Design Of Polynomial-based Filters For Continuously Variable Sample Rate Conversion With Applications In Synthetic Instrumentati

Matthew Hunter
University of Central Florida

DESIGN OF POLYNOMIAL-BASED FILTERS FOR CONTINUOUSLY VARIABLE SAMPLE RATE CONVERSION WITH APPLICATIONS IN SYNTHETIC INSTRUMENTATION AND SOFTWARE DEFINED RADIO

by

MATTHEW T. HUNTER
M.S. University of Central Florida, 2005
B.S. University of Central Florida, 2004

A dissertation submitted in partial fulfillment of the requirements
for the degree of Doctor of Philosophy
in the School of Electrical Engineering and Computer Science
in the College of Engineering and Computer Science
at the University of Central Florida
Orlando, Florida

Summer Term
2008

Major Professor: Wasfy B. Mikhael

ABSTRACT

In this work, the design and application of Polynomial-Based Filters (PBF) for continuously variable Sample Rate Conversion (SRC) is studied. The major contributions of this work are summarized as follows.

First, an explicit formula for the Fourier Transform of both a symmetrical and nonsymmetrical PBF impulse response with variable basis function coefficients is derived. In the literature only one explicit formula is given, and that for a symmetrical even length filter with fixed basis function coefficients.

The frequency domain optimization of PBFs via linear programming has been proposed in the literature, however, the algorithm was not detailed nor were explicit formulas derived. In this contribution, a minimax optimization procedure is derived for the frequency domain optimization of a PBF with time-domain constraints. Explicit formulas are given for direct input to a linear programming routine. Additionally, accompanying Matlab code implementing this optimization in terms of the derived formulas is given in the appendix.

In the literature, it has been pointed out that the frequency response of the Continuous-Time (CT) filter decays as frequency goes to infinity. It has also been observed that when implemented in SRC, the CT filter is sampled resulting in CT frequency response aliasing. Thus, for example, the stopband sidelobes of the Discrete-Time (DT) implementation rise above the CT designed level. Building on these observations, it is shown how the rolloff rate of the frequency response of a PBF can be *adjusted* by adding continuous derivatives to the impulse response. This is of great advantage, especially when the PBF is used for decimation as the aliasing band attenuation can be made to increase with frequency. It is shown how this technique can be used to dramatically reduce the effect of alias build up in the passband. In addition, it is shown that as the number of continuous derivatives of the PBF increases the resulting DT implementation more closely matches the Continuous-Time (CT) design.

When implemented for SRC, samples from a PBF impulse response are computed by evaluating the polynomials using a so-called fractional interval, μ. In the literature, the effect of quantizing μ on the frequency response of the PBF has been studied. Formulas have been derived to determine the number of bits required to keep frequency response distortion below prescribed bounds. Elsewhere, a formula has been given to compute the number of bits required to represent μ to obtain a given SRC accuracy for rational factor SRC. In this contribution, it is shown how these two apparently competing requirements are quite independent. In fact, it is shown that the wordlength required for SRC accuracy need only be kept in the μ generator which is a single accumulator. The output of the μ generator may then be truncated prior to polynomial evaluation. This results in significant computational savings, as polynomial evaluation can require several multiplications and additions.

Under the heading of applications, a new Wideband Digital Downconverter (WDDC) for Synthetic Instruments (SI) is introduced. DDCs first tune to a signal's center frequency using a numerically controlled oscillator and mixer, and then zoom-in to the bandwidth of interest using SRC. The SRC is required to produce continuously variable output sample rates from a fixed input sample rate over a large range. Current implementations accomplish this using a pre-filter, an arbitrary factor resampler, and integer decimation filters. In this contribution, the SRC of the WDDC is simplified reducing the computational requirements to a factor of three or more. In addition to this, it is shown how this system can be used to develop a novel computationally efficient FFT-based spectrum analyzer with continuously variable frequency spans.

Finally, after giving the theoretical foundation, a real Field Programmable Gate Array (FPGA) implementation of a novel Arbitrary Waveform Generator (AWG) is presented. The new approach uses a fixed Digital-to-Analog Converter (DAC) sample clock in combination with an arbitrary factor interpolator. Waveforms created at any sample rate are interpolated to the fixed DAC sample rate in real-time. As a result, the additional lower performance analog hardware required in current approaches, namely, multiple reconstruction filters and/or additional sample clocks, is avoided. Measured results are given confirming the performance of the system predicted by the theoretical design and simulation.

ACKNOWLEDGEMENTS

First and foremost I would like to thank the Triune God of the Bible, the one true God. I praise Him for creating the wondrous complexity, beauty, and harmony of the universe, that He uses to display His supreme majesty. He has given His creation for humanity to explore and discover through research and inquiry. This work was done not only in an effort to advance the state of the art, but to think God's thoughts after Him and to labor towards man's chief end, namely, to glorify God and enjoy Him forever. All glory and credit are due to my Savior and Lord Jesus Christ who is the only way, the truth, and the life.

I would like to thank my beautiful, long-suffering wife who is my soul mate and best friend. She has patiently endured the Saturdays I had to be away from her and my children as I worked on my research. I thank my precious children, Taylor and Audrey, as well for their love and patience with their busy dad. My parents Carl and Laura Hunter as well as my wife's parents, Nick and Audrey Ainsworth, tirelessly gave their prayerful and financial support without which this work probably would not have come to fruition.

My advisor Dr. Wasfy B. Mikhael instilled in me a love for signal processing. He also taught me to always look at the beauty of how all things are related, and that there is nothing new under the sun. The person who inspired me to start my journey into engineering was Professor Greg Dietrich. He would not settle for mediocrity, but challenged me to excel. For these two men I am truly grateful.

Finally, I would like to thank DME corporation, and especially Craig Hughes, the Engineering Director. They gave me the opportunity to apply my research to real world applications and provided financial support for my tuition.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBREVIATIONS

ADC	–	Analog-to-Digital Converter
AWG	–	Arbitrary Waveform Generator
BPF	–	BandPass Filter
CIC	–	Cascaded-Integrator-Comb
CT	–	Continuous-Time
CVSRC	–	Continuously Variable Sample Rate Conversion
DAC	–	Digital-to-Analog Converter
DC	–	Down Converter
DDS	–	Direct Digital Synthesis
DT	–	Discrete-Time
FIR	–	Finite Impulse Response
IF	–	Intermediate Frequency
ISL	–	Integrated Sidelobe Level
LO	–	Local Oscillator
MPIS	–	Multiplications Per Input Sample
MPOS	–	Multiplications Per Output Sample
NCO	–	Numerically Controlled Oscillator
PBF	–	Polynomial-Based Filter
SDR	–	Software Defined Radio
SI	–	Synthetic Instrumentation
SRC	–	Sample Rate Conversion
UC	–	Up Converter
ZOH	–	Zero Order Hold
ZP	–	Zero Phase

CHAPTER 1: INTRODUCTION

1.1 Motivation

The two motivating applications for this work are Synthetic Instrumentation (SI) [1,2,3,4,5,6,7,8] and Software Defined Radio (SDR) [9,10,11,12]. One of the main goals of SDR is to replace as many of the analog and hard-wired digital circuits as possible with programmable devices. This makes a radio (any wireless communication device, e.g. cell phone, walkie-talkie) more flexible in the sense that it can be reconfigured to handle a different type of communication waveform simply by changing its programming. This concept is illustrated in Fig. 1 where the single SDR on the right can handle all of the waveforms produced by the multiple hardware defined radios on the left. This type of reprogrammability is being driven by many factors including the desire for multi-mode terminals — we want bluetooth and wireless internet in the same device — as well as the major problem of different branches of the military not being able to communicate because they have different radios.

The SI movement takes the same approach in that it seeks to use flexible Digital Signal Processing (DSP) based architectures to provide many functions on a single platform. Multiple measurement functions can be synthesized from a limited set of "generic" SI components as opposed to discrete instrument types such as a spectrum analyzer [13]. This concept is shown in Fig. 2, where the SI platform on the right performs the same functions as the discrete instruments on the left. In the figure, the example SI platform consists of a monitor for display on top of a card

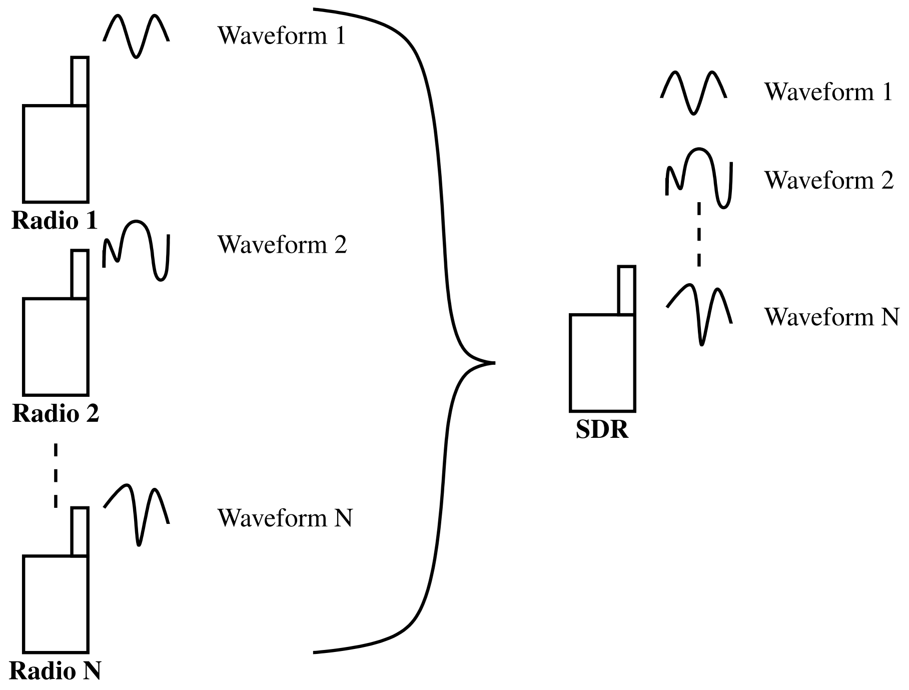

Figure 1: Software Defined Radio Concept

cage housing the generic SI components. Examples of generic SI components include frequency downconverters, digitizers, and frequency upconverters.

The differences between SDR and SI are quite small in the area of RF and communications instrumentation, such as the Vector Signal Analyzer (VSA) and Vector Signal Generator (VSG). In fact the SI implementation of the VSA can be thought of as a gold standard SDR receiver [7]. Hence, advances in SDR can be taken advantage of in the development of new SI's and vice-versa. In both areas, DSP plays a prominent role. One of the major tasks being Sample Rate Conversion (SRC) or resampling [14, 15]. SDR's and SI's use sampled data modems. That is, an incoming signal passes through an Analog to Digital Converter (ADC) sampling at a fixed rate before the information symbols are recovered. In order to recover the symbols, the data must be sampled

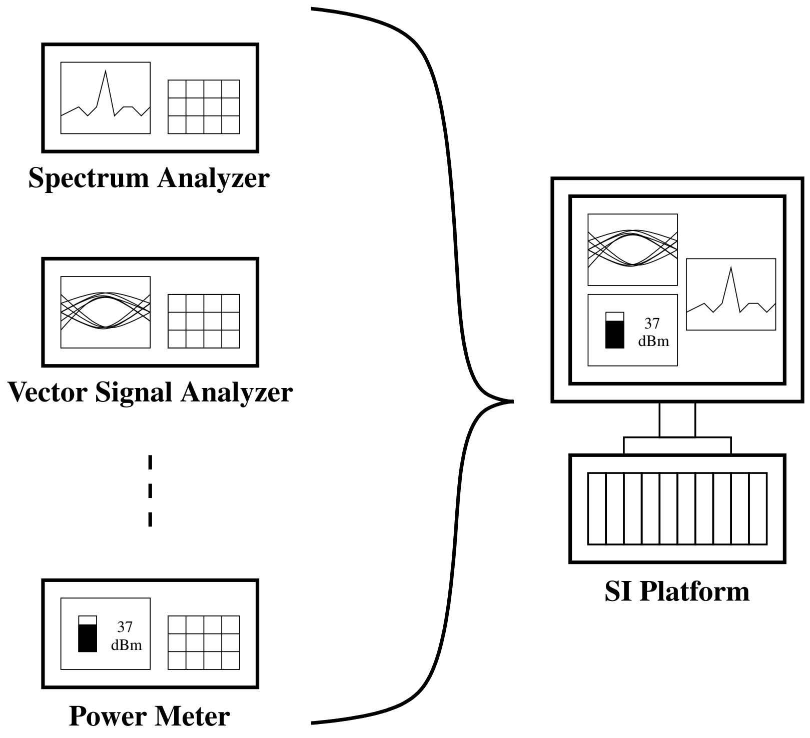

Figure 2: Synthetic Instrument Concept

at the symbol rate or an integer multiple thereof. Since different communications standards use different symbol rates, a method is needed to adjust the sample rate entirely in the discrete-time domain. Moreover, since the sample rate may not be an integer multiple of the symbol rate, the SRC must be able to convert between *arbitrary* sampling rates [16, 17, 18]. Another SI application is FFT-based spectral analysis. Here, it's desired to compute the spectrum of a signal over a certain bandwidth or span using FFT techniques. In order to make the choice of spans continuously variable (like in a traditional spectrum analyzer), continuously variable SRC is needed [5]. There are many other applications of continuously variable SRC including image zooming [19], digital audio resampling [20], reconstruction of non-uniformly sampled data [21,22], and continuous-time signal processing [23].

1.2 Introduction to Sample Rate Conversion

A CT signal, $x_a(t)$, can be perfectly reconstructed from its samples given by

$$x(kT_s) = x_a(t)\Big|_{t=kT_s}, \tag{1.1}$$

where T_s is the sampling period in seconds and

$$F_s = \frac{1}{T_s} \tag{1.2}$$

is the sample rate in Hz. Given that $x_a(t)$ has a strictly bandlimited Fourier Transform $X_a(F)$, or

$$\Big|X_a(F)\Big| = 0, \quad \text{for} \quad |F| > B. \tag{1.3}$$

The sample rate required to perfectly reconstruct $x_a(t)$ from its samples, $x(kT_s)$, must follow

$$F_s \geq 2B \tag{1.4}$$

This is the sampling theorem [24]. The constraint in 1.4 prevents aliasing [25]. In practice, the sample rate must, in most cases, be slightly greater than the minimum bound given in (1.4). Extensions of the sampling theorem to other types of signals, e.g. bandpass signals, can be found in [26, 27, 28].

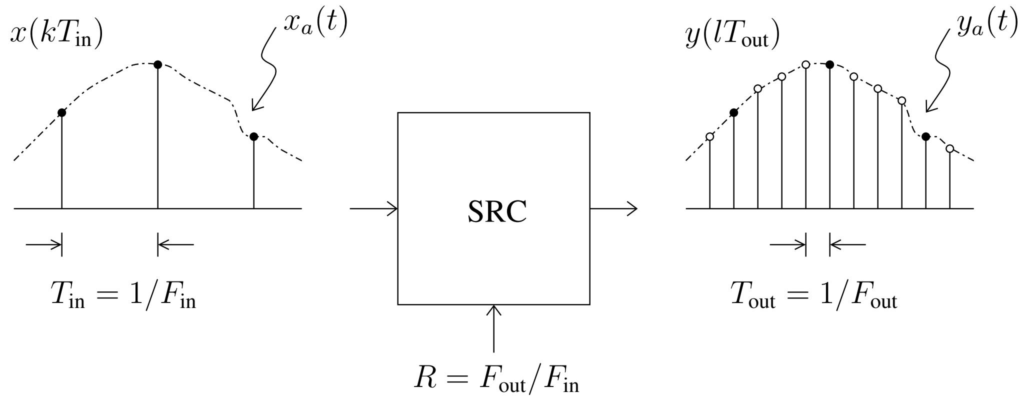
$x(kT_{\text{in}})$
$x_a(t)$
$T_{\text{in}} = 1/F_{\text{in}}$
SRC
$R = F_{\text{out}}/F_{\text{in}}$
$y(lT_{\text{out}})$
$y_a(t)$
$T_{\text{out}} = 1/F_{\text{out}}$

Figure 3: Sample Rate Conversion

Figure 3 depicts the sample rate *conversion* of a discrete-time input signal, $x(kT_{\text{in}})$, having sample rate F_{in} and corresponding sample period T_{in}. The output of the system is the signal $y(lT_{\text{out}})$ having sample rate F_{out} and corresponding sample period T_{out}. Also, $x_a(t)$ is the underlying CT signal from which the samples $x(kT_{\text{in}})$ are generated, while $y_a(t)$ is the underlying CT signal from which the output samples $y(lT_{\text{out}})$ are generated.

The sample rate conversion factor is given by

$$R = F_{\text{out}}/F_{\text{in}}, \tag{1.5}$$

such that the output sample rate is given by,

$$F_{\text{out}} = RF_{\text{in}}. \tag{1.6}$$

When $R > 1$ the SRC process is referred to as interpolation, while the $0 < R < 1$ case is referred to as decimation.

