{b_{2i}^*}{2} x_i(n) - \frac{a_{ii}}{2} y_i(n) \right) \right] \right]_w
\]

(4.27c)

\[
y(n) = \left[ \left[ 2 \cdot \frac{B_0^*}{2} \cdot y_L(n) \right] \right]_w
\]

(4.27d)

where

- \( B_0^* \) = Input Scaling Gain
- \( a_i \) = Denominator Coefficients
- \( b_i^* \) = Scaled Numerator Coefficients
- \( L \) = Number of 2nd Order Sections
- \( w \) = Wordlength
Equations (4.27a, b, c) show that each Direct Form I filter section contains three summation nodes compared to the Direct Form II structure which contains only a single summation node per section. These two filters also differ in their pole and zero ordering, which will contribute to roundoff noise performance differences. Like the Direct Form II Cascaded structure in Section 4.2.1, the Direct Form I Cascaded structure possesses the advantage of decoupling the poles of the filter to reduce coefficient quantization sensitivity. The guidelines for pole-zero pairing and section ordering in Section 4.2.1 also apply to the Direct Form I Cascaded Structure. A major difference between the Direct Form I and Direct Form II sub-filter structures is the number of summation nodes in the difference equation.

### 4.4.1 Overflow Scaling

The Cascaded 2nd Order Direct Form I is scaled according to the methods described by L.B. Jackson [10]. Referring to Figure 4.14, the transfer functions from the input $x(n)$, to the summation node outputs $s_{1i}(n)$ and $s_{2j}(n)$, are defined as $F_{i}^*(z)$, and given in (4.28) below, where the asterisk indicates that scaling is included in the transfer function.

$$F_{i}^*(z) = \prod_{k=1}^{t} H_{k}^*(z)$$  \hspace{1cm} (4.28)
Figure 4.14: Cascaded 2nd Order Direct Form I Structure Signal Flow Graph.
Notice that \( F_i^*(z) \) is equivalent to the transfer function from the input to the output of the \( i \)th sub-filter. As with the Cascaded Direct Form II structure, scaling is performed such that the norm of each \( F_i^* \) is bounded by unity to prevent internal overflows in each second order section. For each cascaded section the following transfer function must be scaled to prevent overflow.

\[
F_i^*(z) = \prod_{k=1}^{i} H_k(z)
\]  

(4.29)

The overflow scaling gains are defined as follows.

\[
K_i = \frac{1}{\|F_i^*\|_p}
\]  

(4.30)

The \( p \)-norm of \( F_i \) is approximated using the method outlined in Section 4.1.2 using (4.9a), (4.9b) or (4.9c). The scaled filter coefficients are computed as follows.

\[
b_{ki}^* = \frac{K_i}{K_{i-1}} b_{kj}, \quad i = 1, \ldots, L \text{ and } K_0 = 1
\]  

(4.31)

\[
B_0^* = \frac{b_0}{K_L}
\]  

(4.32)

The Motorola DSP56002 employs an automatic scaling mode for the implementation of second order filter sections. This mode is used to compensate for the coefficient scaling of 1/2 by automatically scaling the accumulator contents by a factor of 2x when
transfers to memory or general purpose registers are performed. The automatic 2x scaling operation is performed prior to rounding and saturation and occurs with no additional processing overhead.

4.4.2 Structure Conversion

The Matlab m-file sos2df1c.m performs the operations of overflow scaling, fractional scaling and coefficient quantization, and also generates a Motorola DSP56002 data file. The flowchart in Figure 4.15 illustrates the order in which each of these procedures is performed and indicates the equations used. The original filter description must exist in Matlab's second order system matrix format under the variable name sos.

The m-file snorm.m calculates the $L_1$, $L_2$ and $L_\infty$ norm approximations defined in (4.9a), (4.9b) and (4.9c) respectively. After scaling and quantization, a text file is generated with all coefficients and necessary parameters in a format which can be directly included into a Motorola DSP56002 assembly file for filter testing.

4.4.3 DSP56002 Implementation

Difference equations (4.27a, b, c and d) are implemented on the Motorola DSP56002 by the assembly code in iird1c1b.asm. The data structures for this code are pictured in Figure 4.16. The coefficients are stored in a circular buffer in Y-memory and pointed to by R4. The state variables $s_1(n-1)$ and $s_1(n-2)$ are stored in a circular buffer in X-memory pointed to by R1 and R2 respectively.

Implementation of Direct Form Filter Structures 67
Figure 4.15: Flowchart for sos2df1c.m and snorm.m.

This assembly program is a combination of an initialization subroutine and the filter subroutine. The filter subroutine enables the 2x scaling mode, and then calculates the difference equations (4.27a, b, and c) for each section and the filter output (4.27d). The filter input is passed to the filter routine in register X0 and the filter output is passed to the main program in register X0. This routine requires a total of 4*N+10 instruction cycles, where N represents the filter order.
4.4.4 Non-Linear Simulation Model

The Matlab file d1cas_nl.m implements the non-linear simulation of (4.27a, b, c, and d) using the convergent rounding and saturation emulation routine developed in Section 2.2. This routine will return the non-linear filtered output signal $y_n$, and filter states $S_n$.
given the input vector $x_i$, scaled and quantized coefficients in an sos1sq matrix, scaling
gain B0sq and nB, and desired wordlength w.

4.4.5 Linear-Stochastic Simulation Model

The cascade of second order Direct Form II structure linear stochastic model is shown
in Figure 4.17. The transfer functions $G^*_i(z)$ represent the roundoff noise transfer
functions from the noise sources $e_i(n)$ to the filter output $y(n)$.

$$G^*_i(z) = \frac{b^*_0}{1 + a_{ui}z^{-1} + a_{zi}z^{-2}} \prod_{k=i+1}^{L} H^*_k(z), \quad i = 1, \ldots, L \tag{4.33}$$

The linear stochastic model is described by the following state equations with roundoff
noise sources.

Input Equation:

$$x(n) = u_i(n) \cdot \tag{4.34a}$$

Section Equation:

$$\begin{bmatrix} s_{ui}(n+1) \\ s_{zi}(n+1) \end{bmatrix} = \begin{bmatrix} -a_{ui} & 1 \\ -a_{zi} & 0 \end{bmatrix} \begin{bmatrix} s_{ui}^*(n) \\ s_{zi}^*(n) \end{bmatrix} + \begin{bmatrix} b^*_y - a_{ui}b^*_w \\ b^*_y - a_{zi}b^*_w \end{bmatrix} \begin{bmatrix} x_i(n) \\ 3e_i(n) \end{bmatrix} \tag{4.34b}$$

$$y_i(n) = \begin{bmatrix} 0 & 1 \end{bmatrix} \begin{bmatrix} s_{ui}^*(n) \\ s_{zi}^*(n) \end{bmatrix} + \begin{bmatrix} b^*_w \\ 3e_i(n) \end{bmatrix} \tag{4.34c}$$
Figure 4.17: Linear Stochastic Model for Cascade of 2nd Order Direct Form I filter structure.
Output Equation:

\[ y(n) = 2^{nb} \frac{B_o}{2^{nb}} \cdot y_L + e_{L,q}(n) \]  (4.34d)

The Matlab m-files `sosdlcss.m` and `lsimd1c.m` were written to form and simulate the state-space model of equations (4.34a, b, c, and d). Routine `lsimd1c.m` returns the linear-stochastic approximation of the filtered output signal \( y_l \), and filter states \( S_l \) given the input vector \( x_i \), scaled and quantized coefficients in an `sos1sq` matrix, scaling gain `B0sq` and `nB`, and desired wordlength \( w \).
4.5 Direct Form FIR

The Direct Form Finite Impulse Response (FIR) filter is described by the following transfer function.

\[ H(z) = b_0 \frac{(z - z_0)(z - z_1) \cdots (z - z_N)}{z^N} = b_0 + b_1 z^{-1} + b_2 z^{-2} + \cdots + b_N z^{-N} \]  \hspace{1cm} (4.35)

The output of the Direct Form FIR structure in (4.35) is expressed as a convolution of a finite set of impulse response coefficients \( b_i \) and a set of past input data values. Fractional scaling of the coefficients \( b_i \) by a power of 2 is necessary for representation in the DSP's \( w \)-bit fixed-point registers. The DSP56002 implementation of (4.35) is shown in Figure 4.18; it consists of a summation of weighted delay line outputs (taps), and a power of 2 output scaling gain to restore the overall filter gain.

