Design of Efficient Digital Interpolation Filters for Integer Upsampling

by

Daniel B. Turek

Submitted to the Department of Electrical Engineering and Computer Science in partial fulfillment of the requirements for the degree of

Master of Engineering

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

June 2004

Author ....
Department of Electrical Engineering and Computer Science
May 3, 2004

Certified by ....
Alan V. Oppenheim
Ford Professor of Engineering
Thesis Supervisor

Accepted by ....
Arthur C. Smith
Chairman, Department Committee on Graduate Students

Design of Efficient Digital Interpolation Filters

for Integer Upsampling

by

Daniel B. Turek

Submitted to the Department of Electrical Engineering and Computer Science
on May 3, 2004, in partial fulfillment of the
requirements for the degree of
Master of Engineering

Abstract

Digital signal interpolation systems can be implemented in a variety of ways. The most basic interpolation system for integer upsampling cascades an expander unit with an interpolation low-pass filter. More complex implementations can cascade multiple expander and low-pass filter pairs. There is also flexibility in the design of interpolation filters.

This thesis explores how digital interpolation systems for integer upsampling can be efficiently implemented. Efficiency is measured in terms of the number of multiplications required for each output sample point. The following factors are studied for their effect on system efficiency: the decomposition of an interpolation system into multiple cascaded stages, the use of recursive and non-recursive interpolation filters, and the use of linear-phase and minimum-phase interpolation filters. In this thesis interpolation systems are designed to test these factors, and their computational costs are calculated. From this data, conclusions are drawn about efficient designs of interpolation systems for integer upsampling.

Thesis Supervisor: Alan V. Oppenheim
Title: Ford Professor of Engineering

Acknowledgments

Foremost, I'd like to express my gratitude to our weekly DSPG group meetings. The core ideas behind my thesis originated in these informal brainstorming sessions, which allowed my thesis to become a reality.

Next to these meetings, I have to thank my thesis advisor, Al Oppenheim. He's critical, demanding, and upfront with his opinions, which led my thesis to becoming the highest quality document I've ever written.

No acknowledgments could be complete without mentioning one's parents. True to this, my family's love and support have flowed constantly throughout my life, for which I cannot express my full appreciation.

And lastly, for Jenn. Your support first made me believe I could complete my thesis.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Interpolation Systems

The process of digital signal interpolation is fundamental to signal processing. It is used in many contexts, most typically for conversion between sampling rates. This thesis explores efficient designs of digital interpolation systems for integer upsample factors.

Interpolation of a signal by an integer upsample factor can be accomplished by processing the signal, $x[n]$, with the cascade of an expander and low-pass filter, as shown in Figure 1-1. If the input signal $x[n]$ has sampling frequency f, this results in the upsampled and interpolated output signal $y[n]$ at the increased sampling frequency Lf. More complex interpolation systems can be designed as the cascade of multiple expanders and low-pass filters. A system containing two expanders and two low-pass filters is shown in Figure 1-2.

Figure 1-1: Interpolation system consisting of an expander and low-pass filter

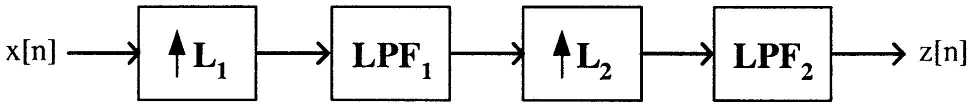

Figure 1-2: Interpolation system consisting of the cascade of two expanders and two low-pass filters

If the parameters of the cascaded interpolator in Figure 1-2 are chosen correctly, namely $L_1 \cdot L_2 = L$ and with appropriate choices of LPF_1 and LPF_2, then this system will perform equivalent interpolation to the system in Figure 1-1. In this case, assuming input sampling frequency f, the interpolated output signals $y[n]$ and $z[n]$ both have sampling frequencies Lf, and more specifically $y[n] = z[n]$. Thus, these two systems are distinct designs accomplishing the same interpolation, and can be compared in terms of computational efficiency.

This thesis studies the tradeoffs in the design of such interpolation systems for integer upsample factors. The metric used for comparison between system designs is computational cost, measured in multiplies per output sample. The following factors in system design are examined for their effect on computational cost:

- Finite impulse response (FIR) and infinite impulse response (IIR) low-pass filter designs.
- Linear-phase and minimum-phase FIR filter designs.
- Cascades of multiple expanders and low-pass filters.
- Distributions of the upsampling factor L over multiple expanders.

1.2 System Specifications

Throughout this thesis, a specific set of specifications for an interpolation system is used to compare varying system designs. These specifications are taken directly from a commercially available Philips Semiconductors DAC chip, [6].

The interpolation system upsamples the input signal by a factor of $L = 128$. The interpolation filter characteristics from the Philips Semiconductors chip, [6], are given in Table 1.1. All interpolation systems compared in this thesis are designed to meet these specifications.

Specification	Frequency Band	Value (dB)
Pass-Band Ripple	$< 0.45 f_s$	0.1
Stop-Band Attenuation	$> 0.55 f_s$	50

Table 1.1: Interpolation filter specifications. f_s is the input sampling frequency.

1.3 Outline of Thesis

Chapter 2 describes the filter classes that will be used as interpolation system low-pass filters. This includes general properties of each filter and how the filters are designed using Matlab. First, Butterworth, Chebyshev, and elliptical IIR filters are discussed, followed by Kaiser and Parks-McClellan FIR linear-phase filters.

Chapter 2 introduces the class of FIR minimum-phase filters. These filters are generated from FIR linear-phase filters using a technique proposed by Schuessler, [2], and hence are referred to as Schuessler filters. Schuessler's method for generating minimum-phase filters is given, and we describe the design of Schuessler filters for use in interpolation systems.

A discussion of interpolation system designs is given in Chapter 3. First, single-stage interpolation systems are considered, which contain an expander unit and a low-pass filter. The single-stage filter specification which satisfies the interpolation system specification in Section 1.2 is calculated. Filter orders for Matlab implementations of these single-stage filters are given.

Chapter 3 considers multiple-stage interpolation systems. Multi-stage systems are designed as the cascade of two or more stages, each containing an expander and a low-pass filter. The general idea behind designing cascaded filters to form an interpolation system is described. Low-pass filter specifications for two-stage, three-stage, and four-stage interpolation systems are then calculated. In addition, these multi-stage systems are implemented in Matlab, and the filter orders are given.

Chapter 4 discusses the computational requirements of interpolation systems, measured in the number of required multiplications per output sample. The chapter begins with a description of polyphase implementations. Next, we derive the computational cost equations for single-stage systems. Using the filter orders from Chapter 3, the numerical costs of single-stage systems are calculated. This is repeated for multi-stage interpolation systems: equations for the computational cost are derived, then evaluated to give the numerical costs of multi-stage systems.

Chapter 5 analyzes the tradeoffs in various interpolation system designs, and draws conclusions about efficient systems. First, we consider the effect of decomposing a system into multiple cascaded stages. It is observed that computational cost generally decreases as a system is decomposed into additional stages. Next, the use of FIR, IIR, linear-phase, and minimum-phase interpolation filters is examined. It is shown that interpolation systems containing a combination of these filter classes attain the lowest computational costs. Finally, the distribution of the expanding factor L over multiple expanders is considered, and we see that well-balanced distributions provide the lowest computational requirements.

Chapter 2

Designs of Interpolation Filters

A variety of filter designs are considered for use in interpolation systems. The broadest distinction lies between FIR and IIR filter designs, which impacts the use of polyphase implementations, and hence the computational cost of a system. The class of FIR filters is further divided into linear-phase and minimum-phase filters. Minimum-phase FIR filters will be derived from linear-phase FIR filters, using the Schuessler Factorization as described in Section 2.3

Matlab is used to model all digital interpolation filters in this thesis. This chapter contains a description of each filter design considered, as well as an explanation of the design techniques and parameters used by Matlab.

2.1 IIR Filters

IIR filters allow flexibility in the location of both poles and zeros in the system function: $H(z) = B(z)/A(z)$, for polynomials $B(z)$ and $A(z)$. Each IIR filter of order N will contain N zeros, and N poles not located at $z = 0$. IIR filter designs generally provide low order filters, though they cannot utilize polyphase implementations to reduce computational cost. We consider three types of IIR filters in our design of the interpolation system: Butterworth, Chebyshev, and elliptical.

Traditionally, because of coefficient quantization, high order IIR filters are best implemented in a cascaded form of second-order sections. In this thesis, all IIR filters are implemented as cascades of second-order filters.

2.1.1 Butterworth Filters

Butterworth filters have monotonic magnitude responses, and for a given set of specifications require the highest orders among the three IIR filters. An N^{th} order Butterworth low-pass filter contains N zeros located at $z = -1$, and N poles inside the unit circle arced around $z = 1$.

To model a Butterworth filter in Matlab, the `buttord` function was supplied with the desired passband and stopband cutoff frequencies f_p and f_s, the allowable passband ripple R_p, and minimum stopband attenuation R_s. These parameters come directly from the specifications in Table 1.1. `buttord` estimates the minimum Butterworth order N that can achieve these specifications, and provides the natural frequency W_n of such a filter. Using N and W_n, Matlab's `butter` function generates the poles and zeros of the desired Butterworth filter[1].