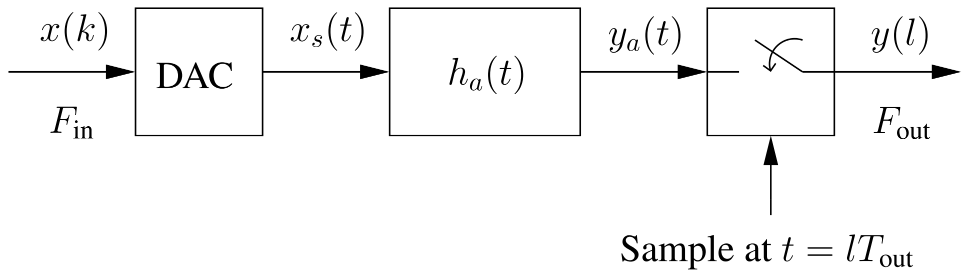

Figure 4: CT Model for Sample Rate Conversion

The process of SRC just described can be modeled with the system in Fig. 4 [24,29,30,31, 32]. In this model, the input samples are reconstructed to form a CT signal. The CT signal is then resampled at a new sampling rate. In Fig. 4, the samples to be reconstructed are input to an ideal DAC followed by filtering with $h_a(t)$, the CT reconstruction filter. The output of the ideal DAC is the impulse train given by

$$x_s(t) = \sum_{k=-\infty}^{\infty} x(kT_{\text{in}})\delta_a(t - kT_{\text{in}}) \tag{1.7}$$

where $\delta_a(t)$ is the Dirac delta function. The next step is to determine how the reconstruction filter should be designed. Taking the Fourier Transform of (1.7) provides some insight, and is given by

$$X_s(F) = \frac{1}{T_{\text{in}}} \sum_{k=-\infty}^{\infty} X_a(F + kF_{\text{in}}) \tag{1.8}$$

From (1.8) it can be seen that the spectrum now contains the original spectrum in addition to an infinite number of spectral replicas centered at integer multiples of the sample rate. These replicas are referred to as images and represent distortion. Therefore, to reconstruct the original signal, one simply needs to design $h_a(t)$ to remove the images and preserve the original lowpass spectrum.

To summarize the filtering requirements for signal reconstruction, given that the signal follows (1.3), the filter should approximate

$$H_a(F) = \begin{cases} 1, & \text{for} \quad |F| \le B \\ 0, & \text{for} \quad kF_{\text{in}} - B < |F| < kF_{\text{in}} + B \\ \text{don't care}, & \text{otherwise} \end{cases}, \tag{1.9}$$

where k is an integer ranging from $-\infty$ to ∞. Figure 5 illustrates the reconstruction process from a spectral point of view.

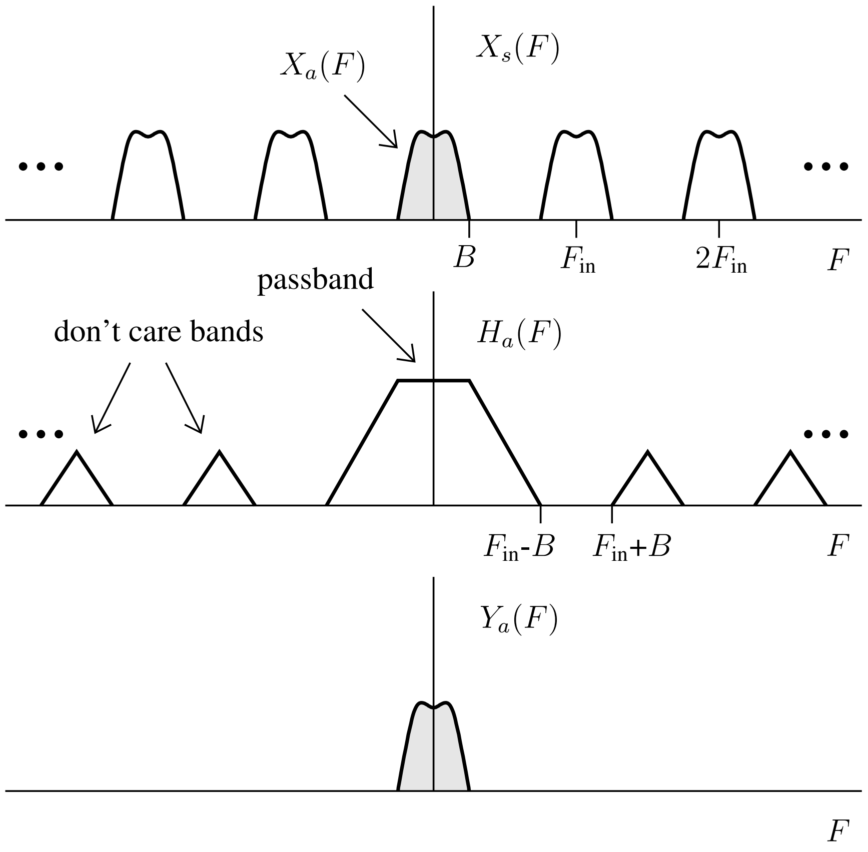

Figure 5: Signal Reconstruction

Returning to the analysis of Fig. 4, the CT output of the filter is given by

$$\begin{aligned} y_a(t) &= \int_{-\infty}^{\infty} x_s(\lambda) h_a(t-\lambda) d\lambda \\ &= \sum_{k=-\infty}^{\infty} x(kT_{\text{in}}) \int_{-\infty}^{\infty} \delta_a(\lambda - kT_{\text{in}}) h_a(t-\lambda) d\lambda \\ &= \sum_{k=-\infty}^{\infty} x(kT_{\text{in}}) h_a(t - kT_{\text{in}}) \end{aligned} \tag{1.10}$$

The CT signal is then resampled at the desired times $t = lT_{\text{out}}$ yielding the Discrete-Time (DT) signal output

$$y(lT_{\text{out}}) = \sum_{k=-\infty}^{\infty} x(kT_{\text{in}}) h_a(lT_{\text{out}} - kT_{\text{in}}). \tag{1.11}$$

This result shows that the computation of the resampled output signal only requires the DT input signal $x(kT_{\text{in}})$, and *samples* from the CT impulse response $h_a(t)$ [24, 29, 30]. No explicit Digital-to-Analog or Analog-to-Digital Conversion is required. All that remains is to design a CT filter such that its samples can be readily computed. Once this is accomplished, the entire resampling operation can take place in the DT domain. Fast, on-line computation of the samples required to perform the resampling operation in (1.11) is readily accomplished by constructing $h_a(t)$ as a piecewise polynomial as shown in Chapter 3.

When $F_{\text{out}} > F_{\text{in}}$ reconstruction of the original signal is accomplished by simply removing the images as shown in Fig. 5. When the signal is resampled, no aliasing will occur, because the new sample rate is higher than the original sample rate. The type of filter that accomplishes this is called an anti-imaging filter. Since the images to be removed reside at integer multiples of the input sample rate, then the filter to remove them must have stopbands at these integer multiples as given in 1.9.

In like manner, when $F_{\text{out}} < F_{\text{in}}$ the filter should remove the spectral images. However, because the output sample rate is less than the original sample rate, it must also be protect the bandwidth of interest from aliasing. Frequency bands centered at multiples of the *output* sample rate will alias into the baseband as a result of sampling. Therefore, the filter should have stopbands located at integer multiples of the output sample rate. Given that the signal to protect is lowpass of single sided bandwidth W, the filtering requirement to prevent aliasing is given by

$$H_a(F) = \begin{cases} 1, & \text{for} \quad |F| \leq W \\ 0, & \text{for} \quad kF_{\text{out}} - W < |F| < kF_{\text{out}} + W \\ \text{don't care,} & \text{otherwise} \end{cases} \cdot \tag{1.12}$$

This type of filtering protects the desired band from aliasing while allowing it in the don't care bands. Alternatively, if no aliasing is allowed the filtering requirement becomes

$$H_a(F) = \begin{cases} 1, & \text{for} \quad |F| \leq W \\ 0, & \text{for} \quad |F| > F_{\text{out}}/2 \end{cases} . \tag{1.13}$$

CHAPTER 2: LITERATURE REVIEW

Digital methods for SRC by integer factors have been studied extensively [25, 33, 24]. This type of SRC falls under two headings: decimation and interpolation. Decimation is the term used for a reduction of the sample rate by an integer factor while interpolation is used for sample rate increase by an integer factor. Structures for efficient realization of integer sample rate conversion are the polyphase filters [34], and, in cases where the signal bandwith is small with respect to the sample rate, the cascaded-integrator-comb (CIC) filters [35]. Interpolation followed by decimation results in SRC by a rational factor. Here the sample rate is first increased by a factor L before being reduced by a factor K to yield the over all SRC factor $R = L/K$. When R is a ratio of two small integers, this operation can be performed efficiently by a single polyphase filter [36]. This technique becomes inefficient when R is a ratio of two large relatively prime integers [18], or when the SRC factor is to be continuously variable.

Upon observing the continuous-time (CT) model for sample rate conversion, it was discovered that the resampling of a signal can be accomplished entirely in the discrete-time (DT) domain [29, 37]. All that is needed is the sequence to be resampled and samples from a CT impulse response used for signal reconstruction. The question then becomes: How does one design such a CT impulse response such that samples from it can be readily computed? Initially, this idea led to the use of classical piecewise polynomial interpolation kernels (e.g. Lagrange) for the CT impulse response [29, 30]. These kernels are attractive because they have explicit formulas for the coefficients, can be evaluated at the proper time instants via a single parameter, and can be implemented in real-time with the Farrow Structure [38, 39]. The drawback to this approach is that the frequency response of the CT filter is governed by the choice of interpolation kernel. This is quite restrictive from a filter design point of view, especially since these kernels often have an unsuitable frequency response for many applications. The solution to this problem came in [40, 31], where it was realized that the zero phase frequency response of a symmetric Polynomial-Based Filter (PBF) is linear with respect to the coefficients. This meant that the coefficients could be optimized to meet a desired frequency response via linear programming [41, 42]. Driven by the Farrow Structure implementation, the PBF impulse response had been constructed with polynomial pieces with length equal to the input sample period T_{in}. This yields a frequency response normalized to the input sample rate $F_{\text{in}} = 1/T_{\text{in}}$. This is what one wants in the case of sample rate increase, but not for sample rate decrease [5]. This was solved by constructing the PBF impulse response with pieces having length equal to the output sample period T_{out} yielding new implementation structures [43, 44]. These fall under the heading of the Transposed Farrow Structure. More general structures have also been proposed which are constructed of PBF's with pieces having different lengths [45], or equal lengths of a multiple of the input or output sample period. These techniques can reduce the number of fixed coefficients needed to implement a desired frequency response at the expense of additional general purpose multipliers needed for polynomial evaluation [46]. Other strucutres based on CIC filters have also been developed which take advantage of oversampling [47, 18].

In practical implementation, the SRC factor is always approximated by a ratio of two, not necessarily small, relatively prime integers. This insight led to the observation that for a particular SRC factor $R = L/K$, there exists a polyphase filter implementation exactly equivalent to the

PBF implementation [48]. When SRC is performed using a PBF the CT impulse response is uniformly sampled yielding a DT FIR filter equivalent. Now, filters could be designed by first designing a DT filter and then converting it to a polynomial-based filter [48]. The advantage to this approach is that in the DT model, filter aliasing is taken into account, thus the designed stopband attenuation is exactly the same as in implementation. When the CT-designed impulse response is sampled, aliasing occurs, thus the actual stopband sidelobes may rise far above the designed level. The disadvantage to the DT-design method is that it only optimizes the filter for a particular rate change, and the performance for other rate changes severely degrades to an unacceptable level. Thus, in [49] it was concluded that for applications where the SRC factor is to be continuously variable the CT design method is sufficient.

The effect of quantizing the fractional interval on the frequency response of the PBF has also been studied in [50, 51]. It was shown that fractional interval quantization effectively applies a Zero Order Hold (ZOH) to the impulse response. This causes image distortion in the frequency response. The relationship between the level of the distortion images caused by quantization and the number of bits used was derived.

CHAPTER 3: DESIGN OF POLYNOMIAL-BASED FILTERS

3.1 Introduction to Polynomial-Based Filters

Polynomial-Based Filters (PBF) are continuous-time finite duration impulse response filters. As the name suggests PBFs are *polynomial-based*, that is, they are constructed of polynomials. To be more precise, PBFs are composed of concatenated (arranged side by side) piecewise polynomial segments of equal length. To introduce this concept consider the following two polynomial pieces.

$$h_0(t) = \begin{cases} t/T, & 0 \leq t < T, \\ 0, & \text{otherwise.} \end{cases} \tag{3.1}$$

$$h_1(t) = \begin{cases} 1 - t/T, & 0 \leq t < T, \\ 0, & \text{otherwise.} \end{cases} \tag{3.2}$$

Equations (3.1) and (3.2) are equal length, linear polynomials in t. Concatenation is performed by shifting $h_1(t)$ to the right and adding the two pieces as follows.

$$h_a(t) = h_0(t) + h_1(t - T) \tag{3.3}$$

This example produces the linear interpolation kernel. A graphical illustration of this process is given in Fig. 6.

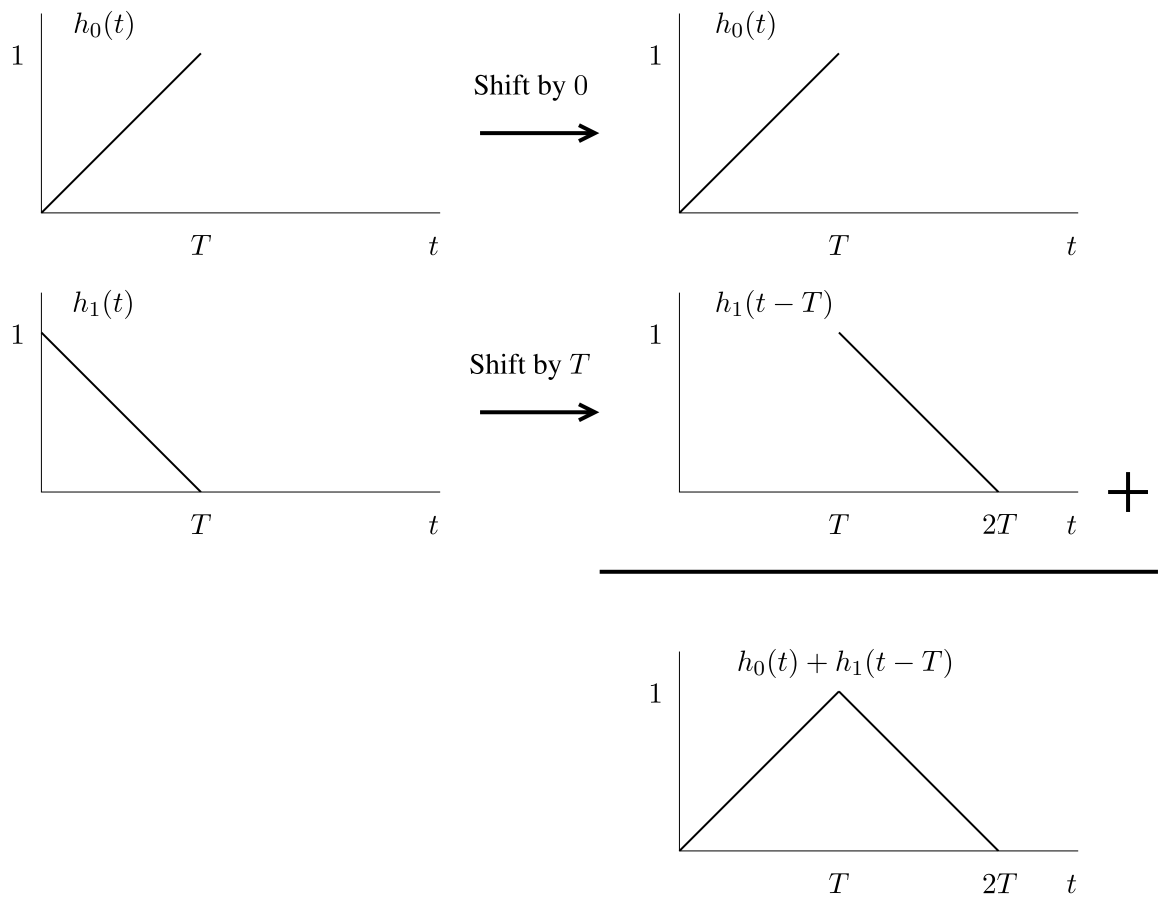

Figure 6: Construction of a PBF

3.2 Impulse Response

In general, $h_a(t)$ can be constructed with N concatenated polynomial pieces of degree M and length T as follows [43, 44, 18, 46]. Define a polynomial piece as

$$h_n(t) = \begin{cases} \sum_{m=0}^{M} c_m(n) \left(a\left(\frac{t}{T}\right) + b\right)^m, & 0 \le t < T, \\ 0, & \text{otherwise.} \end{cases} \tag{3.4}$$

Equation (3.4) describes an Mth order polynomial in t of length T. The $c_m(n)$'s are the polynomial coefficients and a and b are constants. The overall impulse response is given by the summation of the shifted pieces

$$\begin{aligned} h_a(t) &= \sum_{n=0}^{N-1} h_n(t - nT) \\ &= \sum_{n=0}^{N-1} \sum_{m=0}^{M} c_m(n)\psi_m(n, T, t), \end{aligned} \tag{3.5}$$

where

$$\psi_m(n, T, t) = \begin{cases} \left(a\left(\frac{t-nT}{T}\right) + b\right)^m & nT \le t < (n+1)T, \\ 0 & \text{otherwise.} \end{cases} \tag{3.6}$$

are the basis functions.