![Figure 4.18: Direct Form FIR Signal Flow Graph.](image)

The difference equation for this structure is shown below.

\[ y(n) = S \left[ R \left[ 2^{nb} \left\{ \frac{b_0}{2^{nb}} s(n) + \frac{b_1}{2^{nb}} s(n-1) + \frac{b_2}{2^{nb}} s(n-2) + \cdots + \frac{b_N}{2^{nb}} s(n-N) \right\} \right] \right] \]  \hspace{1cm} (4.36)

where, \( b_i = \) Numerator Coefficients
\( 2^{nb} = \text{Numerator Coefficient Magnitude Scale Factor} \)

\( W = \text{Wordlength} \).

### 4.5.1 DSP56002 Implementation

The Motorola DSP56002 assembly code in file `dfirb.asm` implements the difference equation in (4.36). The data structures for this code are pictured in Figure 4.19. The set of coefficients is stored in a circular buffer located in Y data memory and indexed by the data address pointer register R4. The filter states \( s(n) \) to \( s(n-N) \) are stored in a circular buffer in X-data memory and indexed by the data address pointer register R1. This routine requires \( N+5 \) instruction cycles, where \( N \) represents the filter order.

![Diagram](image.png)

**Figure 4.19:** Data Structures for `dfirb.asm`, Direct Form FIR DSP56002 implementation.

### 4.5.2 Non-Linear and Linear Simulation Models

Both linear and non-linear models have been incorporated into the Matlab m-file `dfirsim.m`. The non-linear simulation implements the difference equation (4.36) using the convergent rounding and saturation model `xquant.m` from Section 2.4. As shown in
Figure 4.20, the linear stochastic model uses the noise source of Section 2.5 to simulate roundoff error.

Figure 4.20: Direct Form FIR Signal Flow Graph with Linear Stochastic Roundoff Noise Input.
5.0 Implementation of Second Order Section Optimal Structure

It is known that a linear system may be represented by a linear state-space description. For this description, Mullis and Roberts [15] derived a set of conditions for which this structure is optimal in terms of output roundoff noise variance. One drawback of this structure is the large number of multiplications required. For example an Nth order filter requires \((N+1)^2\) multiplications. However, they also showed that a nearly optimal structure requiring only \(4N+1\) multiplications can be realized by decomposing the filter into a cascaded set of second order optimal state-space sections. The synthesis, implementation and modeling of this section optimal structure is presented below.

5.1 Description of Structure and Overflow Scaling

The section optimal structure is formed by first factoring the filter transfer function \(H(z)\) into second order sections \(H_i(z)\) as expressed below.

\[
H(z) = \prod_{i=1}^{L} H_i(z), \quad i = 1, 2, \ldots, L
\]  

(5.1)

Each second order filter section should be formed by pairing complex pole pairs and their nearest complex zero pairs as discussed in Section 4.2.1. Each second order section is defined to have the following form.

\[
H_i(z) = d + \frac{\gamma_1 z^{-1} + \gamma_2 z^{-2}}{1 + \alpha_1 z^{-1} + \alpha_2 z^{-2}}
\]  

(5.2)
A set of synthesis equations is developed to transform (5.2) to the following second order state-space description.

\[
\begin{bmatrix}
  s_1(n+1) \\
  s_2(n+1)
\end{bmatrix} =
\begin{bmatrix}
  a_{11} & a_{12} \\
  a_{21} & a_{22}
\end{bmatrix}
\begin{bmatrix}
  s_1(n) \\
  s_2(n)
\end{bmatrix} +
\begin{bmatrix}
  b_1 \\
  b_2
\end{bmatrix} x(n)
\] (5.3a)

\[y(n) = [c_1 \ c_2] \begin{bmatrix}
  s_1(n) \\
  s_2(n)
\end{bmatrix} + dx(n)\] (5.3b)

Jackson et. al. [16] also derived a set of conditions necessary for a second-order state-space digital filter with L2 scaling to be optimal in terms of output roundoff noise. The structure in (5.3) was shown to be optimal in regard to output roundoff noise if the following two conditions were satisfied.

1) \[a_{11} = a_{22}\]

2) \[c_1 / c_2 = b_2 / b_1\]

A simple synthesis procedure was also outlined for this structure.

\[a_{11} = a_{22} = -\alpha_1 / 2\]
\[a_{12} = (1 + \gamma_2)(K_1 \pm K_2) / \gamma_1^2\]
\[a_{21} = (K_1 \mp K_2) / (1 + \gamma_2)\]
\[b_1 = 0.5(1 + \gamma_2)\]
\[b_2 = 0.5\gamma_1\]
\[c_1 = \gamma_1 / (1 + \gamma_2)\]
\[c_2 = 1\]
\[K_1 = \gamma_2 - 0.5\alpha_1\gamma_1\]
\[K_2 = \sqrt{\gamma_2^2 - \gamma_1\gamma_2\alpha_1 + \gamma_1^2\alpha_2}\] (5.4)
Note that conditions 1 and 2 are satisfied in (5.4).

For each optimal state-space second order section \((A_i, B_i, C_i, d_i)\) defined according to (5.3-4), scaling is performed to ensure that both the section output and internal states do not overflow. The input to each section is scaled such that the section's output does not overflow. Referring to Figure 5.1, the input to section \(H_s\) is scaled by the gain \(k_i\) which is defined as the \(L_2\) norm of \(H_i(z)\).

\[
k_i = \frac{1}{\|H_i(z)\|_2}
\]

\hspace{1cm} (5.5)

This scaling gain can be incorporated into \(B_i\) and \(d_i\) of \(H_i(z)\).

\[
B_i' = k_i B_i \hspace{1cm} (5.6a)
\]

\[
d_i' = k_i d_i \hspace{1cm} (5.6b)
\]

The overall gain of the filter is restored by the output gain \(B_0\), defined as follows.

\[
B_0 = \prod_{i=1}^{L} \frac{1}{k_i}
\]

\hspace{1cm} (5.7)
Overflow scaling is now performed for each section's internal state vector by a linear transformation of the state vector $s_i(n)$ using the scaling matrix $T_i$ in (5.8) below.

\[
T_i = \frac{1}{\delta} \begin{bmatrix}
\|F_{\iota i}\|^{-1}
& 0 \\
0
& \|F_{2i}\|^{-1}
\end{bmatrix}
\]  

(5.8)

with

\[
F_i = \begin{bmatrix}
F_{\iota i} \\
F_{2i}
\end{bmatrix} = (zI - A_i)^{-1} B_i'
\]  

(5.6)

where $F_i$ is defined as the response from the input $x(n)$ to the state $s_i$ of the $i$th section.

The scaling factor $\delta$ in (5.8) weights the $L_p$-norm allowing the designer to adjust the probability of overflow. It should be noted that the state-space sections are only optimal in terms of output roundoff noise if the $L_2$ scaling norm is used. Other norms may be used but the resulting structure is not necessarily optimal in the sense of minimum output roundoff noise. For the $L_\infty$-norm, the structure would be optimal in terms of minimizing the peak output roundoff noise power spectral density [10, p. 334].

For the $i$th section, the scaled set of second order optimal state-space equations are given below.

\[
A_i^* = T_i A_i T_i^{-1}
\]  

(5.10a)

\[
B_i^* = T_i B_i'
\]  

(5.10b)
\[ C_i^* = C_i T_i^{-1} \]  \hspace{1cm} (5.10c)

\[ d_i^* = d_i' \]  \hspace{1cm} (5.10d)

The final step is to fractionally scale each set of coefficients to less than unity. It is difficult to predict the coefficient's full magnitude range due the scaling variable \(\delta\). So far it has been sufficient to scale each coefficient by 2 and use the DSP56002's automatic scaling mode, with the exception of output gain \(B_0\). To realize filter gains greater than unity \(B_0\) is divided by an power of 2 and the output is rescaled by an left shifts.

The synthesis and scaling operations for the section optimal structure are performed by the m-file \texttt{sos2ssvec.m}. This m-file takes a second order system matrix and creates the set of scaled section matrices \((A_i^*, B_i^*, C_i^*, d_i^*)\) and stores them along the diagonal of the matrices \(A_0, B_0, C_0,\) and \(D_0\) in the Matlab work space. The m-file \texttt{secopsim.m} was written to simulate this structure using the non-linear and linear stochastic models described below.