2.1.2 Chebyshev Filters

For fixed specifications, Chebyshev filters provide the smallest step response settling time of the IIR filters considered, and come in two types: Chebyshev Type-1 and Chebyshev Type-2. Type-1 filters attain equal-ripple behavior in the passband. Similar to Butterworth filters, an N^{th} order Chebyshev Type-1 low-pass filter has N zeros located at $z = -1$, but differs by having N poles inside the unit circle arced away from $z = 1$. Type-2 filters attain equal-ripple behavior in the stopband. This is achieved by distributing N zeros in the stopband region of the unit circle, and arcing N poles inside the unit circle around $z = 1$.

In Matlab, either the `cheb1ord` or `cheb2ord` functions accept the f_p, f_s, R_p, and R_s parameters, to estimate the Chebyshev order N and natural frequency W_n. Using these values, `cheby1` and `cheby2` generate the poles and zeros of Chebyshev Type-1 and Type-2 low-pass filters[2].

2.1.3 Elliptical Filters

By allowing both passband and stopband ripple, elliptical filters provide the lowest orders of these three IIR filters. The passband and stopband ripple results from having N zeros in the stopband

[1]For IIR filter design, Matlab first designs the corresponding analog filter, then uses a bilinear transformation, [3], to produce a digital filter. The IIR filter order estimation functions used by Matlab are described in [7].

[2]The `cheby1` function minimizes the filter's stopband frequency edge for the given order N and fixed passband edge f_p. In contrast, `cheby2` maximizes the passband frequency edge for the provided filter order N and fixed stopband edge f_s. Again, Matlab first designs analog IIR filters, then transforms these into digital filters using a bilinear transformation, [3].

region of the unit circle, as Chebyshev Type-2 filters, and N poles inside the unit circle arced away from $z = 1$, as Chebyshev Type-1 filters.

The Matlab design of elliptical filters is similar to that of Butterworth and Chebyshev filters. The `ellipord` function estimates the elliptical filter order N and natural frequency W_n meeting the given specifications. Matlab's `ellip` function uses these parameters to generate the poles and zeros of the desired elliptical filter[3].

2.2 FIR Linear-Phase Filters

Causal FIR filter designs fix the pole locations at $z = 0$. A causal FIR filter of order N will have polynomial system function $H(z)$, which contains exactly N zeros in the z-plane and N poles located at $z = 0$. The poles at $z = 0$ act as delay elements, and will not increase the required computation. Linear-phase FIR filters have their zeros appearing in symmetric pairs about the unit circle, or on the unit circle.

FIR linear-phase filters will generally require higher orders than IIR filters meeting the same specifications. However, when used for sampling rate conversion FIR filters allow for a polyphase implementation, which can considerably reduce the cost of computation. An analysis of this technique accompanies the discussion of computation in Chapter 4.

2.2.1 Kaiser Filters

Kaiser linear-phase filters are generated using the windowing method of FIR filter design, with a Kaiser window. Kaiser windows are generated through the use of Bessel functions, with two parameters: the window length M, and the shape parameter β, [3]. By varying M and β, the frequency domain main lobe width and side lobe amplitudes of Kaiser low-pass filters can be adjusted.

To generate a Kaiser filter in Matlab, the `kaiserord` function is used to estimate the Kaiser window parameters M and β, the Kaiser filter order N, and the normalized cutoff frequency W_n. `kaiserord` requires several arguments: F is a vector of low-pass filter band edge frequencies, $F = [f_p, f_s]$. A specifies the desired filter's amplitude on the bands defined by F. To generate a

[3]For elliptical filter design, Matlab uses the algorithm described in [5] to design an analog elliptical filter, then a bilinear transformation, [3], to produce a digital elliptical filter. The elliptical filter order estimation functions used by Matlab are described in [7].

Kaiser low-pass filter, $A = [1, 0]$. The *dev* parameter gives the maximum allowable ripple in the passband and stopband. Since *dev* is specified in amplitude but R_p is given in dB, $dev = 10^{R_p/20}$. Finally, the sampling frequency parameter is normalized to $F_s = 1$.

The Matlab `kaiser` function accepts the M and β parameters, to produce the desired Kaiser window. The `fir1` function is used to generate the FIR filter coefficients using the windowing method, [1]. This function accepts the Kaiser window generated by `kaiser`, N, and W_n as arguments.

2.2.2 Parks-McClellan Filters

The Parks-McClellan algorithm is a method for optimum approximation of FIR filters. It is an iterative algorithm, in which the filter order N, the passband and stopband edge frequencies f_p and f_s, and the ratio of passband and stopband ripple δ_1/δ_2 are specified. After termination, the algorithm produces an FIR linear-phase approximation to the given system parameters, [3].

For design of Parks-McClellan filters in Matlab, the `remezord` function accepts design parameters F, A, *dev*, and F_s, exactly as `kaiserord`. This function outputs the approximate filter order N, normalized cutoff frequency F_0, and frequency band magnitudes A_0 and weights W. These exact outputs are used by the `remez` function, which implements the Parks-McClellan iterative algorithm in [1], and generates FIR approximation filter coefficients.

2.3 FIR Minimum-Phase Schuessler Filters

In [2] W. Schuessler proposes a method for transforming FIR low-pass filters with linear-phase into minimum-phase FIR filters of half the degree, while maintaining the same passband and stopband edge frequency characteristics. This transformation is possible because the zeros of linear-phase FIR filters appear on the unit circle or in symmetric pairs about the unit circle. With appropriate modification, this allows a decomposition into minimum-phase and maximum-phase components, say $G(z) \cdot G(z^{-1})$, where $|G(e^{j\omega})| = |G(e^{-j\omega})|$ on the unit circle. This decomposition will be called the Schuessler Factorization, and the resultant minimum-phase FIR filters are called Schuessler filters.

2.3.1 The Schuessler Factorization

The design of a minimum-phase Schuessler filter is accomplished in three steps: First, the frequency response is raised by a constant offset to eliminate any single zeros on the unit circle. This is followed by designing a new filter from the minimum-phase component of the raised filter. This minimum-phase filter is then normalized to produce the desired unit gain in the passband.

Raising the Frequency Response

Raising the frequency response of a filter means adding a constant offset to the entire frequency response. If the initial linear-phase filter contains single zeros on the unit circle, then the frequency response must be raised before the filter can be factored into minimum-phase and maximum-phase components. We will raise the frequency response sufficiently to move all single zeros off the unit circle into symmetric pairs, or relocate them to generate double zeros on the unit circle. The raised filter can then be factored as $G(z) \cdot G(z^{-1})$.

The Schuessler Factorization will be performed exclusively on Parks-McClellan equal-ripple filters. To raise the frequency response of a filter, consider a linear-phase FIR filter $H(e^{j\omega})$, with impulse response $h[n]$ symmetric about the point $n = n_0$. Define the real valued function $H_0(e^{j\omega}) = e^{j\omega n_0} \cdot H(e^{j\omega})$. Assume that $H_0(e^{j\omega})$ has maximum deviation from unity of δ_1 in the passband and maximum deviation from zero of δ_2 in the stopband, as shown in Figure 2-1.

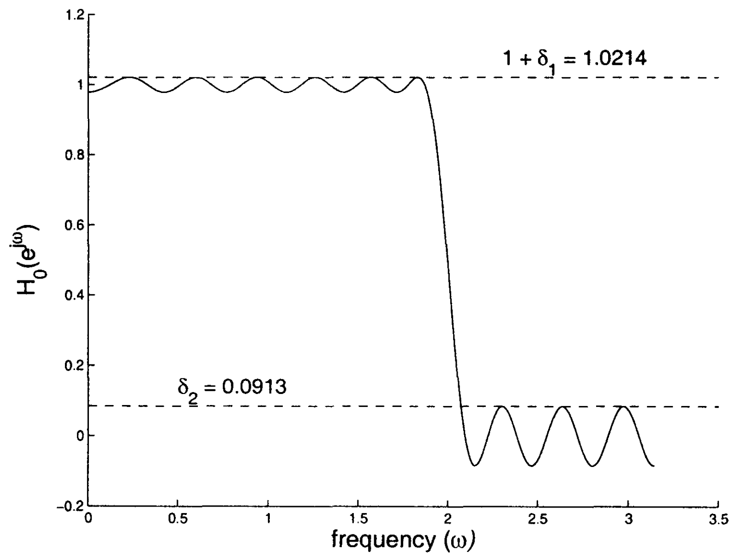

Figure 2-1: Frequency response of $H_0(e^{j\omega})$

A new transfer function is defined as $H_1(z) = H(z) + \delta_2 \cdot z^{-n_0}$, which has the raised frequency response $|H_1(e^{j\omega})| = H_0(e^{j\omega}) + \delta_2$. The magnitude frequency response $|H_1(e^{j\omega})|$ and associated pole-zero diagram of $H_1(e^{j\omega})$ are shown in Figure 2-2. As indicated, the magnitude frequency response has been raised by δ_2, and the filter has zeros of second order in the stopband.