3.3 Frequency Response

Given a PBF consisting of N polynomial pieces of order M,

$$h_a(t) = \sum_{n=0}^{N-1} \sum_{m=0}^{M} c_m(n) \psi_m(n, T, t), \tag{3.7}$$

where,

$$\psi_m(n, T, t) = \begin{cases} \left(a\left(\frac{t-nT}{T}\right) + b\right)^m, & nT \leq t < (n+1)T, \\ 0, & \text{otherwise.} \end{cases} \tag{3.8}$$

The Fourier Transform is given by

$$\begin{aligned} H_a(\Omega) &= \sum_{n=0}^{N-1} \int_{nT}^{(n+1)T} \left(\sum_{m=0}^{M} c_m(n) \psi_m(n, T, t) \right) e^{-j\Omega t} dt \\ &= \sum_{n=0}^{N-1} \sum_{m=0}^{M} c_m(n) \int_{nT}^{(n+1)T} \psi_m(n, T, t) e^{-j\Omega t} dt \\ &= \sum_{n=0}^{N-1} \sum_{m=0}^{M} c_m(n) \Psi_m(n, T, \Omega), \end{aligned} \tag{3.9}$$

where,

$$\begin{aligned} \Psi_m(n, T, \Omega) &= \int_{nT}^{(n+1)T} \psi_m(n, T, t) e^{-j\Omega t} dt \\ &= \int_{nT}^{(n+1)T} \left(a\left(\frac{t-nT}{T}\right) + b \right)^m e^{-j\Omega t} dt. \end{aligned} \tag{3.10}$$

Making a change of variables, we let

$$\begin{aligned} u &= -j\Omega t - j\Omega T \frac{b}{a} + j\Omega nT \\ \rightarrow du &= -j\Omega dt. \end{aligned} \tag{3.11}$$

This gives,

$$\begin{aligned} t =& -\frac{u}{j\Omega} - \frac{b}{a}T + nT \\ \rightarrow dt =& -\frac{du}{j\Omega}. \end{aligned} \tag{3.12}$$

The new integration limits become

$$\begin{aligned} u_1 =& -j\Omega t_1 - j\Omega T\frac{b}{a} + j\Omega nT = -j\Omega T\frac{b}{a} \\ u_2 =& -j\Omega t_2 - j\Omega T\frac{b}{a} + j\Omega nT = -j\Omega T\left(1 + \frac{b}{a}\right). \end{aligned} \tag{3.13}$$

Substituting (3.11), (3.12), and (3.13) into (3.10) gives

$$\begin{aligned} \Psi_m(n,T,\Omega) =& \int_{u_1}^{u_2} \left(\frac{au}{-j\Omega T}\right)^m e^u e^{-j\Omega nT} e^{j\Omega T\frac{b}{a}} \frac{du}{-jw} \\ =& \frac{e^{-j\Omega nT} a^m e^{j\Omega T\frac{b}{a}}}{-j\Omega(-j\Omega T)^m} \int_{u_1}^{u_2} u^m e^u du \\ =& \frac{e^{-j\Omega nT} a^m e^{j\Omega T\frac{b}{a}}}{-j\Omega(-j\Omega T)^m} \left[e^u \sum_{k=0}^{m} (-1)^k \frac{m!}{(m-k)!} u^{m-k} \Big|_{u1}^{u2} \right] \\ =& \frac{e^{-j\Omega nT} a^m e^{j\Omega T\frac{b}{a}}}{-j\Omega(-j\Omega T)^m} \left[\sum_{k=0}^{m} (-1)^k \frac{m!}{(m-k)!} \left(e^{u_2} {u_2}^{m-k} - e^{u_1} {u_1}^{m-k} \right) \right]. \end{aligned} \tag{3.14}$$

Simplifying (3.14) gives

$$\Psi_m(n,T,\Omega) = Te^{-j\Omega nT} \sum_{k=0}^{m} \frac{a^k m!}{(m-k)!} \left(\frac{1}{j\Omega T}\right)^{k+1} \Big(b^{m-k} - e^{-j\Omega T}(a+b)^{m-k} \Big). \tag{3.15}$$

The final form is then given by

$$H_a(\Omega) = \sum_{n=0}^{N-1} \sum_{m=0}^{M} c_m(n) \Psi_m(n,T,\Omega), \tag{3.16}$$

where $\Psi_m(n,T,\Omega)$ is given by (3.15).

A more useful form for design purposes is the normalized frequency response given by

$$\begin{aligned} H_a(\omega) &= \left. \frac{H_a(\Omega)}{T} \right|_{\Omega=\omega/T} \\ &= \sum_{n=0}^{N-1} \sum_{m=0}^{M} c_m(n) \Psi_m(n,\omega), \end{aligned} \tag{3.17}$$

where

$$\begin{aligned} \Psi_m(n,\omega) &= \left. \frac{\Psi_m(n,T,\Omega)}{T} \right|_{\Omega=\omega/T} \\ &= e^{-j\omega n} \sum_{k=0}^{m} \frac{a^k m!}{(m-k)!} \left(\frac{1}{j\omega} \right)^{k+1} \Big(b^{m-k} - e^{-j\omega}(a+b)^{m-k} \Big). \end{aligned} \tag{3.18}$$

3.3.1 Linear Phase Filters

Linear phase filters have the property of impulse response symmetry. This condition can be imposed on the PBF in the design process by restricting the impulse response to be symmetric. This has the added benefit of giving a *real-valued* zero-phase frequency response, which eases the task of PBF design in the frequency domain. The causal PBF $h_a(t)$ centered at $t = NT/2$ is symmetric if

$$h_a(t) = h_a(NT - t). \tag{3.19}$$

Equations (3.7) and (3.8) may be written in terms of each polynomial segment, $h_n(t)$, as

$$h_a(t) = \sum_{n=0}^{N-1} h_n(t - nT), \tag{3.20}$$

where

$$h_n(t) = \begin{cases} \sum_{m=0}^{M} c_m(n) \left(a\left(\frac{t}{T}\right) + b\right)^m, & 0 \leq t < T, \\ 0, & \text{otherwise.} \end{cases} \tag{3.21}$$

Equation (3.19) is satisfied if

$$h_n(t) = h_{N-n-1}(T - t), \tag{3.22}$$

where,

$$\begin{cases} n = 0, 1, \ldots, \frac{N-1}{2}, & \text{for } N \text{ odd} \\ n = 0, 1, \ldots, \frac{N}{2} - 1, & \text{for } N \text{ even.} \end{cases} \tag{3.23}$$

Substituting Eq. (3.4) into Eq. (3.22) gives

$$\begin{aligned}
\sum_{m=0}^{M} c_m(n)\left(a\left(\frac{t}{T}\right)+b\right)^m &= \sum_{m=0}^{M} c_m(N-n-1)\left(a\left(\frac{T-t}{T}\right)+b\right)^m \\
&= \sum_{m=0}^{M} c_m(N-n-1)\left(-a\left(\frac{t}{T}\right)+b+a\right)^m \\
&= \sum_{m=0}^{M} (-1)^m c_m(N-n-1)\left(a\left(\frac{t}{T}\right)-b-a\right)^m .
\end{aligned}$$

Letting

$$a = -2b, \tag{3.24}$$

gives

$$\sum_{m=0}^{M} c_m(n)\left(a\left(\frac{t}{T}\right)+b\right)^m = \sum_{m=0}^{M} (-1)^m c_m(N-n-1)\left(a\left(\frac{t}{T}\right)+b\right)^m , \tag{3.25}$$

such that,

$$c_m(n) = (-1)^m c_m(N-n-1). \tag{3.26}$$

The coefficient symmetry given in (3.26) has the major implementation benefit of reducing the required number of coefficient multiplications.

Given that the symmetry condition is imposed, the zero phase PBF is obtained by shifting the center of the causal impulse response to the origin as follows

$$h_{\text{zp}}(t) = h_a(t + NT/2). \tag{3.27}$$

Because the impulse response is now symmetric about the origin, it is by definition an even function giving

$$h_{\text{zp}}(t) = \frac{1}{2}\Big[\, h_{\text{zp}}(t) + h_{\text{zp}}(-t)\,\Big]. \tag{3.28}$$

Now,

$$H_{\text{zp}}(\Omega) = e^{j\frac{\Omega NT}{2}} H_a(\Omega), \tag{3.29}$$

but, from (3.28), (3.29) can be written as

$$\begin{aligned} H_{\text{zp}}(\Omega) &= \frac{1}{2}\Big[\, H_{\text{zp}}(\Omega) + H^*_{\text{zp}}(\Omega)\,\Big] \\ &= \frac{1}{2}\Big[\, 2\Big(\, \Re\Big\{ H_{\text{zp}}(\Omega)\,\Big\}\,\Big)\,\Big] \\ &= \Re\Big\{\, e^{j\frac{\Omega NT}{2}} H_a(\Omega)\,\Big\}. \end{aligned} \tag{3.30}$$

Finally, substituting (3.17) into (3.30) gives

$$H_{\text{zp}}(\Omega) = \sum_{n=0}^{N-1} \sum_{m=0}^{M} c_m(n) \Re\Big\{\, e^{j\Omega\frac{NT}{2}} \Psi_m(n, T, \Omega)\,\Big\}, \tag{3.31}$$

which for N even can be reduced to

$$H_{\text{zp}}(\Omega) = 2 \sum_{n=0}^{\frac{N}{2}-1} \sum_{m=0}^{M} c_m(n) \Re\Big\{\, e^{j\Omega\frac{NT}{2}} \Psi_m(n, T, \Omega)\,\Big\}. \tag{3.32}$$

The key observation at this point is that because the frequency response is real-valued (i.e. we do not need to take the magnitude), it is linear with respect to the coefficients $c_m(n)$. Because of this, linear programming can be used to optimize the coefficients to achieve a desired frequency response.

3.3.2 Example

Before turning to PBF optimization, we pause to illustrate the use of the formulas via the simple example given in the introduction to this chapter. The example PBF from (3.3) has $N = 2$ polynomial pieces of order $M = 1$. Each piece can be mapped to (3.4) with $a = 1$ and $b = 0$ as follows,

$$h_0(t) = \begin{cases} \sum_{m=0}^{1} c_m(0) \left(\frac{t}{T}\right)^m, & 0 \leq t < T, \\ 0, & \text{otherwise.} \end{cases} \tag{3.33}$$

$$h_1(t) = \begin{cases} \sum_{m=0}^{1} c_m(1) \left(\frac{t}{T}\right)^m, & T \leq t < 2T, \\ 0, & \text{otherwise.} \end{cases} \tag{3.34}$$

The overall impulse response is given by the summation

$$\begin{aligned} h_a(t) &= \sum_{n=0}^{1} h_n(t - nT) \\ &= \sum_{n=0}^{1} \sum_{m=0}^{1} c_m(n) \psi_m(n, T, t). \end{aligned} \tag{3.35}$$

The coefficients can be expressed compactly using a coefficient matrix given by

$$\mathbf{C}_{N \times (M+1)} = \begin{pmatrix} c_0(0) & c_1(0) & \cdots & c_M(0) \\ c_0(1) & c_1(1) & \cdots & c_M(1) \\ \vdots & \vdots & & \vdots \\ c_0(N-1) & c_1(N-1) & \cdots & c_M(N-1) \end{pmatrix}. \tag{3.36}$$

For this example (3.36) becomes

$$\mathbf{C} = \begin{pmatrix} 0 & 1 \\ 1 & -1 \end{pmatrix} \tag{3.37}$$

The frequency response can be determined by using (3.15) with $a = 1$ and $b = 0$ giving

$$\Psi_m(n, T, \Omega) = Te^{-j\Omega nT} \sum_{k=0}^{m} \frac{m!}{(m-k)!} \left(\frac{1}{j\Omega T}\right)^{k+1} \left(0^{m-k} - e^{-j\Omega T} \right). \tag{3.38}$$

Then, substituting (3.38) into (3.17) gives

$$\begin{aligned} H_a(\Omega) &= \sum_{n=0}^{1} \sum_{m=0}^{1} c_m(n) \Psi_m(n, T, \Omega) \\ &= \sum_{n=0}^{1} \Big[c_0(n)\Psi_0(n, T, \Omega) + c_1(n)\Psi_1(n, T, \Omega) \Big] \\ &= c_0(0)\Psi_0(0, T, \Omega) + c_1(0)\Psi_1(0, T, \Omega) + c_0(1)\Psi_0(1, T, \Omega) + c_1(1)\Psi_1(1, T, \Omega) \\ &= \Psi_1(0, T, \Omega) + \Psi_0(1, T, \Omega) - \Psi_1(1, T, \Omega) \\ &= Te^{-j\Omega T} \left(\frac{\sin\left(\frac{\Omega T}{2}\right)}{\frac{\Omega T}{2}} \right)^2 . \end{aligned} \tag{3.39}$$

This is the expected result of the Fourier transform of a triangle function [52]. Finally, the zero phase frequency response is obtained by shifting the center of the impulse response to the origin. Since this filter is symmetric, it is linear phase and has a real-valued zero-phase frequency response given by

$$\begin{aligned} H_{\text{lin}}(\Omega) &= e^{j\Omega T} H_a(\Omega) \\ &= T \left(\frac{\sin\left(\frac{\Omega T}{2}\right)}{\frac{\Omega T}{2}} \right)^2 . \end{aligned} \tag{3.40}$$

Here, we use the subscript "lin" as this is the frequency response of the so called *linear interpolator*. Equation (3.40) is plotted in Fig. 7.

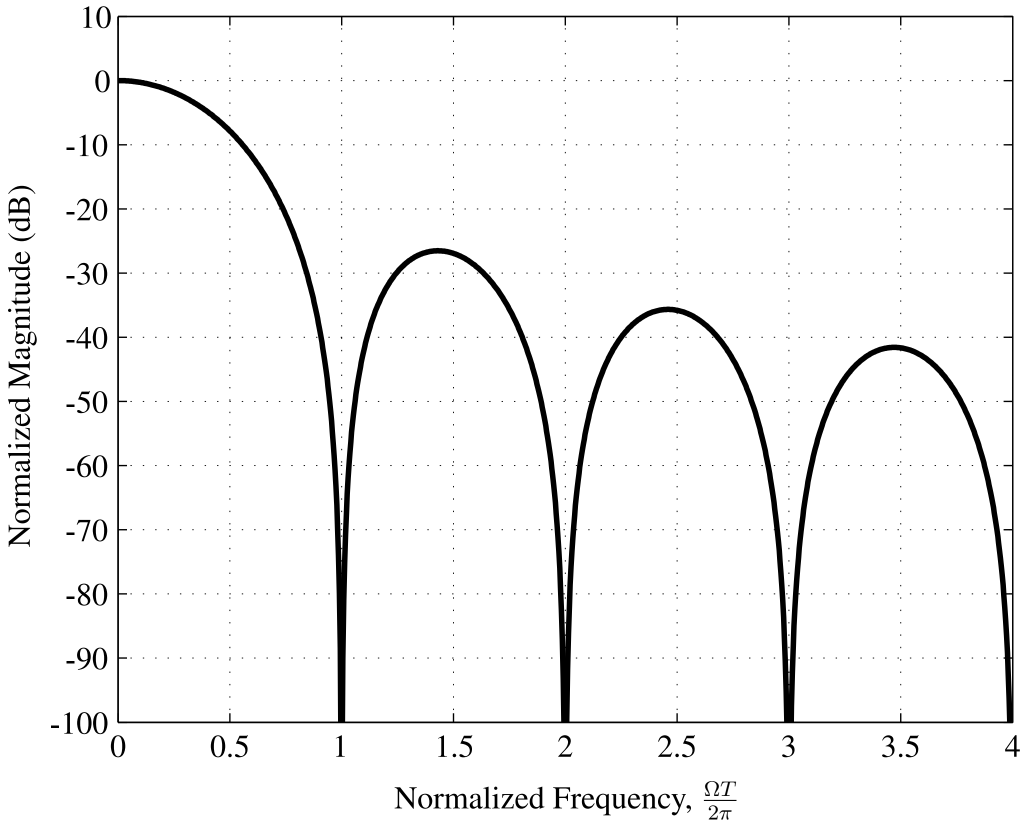

Figure 7: Frequency Response of Linear Interpolator

The linear interpolator is actually a B-spline function, see [53, 54, 55] for the background on B-splines. A B-spline of order M, denoted $\beta^M(t)$, is obtained by the M-fold convolution of the rectangle function with itself. Thus, a zeroth order B-spline is a rectangle function (also called the nearest neighbor interpolator), the first order B-spline is a triangle function, and so on. The repeated convolution of length T rectangle functions yields an Mth order impulse response of length $NT = (M+1)T$. For $h_a(t) = \beta^M(t)$, the coefficients are given by,

$$c_m(n) = \sum_{k=0}^{n} (-1)^k \binom{N}{k} \frac{(n-k)^{M-m}}{(M-m)!M!}, \tag{3.41}$$

for $m = 0, 1, \ldots, M$ and $n = 0, 1, \ldots, N-1$, where $N = M + 1$.

The impulse and frequency response of the B-spline functions up to order $M = 4$ are given in Fig. 8 and Fig. 9, respectively.