### 5.2 DSP56002 Implementation and Non-Linear Model

The Section Optimal State-Space structure is illustrated in Figure 5.2, and its implementation is described by the following non-linear difference equations.

\[ y_i(n) = \sum R \left[ 2 \left\{ x_i(n) \frac{d_i^*}{2} + s_{ii}^* (n-1) \frac{c_{ii}^*}{2} + s_{ii}^* (n-1) \frac{c_{ii}^*}{2} \right\} \right] \]  \hspace{1cm} (5.11a)
\[ s_{1i}^* (n) = S_R \left[ 2 \left\{ s_{1j}^* (n-1) \frac{a_{11j}^*}{2} + s_{2i}^* (n-1) \frac{a_{12j}^*}{2} + x_i(n) \frac{b_{2i}^*}{2} \right\} \right] \tag{5.11b} \]

\[ s_{2i}^* (n) = S_R \left[ 2 \left\{ s_{2j}^* (n-1) \frac{a_{22j}^*}{2} + s_{1i}^* (n-1) \frac{a_{21j}^*}{2} + x_i(n) \frac{b_{1i}^*}{2} \right\} \right] \tag{5.11c} \]

The DSP56002 assembly file `secopt.asm` was written to implement the difference equations (5.11a, b, and c). The data structure for the DSP56002 implementation is shown in Figure 5.3.

![Diagram](image)

Figure 5.2: Second order state-space optimal structure flow graph.

Implementation of 2nd Order Section Optimal Structure
The m-file `sec_n1.m` uses the non-linear fixed-point model from Section 2.4 to simulate the difference equations in (5.11). This structure requires \((11/2)N+6\) instruction cycles, where \(N\) equals the filter order.

```
Delay Line
X:(R2)

\[
\begin{array}{c}
s_{11}(n-1) \\
\vdots \\
s_{1L}(n-1) \\
s_{21}(n-1) \end{array}
\]

Coefficient Buffer
Y:(R4)

\[
\begin{array}{c}
d_{/2} \\
c_{1,1}/2 \\
c_{2,1}/2 \\
b_{1,1}/2 \\
a_{1,1}/2 \\
a_{1,2}/2 \\
b_{2,1}/2 \\
a_{2,2}/2 \\
a_{2,1}/2 \\
\vdots \\
B_0 \end{array}
\]
```

Figure 5.3: Data Structures for `secopt.asm`, section optimal DSP56002 implementation.

### 5.3 Linear-Stochastic Simulation Model

The linear-stochastic model is formed by replacing the non-linear operators by stochastic roundoff noise models as shown in Figure 5.4. The linear state-space equations for the model in Figure 5.4 are as follows.
\[
\begin{bmatrix}
  s_{1,j}^*(n+1) \\
  s_{2,j}^*(n+1)
\end{bmatrix} =
\begin{bmatrix}
  a_{11,j}^* & a_{12,j}^* \\
  a_{21,j}^* & a_{22,j}^*
\end{bmatrix}
\begin{bmatrix}
  s_{1,j}^*(n) \\
  s_{2,j}^*(n)
\end{bmatrix} +
\begin{bmatrix}
  b_{1,j}^* \\
  b_{2,j}^*
\end{bmatrix}
\begin{bmatrix}
  1 & 0 & 0 \\
  0 & 1 & 0
\end{bmatrix}
\begin{bmatrix}
  x_{i}(n) \\
  e_{1,i}(n) \\
  e_{2,i}(n) \\
  e_{3,i}(n)
\end{bmatrix}
\] (5.12a)

\[
y_{i}(n) =
\begin{bmatrix}
  c_{1,j}^* & c_{2,j}^*
\end{bmatrix}
\begin{bmatrix}
  s_{1,j}^*(n) \\
  s_{2,j}^*(n)
\end{bmatrix} +
\begin{bmatrix}
  d_{i}^* \\
  0 & 0 & 1
\end{bmatrix}
\begin{bmatrix}
  x_{i}(n) \\
  e_{1,i}(n) \\
  e_{2,i}(n) \\
  e_{3,i}(n)
\end{bmatrix}
\] (5.12b)

\[
y(n) = B_0 \cdot y_L(n) + e_L(n)
\] (5.12c)

This linear model is implemented in the m-file Isimsec.m.

---

**Figure 5.4**: Optimal state-space section linear stochastic model signal flow graph.
6.0 Implementation of the MA Lattice Structure

The lattice structures are used extensively in the implementation of linear prediction filters, speech processing models, and adaptive filters [17]. This chapter discusses the fixed-point scaling and implementation of the Moving Average lattice structure.

The Moving Average (MA) lattice structure implements the following FIR filter transfer function.

\[
A(z) = \frac{Y(z)}{X(z)} = 1 + \sum_{i=1}^{M} a_i z^{-i}
\]

(6.1)

The lattice structure realizes (6.1) using a cascade of first order sections as shown in Figures 6.1a and b for a fixed-point implementation. The signals marked with an asterisk denote that they have been scaled to prevent overflow.

The names given to the coefficients and signal nodes of this structure come from their linear prediction interpretation. The coefficients \( k_i \) are called reflection coefficients, the node signals along the top path \( e^f \) are the forward prediction errors, and the node signals along the bottom path \( e^b \) are the backward prediction errors [10, pp. 281-287].

The reflection coefficients \( k_i \) are obtained from the \( a_i \) coefficients of (6.1) using the step-down algorithm [10 p.263]. This recursive algorithm is defined as follows.

\[
k_i = a_i^{(i)}
\]

(6.2a)

\[
a_m^{(i-1)} = \frac{a_m^{(i)} - k_i a_{i-m}^{(i)}}{1 - k_i^2}
\]

(6.2b)
where \( m = 1, 2, \ldots, i \) and \( i = 1, 2, \ldots, N \).

Figure 6.1: MA lattice fixed-point implementation. (a) General lattice filter; (b) first order MA lattice section.

Ignoring scaling, the transfer functions from \( x(n) \) to the forward prediction error nodes \( e^f_i \) and from \( x(n) \) to the backward prediction error nodes \( e^b_i \) are given as follows [10, p.283].

\[
A^f_i(z) = \frac{E^f_i(z)}{X(z)} = 1 + \sum_{j=1}^{i} a^f_j z^{-j} \tag{6.3a}
\]
\[ A_i^b(z) = \frac{E_i^b(z)}{X(z)} = z^{-i} A_i^f(z^{-1}) \]  

(6.3b)

where \( N \) is the filter order. Both forward and backward prediction error nodes must be scaled to prevent overflow.

For the fixed-point MA lattice implementation both fractional coefficient scaling and overflow scaling must be performed. The reflection coefficients of a minimum phase system are guaranteed to be less than unity, therefore fractional scaling is not necessary. However, this structure can also implement non-minimum phase filters for which some of the reflection coefficients \( k_i \) exceed unity and must be fractionally scaled by \( 1/2^{n_k(i)} \). As seen in Figure 6.1b, fractional scaling requires that the product \( k_i/2^{n_k(i)} \) and the prediction error signal be left-shifted by \( n_k(i) \) bits to compensate for this scaling.

Overflow scaling is also performed at the output of each section by the factor \( 2^{n_k(i)} \). This scale factor is computed such that the response from \( x(n) \) to the output \( e_i^f \) and \( e_i^b \), given by (6.3a) and (6.3b) respectively, is bounded with respect to some \( L_p \)-norm. Since (6.3a) and (6.3b) have the same magnitude response, the overflow scaling factor \( 2^{n_k(i)} \) is the same for outputs \( e_i^b(n) \) and \( e_i^f(n) \). Each \( n_s(i) \) is computed using the \( p \)-norm of (6.3a) as follows.

\[ n_s(i) = \text{ceil} \left\lfloor \log_2 \left( \frac{1}{\| A_i^f \|_p} \right) \right\rfloor - n_s(i-1), \quad i = 1, \ldots, N. \]  

(6.4)
where \(ns(0) = 0\). Equation (6.4) takes into consideration the scaling factor of the previous sections, so that the overall gain of the filter is maintained.

The difference equations for Figure 6.1 are as follows.

\[ e^e(n) - e^b(n) = x(n) \quad (6.5a) \]

\[ e^{*e}(n) = S \left[ R \left[ 2^{ns(i)} \left\{ e^{i-1}(n) + 2^{nk(i)} \cdot k \cdot e^{i-1}(n - 1) \right\} \right] \right] \quad (6.5b) \]

\[ e^{*b}(n) = S \left[ R \left[ 2^{ns(i)} \left\{ e^{i-1}(n - 1) + 2^{nk(i)} \cdot k \cdot e^{i-1}(n) \right\} \right] \right] \quad (6.5c) \]

\[ y(n) = e^{*e}(n) \quad (6.5d) \]

The m-file `malatsim.m` was written to compute and fractionally scale the reflection coefficients and compute the overflow scale factor \(ns(i)\). The reflection coefficients may be computed in Matlab using the m-file `poly2rc.m`; however, this algorithm does not converge if (6.1) contains zeros outside the unit circle. The m-file `malatsim.m` is also used to perform the non-linear and linear stochastic simulations.