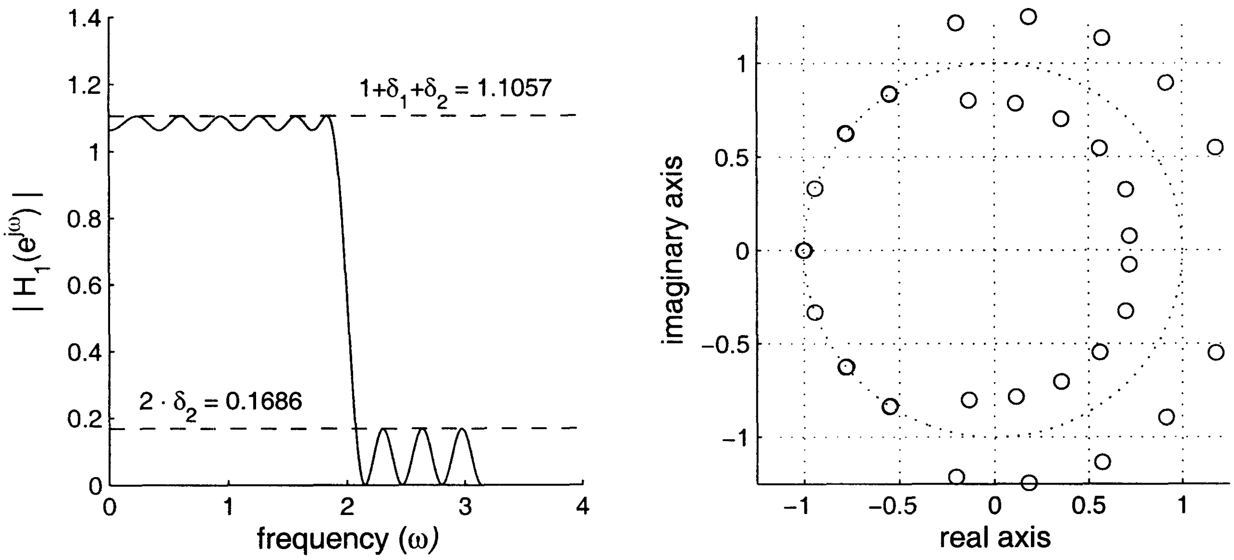

Figure 2-2: Magnitude frequency response and pole-zero plot of $H_1(e^{j\omega})$

Factoring $H_1(z)$

As evident in the pole-zero diagram in Figure 2-2, $H_1(z)$ can be factored as $H_1(z) = z^{-n_0} \cdot H_2(z) \cdot H_2(1/z)$, where $H_2(z)$ has all its zeros inside or on the unit circle, and corresponds to some real impulse response $h_2[n]$. Hence $H_2(e^{j\omega})$ is a minimum-phase low-pass filter, with half the degree of the original $H(e^{j\omega})$.

Normalization to Produce Minimum-Phase Filter

The minimum-phase filter $H_2(e^{j\omega})$ does not oscillate about unity in the passband. Since the original frequency response was raised by δ_2, $H_2(e^{j\omega})$ must be normalized by a factor of $\sqrt{1+\delta_2}$. We define $H_{min}(z) = H_2(z)/\sqrt{1+\delta_2}$ as our final minimum-phase Schuessler filter. The magnitude frequency response of $H_{min}(e^{j\omega})$ is shown in Figure 2-3.

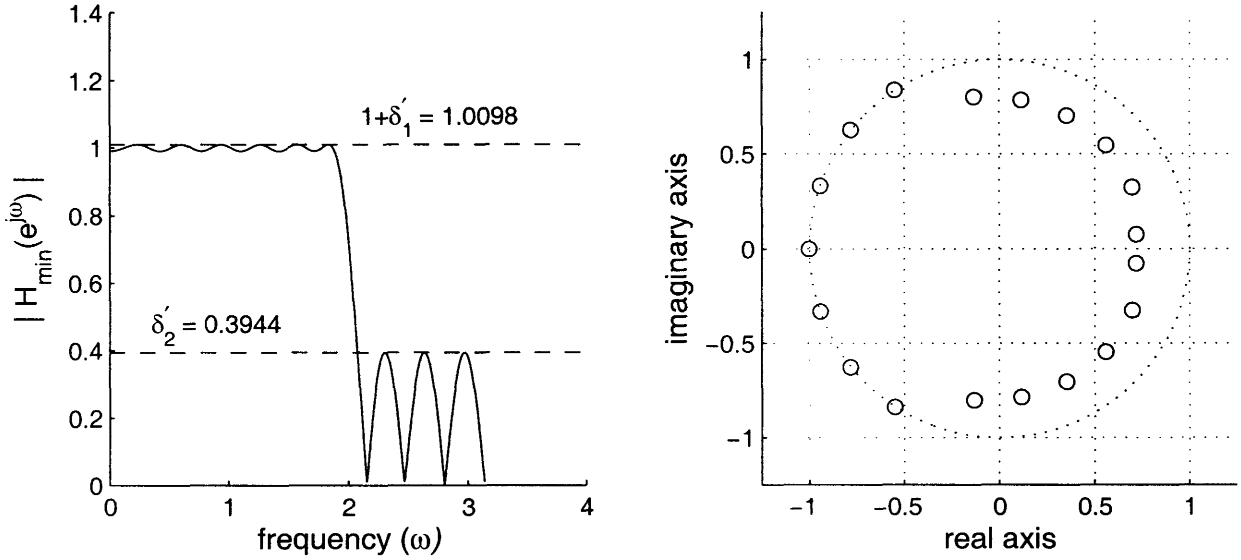

Figure 2-3: Magnitude frequency response and pole-zero plot of $H_{min}(e^{j\omega})$

The minimum-phase filter $H_{min}(e^{j\omega})$ maintains the same passband and stopband edge frequencies as the original filter, $H(e^{j\omega})$. However, due to the factorization of $H_1(z) = z^{-n_0} \cdot H_2(z) \cdot H_2(1/z)$ and normalization by $\sqrt{1+\delta_2}$, the resultant filter has new passband and stopband deviations: δ_1' in the passband and δ_2' in the stopband, as indicated in Figure 2-3. These deviations can be calculated in terms of the original passband and stopband deviations, δ_1 and δ_2 respectively, as

$$\delta_1' = \sqrt{1 + \frac{\delta_1}{1+\delta_2}} - 1 \tag{2.1}$$

$$\delta_2' = \sqrt{\frac{2\delta_2}{1+\delta_2}} \tag{2.2}$$

2.3.2 Design of Schuessler Filters for Interpolation Systems

Low-pass filters can be generated using the Schuessler Factorization technique which meet the interpolation system specifications in Table 1.1. We require the resultant Schuessler minimum-phase filter, $H_{min}(e^{j\omega})$, to possess 50dB attenuation in the stopband. This implies a maximum stopband deviation of $\delta_2' = 3.162 \cdot 10^{-3}$. From equation (2.2), this specifies that the initial linear-phase filter $H(e^{j\omega})$ must have stopband deviation of $\delta_2 = 5 \cdot 10^{-6}$, or about 106dB attenuation.

Parks-McClellan FIR filters exhibit the linear-phase property necessary for the Schuessler Factorization, so Parks-McClellan filters are used as the initial low-pass filters, $H(e^{j\omega})$. To design a Schuessler filter, a Parks-McClellan filter is first over-designed to have 106dB attenuation in the stopband. This design is accomplished as described in Section 2.2.2, except the *dev* ripple parameter is reduced to allow less stopband ripple. After applying the Schuessler Factorization to this over-designed Parks-McClellan filter, the resultant Schuessler filter $H_{min}(e^{j\omega})$ achieves the desired 50dB stopband attenuation. Frequency plots of $|H(e^{j\omega})|$ and $|H_{min}(e^{j\omega})|$ are given in Figure 2-4.

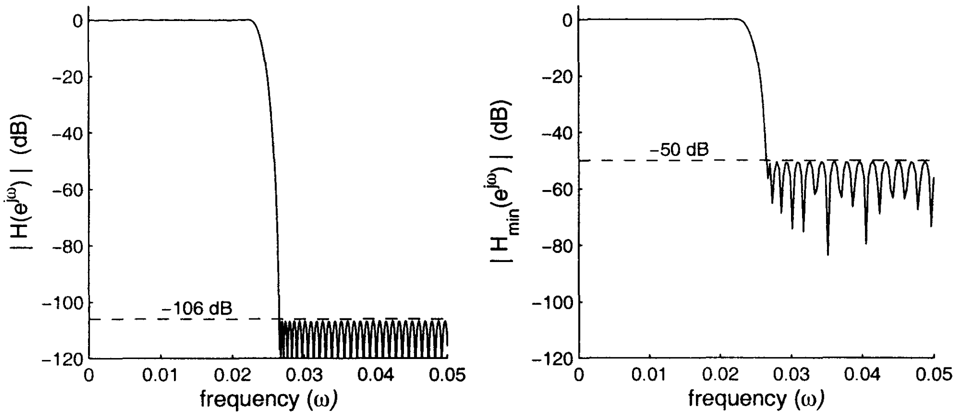

Figure 2-4: Magnitude frequency responses of $H(e^{j\omega})$ and $H_{min}(e^{j\omega})$

2.3.3 Linear-Phase Impulse Response Folding

The Schuessler Factorization is found to provide approximately a 15% reduction in FIR filter order, but at the cost of surrendering the linear-phase property of an FIR filter. When implementing a linear-phase FIR filter, an impulse response folding technique can be used to take advantage of the symmetric impulse response, and generate a 50% savings in computation, [3]. However, if the Schuessler Factorization is used to produce a minimum-phase filter, then impulse response folding can no longer be used.