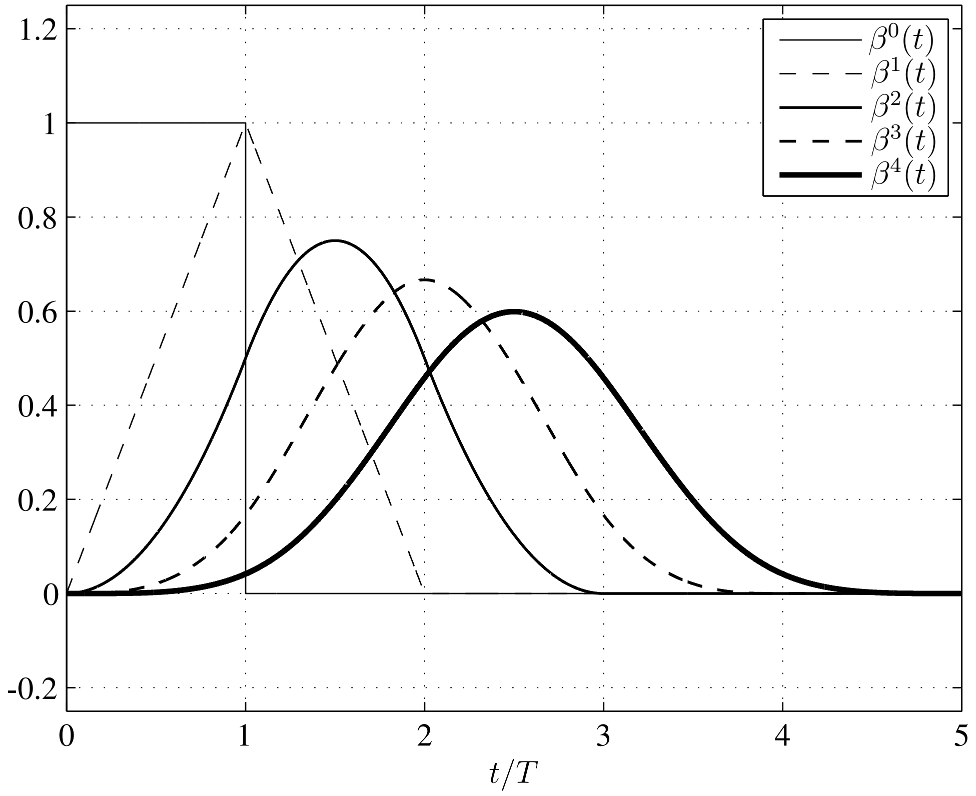

Figure 8: Impulse Response of B-spline Functions

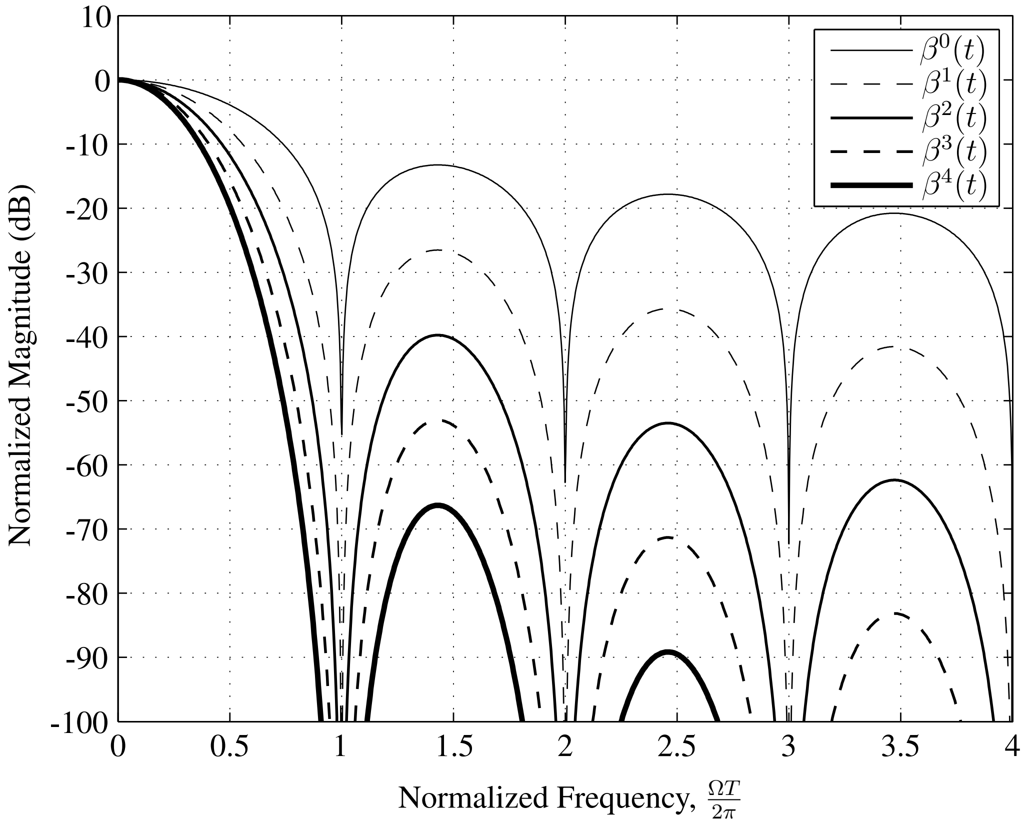

Figure 9: Frequency Response of B-spline Functions

3.3.3 The Matrix Formulation

Before proceeding with design techniques, it will be useful to develop some more notation. First, it should be pointed out that evaluating each polynomial piece, $h_n(t)$ over the interval $t \in [0, T)$ before shifting is equivalent to evaluating the shifted polynomial, $h_n(t - nT)$, over the interval $t \in [nT, (n+1)T)$, or

$$h_n(t) \equiv h_n(t - nT)\Big|_{t=t+nT}. \tag{3.42}$$

Therefore, we define a new variable

$$\mu = \frac{t}{T}. \tag{3.43}$$

Substituting (3.43) into (3.4) gives the polynomial piece in terms of μ as

$$h_n(\mu) = \begin{cases} \sum_{m=0}^{M} c_m(n)\,(a\mu + b)^m, & 0 \le \mu < 1, \\ 0, & \text{otherwise.} \end{cases} \tag{3.44}$$

Now, we define the uniform sampling of $h_a(t)$ with spacing T and offset μ as

$$\begin{aligned} h_a(n, \mu) &= h_a((n + \mu)T) \\ &= \sum_{m=0}^{M} c_m(n)\,(a\mu + b)^m, \end{aligned} \tag{3.45}$$

and the vector $\mathbf{h}_a(\mu)$ to be the set of T-spaced impulse response samples

$$\mathbf{h}_a(\mu) = \begin{pmatrix} h_0(\mu) & h_1(\mu) & \cdots & h_{N-1}(\mu) \end{pmatrix}^{\mathrm{T}} \tag{3.46}$$

This is illustrated in Fig. 10.

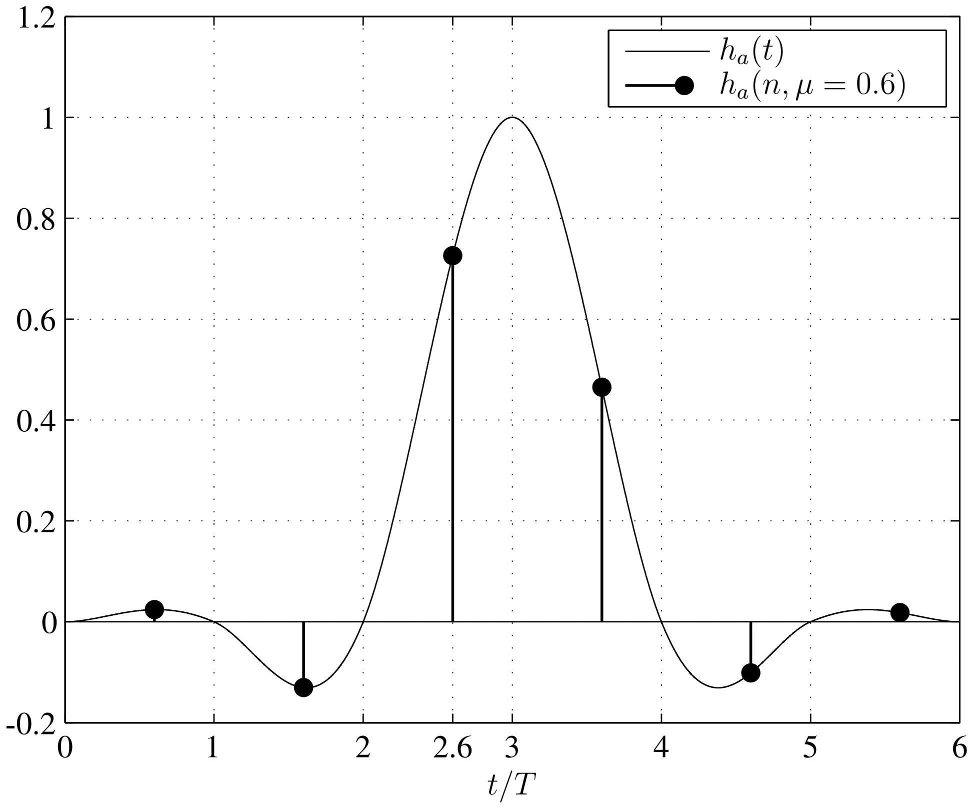

Figure 10: Uniform Sampling of $h_a(t)$

The length N vector of impulse response samples, $\mathbf{h}_a(\mu)$, is readily generated using the coefficient matrix from (3.36) as follows

$$\begin{aligned}\mathbf{h}_a(\mu) &= \mathbf{C}\boldsymbol{\mu} \\ &= \begin{pmatrix} h_0(\mu) & h_1(\mu) & \cdots & h_{N-1}(\mu) \end{pmatrix}^{\mathrm{T}} \\ &= \begin{pmatrix} c_0(0) & c_1(0) & \cdots & c_M(0) \\ c_0(1) & c_1(1) & \cdots & c_M(1) \\ \vdots & \vdots & & \vdots \\ c_0(N-1) & c_1(N-1) & \cdots & c_M(N-1) \end{pmatrix} \begin{pmatrix} 1 \\ (a\mu+b) \\ \vdots \\ (a\mu+b)^M \end{pmatrix},\end{aligned} \tag{3.47}$$

where

$$\boldsymbol{\mu} = \begin{pmatrix} 1 \\ (a\mu + b) \\ \vdots \\ (a\mu + b)^M \end{pmatrix}. \tag{3.48}$$

A nice property of the PBF is that since it is constructed of polynomials, its derivative has a simple closed form. From (3.45) the i^{th} derivative is given by

$$h_a^{(i)}(n, \mu) = \sum_{m=i}^{M} c_m(n) \, (a\mu + b)^{m-i} \, \frac{a^i m!}{(m-i)!}, \tag{3.49}$$

Equation 3.49 is valid for each polynomial piece. Border discontinuities between each polynomial piece give rise to impulses in the derivative. This is of no consequence for our purposes, see [56] for more on the derivative of piecewise polynomials with border discontinuity. Using (3.47) and (3.49) the vector of samples of the derivative impulse response is given by

$$\mathbf{h}_a^i(\mu) = \mathbf{C}\boldsymbol{\Delta}^{(i)}\boldsymbol{\mu}, \tag{3.50}$$

where

$$\boldsymbol{\Delta}^{(i)} = \begin{pmatrix} \boldsymbol{\delta}_i^{(i)} & \boldsymbol{\delta}_{i+1}^{(i)} & \cdots & \boldsymbol{\delta}_{M-i}^{(i)} \end{pmatrix}_{(M+1)\times(M+1)}, \tag{3.51}$$

and

$$\boldsymbol{\delta}_m^{(i)} = \begin{pmatrix} m \text{ zeros} \\ \Delta^{(i)}(m) \\ M - m \text{ zeros} \end{pmatrix}_{(M+1)\times 1}, \tag{3.52}$$

where

$$\Delta^{(i)}(m) = \begin{cases} \frac{a^i m!}{(m-i)!}, & m \geq i, \\ 0, & \text{otherwise}. \end{cases} \tag{3.53}$$

The frequency response given in (3.17) can be represented as

$$H_a(\omega) = \mathrm{Tr}\left(\mathbf{C}\boldsymbol{\Psi}(\omega)\right), \tag{3.54}$$

where,

$$\boldsymbol{\Psi}(\omega) = \begin{pmatrix} \Psi_0(0,\omega) & \Psi_0(1,\omega) & \cdots & \Psi_0(N-1,\omega) \\ \Psi_1(0,\omega) & \Psi_1(1,\omega) & \cdots & \Psi_1(N-1,\omega) \\ \vdots & \vdots & & \vdots \\ \Psi_M(0,\omega) & \Psi_M(1,\omega) & \cdots & \Psi_M(N-1,\omega) \end{pmatrix}. \tag{3.55}$$

Alternatively, (3.54) can be expressed as

$$H_a(\omega) = \boldsymbol{\psi}(\omega)\mathbf{c}, \tag{3.56}$$

where,

$$\boldsymbol{\psi}(\omega) = \left(\begin{array}{c} \Psi_0(0,\omega) \\ \Psi_1(0,\omega) \\ \vdots \\ \Psi_M(0,\omega) \\ \hline \Psi_0(1,\omega) \\ \Psi_1(1,\omega) \\ \vdots \\ \Psi_M(1,\omega) \\ \hline \vdots \\ \hline \Psi_0(N-1,\omega) \\ \Psi_1(N-1,\omega) \\ \vdots \\ \Psi_M(N-1,T,\omega) \end{array}\right)^{\mathrm{T}}_{1\times N(M+1)}, \tag{3.57}$$

and,

$$\mathbf{c} = \left(\begin{array}{c} c_0(0) \\ c_1(0) \\ \vdots \\ c_M(0) \\ \hline c_0(1) \\ c_1(1) \\ \vdots \\ c_M(1) \\ \hline \vdots \\ \hline c_0(N-1) \\ c_1(N-1) \\ \vdots \\ c_M(N-1) \end{array}\right)_{N(M+1)\times 1} . \tag{3.58}$$

Now, define a length P frequency response vector which contains the values of $H_a(\omega)$ at P frequency points, $\omega = \omega_p$ as

$$\begin{aligned} \mathbf{h}_a(\omega_p) &= \hat{\mathbf{\Psi}}(\omega_p)\mathbf{c}, \\ &= \begin{pmatrix} H_a(\omega_0) \\ H_a(\omega_1) \\ \vdots \\ H_a(\omega_{P-1}) \end{pmatrix}_{P\times 1} , \end{aligned} \tag{3.59}$$

where,

$$\hat{\mathbf{\Psi}}(\omega_p) = \begin{pmatrix} \boldsymbol{\psi}(\omega_0) \\ \boldsymbol{\psi}(\omega_1) \\ \vdots \\ \boldsymbol{\psi}(\omega_{P-1}) \end{pmatrix}_{P\times N(M+1)} . \tag{3.60}$$

Likewise, the impulse response can be written as

$$\mathbf{h}_{\mathbf{a}}(\mu) = \mathbf{M}(\mu)\mathbf{c}, \tag{3.61}$$

where, using (3.48),

$$\mathbf{M}(\mu) = \begin{pmatrix} \boldsymbol{\mu}^{\mathrm{T}} & 0 & 0 & \cdots & 0 \\ \mathbf{z} & \boldsymbol{\mu}^{\mathrm{T}} & 0 & \cdots & 0 \\ \mathbf{z} & \mathbf{z} & \boldsymbol{\mu}^{\mathrm{T}} & \ddots & 0 \\ \vdots & \vdots & \ddots & \ddots & 0 \\ \mathbf{z} & \mathbf{z} & \cdots & \mathbf{z} & \boldsymbol{\mu}^{\mathrm{T}} \end{pmatrix}_{N\times N(M+1)} , \tag{3.62}$$

and $\mathbf{z}$ is a length $M+1$ zero vector,

$$\mathbf{z} = \begin{pmatrix} 0 & 0 & \cdots & 0 \end{pmatrix}_{1\times(M+1)} \tag{3.63}$$

The derivative of the impulse response can be arranged in a similar manner giving

$$\mathbf{h}_{a}^{(i)}(\mu) = \mathbf{M}^{(i)}(\mu)\mathbf{c}, \tag{3.64}$$

where,

$$\mathbf{M}^{(i)}(\mu) = \begin{pmatrix} \boldsymbol{\Delta}^{(i)}\boldsymbol{\mu}^{\mathrm{T}} & 0 & 0 & \cdots & 0 \\ \mathbf{z} & \boldsymbol{\Delta}^{(i)}\boldsymbol{\mu}^{\mathrm{T}} & 0 & \cdots & 0 \\ \mathbf{z} & \mathbf{z} & \boldsymbol{\Delta}^{(i)}\boldsymbol{\mu}^{\mathrm{T}} & \ddots & 0 \\ \vdots & \vdots & \ddots & \ddots & 0 \\ \mathbf{z} & \mathbf{z} & \cdots & \mathbf{z} & \boldsymbol{\Delta}^{(i)}\boldsymbol{\mu}^{\mathrm{T}} \end{pmatrix}_{N\times N(M+1)} , \tag{3.65}$$

3.3.3.1 The Linear Phase Case

For the linear phase PBF, the N even case yields

$$H_a(\omega) = \boldsymbol{\psi}_E(\omega)\mathbf{c}_E, \tag{3.66}$$

where,

$$\boldsymbol{\psi}_E(\omega) = \begin{pmatrix} 2\Re\{\Psi_0(0,\omega)\} \\ 2\Re\{\Psi_1(0,\omega)\} \\ \vdots \\ 2\Re\{\Psi_M(0,\omega)\} \\ \hline 2\Re\{\Psi_0(1,\omega)\} \\ 2\Re\{\Psi_1(1,\omega)\} \\ \vdots \\ 2\Re\{\Psi_M(1,\omega)\} \\ \hline \vdots \\ \hline 2\Re\{\Psi_0(N/2-1,\omega)\} \\ 2\Re\{\Psi_1(N/2-1,\omega)\} \\ \vdots \\ 2\Re\{\Psi_M(N/2-1,\omega)\} \end{pmatrix}^{\mathrm{T}}_{1\times(N/2)(M+1)}, \tag{3.67}$$

and

$$\mathbf{c}_E = \left(\begin{array}{c} c_0(0) \\ c_1(0) \\ \vdots \\ c_M(0) \\ \hline c_0(1) \\ c_1(1) \\ \vdots \\ c_M(1) \\ \hline \vdots \\ \hline c_0(N/2-1) \\ c_1(N/2-1) \\ \vdots \\ c_M(N/2-1) \end{array}\right)_{(N/2)(M+1)\times 1} . \tag{3.68}$$

The P length frequency response vector is now given by

$$\begin{aligned} \mathbf{h}_a(\omega_p) &= \hat{\mathbf{\Psi}}_E(\omega_p)\mathbf{c}_E, \\ &= \begin{pmatrix} H_a(\omega_0) \\ H_a(\omega_1) \\ \vdots \\ H_a(\omega_{P-1}) \end{pmatrix}_{P\times 1}, \end{aligned} \tag{3.69}$$

where,

$$\hat{\mathbf{\Psi}}_E(\omega_p) = \begin{pmatrix} \boldsymbol{\psi}_E(\omega_0) \\ \boldsymbol{\psi}_E(\omega_1) \\ \vdots \\ \boldsymbol{\psi}_E(\omega_{P-1}) \end{pmatrix}_{P\times(N/2)(M+1)} . \tag{3.70}$$