### 6.1 DSP56002 MA Lattice Implementation

The difference equations in (6.5a, b, c, and d) have been implemented on the DSP56002 using the assembly code subroutine `malat.asm`. The data structures for this code are pictured in Figure 6.2. The reflection coefficients are stored in a circular buffer.
located in Y data memory and indexed by the data address pointer register R4. The backward prediction errors $e_i^{b^*}$ are stored to a circular buffer in X-data memory and indexed by the data address pointer register R1. The forward prediction errors $e_i^{f^*}$ are held in the B accumulator. The scaling gains $2^{n_k(0)}$ and $2^{n_k(0)}$ are implemented by left or right shifting the accumulator contents before rounding.

![Diagram](image)

Figure 6.2: Data Structures for `iird2b.asm`, High Order Direct Form II DSP56002 implementation.

### 6.2 Non-Linear Simulation Model

The Matlab m-file `malat_nl.m` was written to implement the difference equations of (6.5a, b, c, and d) using the convergent rounding and saturation model from Section 2.4.

The following input variables are required.

- $x_i$ = Input signal vector (quantized to $w$-bits).
- $k_{sq}$ = Vector containing the fractionally scaled and quantized reflection coefficients.
- $n_k$ = Fractional coefficients scaling integers.
- $n_s$ = Vector of overflow scaling integers.
- $w$ = Register wordlength, including sign bit.

The following variables are written to the workspace.

Implementation of Lattice Structures
\[ y_n = \text{non-linear filter output.} \]
\[ e_{fn} = \text{matrix of forward prediction errors, the ith row contains the time response for the error at the ith section.} \]
\[ e_{bn} = \text{matrix of backward prediction errors, same format as } e_{fn}. \]

### 6.3 Linear-Stochastic Model

The linear-stochastic model of the MA lattice section in Figure 6.1 is shown in Figure 6.3. As described in Section 2.3, each non-linear quantization and rounding operator \( \mathbf{R}[\cdot] \) has been replaced by a uniform uncorrelated noise source denoted by \( e_i(n) \). The following difference equations for this model were derived from (6.5a, b, c, and d).

\[
e^{f}_{0}(n) = e^{b}_{0}(n) = x(n) \tag{6.6a}
\]

\[
e^{f*}_{i}(n) = 2^{n_k(i)} \left\{ e^{f*}_{i-1}(n) + \left( 2^{-n_k(i)} \right) \cdot \frac{k_i}{2^{n_k(i)}} \cdot e^{b*}_{i-1}(n-1) \right\} + e_{2,i}(n) \tag{6.6b}
\]

\[
e^{b*}_{i}(n) = 2^{n_k(i)} \left\{ e^{b*}_{i-1}(n-1) + \left( 2^{-n_k(i)} \right) \cdot \frac{k_i}{2^{n_k(i)}} \cdot e^{f*}_{i-1}(n) \right\} + e_{2,i}(n) \tag{6.6c}
\]

\[
y(n) = e^{f}_{x}(n) \tag{6.6d}
\]

Figure 6.3: MA lattice section, Linear-Stochastic Model.
The Matlab m-file malsim.m was written for the linear-stochastic simulation using the difference equations in (6.6a, b, c, and d) and the linear-stochastic rounding model discussed in Section 2.5. The input and output variables are the same as those of the nonlinear simulation in Section 6.2.

The following input variables are required.

\[
\begin{align*}
\textbf{xi} & = \text{input signal vector (quantized to } w\text{-bits).} \\
\textbf{ksq} & = \text{vector containing the fractionally scaled and quantized reflection coefficients.} \\
\textbf{nk} & = \text{fractional coefficients scaling integers.} \\
\textbf{ns} & = \text{vector of overflow scaling integers.} \\
\textbf{w} & = \text{register wordlength, including sign bit.}
\end{align*}
\]

The following variables are written to the workspace.

\[
\begin{align*}
\textbf{yl} & = \text{linear filter output.} \\
\textbf{efl} & = \text{matrix of forward prediction errors, the } i\text{th row contain the time response for the error at the } i\text{th sections.} \\
\textbf{ebl} & = \text{matrix of backward prediction errors, the } i\text{th row contain the time response for the error at the } i\text{th sections.}
\end{align*}
\]
7.0 Filter Structure Performance Comparison

To validate the simulation models developed in Chapters 4, 5, and 6, the performance of each filter structure has been compared to its actual DSP56002 implementation. The performance measures used in this analysis are: the filter magnitude response error, roundoff noise power, the roundoff noise lower bound, and the number of operations required to implement the filter. The latter is simple to compute from the number of operations per filter order given in Chapters 4, 5, and 6. Computation of the other metrics is discussed below.

7.1 Magnitude Response Error and Roundoff Noise Power Calculations

The error between an ideal digital filter and its hardware implementation is a combination of the roundoff noise introduced by finite wordlength arithmetic and the perturbation of its frequency response due to coefficient quantization. These errors are measured independently. The effect of coefficient quantization is observed by computing the frequency responses for both the unquantized coefficients $H_u$ and the quantized coefficients $H_q$. The coefficient magnitude error $H_e$ is displayed as the difference between the magnitude responses of these two systems.

$$H_e(e^{j\omega}) = |H_u(e^{j\omega})| - |H_q(e^{j\omega})|$$

(7.1)

The following method of computing roundoff noise power is used for both simulated and actual DSP56002 implementations. Given the input sequence $x_i$ and using double
precision floating point arithmetic, the time response of the quantized system $y_i$ is computed. Then using either the simulation or DSP56002, the time response of this same quantized system is computed. The filter's time response error $e$ due to roundoff noise is computed as the difference between the "ideal response" $y_i$ and the simulation or DSP56002 response. From this error sequence the roundoff noise power is computed as follows.

$$ P_e = \sigma_e^2 + m_e^2 $$  \hspace{1cm} (7.2)

where $\sigma_e^2$ is the variance of $e$ and $m_e$ is the mean of $e$.

### 7.2 Computation of Roundoff Noise Lower Bounds

It has been shown by L.B. Jackson [19] that a filter structure's roundoff error is directly related to its coefficient quantization sensitivity. Jackson derived lower bounds for a structure's roundoff noise variance $\sigma_e^2$. These bounds are based on the unscaled filter sensitivity function $S(x)$, which is a measure of the sensitivity of $H(x)$ to quantization perturbations in the feedback coefficients. For $L_2$ scaling the lower bound on the roundoff noise is as follows.

$$ \sigma_e^2 \geq \sigma_o^2 \left[ k_{L,1} + \sum_{i=1}^{L} k_i \| S_i \|_1^2 \right] $$  \hspace{1cm} (7.3)

For $L_\infty$ scaling the lower bound on the roundoff noise is given below.
\[ \sigma_i^2 \geq \sigma_o^2 \left[ k_{L+1} + \sum_{i=1}^{L} k_i \| S_i \|_2 \right] \]  

(7.4)

Given a filter structure's state space description \((A, B, C, D)\), the filter sensitivity function \(S_i(z)\) is computed as follows:

\[ S_i(z) = F_i(z)G_i(z) \]  

(7.5a)

\[ F_i(z) = c_i(zI - A)^{-1}B \]  

(7.5b)

\[ G_i(z) = C(zI - A)^{-1}b_i \]  

(7.5c)

where \(c_i\) is a row vector and \(b_i\) is a column vector; each having a one in the \(i\)th position and zeros elsewhere.

In (7.3) and (7.4) the variance of a single rounding operation at a branch node is represented by \(\sigma_o^2\) and is defined as,

\[ \sigma_o^2 = \frac{2^{-(w-1)}}{12} \]  

(7.6)

where \(w\) is the total wordlength in bits. Also, in (7.3) and (7.4) the number of rounding operations at the \(i\)th branch node is represented by \(k_i\).

The following m-files were written to compute these lower bounds given the unscaled filter coefficients in the matrix \(\text{sos}\).

- `df2_lb.m` - High Order Direct Form II Structure.
- `df2c_lb.m` - Cascaded Direct Form II Structure.
- `df1c_lb.m` - Cascaded Direct Form I Structure.