In our application of low-pass filters as interpolation filters for integer upsampling, the low-pass filters will always follow expander units. In this configuration a polyphase implementation can be used in the implementation of the filter, regardless of whether the filter is linear-phase or minimum-phase. For an expander factor L, the polyphase implementation provides a savings factor of $\frac{1}{L}$, as compared to the $\frac{1}{2}$ savings from impulse response folding. Assuming an expander with $L \geq 2$, impulse response folding can never computationally outperform a polyphase implementation.

Chapter 3

Interpolation System Designs

As described in Chapter 1, interpolation systems can be designed as an expander and low-pass filter pair, or as the cascade of multiple expander and low-pass filter stages. Systems containing a single expander and interpolation filter will be called single-stage systems. A cascade of two or more expander and filter stages will be called a multiple-stage interpolation system.

This chapter analyzes the filter specifications necessary for single-stage and multiple-stage interpolation systems. First, single-stage systems are discussed, and filter orders for single-stage interpolation filters are given. Next, the general approach for designing cascaded filters is explained, using the example of a two-stage interpolation system. Finally, two-stage, three-stage, and four-stage interpolation systems are analyzed.

3.1 Single-Stage Interpolation Systems

Consider an interpolation system consisting of an expander followed by a low-pass filter. The specific case of expanding by a factor of 128, then low-pass filtering to meet the specifications given in Table 1.1 are used.

3.1.1 Filter Specification Analysis

Considering that upsampling a signal by a factor of L compresses the signal's frequency response by that same factor of L, we see that the passband and stopband edges of a low-pass filter following an expander must also be compressed by a factor of L. Hence in the single-stage implementations of a filter meeting the specifications of Table 1.1 and following expanding by a factor of 128, the filter's frequency edges must be

$$\text{single-stage H}_1 : f_{passband} = \frac{0.45 f_s}{128} \tag{3.1}$$

$$\text{single-stage H}_1 : f_{stopband} = \frac{0.55 f_s}{128} \tag{3.2}$$

3.1.2 System Implementations

We consider implementing this interpolation filter using the IIR filters described in Section 2.1, the FIR linear-phase filters described in Section 2.2, and Schuessler filters as described in Section 2.3. Each of these filters were designed to meet the specifications in Table 1.1, by using the passband and stopband edge frequencies given in equations (3.1) and (3.2). The minimum required filter orders obtained by Matlab are given below in Table 3.1.

Filter Type	H_1
FIR Kaiser Window	4601
FIR Parks-McClellan	3906
FIR Schuessler	3341
IIR Butterworth	43
IIR Chebyshev	14
IIR Elliptical	7

Table 3.1: Single-stage system filter orders

3.2 Design of Cascaded Low-Pass Filters

In this section we examine the design of a two-stage cascaded interpolation system, to demonstrate the design approach of cascaded low-pass filters. Consider a two-stage interpolation system, consisting of two expanders and two low-pass filters as indicated in Figure 3-1.

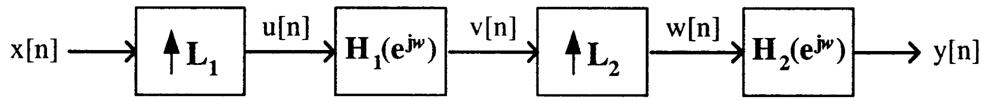

Figure 3-1: Two-stage interpolation system

The input signal $x[n]$ is first expanded by L_1 to produce $u[n]$. $u[n]$ is then filtered by $H_1(e^{j\omega})$, resulting in the intermediate signal $v[n]$. If we take the input $x[n] = \delta[n]$, then $v[n]$ is the impulse response $h_1[n]$, or in the frequency domain $V(e^{j\omega}) = H_1(e^{j\omega})$. Choosing a low-pass filter design of $H_1(e^{j\omega})$, the Fourier transform $V(e^{j\omega})$ is shown in Figure 3-2.

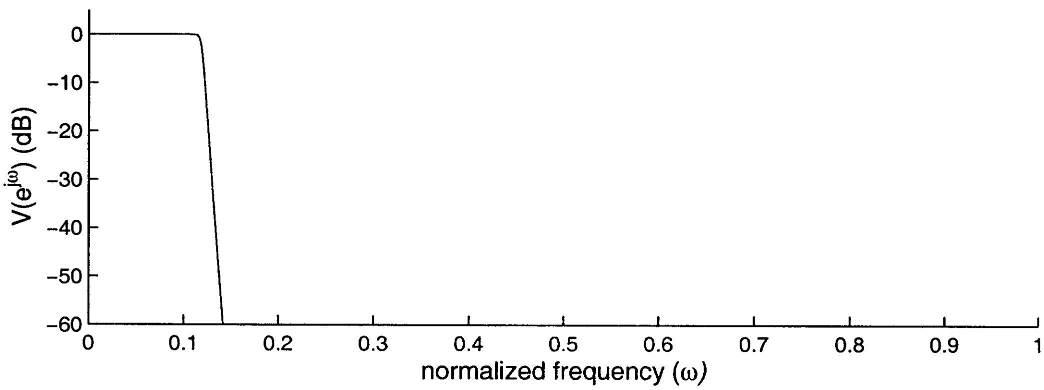

Figure 3-2: Fourier transform $V(e^{j\omega})$

The signal $v[n]$ is then expanded by L_2, producing the signal $w[n]$. Correspondingly in the frequency domain, $V(e^{j\omega})$ is compressed by a factor of L_2, resulting in $W(e^{j\omega})$. Taking the case $L_2 = 16$, the Fourier transform of $W(e^{j\omega})$ is shown in Figure 3-3, appearing as compressed copies of $V(e^{j\omega})$.

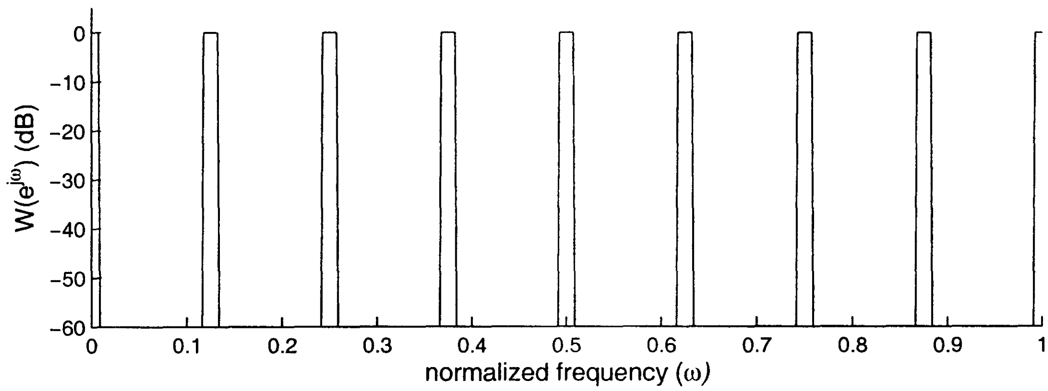

Figure 3-3: Fourier transform $W(e^{j\omega})$

We observe that Figure 3-3 consists of copies of a single "lobe", generated by expanding $V(e^{j\omega})$. Our general design approach is to choose successive low-pass filters with unit gain over the first lobe of the input Fourier transform, and to attenuate all higher frequency copies of this lobe. In the time domain, this interpolates the intermediate signal after each expander. Each expander and low-pass filter stage actually interpolates the input signal by some upsampling factor L_i, where L_i is a factor of $L = 128$.

Following this approach, $H_2(e^{j\omega})$ is chosen to have unit gain over the first lobe of $W(e^{j\omega})$, roughly in the range of normalized frequencies $\omega \in [0, 0.01]$, and to achieve 50dB attenuation over all successive lobes of $W(e^{j\omega})$, roughly over the normalized frequency range $\omega \in [0.11, 1]$. Designed such, $H_2(e^{j\omega})$ preserves the first lobe of $W(e^{j\omega})$ and removes all successive lobes. A possible frequency response of $H_2(e^{j\omega})$ is shown in Figure 3-4.

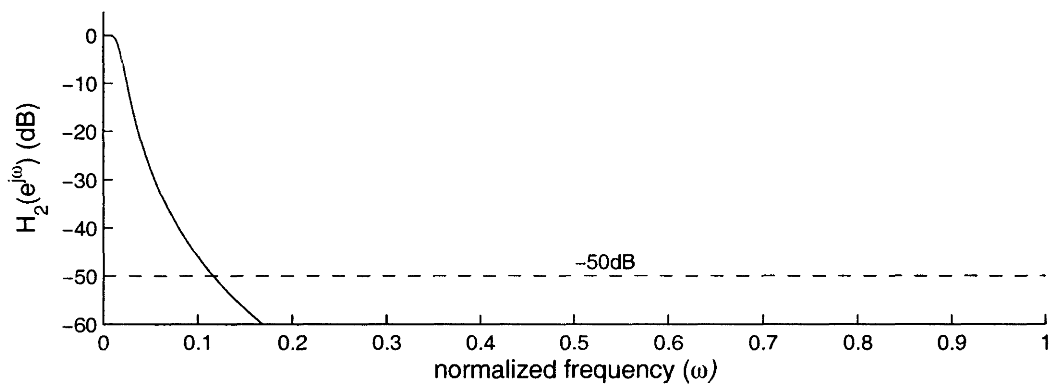

Figure 3-4: Frequency response of $H_2(e^{j\omega})$

Finally, as $H_2(e^{j\omega})$ filters the intermediate signal $w[n]$, the frequency response $H_2(e^{j\omega})$ and Fourier transform $W(e^{j\omega})$ are multiplied to produce the output $Y(e^{j\omega})$. This output Fourier transform is given in Figure 3-5.