Likewise, the impulse response for $n = 0, 1, \ldots, N/2-1$ can be written as

$$\mathbf{h}_{aE}(\mu) = \mathbf{M}_E(\mu)\mathbf{c}_E, \tag{3.71}$$

where, using (3.48),

$$\mathbf{M}_E(\mu) = \begin{pmatrix} \boldsymbol{\mu}^{\mathrm{T}} & 0 & 0 & \cdots & 0 \\ \mathbf{z} & \boldsymbol{\mu}^{\mathrm{T}} & 0 & \cdots & 0 \\ \mathbf{z} & \mathbf{z} & \boldsymbol{\mu}^{\mathrm{T}} & \ddots & 0 \\ \vdots & \vdots & \ddots & \ddots & 0 \\ \mathbf{z} & \mathbf{z} & \cdots & \mathbf{z} & \boldsymbol{\mu}^{\mathrm{T}} \end{pmatrix}_{N/2\times N/2(M+1)}, \tag{3.72}$$

The derivative of the impulse response for $n = 0, 1, \ldots, N/2 - 1$ can be arranged in a similar manner giving

$$\mathbf{h}_{aE}^{(i)}(\mu) = \mathbf{M}_E^{(i)}(\mu)\mathbf{c}_E, \tag{3.73}$$

where,

$$\mathbf{M}_E^{(i)}(\mu) = \begin{pmatrix} \boldsymbol{\Delta}^{(i)}\boldsymbol{\mu}^{\mathrm{T}} & 0 & 0 & \cdots & 0 \\ \mathbf{z} & \boldsymbol{\Delta}^{(i)}\boldsymbol{\mu}^{\mathrm{T}} & 0 & \cdots & 0 \\ \mathbf{z} & \mathbf{z} & \boldsymbol{\Delta}^{(i)}\boldsymbol{\mu}^{\mathrm{T}} & \ddots & 0 \\ \vdots & \vdots & \ddots & \ddots & 0 \\ \mathbf{z} & \mathbf{z} & \cdots & \mathbf{z} & \boldsymbol{\Delta}^{(i)}\boldsymbol{\mu}^{\mathrm{T}} \end{pmatrix}_{N/2\times N/2(M+1)}, \tag{3.74}$$

3.4 Filter Optimization

The optimal filter is the one for which the maximum of the error, $E(\omega)$, from a desired frequency response $D(\omega)$ is minimized over all ω, where

$$E(\omega) = W(\omega)\left[D(\omega) - H_a(\omega)\right]. \tag{3.75}$$

In (3.75) $H_a(\omega)$ is the actual frequency response and $W(\omega)$ is a weighting function allowing tuning of the relative size of the error over frequency. Thus, the weighting function may be used for example to emphasize stopband attenuation over passband ripple. If δ is the maximum error, then the problem can be stated as

$$-\delta \leq E(\omega) \leq \delta, \tag{3.76}$$

where the goal is to minimize the maximum error [41]. This is why this type of optimization is termed Minimax Optimization. Substituting (3.75) into (3.76) gives

$$-\delta \leq W(\omega)\left[D(\omega) - H_a(\omega)\right] \leq \delta, \tag{3.77}$$

Equation (3.77) can be re-written as

$$D(\omega) - \frac{\delta}{W(\omega)} \leq H_a(\omega) \leq D(\omega) + \frac{\delta}{W(\omega)}. \tag{3.78}$$

In order to perform filter optimization, the frequency variable ω must be discretized. Therefore, (3.78) is written in its discretized form using (3.69) as

$$\mathbf{d}(\omega_p) - \delta\hat{\mathbf{w}}(\omega_p) \leq \mathbf{h}_a(\omega_p) \leq \mathbf{d}(\omega_p) + \delta\hat{\mathbf{w}}(\omega_p), \tag{3.79}$$

where,

$$\mathbf{d}(\omega_p) = \begin{pmatrix} D(\omega_0) \\ D(\omega_1) \\ \vdots \\ D(\omega_{P-1}) \end{pmatrix}_{P\times 1}, \tag{3.80}$$

and,

$$\hat{\mathbf{w}}(\omega_p) = \begin{pmatrix} W(\omega_0)^{-1} \\ W(\omega_1)^{-1} \\ \vdots \\ W(\omega_{P-1})^{-1} \end{pmatrix}_{P\times 1}, \tag{3.81}$$

Writing (3.69) in terms of in (3.68) gives

$$\mathbf{d}(\omega_p) - \delta\hat{\mathbf{w}}(\omega_p) \leq \hat{\mathbf{\Psi}}_E(\omega_p)\mathbf{c}_E \leq \mathbf{d}(\omega_p) + \delta\hat{\mathbf{w}}(\omega_p), \tag{3.82}$$

This step exposes the linearity of the frequency response with respect to the coefficients. It is this fact that allows the optimization to be carried out via Linear Programming.

3.4.1 Linear Programming

Linear Programming solves the problem (see [57] and [41])

$$\min_{\mathbf{x}} \mathbf{g}^{\mathrm{T}}\mathbf{x}, \qquad \text{such that} \qquad \begin{cases} \mathbf{A}\mathbf{x} \leq \mathbf{b} \\ \mathbf{A}_{\text{eq}}\mathbf{x} = \mathbf{b}_{\text{eq}} \\ lb \leq \mathbf{x} \leq ub \end{cases} . \tag{3.83}$$

The minimax problem given in the last section can be modified to fit this form as follows. First, split (3.82) into two inequalities given by

$$\begin{aligned} \hat{\boldsymbol{\Psi}}_E(\omega_p)\mathbf{c}_E &\leq \mathbf{d}(\omega_p) + \delta\hat{\mathbf{w}}(\omega_p) \\ -\hat{\boldsymbol{\Psi}}_E(\omega_p)\mathbf{c}_E &\leq -\mathbf{d}(\omega_p) + \delta\hat{\mathbf{w}}(\omega_p) \end{aligned} \tag{3.84}$$

Then, rearrange them as

$$\begin{aligned} \hat{\boldsymbol{\Psi}}_E(\omega_p)\mathbf{c}_E - \delta\hat{\mathbf{w}}(\omega_p) &\leq \mathbf{d}(\omega_p) \\ -\hat{\boldsymbol{\Psi}}_E(\omega_p)\mathbf{c}_E - \delta\hat{\mathbf{w}}(\omega_p) &\leq -\mathbf{d}(\omega_p). \end{aligned} \tag{3.85}$$

The two inequalities in (3.85) can be combined in matrix form as

$$\begin{pmatrix} \hat{\boldsymbol{\Psi}}_E(\omega_p) & -\hat{\mathbf{w}}(\omega_p) \\ -\hat{\boldsymbol{\Psi}}_E(\omega_p) & -\hat{\mathbf{w}}(\omega_p) \end{pmatrix} \begin{pmatrix} \mathbf{c}_E \\ \delta \end{pmatrix} \leq \begin{pmatrix} \mathbf{d}(\omega_p) \\ -\mathbf{d}(\omega_p) \end{pmatrix} \tag{3.86}$$

Equation (3.86) now is in the form of (3.83) with

$$\mathbf{A} = \begin{pmatrix} \hat{\mathbf{\Psi}}_E(\omega_p) & -\hat{\mathbf{w}}(\omega_p) \\ -\hat{\mathbf{\Psi}}_E(\omega_p) & -\hat{\mathbf{w}}(\omega_p) \end{pmatrix}_{2P\times(N/2(M+1)+1)} \tag{3.87}$$

$$\mathbf{x} = \begin{pmatrix} \mathbf{c}_E \\ \delta \end{pmatrix}_{(N/2(M+1)+1)\times 1} \tag{3.88}$$

$$\mathbf{b} = \begin{pmatrix} \mathbf{d}(\omega_p) \\ -\mathbf{d}(\omega_p) \end{pmatrix}_{2P\times 1} \tag{3.89}$$

$$\mathbf{g} = \begin{pmatrix} 0 \\ 0 \\ \vdots \\ 0 \\ 1 \end{pmatrix}_{(N/2(M+1)+1)\times 1} \tag{3.90}$$

Using (3.90) and (3.88), it is clear that the minimization in (3.83) now becomes

$$\min_{\mathbf{x}} \mathbf{g}^{\mathrm{T}}\mathbf{x} = \min_{\mathbf{x}} \delta. \tag{3.91}$$

This is the desired result, namely, minimization of the maximum error.

The linear programming problem in (3.83) also allows inclusion of equality constraints given by

$$\mathbf{A}_{\text{eq}}\mathbf{x} = \mathbf{b}_{\text{eq}}, \tag{3.92}$$

and bounds on $\mathbf{x}$ given by

$$lb \le \mathbf{x} \le ub. \tag{3.93}$$

These can be left out of the optimization, but (3.92) proves quite useful in the inclusion of simultaneous time domain constraints during minimax optimization. This will be demonstrated in a later section.

To summarize, given a desired frequency response and weighting function, the matrix $\mathbf{A}$ and the vectors $\mathbf{g}$ and $\mathbf{b}$ can be constructed and input to a linear programming routine. One such routine is Matlab's `linprog` routine. The output is the vector $\mathbf{x}$ containing the optimized coefficients $\mathbf{c}_E$ and the minimized maximum error, δ. An example will be given in the next section for clarification.

3.4.2 Example

The frequency domain optimization of a PBF using linear programming will now be illustrated by an example. A filter is to be designed with the following parameters

1. Normalized passband edge: $\omega_{\text{pass}} = 2\pi f_{\text{pass}} = 2\pi(.2)$
2. Normalized stopband edge: $\omega_{\text{stop}} = 2\pi f_{\text{stop}} = 2\pi(.8)$
3. Number of polynomial pieces: $N = 6$
4. Order of each piece: $M = 3$
5. Basis function constants: $a = 1$, $b = -1/2$
6. Passband weight: $K_{\text{pass}} = 10$
7. Stopband weight: $K_{\text{stop}} = 1$

Now, two uniformly spaced frequency grids are chosen for the passband and stopband of 100 and 500 points respectively. This results in a total of

$$P = 100 + 500 = 600 \tag{3.94}$$

points for optimization. These are the only points optimized. The transition band is a "don't care" band. The passband extends from 0 to ω_{pass}, while the stopband extends from ω_{stop} to ∞.

The infinite extent of the stopband is not really a problem as far as optimization is concerned. As will be shown in a later section, the PBF frequency response rolls-off as $\omega \to \infty$ with a decay proportional to the smoothness of the PBF impulse response. In practice, the optimization need only be carried out to $\omega \approx 8\pi$. The response at higher frequencies can be inspected after optimization to ensure good behavior.

A Matlab script implementing this optimization is given in the Appendix. The optimization results in a minimized maximum error of

$$\delta = 0.0020, \tag{3.95}$$

and optimized coefficients

$$\mathbf{c}_E = \begin{pmatrix} 0.0138 \\ 0.0687 \\ -0.0079 \\ -0.1415 \\ -0.1066 \\ -0.2875 \\ 0.3480 \\ 0.8925 \\ 0.5923 \\ 1.5384 \\ -0.3324 \\ -1.7383 \end{pmatrix}. \tag{3.96}$$

The frequency response of the filter is given in Fig. 11.

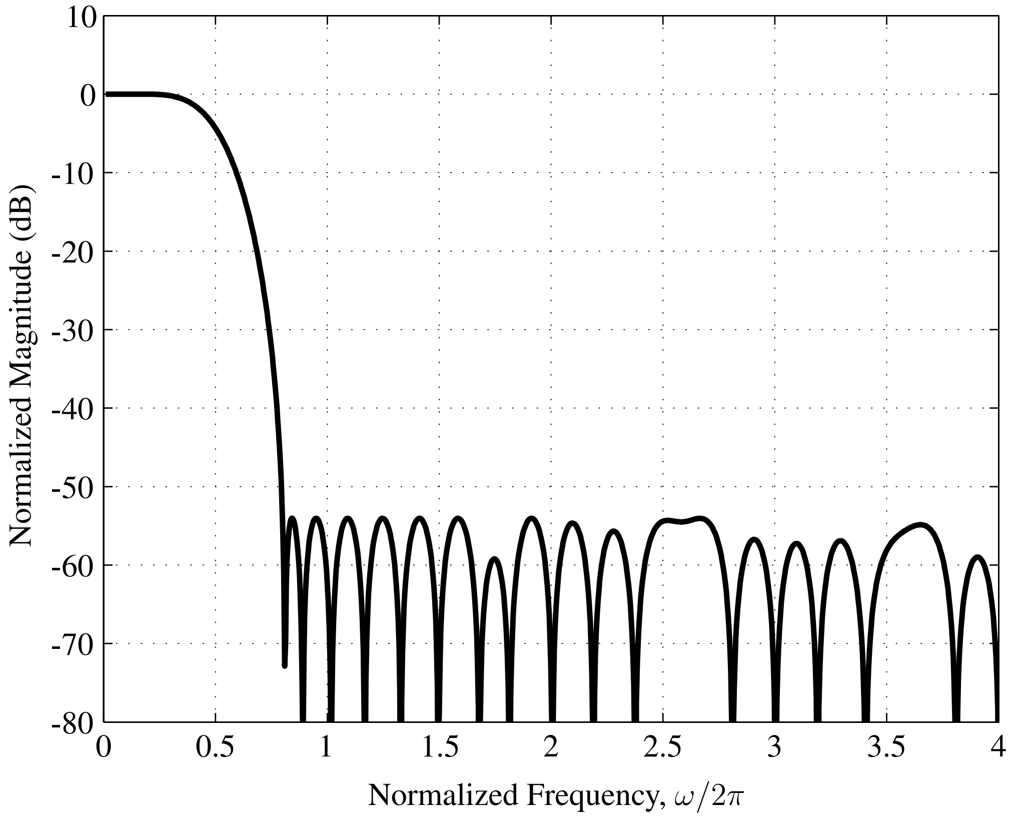

Figure 11: Frequency Response of Optimization Example Filter

The vector $\mathbf{c}_E$ contains only half the coefficients of the linear phase PBF since this is all that is needed for optimization. The complete coefficient matrix of (3.36) can be constructed using (3.26), which yields

$$\mathbf{C} = \begin{pmatrix} 0.0138 & 0.0687 & -0.0079 & -0.1415 \\ -0.1066 & -0.2875 & 0.3480 & 0.8925 \\ 0.5923 & 1.5384 & -0.3324 & -1.7383 \\ 0.5923 & -1.5384 & -0.3324 & 1.7383 \\ -0.1066 & 0.2875 & 0.3480 & -0.8925 \\ 0.0138 & -0.0687 & -0.0079 & 0.1415 \end{pmatrix} \tag{3.97}$$

The vector uniformly spaced impulse response samples can now be computed for any μ using (3.47). For example, for $\mu = .25$, (3.47) gives

$$\mathbf{h}_a(\mu = 0.25) = \begin{pmatrix} 0.0138 & 0.0687 & -0.0079 & -0.1415 \\ -0.1066 & -0.2875 & 0.3480 & 0.8925 \\ 0.5923 & 1.5384 & -0.3324 & -1.7383 \\ 0.5923 & -1.5384 & -0.3324 & 1.7383 \\ -0.1066 & 0.2875 & 0.3480 & -0.8925 \\ 0.0138 & -0.0687 & -0.0079 & 0.1415 \end{pmatrix} \begin{pmatrix} 1.0000 \\ -0.2500 \\ 0.0625 \\ -0.0156 \end{pmatrix} \tag{3.98}$$

$$= \begin{pmatrix} -0.0017 \\ -0.0269 \\ 0.2140 \\ 0.9289 \\ -0.1427 \\ 0.0282 \end{pmatrix} \tag{3.99}$$

These values are plotted on top of a more densely sampled version of the impulse response in Fig. 12.

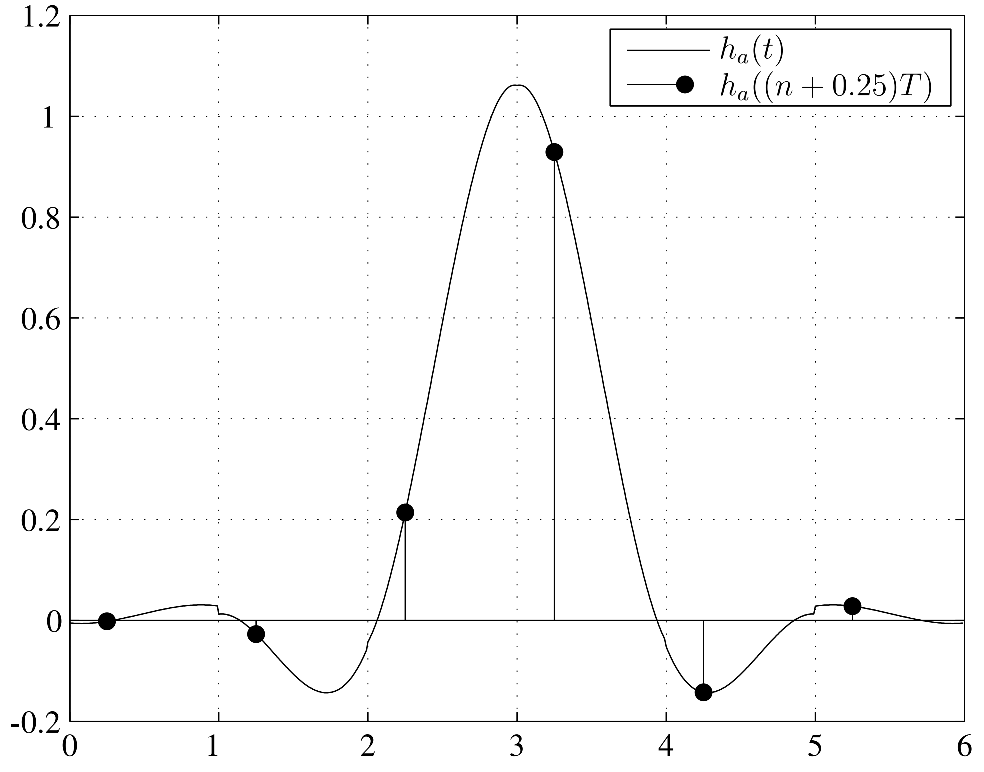

Figure 12: Impulse Response of Optimization Example Filter

3.4.3 Time Domain Constraints

When optimizing PBFs without time domain constraints, the impulse response exhibits only *piecewise* smoothness. This phenomenon is easily observed by zooming in on one of the borders between adjacent polynomial pieces. Figure 13 shows the impulse response of Fig. 12 zoomed in on the border between $h_a(0,\mu)$ and $h_a(1,\mu)$. These border discontinuities may or may not be acceptable depending on the application.