Filter Structure Performance Comparison 93
7.3 12th Order Chebyshev II Bandpass Filter

In this section a 12th order Chebyshev II bandpass filter will be used to compare the finite wordlength performance of each IIR filter structure. This section also validates the non-linear and linear simulations by comparing their results to the DSP56002 implementation. The example filter is a cascade of six second order sections as follows.

\[ H(z) = \prod_{j=1}^{6} \frac{b_{0,j}(1 + b_{1,j}z^{-1} + z^{-2})}{1 + a_{1,j}z^{-1} + a_{2,j}z^{-2}} \]  

(7.7)

The parameters for each filter section are listed in Table 7.1 and the coefficients for (7.7) are listed in Table 7.2.

<table>
<thead>
<tr>
<th>Table 7.1: 12th Order Chebyshev II Bandpass Filter Parameters.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Sample Rate</strong> f_s</td>
</tr>
<tr>
<td>Lower Stopband Cutoff Frequency</td>
</tr>
<tr>
<td>Lower Passband Cutoff Frequency</td>
</tr>
<tr>
<td>Upper Passband Cutoff Frequency</td>
</tr>
<tr>
<td>Upper Stopband Cutoff Frequency</td>
</tr>
<tr>
<td>Passband Ripple</td>
</tr>
<tr>
<td>Stopband Ripple</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Table 7.2: 12th Order Chebyshev II Bandpass Filter Coefficients.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Section i</strong></td>
</tr>
<tr>
<td>----------------</td>
</tr>
<tr>
<td>1</td>
</tr>
<tr>
<td>2</td>
</tr>
<tr>
<td>3</td>
</tr>
<tr>
<td>4</td>
</tr>
<tr>
<td>5</td>
</tr>
<tr>
<td>6</td>
</tr>
</tbody>
</table>
The magnitude response is shown in Figure 7.1, the impulse response in Figure 7.2 and the pole-zero map in Figure 7.3.

![Frequency Response Chart]

**Figure 7.1**: Chebyshev 12th-order Bandpass filter magnitude response.

![Impulse Response Chart]

**Figure 7.2**: Chebyshev 12th-order Bandpass filter impulse response.
A comparison of the frequency response magnitude error for each structure is made in Figure 7.4 for a wordlength of 16-bits. This comparison shows the Parallel Direct Form having the least error at the corner frequencies and in the passband but the largest error in the stop bands. An interesting observation is that the Section Optimal structure exhibited the maximum error, at the lower passband.

The roundoff noise measurements for each filter structure simulation and DSP56002 implementation are now compared. For each filter structure and wordlength tested the input sequence was chosen as a 4096 point normally distributed pseudorandom sequence with variance 0.0989 and zero mean. Each structure has been scaled using the $L_2$ norm and a safety factor of 3 was used for this case.
Figure 7.4: Chebyshev 12th-order Bandpass filter magnitude response errors for a wordlength of 16-bits.

The roundoff noise measurements for the High Order Direct Form II structure are presented in Table 7.3. This structure is only stable for wordlengths greater than or equal to 24-bits. The roundoff noise power results presented in Tables 7.4 through 7.7 are for the Cascaded Direct Form II, Cascaded Direct Form I, Parallel Direct Form II, and Section Optimal structures respectively. The roundoff noise powers in Tables 7.3 through 7.7 are also plotted in Figures 7.5 through 7.9 for the respective structures.
Table 7.3: Roundoff noise power results for the High Order Direct Form II.

<table>
<thead>
<tr>
<th>Word Size</th>
<th>Lower Bound</th>
<th>Linear Simulation Avg = 20</th>
<th>Non-Linear Simulation</th>
<th>DSP56k</th>
</tr>
</thead>
<tbody>
<tr>
<td>32</td>
<td>-152.31</td>
<td>-137.770</td>
<td>-137.72</td>
<td>NA</td>
</tr>
<tr>
<td>28</td>
<td>-128.23</td>
<td>-113.78</td>
<td>-113.86</td>
<td>NA</td>
</tr>
<tr>
<td>24</td>
<td>-104.14</td>
<td>-89.97</td>
<td>-89.97</td>
<td>-89.97</td>
</tr>
<tr>
<td>20</td>
<td>NA</td>
<td>unstable</td>
<td>unstable</td>
<td>unstable</td>
</tr>
<tr>
<td>16</td>
<td>NA</td>
<td>unstable</td>
<td>unstable</td>
<td>unstable</td>
</tr>
<tr>
<td>12</td>
<td>NA</td>
<td>unstable</td>
<td>unstable</td>
<td>unstable</td>
</tr>
<tr>
<td>8</td>
<td>NA</td>
<td>unstable</td>
<td>unstable</td>
<td>unstable</td>
</tr>
</tbody>
</table>

Table 7.4: Roundoff noise power results for the Cascaded Direct Form II.

<table>
<thead>
<tr>
<th>Word Size</th>
<th>Lower Bound</th>
<th>Linear Simulation Avg = 20</th>
<th>Non-Linear Simulation</th>
<th>DSP56k</th>
</tr>
</thead>
<tbody>
<tr>
<td>32</td>
<td>-190.71</td>
<td>-178.76</td>
<td>-178.89</td>
<td>NA</td>
</tr>
<tr>
<td>28</td>
<td>-166.63</td>
<td>-154.69</td>
<td>-155.06</td>
<td>NA</td>
</tr>
<tr>
<td>24</td>
<td>-142.55</td>
<td>-130.60</td>
<td>-130.80</td>
<td>-130.80</td>
</tr>
<tr>
<td>20</td>
<td>-118.46</td>
<td>-106.54</td>
<td>-106.61</td>
<td>-106.61</td>
</tr>
<tr>
<td>16</td>
<td>-94.38</td>
<td>-82.43</td>
<td>-82.75</td>
<td>-82.75</td>
</tr>
<tr>
<td>12</td>
<td>-70.30</td>
<td>-58.39</td>
<td>-58.23</td>
<td>-58.23</td>
</tr>
<tr>
<td>8</td>
<td>-46.23</td>
<td>-34.31</td>
<td>-34.22</td>
<td>-34.82</td>
</tr>
</tbody>
</table>

Table 7.5: Roundoff noise power results for the Cascaded Direct Form I.

<table>
<thead>
<tr>
<th>Word Size</th>
<th>Lower Bound</th>
<th>Linear Simulation Avg = 20</th>
<th>Non-Linear Simulation</th>
<th>DSP56k</th>
</tr>
</thead>
<tbody>
<tr>
<td>32</td>
<td>-185.75</td>
<td>-170.68</td>
<td>-170.82</td>
<td>NA</td>
</tr>
<tr>
<td>28</td>
<td>-161.67</td>
<td>-146.73</td>
<td>-146.85</td>
<td>NA</td>
</tr>
<tr>
<td>24</td>
<td>-137.58</td>
<td>-122.43</td>
<td>-122.93</td>
<td>-122.93</td>
</tr>
<tr>
<td>20</td>
<td>-113.50</td>
<td>-98.44</td>
<td>-98.89</td>
<td>-98.89</td>
</tr>
<tr>
<td>16</td>
<td>-89.42</td>
<td>-74.39</td>
<td>-74.14</td>
<td>-74.14</td>
</tr>
<tr>
<td>12</td>
<td>-65.33</td>
<td>-50.21</td>
<td>-50.58</td>
<td>-50.58</td>
</tr>
<tr>
<td>8</td>
<td>-41.25</td>
<td>-25.42</td>
<td>-26.11</td>
<td>-26.86</td>
</tr>
</tbody>
</table>

Filter Structure Performance Comparison
### Table 7.6: Roundoff noise power results for the Parallel Direct Form II.

<table>
<thead>
<tr>
<th>Word Size</th>
<th>Roundoff Noise Power (dB)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Lower Bound</td>
</tr>
<tr>
<td>32</td>
<td>-189.69</td>
</tr>
<tr>
<td>28</td>
<td>-165.61</td>
</tr>
<tr>
<td>24</td>
<td>-141.53</td>
</tr>
<tr>
<td>20</td>
<td>-117.45</td>
</tr>
<tr>
<td>16</td>
<td>-93.36</td>
</tr>
<tr>
<td>12</td>
<td>-69.28</td>
</tr>
<tr>
<td>8</td>
<td>-45.20</td>
</tr>
</tbody>
</table>

### Table 7.7: Roundoff noise power results for the Section Optimal.

<table>
<thead>
<tr>
<th>Word Size</th>
<th>Roundoff Noise Power (dB)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Lower Bound</td>
</tr>
<tr>
<td>32</td>
<td>-193.74</td>
</tr>
<tr>
<td>28</td>
<td>-169.66</td>
</tr>
<tr>
<td>24</td>
<td>-145.57</td>
</tr>
<tr>
<td>20</td>
<td>-121.49</td>
</tr>
<tr>
<td>16</td>
<td>-97.41</td>
</tr>
<tr>
<td>12</td>
<td>-73.33</td>
</tr>
<tr>
<td>8</td>
<td>-49.24</td>
</tr>
</tbody>
</table>
Figure 7.5: High Order Direct Form II Structure.