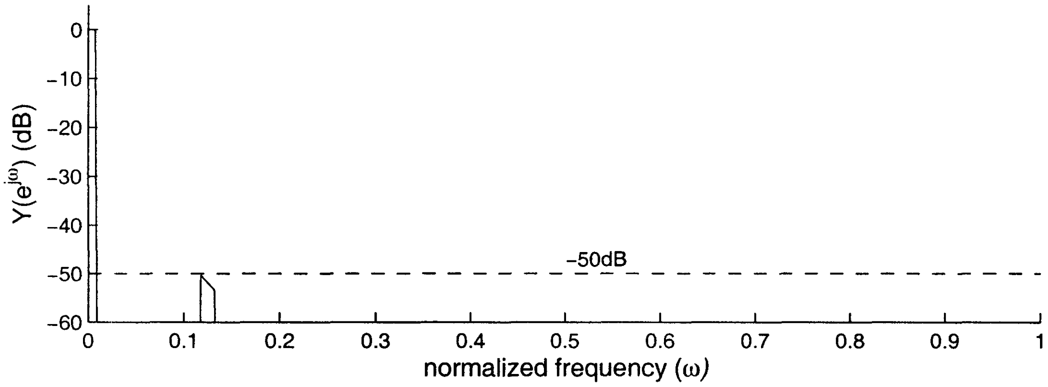

Figure 3-5: Fourier transform $Y(e^{j\omega})$

Having analyzed the entire two-stage interpolation system from Figure 3-1, we now see how this multi-stage design approach can utilize two low-pass filters to achieve the effect of a single low-pass filter. In this example, the Fourier transform of the overall system's impulse response has unit gain over the passband $\omega \in [0, 0.01]$, and achieves 50dB attenuation in the stopband $\omega > 0.01$.

This idea can be extended to any number of cascaded expander and filter stages, where each low-pass filter removes the additional lobes added by the previous expansion. This approach is used to design all multi-stage interpolation systems discussed through this chapter.

3.3 Two-Stage Interpolation Systems

First we examine two-stage designs of interpolation systems, where the entire system appears as in Figure 3-1.

This breakdown into two distinct stages has some advantages over the single-stage implementations discussed in Section 3.1. Similar to the single-stage implementations, the first filter $H_1(e^{j\omega})$ still must satisfy the original specification in Table 1.1. However, H_1 operates at a lower sampling rate than the single-stage filter, and hence its cutoff bands will be shallower. Since H_1 has a shallower cutoff, it will require a lower filter order and less computation.

Similarly, the specifications on the second filter $H_2(e^{j\omega})$ are relaxed substantially. The stopband edge of H_2 must only remove the additional lobes of the expanded filter H_1, and hence will involve a much shallower cutoff. Again, this relaxation in the filter's cutoff bands leads to a lower order for H_2 and less computation.

Dividing the system into stages also has a downside. Introducing a second filter may require more computation, simply on account of filtering the signal twice. The overall computational requirements will be dependent on the expander values L_1 and L_2, and the filter orders of H_1 and H_2. A study of the required computation appears in Chapter 4.

3.3.1 Filter Specification Analysis

In a cascaded interpolation system, the passband and stopband specifications of the low-pass filters H_i become dependent on the expander values L_i. Recalling the two-stage interpolation system in Figure 3-1, the output of the second expander $W(e^{j\omega})$ must have its lowest frequency lobe satisfying the specifications in Table 1.1. Note that in a two-stage system, $L_1 \cdot L_2 = 128$. Since $W(e^{j\omega})$ consists of copies of $V(e^{j\omega})$ compressed by a factor of L_2, the first low-pass filter H_1 must have frequency cutoff edges:

$$\text{two-stage H}_1 : f_{passband} = \frac{0.45 f_s}{L_1}$$

$$\text{two-stage H}_1 : f_{stopband} = \frac{0.55 f_s}{L_1}$$

The second low-pass filter, H_2, must be designed so as not to interfere with the lowest frequency lobe of $W(e^{j\omega})$, but must eliminate all higher frequency lobes. The first lobe of $W(e^{j\omega})$ extends through the frequency $\frac{0.45 f_s}{128}$, which therefore defines the passband edge of H_2. The second lobe of $W(e^{j\omega})$ is centered around the frequency $\frac{f_s}{L_2}$, and hence to remove this second lobe, H_2 must have its stopband edge $\frac{0.55 f_s}{128}$ less than $\frac{f_s}{L_2}$. Consequently, we arrive at the passband and stopband specifications of H_2 as

$$\text{two-stage H}_2 : f_{passband} = \frac{0.45 f_s}{128}$$

$$
\begin{aligned}
\text{two-stage H}_2 : f_{stopband} &= \frac{f_s}{L_2} - \frac{0.55 f_s}{128} \\
&= \frac{f_s}{128} \cdot (L_1 - 0.55)
\end{aligned}
$$

3.3.2 System Implementations

In the two-stage design, the overall expansion factor L is divided between L_1 and L_2. Depending on the choice of these expander values, the specifications and the orders of the filters H_1 and H_2 will change. Designs for all possible values of L_1 and L_2 were experimented with, resulting in a large number of possible orders for H_1 and H_2. Only a small subset of this data is given here, to give an idea of the results. However, later analysis will involve all possible expander values and filter orders.

The following table gives the orders of H_1 and H_2, for two different choices of the expander values L_1 and L_2, and for three different choices of filters: the FIR Parks-McClellan, FIR Schuessler, and IIR Butterworth filters. This table should be compared with the single-stage results in Table 3.1, which is suggestive of the effect of splitting the system into distinct stages: the filter orders are substantially lower, but the system now requires two filters. A detailed analysis of how these filter orders affect the computational cost will follow.

Filter Type	Expander Values	H_1	H_2
FIR Parks-McClellan	$(L_1, L_2) = (2, 64)$	54	344
	$(L_1, L_2) = (16, 8)$	430	22
FIR Schuessler	$(L_1, L_2) = (2, 64)$	42	268
	$(L_1, L_2) = (16, 8)$	335	18
IIR Butterworth	$(L_1, L_2) = (2, 64)$	25	7
	$(L_1, L_2) = (16, 8)$	38	3

Table 3.2: Two-stage system filter orders

3.4 Three-Stage Interpolation Systems

In a manner similar to the two-stage approach described, the overall interpolation system can be decomposed into a cascade of three-stages, each consisting of an expander and a low-pass filter. The three expansion values are L_1, L_2, and L_3, where $L_1 \cdot L_2 \cdot L_3 = 128$, and the three low-pass filters will have frequency responses $H_1(e^{j\omega})$, $H_2(e^{j\omega})$, and $H_3(e^{j\omega})$. The approach for designing cascaded filters is repeated for each successive cascade:

- H_1, after being expanded by L_1 and L_2, will satisfy the interpolation filter specifications.
- H_2 will remove the repeated lobes of the expanded output of the filter H_1.
- H_3 will remove the repeated lobes of the expanded output of the filter H_2.

Introducing a third stage in this cascade has advantages and disadvantages similar to those of dividing the original single-stage system into two-stages. Having three distinct stages allows for gentler cutoffs in all three filters, and hence lower filter orders and less computation in each filter. However, the presence of three filters can potentially also increase the required computation, since the signal must be processed by three distinct filters.

3.4.1 Filter Specification Analysis

The cutoff frequencies of H_1, H_2, and H_3 can be calculated in a manner similar to Section 3.3.1. The first low-pass filter, H_1, will be expanded by L_2 and L_3, and then must meet the overall system specifications in Table 1.1.

$$\text{three-stage H}_1 : f_{passband} = \frac{0.45 f_s}{L_1}$$

$$\text{three-stage H}_1 : f_{stopband} = \frac{0.55 f_s}{L_1}$$

H_2 must maintain the first lobe of the expanded H_1, but remove all successive copies.

$$\text{three-stage H}_2 : f_{passband} = \frac{0.45 f_s}{L_1 L_2}$$

$$\text{three-stage H}_2 : f_{stopband} = \frac{f_s}{L_2} - \frac{0.55 f_s}{L_1 L_2} = \frac{f_s}{L_1 L_2} \cdot (L_1 - 0.55)$$

Similarly, H_3 must preserve the first lobe of H_2 after being expanded by L_3, but remove all repeated lobes.

$$\text{three-stage H}_3 : f_{passband} = \frac{0.45 f_s}{128}$$

$$\text{three-stage H}_3 : f_{stopband} = \frac{f_s}{L_3} - \frac{0.55 f_s}{128} = \frac{f_s}{128} \cdot (L_1 L_2 - 0.55)$$

3.4.2 System Implementations

Again, there are a large number of choices for the expander values L_1, L_2 and L_3. To give an idea of the results of this three-stage breakdown, filter orders for the FIR Parks-McClellan, FIR Schuessler, and IIR Butterworth filters are given, for two different sets of expander values. These should be compared with the two-stage results in Table 3.2.