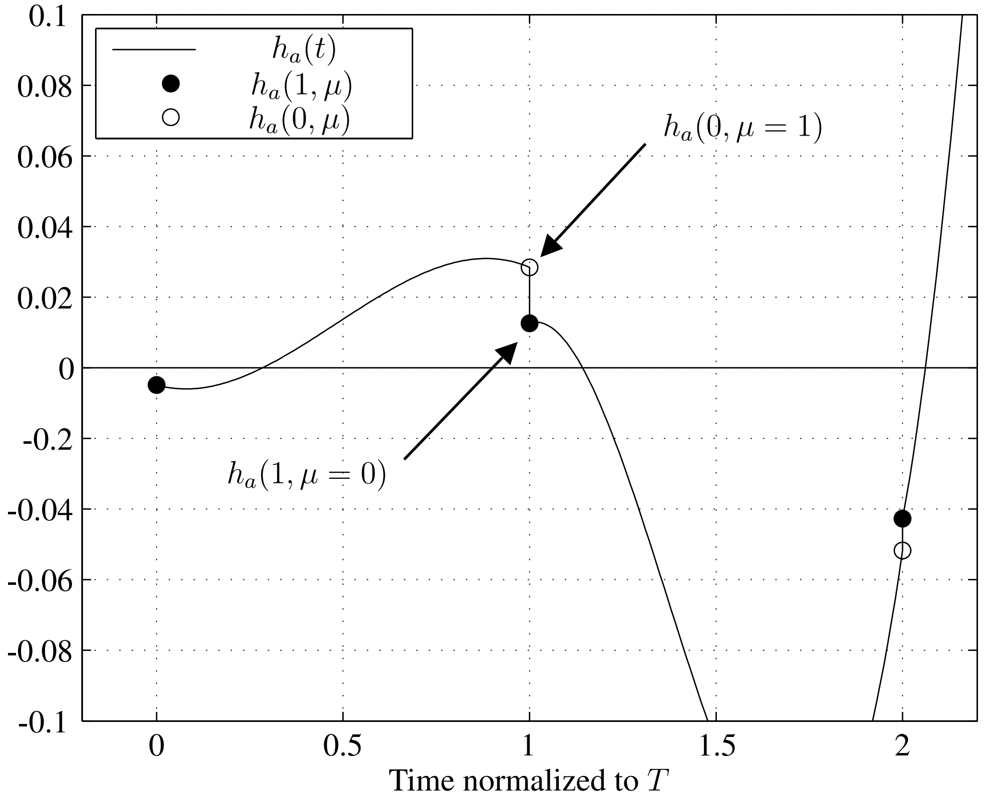

Figure 13: Zoomed Impulse Response of Optimization Example Filter

An observation of Fig. 13 reveals the key to making the impulse response, or its i^{th} derivative continuous across segment borders, namely,

$$h_a^{(i)}(n,1) = h_a^{(i)}(n+1,0), \text{ for } 0 \leq n \leq N-2. \tag{3.100}$$

Because the first and last piece have no adjacent piece to the left and right, respectively, the following condition must also be imposed

$$h_a^{(i)}(0,0) = h_a^{(i)}(N-1,1) = 0. \tag{3.101}$$

Finally,

$$h_a^{(i)}(N/2-1,1) = 0, \text{ for } i \text{ odd}. \tag{3.102}$$

Due to symmetry in the linear phase case, (3.100) becomes

$$h_a^{(i)}(n,1) = h_a^{(i)}(n+1,0), \text{ for } 0 \leq n \leq N/2-2, \tag{3.103}$$

and (3.101) becomes

$$h_a^{(i)}(0,0) = 0. \tag{3.104}$$

As mentioned in section 3.4.1, these constraints can be included in the optimization via (3.92). The first step is to rearrange the constraints of (3.103) as

$$h_a^{(i)}(n,1) - h_a^{(i)}(n+1,0) = 0, \text{ for } 0 \leq n \leq N/2-2. \tag{3.105}$$

Equation (3.105) can be written as

$$\begin{pmatrix} h_a^{(i)}(0,1) \\ h_a^{(i)}(1,1) \\ \vdots \\ h_a^{(i)}(N/2-2,1) \\ 0 \end{pmatrix} - \begin{pmatrix} h_a^{(i)}(1,0) \\ h_a^{(i)}(2,0) \\ \vdots \\ h_a^{(i)}(N/2-1,0) \\ 0 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 0 \\ \vdots \\ 0 \end{pmatrix} \tag{3.106}$$

In (3.106), an extra zero has been appended to the end of each vector so that each one is of length $N/2$. This was done in order to construct equations in terms of (3.73).

Before proceeding, it is useful to define

$$\mathbf{e}_1, \mathbf{e}_2, \ldots, \mathbf{e}_l \tag{3.107}$$

as the standard basis of $\mathbb{R}^l$, where $\mathbf{e}_j$ is a length l column vector with a 1 in the j^{th} position as the only nonzero entry. For example, for $\mathbb{R}^3$, the space of length $l = 3$ vectors over the real numbers, the standard basis is given by

$$\mathbf{e}_1 = \begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix}, \mathbf{e}_2 = \begin{pmatrix} 0 \\ 1 \\ 0 \end{pmatrix}, \mathbf{e}_3 = \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix}. \tag{3.108}$$

Also define

$$\mathbf{0} = \begin{pmatrix} 0 \\ 0 \\ \vdots \\ 0 \end{pmatrix}_{l \times 1}, \tag{3.109}$$

to be a length l column vector consisting of only zero entries.

Using (3.73), (3.106) becomes

$$\mathbf{P}\mathbf{M}_E^{(i)}(1)\mathbf{c}_E - \mathbf{Q}\mathbf{M}_E^{(i)}(0)\mathbf{c}_E = \mathbf{0} \tag{3.110}$$

where, in $\mathbb{R}^{N/2}$

$$\mathbf{P} = \begin{pmatrix} \mathbf{e}_1 & \mathbf{e}_2 & \cdots & \mathbf{e}_{N/2-1} & \mathbf{0} \end{pmatrix}_{N/2 \times N/2}, \tag{3.111}$$

and,

$$\mathbf{Q} = \begin{pmatrix} \mathbf{0} & \mathbf{e}_1 & \mathbf{e}_2 & \cdots & \mathbf{e}_{N/2-1} \end{pmatrix}_{N/2 \times N/2}. \tag{3.112}$$

Simplifying (3.110) gives

$$\begin{aligned} \left(\mathbf{P}\mathbf{M}_E^{(i)}(1) - \mathbf{Q}\mathbf{M}_E^{(i)}(0)\right)\mathbf{c}_E &= \mathbf{0} \\ \tilde{\mathbf{A}}_1\mathbf{c}_E &= \tilde{\mathbf{b}}_1 \end{aligned} \tag{3.113}$$

Equation (3.104) can also be written in terms of (3.73) as

$$\begin{aligned} \mathbf{R}\mathbf{M}_E^{(i)}(0)\mathbf{c}_E &= \mathbf{0} \\ \tilde{\mathbf{A}}_2\mathbf{c}_E &= \tilde{\mathbf{b}}_2, \end{aligned} \tag{3.114}$$

where

$$\mathbf{R} = \begin{pmatrix} \mathbf{e}_1 & \mathbf{0} & \mathbf{0} & \cdots & \mathbf{0} \end{pmatrix}_{N/2 \times N/2}. \tag{3.115}$$

Equation (3.102) can also be written in terms of (3.73) as

$$\begin{aligned} \mathbf{S}\mathbf{M}_E^{(i)}(1)\mathbf{c}_E &= \mathbf{0} \\ \tilde{\mathbf{A}}_3\mathbf{c}_E &= \tilde{\mathbf{b}}_3, \end{aligned} \tag{3.116}$$

where

$$\mathbf{S} = \begin{pmatrix} \mathbf{0} & \mathbf{0} & \cdots & \mathbf{0} & \mathbf{e}_{N/2} \end{pmatrix}_{N/2 \times N/2}. \tag{3.117}$$

Finally, (3.113), (3.114), and (3.116) can be combined into the form of 3.92 as

$$\mathbf{A}_{\text{eq}}\mathbf{x} = \mathbf{b}_{\text{eq}}$$
$$\begin{pmatrix} \tilde{\mathbf{A}} & \mathbf{0} \end{pmatrix} \begin{pmatrix} \mathbf{c}_E \\ \delta \end{pmatrix} = \tilde{\mathbf{b}} \tag{3.118}$$

where,

$$\tilde{\mathbf{A}} = \begin{pmatrix} \tilde{\mathbf{A}}_1 \\ \tilde{\mathbf{A}}_2 \\ \tilde{\mathbf{A}}_3 \end{pmatrix}, \tag{3.119}$$

and,

$$\tilde{\mathbf{b}} = \begin{pmatrix} \tilde{\mathbf{b}}_1 \\ \tilde{\mathbf{b}}_2 \\ \tilde{\mathbf{b}}_3 \end{pmatrix}, \tag{3.120}$$

where $\tilde{\mathbf{A}}_3$ and $\tilde{\mathbf{b}}_3$ are only included for i odd. A Matlab function that generates these constraint matrices can be found in the Appendix.

3.4.4 Example with Constraints

In this section, we demonstrate the use of the time domain constraints developed in the preceding section. The optimization parameters are the same as the example from section (3.4.2), except that impulse response continuity is imposed. A Matlab function implementing this example optimization can be found in the Appendix. The optimization results in a minimized maximum error of

$$\delta = 0.0029, \tag{3.121}$$

and optimized coefficients

$$\mathbf{c}_E = \begin{pmatrix} 0.0172 \\ 0.0649 \\ -0.0489 \\ -0.2194 \\ -0.1146 \\ -0.2192 \\ 0.4413 \\ 0.7620 \\ 0.5967 \\ 1.4215 \\ -0.3829 \\ -1.5289 \end{pmatrix}. \tag{3.122}$$

The frequency response of the filter is given in Fig. 14.

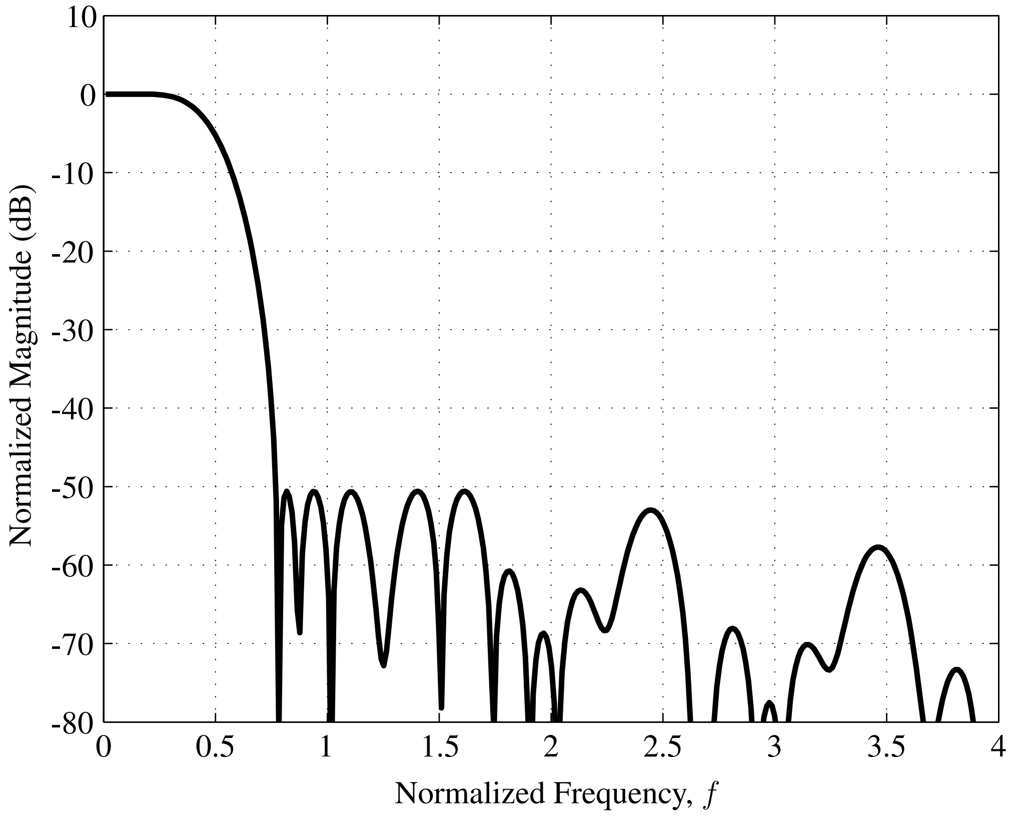

Figure 14: Frequency Response of Optimization Example Filter

The vector $\mathbf{c}_E$ contains only half the coefficients of the linear phase PBF since this is all that is needed for optimization. The complete coefficient matrix of (3.36) can be constructed using (3.26), which yields

$$\mathbf{C} = \begin{pmatrix} 0.0172 & 0.0649 & -0.0489 & -0.2194 \\ -0.1146 & -0.2192 & 0.4413 & 0.7620 \\ 0.5967 & 1.4215 & -0.3829 & -1.5289 \\ 0.5967 & -1.4215 & -0.3829 & 1.5289 \\ -0.1146 & 0.2192 & 0.4413 & -0.7620 \\ 0.0172 & -0.0649 & -0.0489 & 0.2194 \end{pmatrix}. \tag{3.123}$$

The impulse response and its first derivative are plotted in Fig. 15.

The effectiveness of the constrained optimization is shown in Fig. 16 where it is clear that the impulse response is now continuous.