Figure 7.6: Cascaded Direct Form II Structure.
Figure 7.7: Cascaded Direct Form I Structure.

Figure 7.8: Parallel Direct Form II Structure.
Figure 7.9: Section Optimal Structure.

For the Example 1 filter the roundoff noise power of the non-linear model agrees almost exactly with the hardware implementation; the only (slight) deviations occur at small wordlength. Also, the linear stochastic simulation results are within 1 dB of those for the actual DSP implementation. Slight discrepancies between the linear stochastic and the measured DSP56002 noise power is due to the assumptions made by the linear stochastic roundoff noise model, see Section 2.5.

Tables 7.3 through 7.7 also list the roundoff noise lower bounds which were calculated as described in Section 7.2. These lower bounds were found to be approximately 10 to 15 dB lower than the actual implementations. Similar results have been obtained by V.
Debrunner [26, p. 45] where roundoff noise powers where computed based on the filter’s $L_2$ sensitivity function. In that study the computed noise powers are as high as 10 dB above the lower bound. These results differ from those of L.B. Jackson [19], who implies that these bounds are within 6 dB of the actual roundoff noise. Although the bounds do not accurately predict the roundoff noise power, they do indicate the relative noise performance of each structure.

For this filter the Parallel Direct Form II structure and the Section Optimal structures exhibit the lowest roundoff noise powers over the entire range of wordlengths (see Tables 7.6 and 7.7). However, the roundoff noise powers for the Cascaded Direct Form II are only 3 dB higher (see Table 7.4). When choosing the best structure for the filter implementation the computational requirements for each structure must be considered as well. See Table 7.8 for a comparison in terms of the number of instruction cycles required for implementation. For this Example 1 filter the Cascaded Direct Form II structure may, practically, be the best choice due to its lower computational requirement (higher throughput rate, or slower-less expensive processor).

<table>
<thead>
<tr>
<th>Structure</th>
<th>Number of instruction Cycles for DSP56002 Implementation</th>
</tr>
</thead>
<tbody>
<tr>
<td>High Order Direct Form II</td>
<td>73</td>
</tr>
<tr>
<td>Cascaded Direct Form II</td>
<td>37</td>
</tr>
<tr>
<td>Cascaded Direct Form I</td>
<td>31</td>
</tr>
<tr>
<td>Parallel Direct Form II</td>
<td>49</td>
</tr>
<tr>
<td>Section Optimal</td>
<td>72</td>
</tr>
</tbody>
</table>
The roundoff noise error's Power Spectral Density (PSD) may also be used to compare the filter structures performance. These roundoff noise PSD plots can be used to determine the frequency content of the noise power. It is desirable for a structure's noise power to be concentrated in the stopband(s) as apposed to the passband(s). For example, the roundoff noise PSD plots for the Cascaded Direct Form II structure and Parallel Direct Form structure II with a wordlength 24-bits are shown in Figures 7.10 and 7.11 respectively. These two structures differ by only 3 dB in noise power, in this case the PSD can be used in selecting the best structure for implementation. The Parallel Direct Form II would be the best structure since it has slightly less noise power in the passband (0.1 to 0.2 Hz) compared to that of the Cascaded Direct Form II.

![Nonlinear Simulation Output Error PSD, \( \text{Nen} \)]

Figure 7.10: Parallel Direct Form II roundoff noise PSD for a 24-bits wordlength.
Figure 7.11: Cascaded Direct Form II roundoff noise PSD for a 24-bit wordlength.
7.4 6th Order Butterworth Filter

In this section a second example is used to compare the IIR structures. This filter is a Butterworth sixth-order Lowpass filter taken from an example by Mullis and Roberts [5, pp. 402-407]. This filter is specified as a cascade of three second order sections.

\[ H(z) = H_1(z)H_2(z)H_3(z) \]

where

\[ H_i(z) = \frac{g_i(z+1)^2}{z^2 + a_{1,i}z + a_{2,i}}, \quad i = 1, 2, 3 \]

The parameters for each filter section are listed in Table 7.9.

<table>
<thead>
<tr>
<th>Section (i)</th>
<th>Gain (g_i)</th>
<th>(a_{1,i})</th>
<th>(a_{2,i})</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>9.800 \times 10^{-4}</td>
<td>-1.9641</td>
<td>0.96802</td>
</tr>
<tr>
<td>2</td>
<td>9.450 \times 10^{-4}</td>
<td>-1.9112</td>
<td>0.91498</td>
</tr>
<tr>
<td>3</td>
<td>9.325 \times 10^{-4}</td>
<td>-1.8819</td>
<td>0.88563</td>
</tr>
</tbody>
</table>

\[ H(z) = 0.0086 \times 10^{-7} \cdot \frac{1 + 0.0518z^{-1} + 0.1295z^{-2} + 0.1727z^{-3} + 0.1295z^{-4} + 0.0518z^{-5} + z^{-6}}{1 - 5.7572z^{-1} + 13.8153z^{-2} - 17.6871z^{-3} + 12.7415z^{-4} - 4.8969z^{-5} + 0.7844z^{-6}} \]

The -3 dB passband cutoff frequency for this filter is at 0.01\(f_s\), where \(f_s\) is the sampling frequency. The filter magnitude response is shown in Figure 7.12, the pole-zero map in Figure 7.13, and the impulse response in Figure 7.14.
Figure 7.12: Butterworth 6th-order Lowpass filter magnitude response.

Figure 7.13: Butterworth 6th-order Lowpass filter pole-zero map.
Figure 7.14: Butterworth 6th-order Lowpass filter impulse response.

As seen in Figures 7.12 and 7.13, this is a very narrow band filter with poles clustered near the unit circle. A comparison of the frequency response magnitude error for each structure is made in Figure 7.15; the Section Optimal structure exhibits the least error.

The roundoff noise measurements for each filter structure simulation and DSP56002 implementation are now compared. The input sequence is the same as that used in Section 6.2, a 4096-point normally distributed pseudorandom sequence with variance 0.0989 and zero mean. Each structure is scaled using the $L_2$ norm with a safety factor of 3.
Figure 7.15: Magnitude error for the Butterworth 6th-order Lowpass with coefficients quantized to 16-bits.

No data is presented for the High Order Direct Form II structure since it was found to be unstable for wordlengths of less than 38-bits. The roundoff noise power results presented in Tables 7.10 through 7.13 are for the Cascaded Direct Form II, Cascaded Direct Form I, Parallel Direct Form II, and Section Optimal structures respectively. The roundoff noise powers in Tables 7.10 through 7.13 are also plotted in Figures 7.16 through 7.19 for the respective structures.
Table 7.10: Roundoff noise power results for the Cascaded Direct Form II.

<table>
<thead>
<tr>
<th>Word Size</th>
<th>Roundoff Noise Power (dB)</th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Lower Bound</td>
<td>Linear Simulation</td>
<td>Non-Linear Simulation</td>
<td>DSP56k</td>
</tr>
<tr>
<td>32</td>
<td>-176.84</td>
<td>-166.81</td>
<td>-166.18</td>
<td>NA</td>
</tr>
<tr>
<td>28</td>
<td>-152.75</td>
<td>-142.14</td>
<td>-143.13</td>
<td>NA</td>
</tr>
<tr>
<td>24</td>
<td>-128.67</td>
<td>-118.36</td>
<td>-117.24</td>
<td>-117.24</td>
</tr>
<tr>
<td>20</td>
<td>-104.59</td>
<td>-94.22</td>
<td>-95.09</td>
<td>-95.09</td>
</tr>
<tr>
<td>16</td>
<td>-80.51</td>
<td>-70.35</td>
<td>-69.96</td>
<td>-69.96</td>
</tr>
<tr>
<td>12</td>
<td>-56.42</td>
<td>-47.66</td>
<td>-37.02</td>
<td>-37.02</td>
</tr>
<tr>
<td>8</td>
<td>-23.51</td>
<td>Unstable</td>
<td>Unstable</td>
<td>Unstable</td>
</tr>
</tbody>
</table>

Table 7.11: Roundoff noise power results for the Cascaded Direct Form I.