Filter Type	Expander Values	H_1	H_2	H_3
FIR Parks-McClellan	$(L_1, L_2, L_3) = (2, 4, 16)$	54	21	49
	$(L_1, L_2, L_3) = (8, 8, 2)$	215	24	3
FIR Schuessler	$(L_1, L_2, L_3) = (2, 4, 16)$	42	16	38
	$(L_1, L_2, L_3) = (8, 8, 2)$	167	19	2
IIR Butterworth	$(L_1, L_2, L_3) = (2, 4, 16)$	25	6	3
	$(L_1, L_2, L_3) = (8, 8, 2)$	38	3	1

Table 3.3: Three-stage system filter orders

3.5 Four-Stage Interpolation Systems

Four-stage interpolation systems were also designed, consisting of four expanders and four low-pass filters.

3.5.1 Filter Specification Analysis

By analogy, the passband and stopband edge frequencies of H_1, H_2, H_3, and H_4 are given as:

$$\text{four-stage H}_1 : f_{passband} = \frac{0.45 f_s}{L_1}$$

$$\text{four-stage H}_1 : f_{stopband} = \frac{0.55 f_s}{L_1}$$

$$\text{four-stage H}_2 : f_{passband} = \frac{0.45 f_s}{L_1 L_2}$$

$$\begin{aligned}\text{four-stage H}_2 : f_{stopband} &= \frac{f_s}{L_2} - \frac{0.55 f_s}{L_1 L_2} \\ &= \frac{f_s}{L_1 L_2} \cdot (L_1 - 0.55)\end{aligned}$$

$$\text{four-stage H}_3 : f_{passband} = \frac{0.45 f_s}{L_1 L_2 L_3}$$

$$\begin{aligned}\text{four-stage H}_3 : f_{stopband} &= \frac{f_s}{L_3} - \frac{0.55 f_s}{L_1 L_2 L_3} \\ &= \frac{f_s}{L_1 L_2 L_3} \cdot (L_1 L_2 - 0.55)\end{aligned}$$

$$\text{four-stage H}_4 : f_{passband} = \frac{0.45 f_s}{128}$$

$$\begin{aligned}\text{four-stage H}_4 : f_{stopband} &= \frac{f_s}{L_4} - \frac{0.55 f_s}{128} \\ &= \frac{f_s}{128} \cdot (L_1 L_2 L_3 - 0.55)\end{aligned}$$

3.5.2 System Implementations

To give an idea of four-stage filter orders, data for FIR Parks-McClellan, FIR Schuessler, and IIR Butterworth filters is given below:

Filter Type	Expander Values	H_1	H_2	H_3	H_4
FIR Parks-McClellan	$(L_1, L_2, L_3, L_4) = (2, 8, 4, 2)$	54	43	9	3
	$(L_1, L_2, L_3, L_4) = (8, 4, 2, 2)$	215	10	3	3
FIR Schuessler	$(L_1, L_2, L_3, L_4) = (2, 8, 4, 2)$	42	34	8	2
	$(L_1, L_2, L_3, L_4) = (8, 4, 2, 2)$	167	9	2	2
IIR Butterworth	$(L_1, L_2, L_3, L_4) = (2, 8, 4, 2)$	25	7	3	1
	$(L_1, L_2, L_3, L_4) = (8, 4, 2, 2)$	38	3	2	1

Table 3.4: Four-stage system filter orders

Chapter 4

Computation

Several strategies for implementing interpolation systems have now been discussed. We have considered FIR and IIR filters, as well as designing the interpolation system as the cascade of multiple stages. This chapter analyzes the computational costs of these various implementations. It begins with a discussion of polyphase implementations. Then, general analytic expressions for the computational cost of each system are derived, and the actual costs of the various systems are calculated.

4.1 Polyphase Implementations

A polyphase implementation is a technique for decomposing a filter into a filter bank, used when a decimator follows a filter or an expander precedes a filter. This technique will be used repeatedly in our study of interpolation filters following expander units. For an expander of value L followed by an FIR filter, the polyphase implementation is shown in Figure 4-1, where each $H_i(z)$ component filter consists of delayed L^{th} samples from the original filter. A detailed explanation of polyphase implementations can be found in [3].

It is important to note that polyphase implementations can only be used on FIR filters. However, a polyphase implementation can be be used on the FIR part of an IIR filter, which will be discussed later.

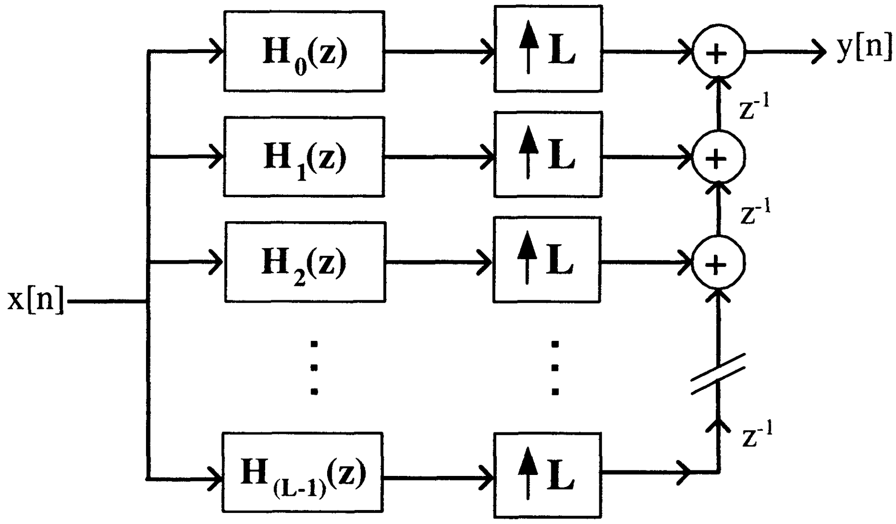

Figure 4-1: Polyphase implementation of an FIR filter

4.1.1 FIR Polyphase Calculation

We first consider the case of an expander with value L followed by an FIR filter of length N. Without polyphase techniques, each output sample will come directly from the filter, and hence require N multiplies to compute. This implementation requires

$$N \text{ multiplies/output sample for FIR filter} \tag{4.1}$$

In the case of the polyphase implementation of an FIR filter, the filter is decomposed into L component filters, each of length N/L. Consider a single input sample to the system. It is processed by L component filters, each of length N/L, and hence $L \cdot N/L = N$ multiplies are required.

For this single input sample, each component filter produces a single output point. Each output point passes through an expander of value L, generating L output points. These output points pass through delay elements and an adder, to produce the final sequence of L output samples. In terms of computation, we required N total multiplies to produce L output points, and hence the computational cost is reduced to

$$\frac{N}{L} \text{ multiplies/output sample for FIR polyphase filter} \tag{4.2}$$

4.1.2 IIR Polyphase Calculation

Instead we begin with an expander followed by an IIR filter. Assume the expander has value L, and the IIR filter has order N, consisting of N poles and N zeros. Without polyphase techniques, each output samples comes directly from the IIR filter, and requires a total of $2N$ multiplies. The computational cost is

$$2N \text{ multiplies/output sample for IIR filter} \tag{4.3}$$

A different approach is required for polyphase implementations of IIR filters, since the polyphase technique cannot directly be applied. Instead, the IIR filter is decomposed into a cascade of two component filters: an FIR filter containing only zeros and an IIR filter containing only poles, where the cascade of these component filters is equivalent to the original IIR filter. This decomposed system appears as the cascade of an expander, an FIR filter, and an IIR filter. If the original IIR filter was order N, consisting of N poles and N zeros, then the component FIR contains N zeros and the component IIR filter contains N poles.

A polyphase implementation can now be applied to the decomposed system. Considering only the expander of value L and the component FIR filter of order N, from equation 4.2 the polyphase implementation will require N/L multiplies/output. Considering only the component IIR filter, no polyphase techniques are possible. Since the component IIR filter has N poles, it will require N multiplies/output. Thus the total computational cost is

$$\frac{N}{L} + N \text{ multiplies/output sample for IIR polyphase filter} \tag{4.4}$$

Equations (4.1), (4.2), (4.3), and (4.4) give the computational cost for FIR and IIR filters, with and without polyphase implementations. These will be used for calculating the total computational cost of single-stage and multi-stage interpolation systems.

4.2 Single-Stage Implementations

We consider the computational costs of single-stage interpolation systems, as discussed in Section 3.1. These implementations always contain a single expander, with value $L = 128$.

Results are given for FIR Parks-McClellan, FIR Schuessler, and IIR Butterworth filters. The single-stage orders of these filters appear in Table 3.1. It will always be beneficial to use polyphase techniques, so only the results for polyphase implementations are given. Using the equations derived in Section 4.1, the computational cost for each of these filters is calculated. These results are given in Table 4.1.

Filter Type	Cost
FIR Parks-McClellan	30.516
FIR Schuessler	26.102
IIR Butterworth	43.366

Table 4.1: Computational costs of single-stage filters in multiplies/output sample

4.3 Multi-Stage Implementations

In a cascaded interpolation system, each filter will operate at a different sampling rate. This causes the computational cost equations to become more complicated. These equations are derived for multi-stage implementations, then applied to several multi-stage designs.