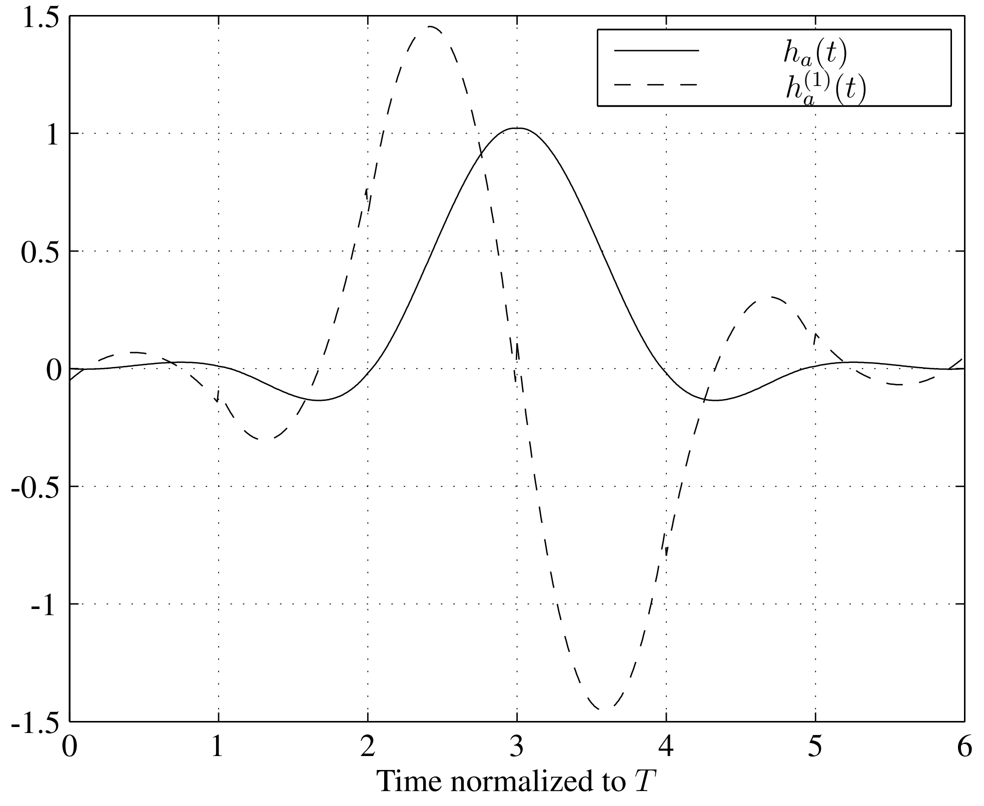

Figure 15: Derivative Impulse Response of Opt. Example with Constraints

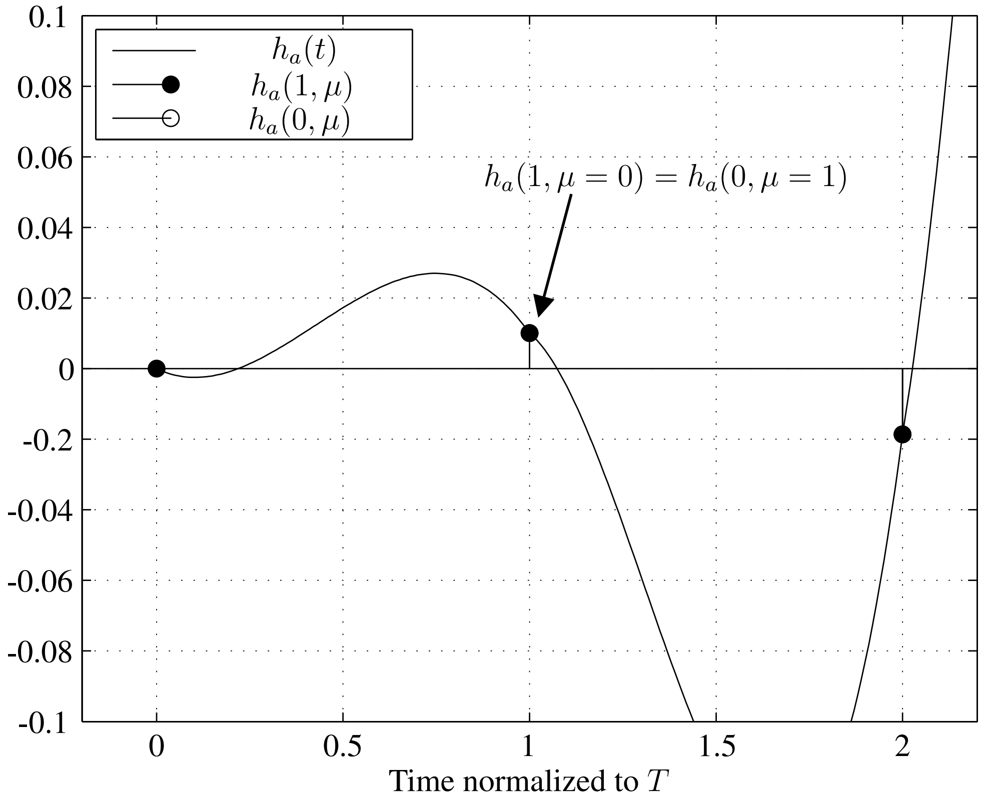

Figure 16: Zoomed Impulse Response of Opt. Example with Constraints

CHAPTER 4: IMPLEMENTATION

In this chapter we derive modified forms of the two main PBF implementation structures, the Farrow Structure and the Transposed Farrow Structure. These structures efficiently implement PBFs having polynomial pieces of equal length. In the case of the Farrow Structure, the length of each polynomial piece is $T = T_{\text{in}}$ while the Transposed Structure has pieces of length $T = T_{\text{out}}$. Other structures have been proposed that implement PBFs with polynomial pieces of different lengths [45] as well as structures that use equal length pieces equal to a multiple of $T = T_{\text{out}}$ or $T = T_{\text{in}}$ [46, 43]. These structures can give some performance enhancement at the cost of additional design and implementation complexity.

Starting with equation (1.11) which is repeated here for convenience

$$y(lT_{\text{out}}) = \sum_k x(kT_{\text{in}})h_a(lT_{\text{out}} - kT_{\text{in}})$$

we observe, as noted in [29], that the resampled signal can be computed *entirely digitally*. All that is needed is the input sequence and samples from the impulse response $h_a(t)$ at the proper time instants. This poses the question: How do we access samples of the continuous-time impulse response $h_a(t)$? Samples of $h_a(t)$ are readily computed on the fly if $h_a(t)$ is constructed as a PBF as detailed in section 3.2.

4.1 The Farrow Structure

The Farrow Structure [38] implements a PBF whose impulse response is composed of length $T = T_{\text{in}}$ polynomial pieces. A general form of the Farrow Structure can be derived as follows. With $t = lT_{\text{out}} - kT_{\text{in}}$

$$\begin{aligned} h_a(t) &= \sum_{n=0}^{N-1} \sum_{m=0}^{M} c_m(n) \psi_m(n, T_{\text{in}}, lT_{\text{out}} - kT_{\text{in}}) \\ &= \sum_{n=0}^{N-1} \sum_{m=0}^{M} c_m(n) \left(a \left(\frac{lT_{\text{out}} - kT_{\text{in}}}{T_{\text{in}}} - n \right) + b \right)^m \end{aligned} \tag{4.1}$$

for

$$nT_{\text{in}} \leq lT_{\text{out}} - kT_{\text{in}} < (n+1)T_{\text{in}} \tag{4.2}$$

and zero otherwise. From (4.2) we see that

$$\left\lfloor \frac{lT_{\text{out}} - kT_{\text{in}}}{T_{\text{in}}} \right\rfloor = n, \tag{4.3}$$

so that,

$$\left(\frac{lT_{\text{out}} - kT_{\text{in}}}{T_{\text{in}}} - n \right) = \frac{lT_{\text{out}} - kT_{\text{in}}}{T_{\text{in}}} - \left\lfloor \frac{lT_{\text{out}} - kT_{\text{in}}}{T_{\text{in}}} \right\rfloor \tag{4.4}$$

Here we see that (4.4) is the fractional part of the time distance between the desired output sample and the current input sample as a fraction of the input sample period. This so-called fractional interval, μ_l, can be simplified to

$$\mu_l = \frac{lT_{\text{out}}}{T_{\text{in}}} - \left\lfloor \frac{lT_{\text{out}}}{T_{\text{in}}} \right\rfloor \tag{4.5}$$

Equation (4.5) can be computed using an overflowing accumulator as

$$\mu_{l+1} = \mu_l + \frac{T_{\text{out}}}{T_{\text{in}}} - \left\lfloor \mu_l + \frac{T_{\text{out}}}{T_{\text{in}}} \right\rfloor \tag{4.6}$$

where from (4.2) we see that $\mu_l \in [0,1)$. In the above, $\lfloor\cdot\rfloor$ and $\lceil\cdot\rceil$ are the floor and ceiling operators, respectively. Substituting (4.5) into (4.1), and using (3.45) gives

$$\begin{aligned} h_a(lT_{\text{out}} - kT_{\text{in}}) &= \sum_{n=0}^{N-1} \sum_{m=0}^{M} c_m(n)\,(a\mu_l + b)^m \\ &= \sum_{n=0}^{N-1} h_a((n+\mu_l)T_{\text{in}}) \end{aligned} \tag{4.7}$$

Here we see that (4.7) represents the impulse response of the continuous time filter $h_a(t)$ uniformly sampled with period T_{in} and offset $\mu_l T_{\text{in}}$. For each output sample, μ_l flags the correct set of impulse response samples of $h_a(t)$ for the computation in (1.11). Substituting (4.7) into (1.11) gives

$$y(lT_{\text{out}}) = \sum_{n=0}^{N-1} x\left((k_l - n)\,T_{\text{in}}\right) h_a((n+\mu_l)T_{\text{in}}) \tag{4.8}$$

$$= \sum_{m=0}^{M} \left[\sum_{n=0}^{N-1} c_m(n) x\left((k_l - n)\,T_{\text{in}}\right)\right] (a\mu_l + b)^m\,, \tag{4.9}$$

where,

$$k_l = \left\lfloor \frac{lT_{\text{out}}}{T_{\text{in}}} \right\rfloor. \tag{4.10}$$

The input/output timing is given in Fig. 17 where it is clear that $x(k_l T_{\text{in}})$ is the input sample occurring just prior to or at the same instant as the desired output sample.

The expression in square brackets in (4.9) can be understood as the parallel filtering of $x(k_l T_{\text{in}})$ with $M+1$, N-tap FIR filters operating at the input sample rate. The filter coefficients are given by the columns of (3.36). The final output is then formed using Horner's rule. The implementation structure is given in Fig. 18. In practice, the structure can be simplified because each subfilter $c_m(n)$ operates on the same set of input samples, thus only one delay line is needed. Also, in the linear phase case coefficient symmetry can be exploited to reduce the number of multiplications required by each subfilter.

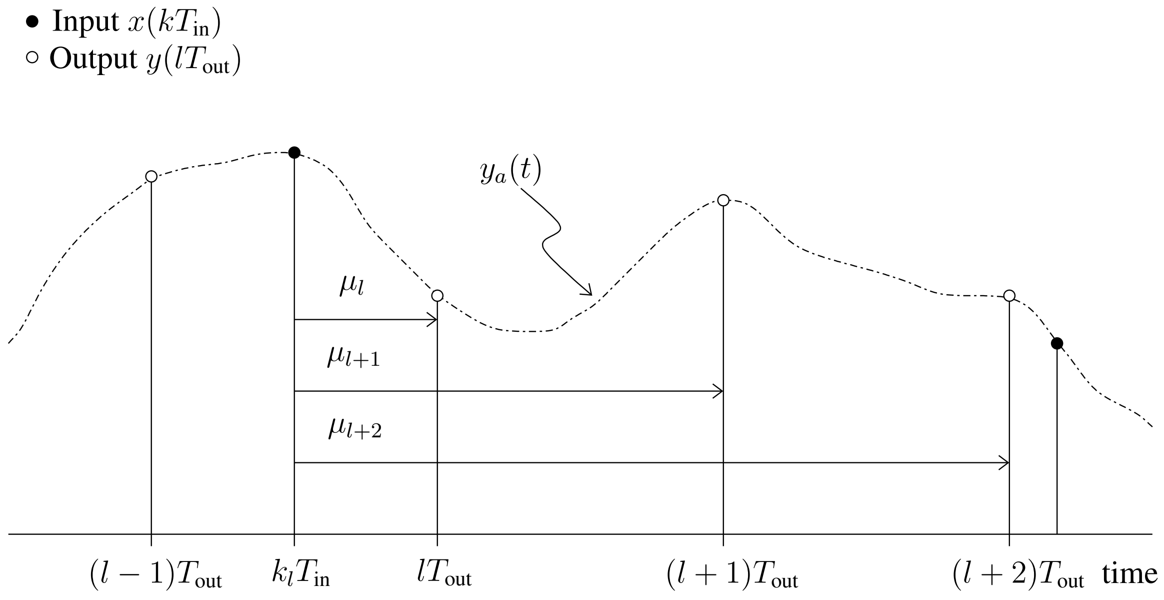

Figure 17: Farrow Structure Input / Output Timing

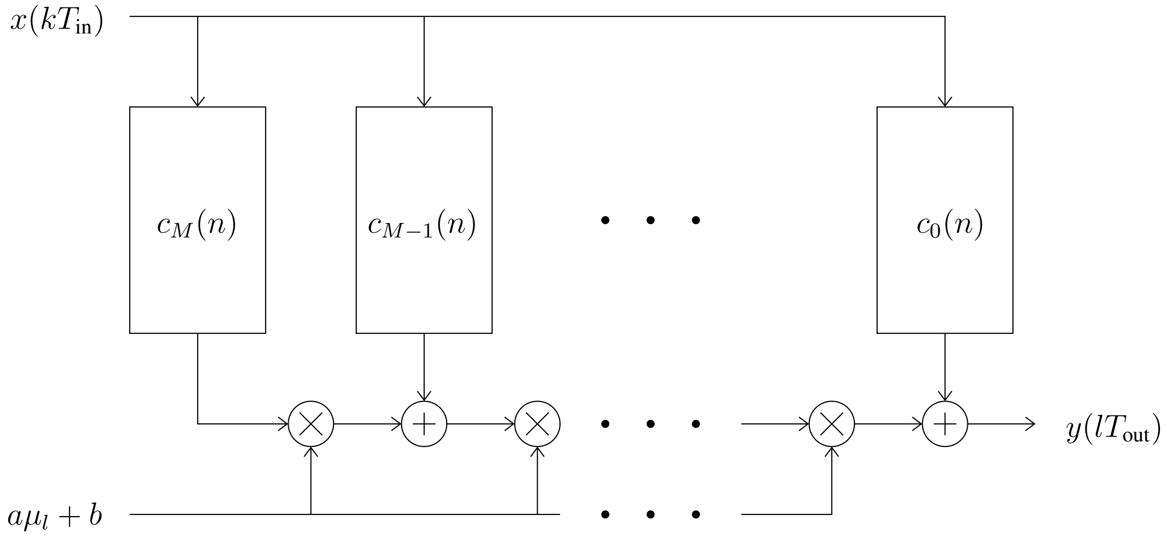

Figure 18: Farrow Structure

4.2 The Transposed Farrow Structure

One of the drawbacks to the original Farrow Structure and traditional signal processing methods for arbitrary sample rate conversion [38, 58, 37] was that they performed poorly for the case of decimation [44, 59]. The reason for this poor performance is that the polynomial segments of $h_a(t)$ are constructed as piecewise polynomials over the input sample intervals T_{in} rather than the output sample intervals T_{out}. This yields a frequency response normalized to the input rather than the output sample rate. Solutions to this problem were given in [43] and [44], yielding transposed forms of the Farrow Structure. Of the two structures, the one found in [44] is the more computationally efficient. In this section, we derive a more general form of this structure. In order to derive the Transposed Farrow Structure, we start with equation (1.11) which is repeated here for convenience

$$y(lT_{\text{out}}) = \sum_k x(kT_{\text{in}})h_a(lT_{\text{out}} - kT_{\text{in}})$$

where, with $t = lT_{\text{out}} - kT_{\text{in}}$

$$\begin{aligned} h_a(t) &= \sum_{n=0}^{N-1}\sum_{m=0}^{M} c_m(n)\psi_m(n, T_{\text{out}}, lT_{\text{out}} - kT_{\text{in}}) \\ &= \sum_{n=0}^{N-1}\sum_{m=0}^{M} c_m(n)\left(a\left(\frac{lT_{\text{out}} - kT_{\text{in}}}{T_{\text{out}}} - n\right) + b\right)^m \end{aligned} \tag{4.11}$$

for

$$nT_{\text{out}} \leq lT_{\text{out}} - kT_{\text{in}} < (n+1)T_{\text{out}} \tag{4.12}$$

and zero otherwise. From (4.12) we see that

$$\left\lfloor \frac{lT_{\text{out}} - kT_{\text{in}}}{T_{\text{out}}} \right\rfloor = n \tag{4.13}$$

so that

$$\left(\frac{lT_{\text{out}} - kT_{\text{in}}}{T_{\text{out}}} - n\right) = \frac{lT_{\text{out}} - kT_{\text{in}}}{T_{\text{out}}} - \left\lfloor \frac{lT_{\text{out}} - kT_{\text{in}}}{T_{\text{out}}} \right\rfloor$$
$$= \mu_k \tag{4.14}$$

Here we see that (4.14) is the fractional part of the time distance between the desired output sample and the current input sample normalized to the output sample period. This so-called fractional interval, μ_k, can be simplified to

$$\mu_k = \left\lceil \frac{kT_{\text{in}}}{T_{\text{out}}} \right\rceil - \frac{kT_{\text{in}}}{T_{\text{out}}} \tag{4.15}$$

where from (4.12) we see that $\mu_k \in [0, 1)$. In the above, $\lfloor\cdot\rfloor$ and $\lceil\cdot\rceil$ are the floor and ceiling operators, respectively. The relationship between μ_k and the input and output samples is illustrated in Fig. 19.

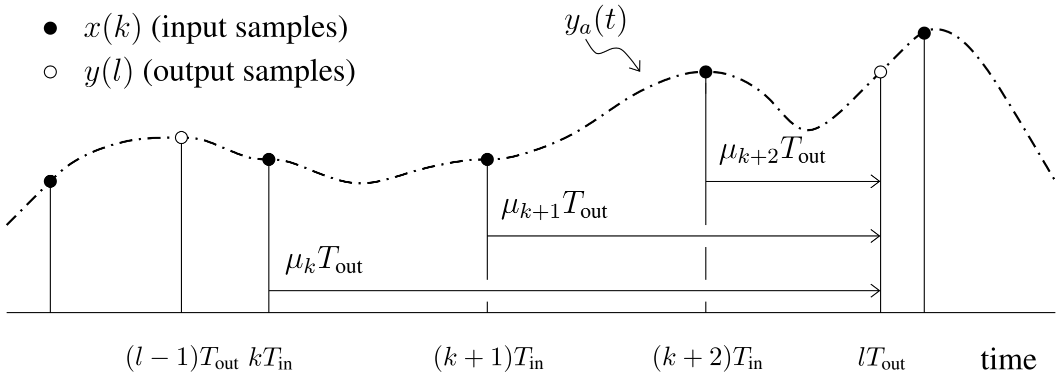

Figure 19: Input / Output Sample Time Relationship

Substituting (4.15) into (4.11) gives

$$h_a(lT_{\text{out}} - kT_{\text{in}}) = \sum_{n=0}^{N-1} \sum_{m=0}^{M} c_m(n) \left(a\mu_k + b\right)^m$$
$$= \sum_{n=0}^{N-1} h_a((n + \mu_k)T_{\text{out}}) \tag{4.16}$$

Here we see that (4.16) represents the impulse response of the continuous time filter $h_a(t)$ uniformly sampled with period T_{out} and offset $\mu_k T_{\text{out}}$. For each input sample, μ_k flags the correct set of impulse response samples of $h_a(t)$ for the computation in (1.11). Also, from (4.14)

$$kT_{\text{in}} = (l - n - \mu_k)\, T_{\text{out}} \tag{4.17}$$

Substituting (4.16) and (4.17) into (1.11) gives

$$y(lT_{\text{out}}) = \sum_{\substack{k,\text{ while}\\ \mu_k \in [0,1)}} \sum_{n=0}^{N-1} x\Big((l - \mu_k - n)T_{\text{out}}\Big) h_a((n + \mu_k)T_{\text{out}}) \tag{4.18}$$

$$= \sum_{n=0}^{N-1} \sum_{m=0}^{M} \left(\sum_{\substack{k,\text{ while}\\ \mu_k \in [0,1)}} x\Big((l - \mu_k - n)T_{\text{out}}\Big) (a\mu_k + b)^m \right) c_m(n) \tag{4.19}$$

An implementation of (4.19) is given in Fig. 20. The I&D (Integrate-and-Dump) blocks perform the summation over k. As can be seen from Fig. 19, multiple input samples can be involved in the computation of each output sample. One way to sum over the correct input samples is to implement the computation of μ_k as a down-counter/subtracting accumulator and use the inherent wrap around to issue a "dump" command to the I&D blocks. In this scenario μ_k is computed upon the arrival of each new input sample as

$$\mu_{k+1} = \mu_k - \frac{T_{\text{in}}}{T_{\text{out}}} - \left\lfloor \mu_k - \frac{T_{\text{in}}}{T_{\text{out}}} \right\rfloor \tag{4.20}$$

Since this sequence is strictly decreasing over each T_{out} interval, the accumulator output can be monitored for the condition $\mu_{k+1} \geq \mu_k$ indicating wrap around. When this condition is met, the outputs of the I&D blocks are passed to the Farrow coefficient network which computes a new output sample by performing the remaining summations over m and n.