<table>
<thead>
<tr>
<th>Word Size</th>
<th>Roundoff Noise Power (dB)</th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Lower Bound</td>
<td>Linear Simulation</td>
<td>Non-Linear Simulation</td>
<td>DSP56k</td>
</tr>
<tr>
<td>32</td>
<td>-172.09</td>
<td>-161.48</td>
<td>-162.96</td>
<td>NA</td>
</tr>
<tr>
<td>28</td>
<td>-148.00</td>
<td>-137.39</td>
<td>-137.08</td>
<td>NA</td>
</tr>
<tr>
<td>24</td>
<td>-123.92</td>
<td>-113.91</td>
<td>-113.91</td>
<td>-113.91</td>
</tr>
<tr>
<td>20</td>
<td>-99.83</td>
<td>-89.84</td>
<td>-89.63</td>
<td>-89.63</td>
</tr>
<tr>
<td>16</td>
<td>-75.75</td>
<td>-65.47</td>
<td>-66.18</td>
<td>-66.18</td>
</tr>
<tr>
<td>12</td>
<td>-51.67</td>
<td>-42.97</td>
<td>-37.90</td>
<td>-37.90</td>
</tr>
<tr>
<td>8</td>
<td>-27.59</td>
<td>Unstable</td>
<td>Unstable</td>
<td>Unstable</td>
</tr>
</tbody>
</table>

Table 7.12: Roundoff noise power results for the Parallel Direct Form II.

<table>
<thead>
<tr>
<th>Word Size</th>
<th>Roundoff Noise Power (dB)</th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Lower Bound</td>
<td>Linear Simulation</td>
<td>Non-Linear Simulation</td>
<td>DSP56k</td>
</tr>
<tr>
<td>32</td>
<td>-162.92</td>
<td>-157.59</td>
<td>-157.36</td>
<td>NA</td>
</tr>
<tr>
<td>28</td>
<td>-143.84</td>
<td>-133.57</td>
<td>-133.85</td>
<td>NA</td>
</tr>
<tr>
<td>24</td>
<td>-119.75</td>
<td>-109.36</td>
<td>-109.80</td>
<td>-109.80</td>
</tr>
<tr>
<td>20</td>
<td>-95.67</td>
<td>-85.33</td>
<td>-85.27</td>
<td>-85.27</td>
</tr>
<tr>
<td>16</td>
<td>-71.59</td>
<td>-61.36</td>
<td>-61.65</td>
<td>-61.65</td>
</tr>
<tr>
<td>12</td>
<td>-47.51</td>
<td>-37.59</td>
<td>-37.19</td>
<td>-37.19</td>
</tr>
<tr>
<td>8</td>
<td>-23.42</td>
<td>Unstable</td>
<td>Unstable</td>
<td>Unstable</td>
</tr>
</tbody>
</table>
Table 7.13: Roundoff noise power results for the Section Optimal.

<table>
<thead>
<tr>
<th>Word Size</th>
<th>Roundoff Noise Power (dB)</th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Lower Bound</td>
<td>Linear Simulation Avg = 20</td>
<td>Non-Linear Simulation</td>
<td>DSP56k</td>
</tr>
<tr>
<td>32</td>
<td>-196.26</td>
<td>-187.52</td>
<td>-187.08</td>
<td>NA</td>
</tr>
<tr>
<td>28</td>
<td>-172.17</td>
<td>-163.51</td>
<td>-163.75</td>
<td>NA</td>
</tr>
<tr>
<td>24</td>
<td>-148.09</td>
<td>-139.54</td>
<td>-139.20</td>
<td>-139.23</td>
</tr>
<tr>
<td>20</td>
<td>-124.01</td>
<td>-115.38</td>
<td>-115.08</td>
<td>-115.08</td>
</tr>
<tr>
<td>16</td>
<td>-99.93</td>
<td>-91.40</td>
<td>-91.83</td>
<td>-91.83</td>
</tr>
<tr>
<td>12</td>
<td>-75.85</td>
<td>-44.82</td>
<td>-30.58</td>
<td>-30.58</td>
</tr>
<tr>
<td>8</td>
<td>-51.76</td>
<td>Unstable</td>
<td>Unstable</td>
<td>Unstable</td>
</tr>
</tbody>
</table>

Figure 7.16: Cascaded Direct Form II Structure.
Figure 7.17: Cascaded Direct Form I Structure.

Figure 7.18: Parallel Direct Form II Structure.
For this Example 2 the Section Optimal structure exhibits at least 20 dB less roundoff noise power than any other structure over the entire range of wordlengths. Roundoff noise powers for the Cascaded Direct Form II are approximately 5 dB less than for the Cascaded Direct Form I structure and almost 9 dB less than for the Parallel Direct Form II. For this filter the roundoff noise lower bounds are approximately 10 dB less than the actual implementation results, similar to the results of Section 7.3.

Note that the noise power results for the linear-stochastic simulation and actual implementation differ by 10 to 20 dB in the Cascaded Direct Form I, II, and the Section Optimal structures which have a wordlength of 12-bits. These discrepancies are a result of a limit cycle condition produced by the non-linear quantization process. The non-linear
simulation input and output of the Example 2 filter for the Cascaded Direct Form II structure with a wordlength of 12 bits is shown in Figure 7.20. The first 4096 point of the input is the same sequence \( x_i \), as used in all above noise power measurements, the last 1024 points are all zeros. The limit cycles result in the dead bands and a constant output for zero input, seen in Figure 7.20. These same dead bands were observed in the Cascaded Direct Form I and Section Optimal structures. As seen in Figure 7.21 on page 116, the Parallel Direct Form II structure does not exhibit significant limit cycles or dead bands. This explains the close agreement between the linear-stochastic simulation and actual noise powers at 12 bit wordlength.

The computational requirements for the different structures are listed in Table 7.14. For this example the additional computational requirement of the Section Optimal structure will provide at least a 20 dB reduction in roundoff noise.

Table 7.14: Computational requirements for 6th Order Lowpass Filter

<table>
<thead>
<tr>
<th>Structure</th>
<th># of instruction Cycles for DSP56002</th>
</tr>
</thead>
<tbody>
<tr>
<td>High Order Direct Form II</td>
<td>37</td>
</tr>
<tr>
<td>Cascaded Direct Form II</td>
<td>22</td>
</tr>
<tr>
<td>Cascaded Direct Form I</td>
<td>19</td>
</tr>
<tr>
<td>Parallel Direct Form II</td>
<td>28</td>
</tr>
<tr>
<td>Section Optimal</td>
<td>39</td>
</tr>
</tbody>
</table>
Figure 7.20: Cascaded Direct Form II non-linear simulation input and output for a 12 bit wordlength.
Figure 7.2i: Parallel Direct Form II non-linear simulation input and output for a 12-bit wordlength.

In this chapter the non-linear and linear stochastic models have been validated for each filter structure by comparison to the actual DSP56002 implementation. This filter, and the example in Section 7.3, illustrate the need to evaluate the filter structure through a
simulation prior to hardware implementation as a part of the design process. It is obvious that no one structure is best for all filter designs.
7.5 12th Order FIR Filter

This section compares the performance of the MA lattice and Direct Form FIR structures using a 12th order filter. This filter is described by the gain $B_0$ of 0.5 and the zero locations listed in Table 7.15.

<table>
<thead>
<tr>
<th>Zero Radius</th>
<th>Zero Angle (Radians)</th>
</tr>
</thead>
<tbody>
<tr>
<td>9.0000e-001</td>
<td>$\pm 2.9452$</td>
</tr>
<tr>
<td>9.0000e-001</td>
<td>$\pm 0.19635$</td>
</tr>
<tr>
<td>8.0000e-001</td>
<td>3.1416</td>
</tr>
<tr>
<td>8.0000e-001</td>
<td>0</td>
</tr>
<tr>
<td>7.5000e-001</td>
<td>$\pm 1.9635$</td>
</tr>
<tr>
<td>7.5000e-001</td>
<td>$\pm 1.1781$</td>
</tr>
<tr>
<td>5.0000e-001</td>
<td>$\pm 1.5708$</td>
</tr>
</tbody>
</table>

This filter's magnitude response is shown in Figure 7.22, the pole-zero map in Figure 7.23, and the impulse response in Figure 7.24.

![FIR filter magnitude response](image)

Figure 7.22: FIR filter magnitude response
As seen in Figure 7.22 this is a wide band filter with all its zeros within the unit circle.
Since this filter is minimum phase, all reflection coefficient magnitudes are less than unity
and fractional scaling is not required. Overflow scaling is required for the MA lattice per
Chapter 6.