4.3.1 Computational Costs of Cascades

Consider a two-stage implementation, cascading an expander of value L_1, a filter of order N_1, a second expander of value L_2, and a second filter of order N_2. We consider a single input sample, which is first expanded to produce L_1 sample points to the first filter. These L_1 samples are processed by the first filter of order N_1, which requires L_1N_1 multiplies and generate L_1 output samples.

These L_1 output samples are then expanded by L_2, producing L_1L_2 samples. Finally, these samples are processed by the second filter of length N_2, which requires $L_1L_2N_2$ multiplies. This filter generates a total of L_1L_2 output samples.

In summary, we have required $L_1 N_1$ multiplies in the first filter, and $L_1 L_2 N_2$ multiplies in the second filter. This produced $L_1 L_2$ output samples, and hence the total multiplies per output sample is $N_1/L_2 + N_2$. This result demonstrates the effect of the first filter operating at a lower sampling rate: It's order N_1 is divided by the second upsampler value L_2.

4.3.2 FIR Computational Costs

In the absence of polyphase implementations, the analysis in Section 4.3.1 gives the cost of a two-stage FIR implementation as

$$\frac{N_1}{L_2} + N_2 \text{ multiplies/output sample for two-stage FIR filter} \tag{4.5}$$

Consider a polyphase implementation of both FIR filters in this cascade. The first filter is preceded by an expander of value L_1, so analogous to equations 4.1 and 4.2, the required multiplies per output becomes $N_1/(L_1 L_2)$. Similarly, the second filter is preceded by an expander of value L_2, so the multiplies per output becomes N_2/L_2. Combining these, we have

$$\frac{N_1}{L_1 L_2} + \frac{N_2}{L_2} \text{ multiplies/output sample for two-stage FIR polyphase filter} \tag{4.6}$$

We also examine three and four-stage implementations. Three-stage cascades contain expanders L_1, L_2, and L_3 and FIR filters of lengths N_1, N_2, and N_3. Four-stage cascades will also include expander L_4 and a fourth FIR filter of length N_4. Following similar logic, we conclude that

$$\frac{N_1}{L_2 L_3} + \frac{N_2}{L_3} + N_3 \text{ multiplies/output sample for three-stage FIR filter} \tag{4.7}$$

$$\frac{N_1}{L_1 L_2 L_3} + \frac{N_2}{L_2 L_3} + \frac{N_3}{L_3} \text{ multiplies/output sample for three-stage FIR polyphase filter} \tag{4.8}$$

$$\frac{N_1}{L_2 L_3 L_4} + \frac{N_2}{L_3 L_4} + \frac{N_3}{L_4} + N_4 \text{ multiplies/output sample for four-stage FIR filter} \tag{4.9}$$

$$\frac{N_1}{L_1L_2L_3L_4} + \frac{N_2}{L_2L_3L_4} + \frac{N_3}{L_3L_4} + \frac{N_4}{L_4} \text{ multiplies/output sample for four-stage FIR polyphase filter} \quad (4.10)$$

Finally, these cost equations are applied to FIR Parks-McClellan and FIR Schuessler filters. Filter orders from Tables 3.2, 3.3, and 3.4 are used to calculate the computational costs. The results for polyphase implementations are given in Table 4.2.

	Expander Values	FIR Parks-McClellan	FIR Schuessler
Two-Stage Designs	$(L_1, L_2) = (2, 64)$	5.797	4.516
	$(L_1, L_2) = (16, 8)$	6.109	4.867
Three-Stage Designs	$(L_1, L_2, L_3) = (2, 4, 16)$	3.813	2.953
	$(L_1, L_2, L_3) = (8, 8, 2)$	4.68	3.492
Four-Stage Designs	$(L_1, L_2, L_3, L_4) = (2, 8, 4, 2)$	3.719	2.859
	$(L_1, L_2, L_3, L_4) = (8, 4, 2, 2)$	4.555	3.367

Table 4.2: Computational costs of FIR multi-stage filters in multiplies/output sample

4.3.3 IIR Computational Costs

Again, the cost equations become more complicated for IIR filters. When using a polyphase implementation, each IIR filter is again decomposed into a cascade of an FIR component filter containing only zeros and an IIR component filter containing only poles. The polyphase technique can then be applied to the FIR component, while the IIR component is unchanged.

The cost equations without using a polyphase implementation are similar to those for FIR filters. We recall that each IIR filter of order N actually consists of N poles and N zeros. Hence, the computational cost for an IIR filter of order N will be twice that of an FIR filter of order N. Recalling equations 4.5, 4.7, and 4.9, we conclude

$$\frac{2N_1}{L_2} + 2N_2 \text{ multiplies/output sample for two-stage IIR filter} \quad (4.11)$$

$$\frac{2N_1}{L_2L_3} + \frac{2N_2}{L_3} + 2N_3 \text{ multiplies/output sample for three-stage IIR filter} \quad (4.12)$$

$$\frac{2N_1}{L_2L_3L_4} + \frac{2N_2}{L_3L_4} + \frac{2N_3}{L_4} + 2N_4 \text{ multiplies/output sample for four-stage IIR filter} \tag{4.13}$$

Next consider using a polyphase implementation: Each IIR filter is decomposed into FIR and IIR components, then a polyphase implementation is performed on the FIR component.

$$\frac{N_1}{L_1L_2} + \frac{N_1 + N_2}{L_2} + N_2 \text{ multiplies/output sample for two-stage IIR polyphase filter} \tag{4.14}$$

$$\frac{N_1}{L_1L_2L_3} + \frac{N_1 + N_2}{L_2L_3} + \frac{N_2 + N_3}{L_3} + N_3 \text{ multiplies/output sample for three-stage IIR polyphase filter} \tag{4.15}$$

$$\frac{N_1}{L_1L_2L_3L_4} + \frac{N_1 + N_2}{L_2L_3L_4} + \frac{N_2 + N_3}{L_3L_4} + \frac{N_3 + N_4}{L_4} + N_4 \text{ multiplies/output for four-stage IIR polyphase filter} \tag{4.16}$$

Using these IIR cost equations, we calculate the computational costs of multi-stage IIR Butterworth filters. Recall Tables 3.2, 3.3, and 3.4 to find the orders of these Butterworth filters. The results using polyphase implementations are given in Table 4.3.

	Expander Values	IIR Butterworth
Two-Stage	$(L_1, L_2) = (2, 64)$	7.695
Designs	$(L_1, L_2) = (16, 8)$	8.422
Three-Stage	$(L_1, L_2, L_3) = (2, 4, 16)$	4.242
Designs	$(L_1, L_2, L_3) = (8, 8, 2)$	5.859
Four-Stage	$(L_1, L_2, L_3, L_4) = (2, 8, 4, 2)$	4.945
Designs	$(L_1, L_2, L_3, L_4) = (8, 4, 2, 2)$	6.609

Table 4.3: Computational costs of IIR Butterworth multi-stage filters in multiplies/output sample

Chapter 5

Analysis

Several approaches to designing interpolation systems for integer upsampling have been discussed. We have considered:

- Using various FIR and IIR filters to implement interpolation filters.
- Utilizing the Schuessler Factorization to generate minimum-phase FIR interpolation filters.
- Decomposing interpolation systems into cascaded stages of expanders and low-pass filters.

For these various approaches, the specific system of expanding by a factor of 128, then low-pass filtering to meet the specifications in Table 1.1 has been implemented, to assess the required computational cost. In this chapter, the tradeoffs between these approaches are examined.

5.1 Multiple-Stage Decompositions

The process of decomposing an interpolation system into a cascade of multiple stages was studied. The specific cases of single-stage, two-stage, three-stage, and four-stage systems have been discussed. Table 5.1 gives the minimum computational costs for these decompositions, for each filter design discussed in Chapter 2.

Filter Type	Single-Stage	Two-Stage	Three-Stage	Four-Stage
FIR Kaiser	35.945	4.75	4.141	4.203
FIR Parks-McClellan	30.516	4.438	3.813	3.625
FIR Schuessler	26.102	3.438	2.953	2.766
IIR Butterworth	43.336	5.492	4.242	4.242
IIR Chebyshev	14.109	3.977	3.188	3.141
IIR Elliptical	7.055	3.18	2.953	3.000

Table 5.1: Minimum computational costs for cascaded stages in multiplies/output sample

We observe that the required computation generally decreases as the number of cascaded stages increases. Specifically, the minimum cost for each filter type occurs in either the three-stage or the four-stage design.

Further, the reduction in cost seems to plateau quickly between the two-stage, three-stage, and four-stage designs. In the cases of FIR Kaiser filters and IIR elliptical filters, the computational cost actually increases in the four-stage design. This suggests that significant additional savings would not be observed in cascades of five or more stages. The minimum possible computational costs are likely to be very similar to the cost of three-stage or four-stage systems.

We conclude that interpolation systems designed as cascaded stages of expanders and low-pass filters experience significant computational savings over single-stage designs. Efficient designs of interpolation systems should consist of two or more cascaded expander and filter pairs.

5.2 FIR versus IIR Interpolation Filters

We have considered interpolation systems involving a variety of interpolation filter designs: Parks-McClellan and Kaiser linear-phase FIR filters, Schuessler minimum-phase FIR filters, and Butterworth, Chebyshev, and elliptical IIR filters. These filter designs are grouped into filter classes as follows:

FIR Linear-Phase consists of Parks-McClellan and Kaiser FIR linear-phase filters.

FIR Schuessler consists of Schuessler minimum-phase filters, generated from Parks-McClellan filters using the Schuessler Factorization.

IIR consists of Butterworth, Chebyshev, and elliptical IIR filters.

Any Combination is not restricted to any specific filters, and consists of filters from the FIR linear-phase, FIR Schuessler, and IIR filter classes described above.

Interpolation systems were implemented for each of these filter classes, using single-stage, two-stage, three-stage, and four-stage decompositions. The minimum computational costs for each of these designs are given in Table 5.2.

Number of Stages	FIR Linear-Phase	FIR Schuessler	IIR	Any Combination
Single-Stage	30.516	26.102	7.055	7.055
Two-Stage	4.438	3.438	3.18	2.867
Three-Stage	3.813	2.953	2.953	2.68
Four-Stage	3.625	2.766	3.000	2.461

Table 5.2: Minimum computational costs for each filter class in multiplies/output sample

As expected, Table 5.2 attains the minimum computational costs using three and four-stage cascades, with two-stage designs only slightly less efficient. This is consistent with the observations in Section 5.1.

More interestingly, we observe that by using any combination of filters, the lowest computational costs are attained. The computationally optimal interpolation system designs use both FIR and IIR filters. In a single-stage design, the single IIR filter obviously provides the least computation; but for multiple-stage designs, using any combination of FIR and IIR filters provides approximately 10% savings over FIR Schuessler or strictly IIR designs.

The optimal system consisting of any combination of filters for each multiple-stage decomposition was examined. In each computationally minimal system, the first interpolation filter is IIR elliptical, and all successive filters are FIR Schuessler. For example, the optimal single-stage system contains a single elliptical filter. The optimal four-stage system cascades a single elliptical filter, followed by three Schuessler filters.

Recalling the cascaded filter specifications derived in Chapter 3, we notice that the later filters in a cascaded system have wider transition bands. Specifically, the i^{th} filter H_i in an n-stage cascaded interpolation system $(1 \leq i \leq n)$ has a transition bandwidth $f_s \cdot (\frac{1}{L_i} - \prod_{k=1}^{k=i} \frac{1}{L_k})$. For larger i, the $\prod_{k=1}^{k=i} \frac{1}{L_k}$ term decreases, and the transition bandwidth generally increases, though this is also dependent on the specific L_i expansion factor.

For these filters with wider transition bands, which appear later in cascaded interpolation systems, Schuessler filters can achieve orders comparable to IIR elliptical filters. However, the Schuessler filters also fully utilize polyphase implementations, making them computationally superior to elliptical filters. Therefore, the later filters in optimal cascaded interpolation systems are implemented using Schuessler filters.

In contrast, the first interpolation filter H_1 in each cascaded system has the narrowest transition bandwidth, of $\frac{0.1 f_s}{L_1}$. For this sharp cutoff low-pass filter, IIR filters can achieve relatively low orders, while FIR implementations require filter lengths many times larger. Despite the polyphase advantages of FIR filters, the low order of an IIR elliptical filter makes it computationally beneficial. Therefore, elliptical filters are used to implement the first filter in each optimal cascaded interpolation system.

5.3 Distribution of Expanding Factor

In this section, we consider various distributions of the expansion factor L. Since the overall system expands by a factor of $L = 128 = 2^7$, these seven factors of 2 can be distributed in many different ways between two or more distinct expanders.

To simplify the analysis slightly, consider the six possible distributions of $L = 128$ between two nontrivial expanding units L_1 and L_2, as in a two-stage decomposition. For each of these six possible distributions of the expansion factor, three optimal interpolation systems were designed: one was restricted to using FIR Schuessler filters, one using IIR filters, and the third using any combination of filters. The minimum computational cost for each of these systems is shown in Figure 5-1, where the values of the expanders L_1 and L_2 are given along the horizontal axis.

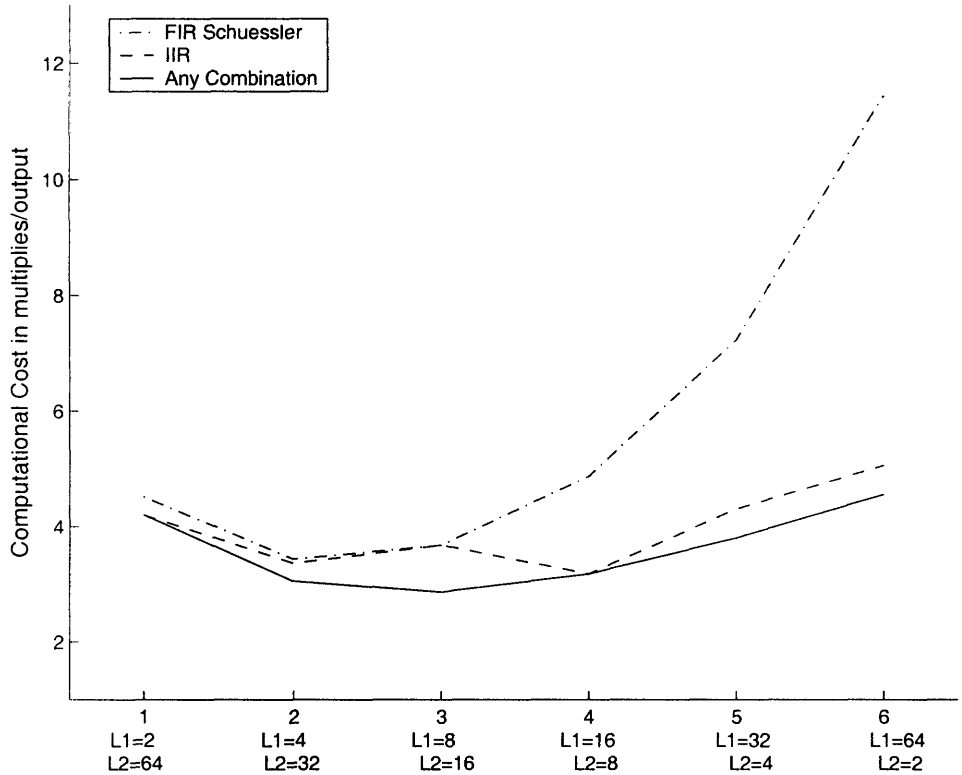

Figure 5-1: Computational costs for each two-stage expander distribution

This graph agrees with the observations in Section 5.2: IIR filters generally outperform FIR Schuessler filters, but a combination of FIR and IIR filters will provide the minimum required computation. We also see that a computational minimum appears around the expanding values $L_1 = 8, L_2 = 16$. Further, for all filter classes the computational cost increases as the expanding factor is pushed toward either extreme, $(L_1 = 2, L_2 = 64)$ or $(L_1 = 64, L_2 = 2)$. An even distribution of the expansion provides the minimum computational requirements.

To understand this result, we consider how distributing the expansion factor affects the cascaded filter specifications. The first filter, H_1, has a transition bandwidth of $\frac{0.1 f_s}{L_1}$. For lower values of L_1 this is a wider bandwidth, and H_1 achieves lower orders. As the expansion factor L_1 increases, this bandwidth reduces by factors of 2, and H_1 requires significantly higher orders.

In contrast, the order of the second low-pass filter H_2 decreases as the first expansion factor L_1 increases. The transition bandwidth of H_2 is $\frac{f_s}{128} \cdot (L_1 - 1)$. As L_1 increases this bandwidth increases proportionally, and the filter order of H_2 decreases. This presents a tradeoff in filter order between H_1 and H_2.

As observed in Figure 5-1, the minimum value of this optimization exists around a relatively equal distribution of the expansion factor: $(L_1 = 8, L_2 = 16)$. Similar, though less intuitive, filter order tradeoffs are witnessed in three-stage and four-stage interpolation systems. In these cascades, the computationally optimal interpolation systems also exhibit a logical distribution of the expansion factor $L = 128$ among the expander units L_i.

Bibliography

[1] *Programs for Digital Signal Processing.* IEEE Press, New York, 1979.

[2] O. Herrmann and W. Schuessler. Design of nonrecursive digital filters with minimum phase. In *Electronics Letters*, volume 6, pages 329–330. Institution of Electrical Engineers, 1970.

[3] Alan V. Oppenheim and Ronald W. Schafer. *Discrete-Time Signal Processing.* Prentice Hall, Upper Saddle River, New Jersey, second edition, 1999.

[4] Alan V. Oppenheim and Alan S. Willsky. *Signals and Systems.* Prentice Hall, Upper Saddle River, New Jersey, second edition, 1997.

[5] T. W. Parks and C. S. Burrus. *Digital Filter Design.* John Wiley and Sons, New York, 1987.

[6] Philips Semiconductors. *DATA SHEET UDA1320ATS*, preliminary specification edition, January 2000.

[7] L. R. Rabiner and B. Gold. *Theory and Application of Digital Signal Processing.* Prentice Hall, Englewood Cliffs, New Jersey, 1975.