Figure 7.25 compares the magnitude error response for both structures. The MA
lattice exhibits the largest magnitude response error at the center of the filter's pass band.
The Direct Form filter has errors equal in magnitude to those of the MA lattice but in the
upper and lower stop bands.

Filter Structure Performance Comparison
119
Figure 7.25: Magnitude response error comparison for the MA lattice and Direct Form FIR with coefficients quantized to 16-bits.

The roundoff noise measurements for each filter structure simulation and DSP56002 implementation are now compared. The input sequence is the same as that used in Sections 6.2 and 6.3, a 4096-point normally distributed pseudorandom sequence with variance 0.0989 and zero mean. The MA lattice structure is scaled using the L₂ norm with a safety factor of 1.3, which was chosen to maximize the output signal to noise ratio while (almost) eliminating overflows. The roundoff noise power results are presented in Table 7.16 for the MA lattice and Table 7.17 for the Direct Form FIR. The roundoff noise powers in Tables 7.16 and 7.17 are also plotted in Figures 7.26 and 7.27 respectively.
Table 7.16: Roundoff noise power results for the MA lattice.

<table>
<thead>
<tr>
<th>Word Size</th>
<th>Roundoff Noise Power (dB)</th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Linear Simulation</td>
<td>Non-Linear Simulation</td>
<td>DSP56k</td>
</tr>
<tr>
<td></td>
<td>Avg = 20</td>
<td></td>
<td></td>
</tr>
<tr>
<td>32</td>
<td>-184.50</td>
<td>-187.13</td>
<td>NA</td>
</tr>
<tr>
<td>28</td>
<td>-160.34</td>
<td>-162.84</td>
<td>NA</td>
</tr>
<tr>
<td>24</td>
<td>-136.19</td>
<td>-139.18</td>
<td>-139.17</td>
</tr>
<tr>
<td>20</td>
<td>-112.25</td>
<td>-114.71</td>
<td>-114.70</td>
</tr>
<tr>
<td>16</td>
<td>-88.34</td>
<td>-90.71</td>
<td>-90.71</td>
</tr>
<tr>
<td>12</td>
<td>-64.01</td>
<td>-66.72</td>
<td>-66.72</td>
</tr>
<tr>
<td>8</td>
<td>-39.84</td>
<td>-42.56</td>
<td>-42.55</td>
</tr>
</tbody>
</table>

Table 7.17: Roundoff noise power results for the Direct Form FIR.

<table>
<thead>
<tr>
<th>Word Size</th>
<th>Roundoff Noise Power (dB)</th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Linear Simulation</td>
<td>Non-Linear Simulation</td>
<td>DSP56k</td>
</tr>
<tr>
<td></td>
<td>Avg = 20</td>
<td></td>
<td></td>
</tr>
<tr>
<td>32</td>
<td>-197.42</td>
<td>-197.48</td>
<td>NA</td>
</tr>
<tr>
<td>28</td>
<td>-173.36</td>
<td>-173.35</td>
<td>NA</td>
</tr>
<tr>
<td>24</td>
<td>-149.27</td>
<td>-149.25</td>
<td>-149.25</td>
</tr>
<tr>
<td>20</td>
<td>-127.17</td>
<td>-125.22</td>
<td>-125.22</td>
</tr>
<tr>
<td>16</td>
<td>-101.09</td>
<td>-100.97</td>
<td>-100.97</td>
</tr>
<tr>
<td>12</td>
<td>-77.01</td>
<td>-77.05</td>
<td>-77.05</td>
</tr>
<tr>
<td>8</td>
<td>-52.93</td>
<td>-52.93</td>
<td>-52.93</td>
</tr>
</tbody>
</table>
Figure 7.26: Noise Power versus wordlength for the MA lattice Structure.

Figure 7.27: Noise Power versus wordlength for the Direct Form Structure.
In this example the Direct Form FIR structure exhibits at least 10 to 11 dB less roundoff noise power than the MA lattice structure. Recall from Section 4.5, page 74, that the direct form FIR is a sum of products with only a single rounding operation performed on the accumulator results. This single rounding source adds noise directly to the output yielding unity noise gain from the noise source to the output. In contrast, the MA lattice structure contains two rounding operations in each lattice section, where the number of sections is equal to the filter order (see Chapter 6). The noise gains from each noise source to the lattice output are plotted in Figure 7.28. All but two noise gains are concentrated near unity or below. The two sections which exhibit noise gains of approximately 4.0 are the sections in which overflow scaling is performed. In both sections the overflow scaling factor is $2^{-1}$.

![Figure 7.28: Noise gains for the individual rounding sources in the MA lattice structure.](image)

Filter Structure Performance Comparison
Figure 7.28 suggests that the individual sections of the lattice are insensitive to rounding since the corresponding noise gains are near, or less than, unity. However, since there is a larger number of noise sources than in the Direct Form structure, the overall noise power of the lattice is greater. Also, the lattice structure is sensitive to the overflow scaling technique used in this implementation. It may be possible to reduce this sensitivity by altering how the scaling factor is implemented.

The computational requirements for this filter are as follows.

- Direct Form FIR \( (N+1)+4 = 17 \) instruction cycles
- MA lattice \( N*5 + 3 = 63 \) instruction cycles

Although the MA lattice requires additional computation and noise power when used for digital filtering implementations, this structure is widely used in linear prediction and adaptive filtering.
8.0 Conclusions and Further Research

The focus of this research is to develop a fixed-point filter simulator for various filter structures employing a variable wordlength. A non-linear fixed-point arithmetic model is developed in Chapter 2 which is based on the Motorola DSP56002 architectures. The architecture of this processor is similar to that for other fixed-point processors such as the Analog Devices ADSP-2101 and AT&T DSP16. Therefore, this model will be useful in many different processing systems.

The non-linear model emulates the operations of convergent or unbiased rounding and overflow saturation. It is shown in Chapter 2 that the probability of a convergent rounding operation occurring is equal to \(2^{-n}\). If the length of the simulation is less than \(2^n\), then the effect of convergent rounding can be ignored. Therefore, conventional rounding can be performed, thus reducing the time required to perform the non-linear simulations.

Descriptions of the filter simulator, the DSP56002 filter implementation code, and the Matlab-to-DSP56002 Development Systems Interface are presented in Chapter 3. Using this simulator the designer can select the desired wordlength and scaling norm, simulate the filter response, automatically generate the DSP56002 assembly code, and evaluate the hardware implementation on the DSP56002 Development System. This simulator allows the filter designer to vary the scaling norm, overflow safety factor, and wordlength to determine the best filter parameters and structure prior to hardware implementation. The DSP56002 fixed-point implementations for each structure are developed in Chapters 4, 5 and 6.

Conclusions and Further Research
In Chapter 7, each structure's simulation for the non-linear and linear models is validated against its DSP56002 hardware implementation. The two filter designs tested show that there is very good agreement between roundoff noise power results from the non-linear simulation and from the actual DSP56002 implementation. The estimated roundoff noise power from the linear simulation is within 1 dB of that for the actual implementation in most cases. The filter examples in Chapter 7 also illustrate that no single structure provides the lowest roundoff noise for all filters. The simulation environment developed in this thesis for useful in evaluating which structure provides the lowest roundoff noise for a particular filter.

There are many possibilities for further research in this area. Currently, the implementation of lattice structures in fixed-point is being investigated. A scaling method has been developed for this structure and its implementation is currently being executed.

Development of a filter simulator for floating-point arithmetic would be useful in comparing fixed to floating-point implementations for a particular filter and structure. A floating-point linear stochastic model has been proposed by B.D. Rao [23]. A complete floating and fixed-point filter simulation could be developed, allowing the designer to determine which processor best approaches, or accomplishes, the design requirements; these constitute a trade-off between the filter magnitude and phase, output quantization noise power and power spectral density, the DSP processors' speed and cost, and the development time for an implementation.
Bibliography


Vita

Daniel Bailey graduated from the Milwaukee School of Engineering with a Bachelor degree in Electrical Engineering in 1987. As a Hardware Design Engineer with the Raytheon Missile Systems Division he was involved in the development of navigation and guidance systems for surface to air missiles. He attended Virginia Polytechnic Institute and State University in Blacksburg, Virginia, completing a Masters of Science in Electrical Engineering specializing in digital signal processing in 1995. He is employed as a Design Engineer with Rockwell Collins Avionics and Communications Division in the Modem and Satellite Communications Group.

His research interests are in the areas of digital filter design, adaptive filtering, and system identification; his hobbies include bicycling, travel, and scuba diving.