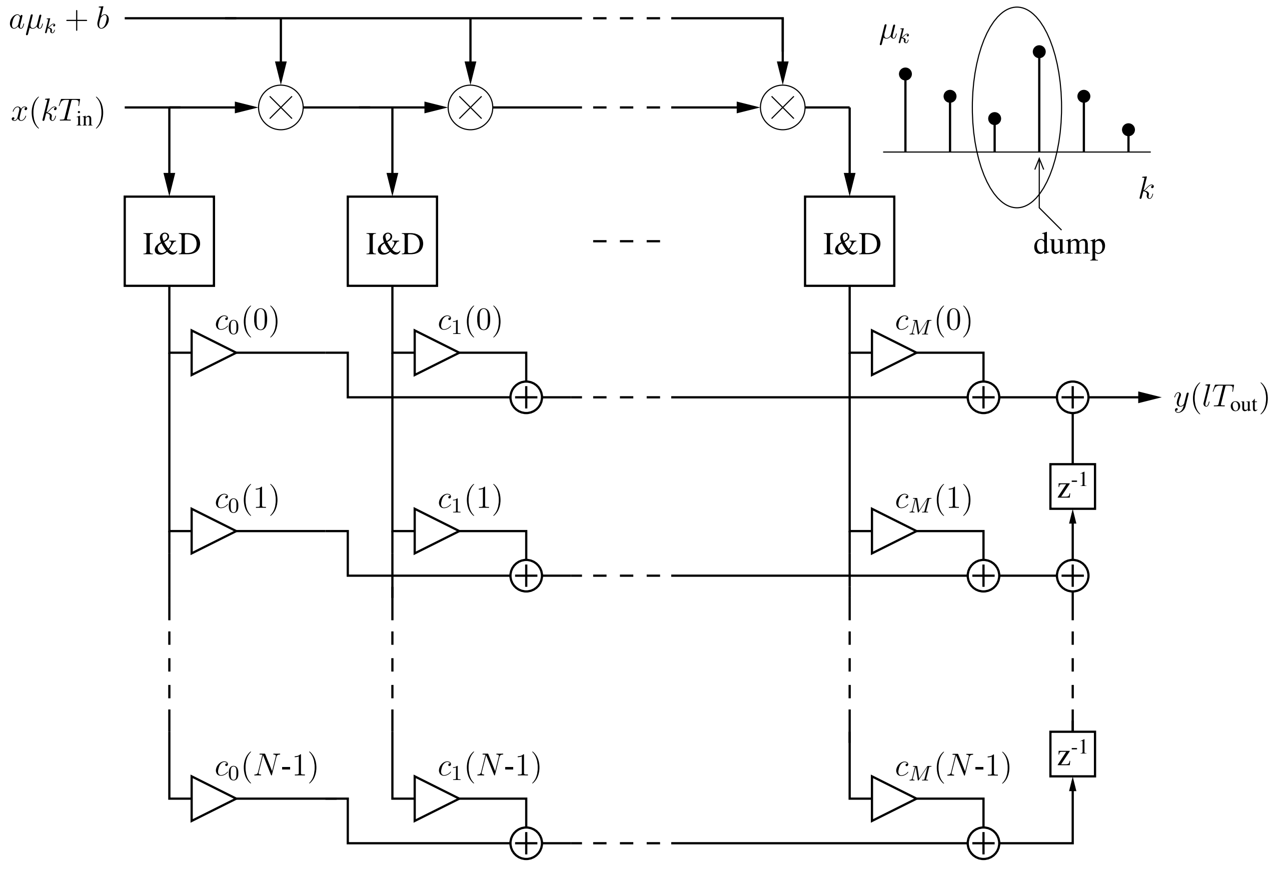

Figure 20: Transposed Farrow Structure

4.3 Quantization of the Fractional Interval

4.3.1 Polynomial Evaluation with a Quantized Fractional Interval

As mentioned in the two previous sections, in implementation, the fractional interval, μ, is computed via an overflowing accumulator. This means that μ must be quantized, or represented with a finite number of bits. The B_q-bit quantized fractional interval is given by (see [50, 51] for a similar derivation)

$$\mu_q = \frac{k_q}{2^{B_q}}, \quad \text{for} \quad 0 \leq k_q \leq 2^{B_q - 1}, \tag{4.21}$$

for $k_q \in \mathbb{Z}$, such that μ_q can take on

$$N_q = 2^{B_q} \tag{4.22}$$

possible values. The effect of quantizing μ on the PBF CT impulse response is shown in Fig. 21, where $B_q = 2$. It is clear that the effect is equivalent to first sampling the CT impulse response with a period of $T_s = T/N_q$ followed by reconstruction with a ZOH. The ZOH has a frequency response given by

$$H_{\text{ZOH}}(\Omega) = \frac{T_s \sin\left(\frac{\Omega T_s}{2}\right)}{\frac{\Omega T_s}{2}}. \tag{4.23}$$

For this particular example, $N_q = 2^2 = 4$, thus the sampled filter has four samples per polynomial piece.

The frequency response of the CT filter, the sampled filter, and the quantized filter are given in Fig. 22, Fig. 23, and Fig. 24, respectively. In [50] it was determined that the maximum level of

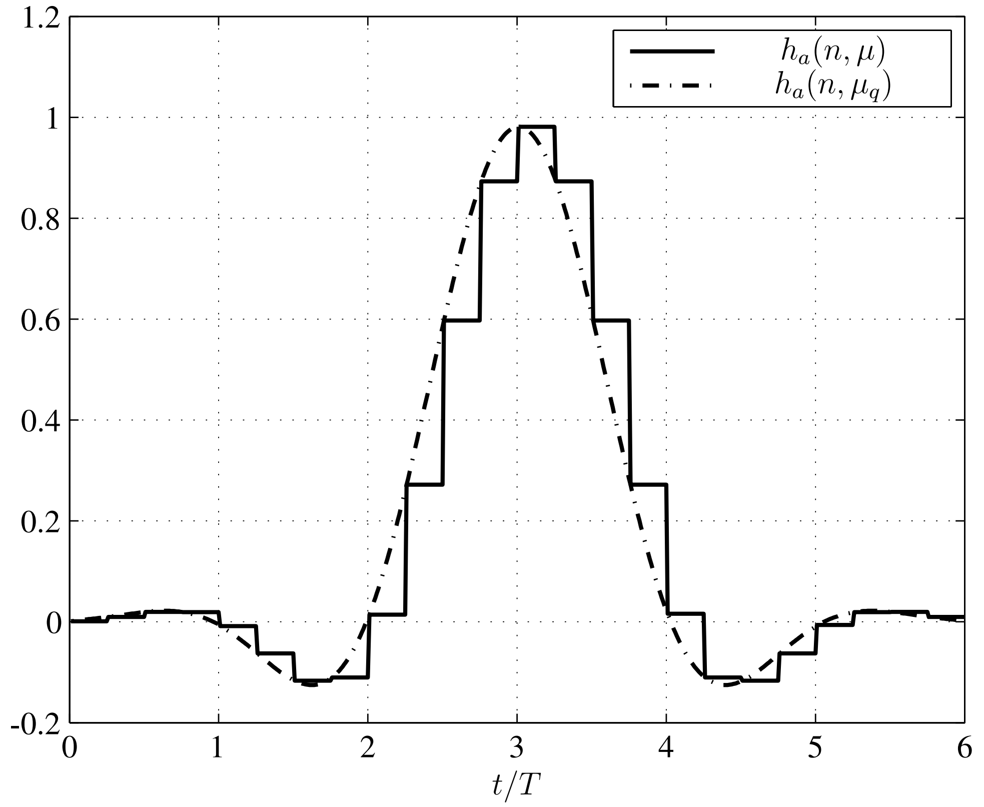

Figure 21: The Effect of 2-bit Quantization of μ on $h_a(t)$

the quantization images relative to the passband in dB can be approximated by

$$\hat{I}_{\text{max}} \approx 20\log_{10}(f_p) - 6.02B_q. \tag{4.24}$$

Alternatively, to keep these images $\hat{I}_{\text{max}}$ dB down, one should use

$$B_q \geq (20\log_{10}(f_p) - \hat{I}_{\text{max}})/6.02. \tag{4.25}$$

bits to represent μ. In the above, f_p is the normalized passband edge frequency.

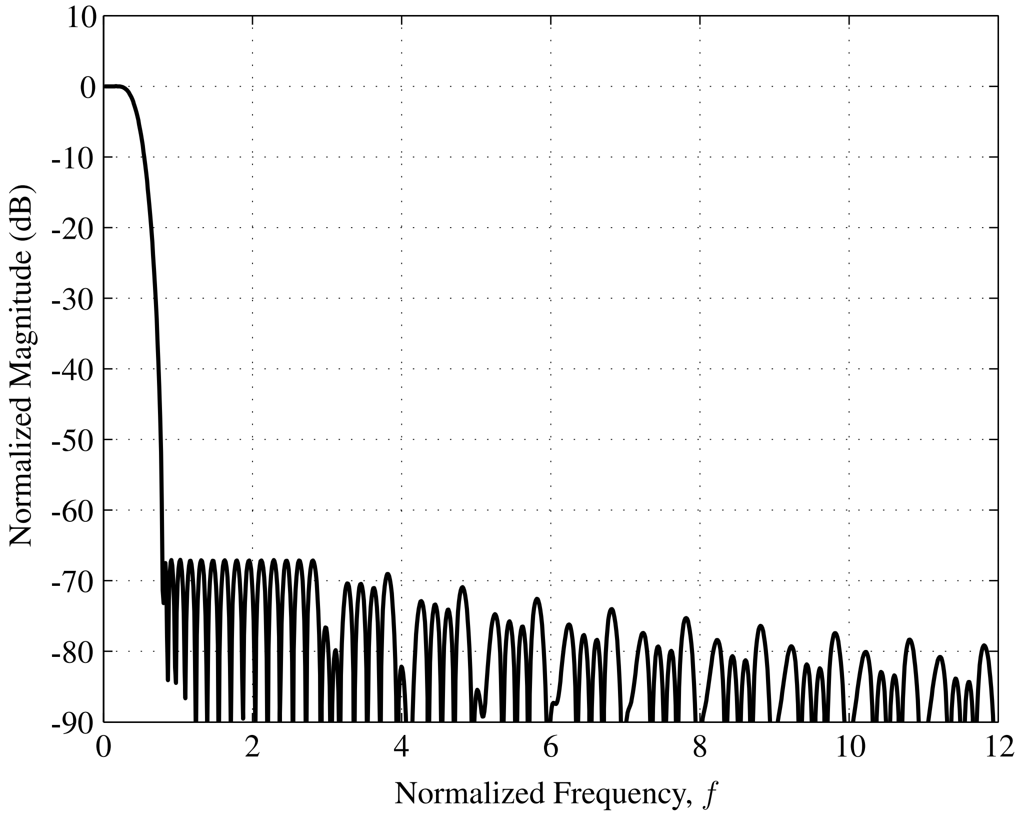

Figure 22: Frequency Response of CT Filter

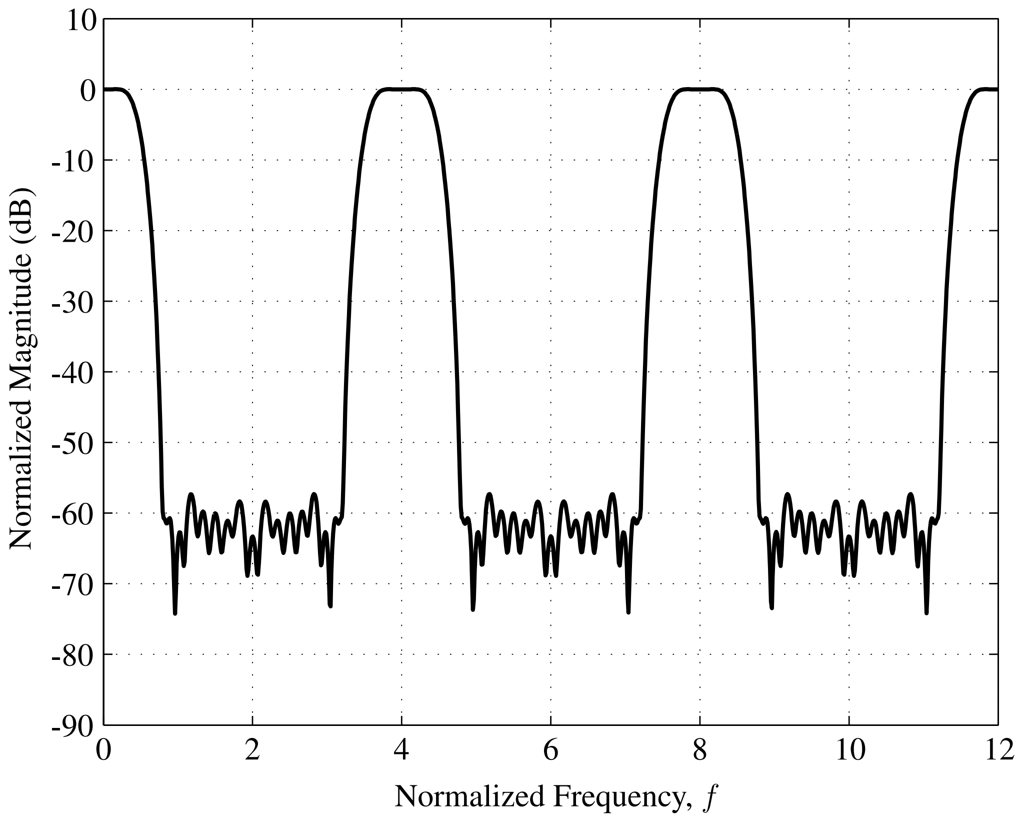

Figure 23: Frequency Response of Sampled Filter

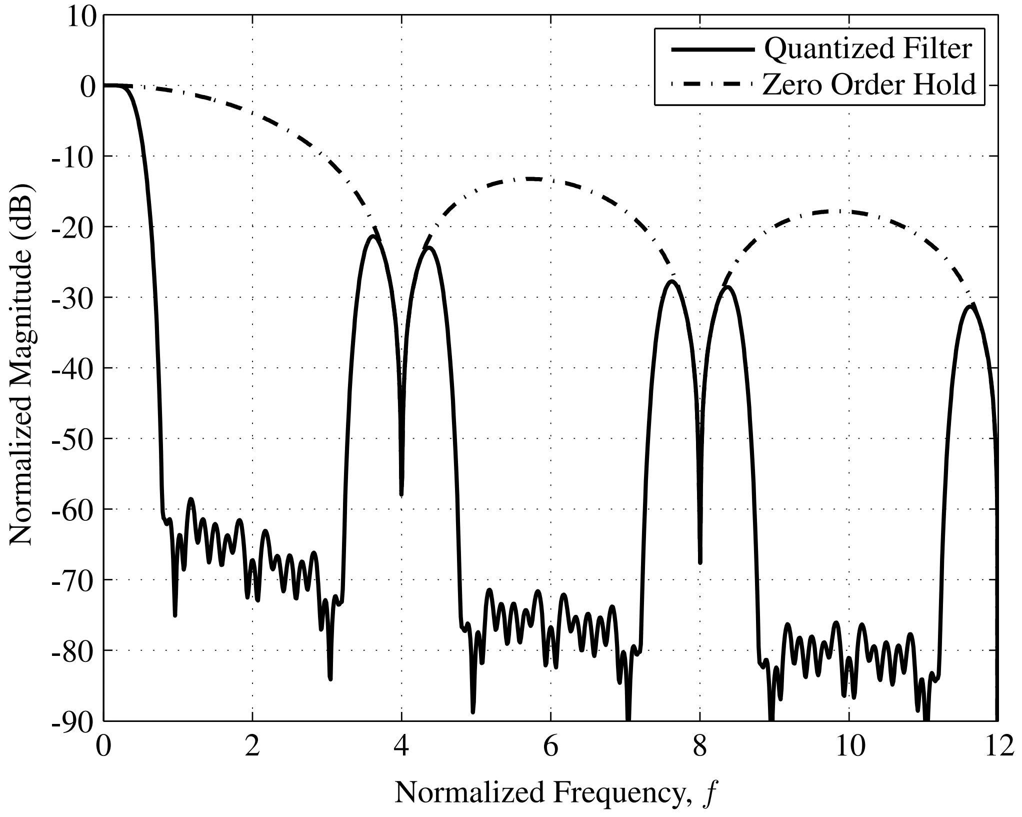

Figure 24: Frequency Response of Quantized Filter

4.3.2 Fractional Interval Generation

From (4.5) the fractional interval, in the case of the Farrow Structure, can be written in terms of the SRC factor (1.5) as

$$\begin{aligned} \mu_l &= \frac{lT_{\text{out}}}{T_{\text{in}}} - \left\lfloor \frac{lT_{\text{out}}}{T_{\text{in}}} \right\rfloor \\ &= \frac{l}{R} - \left\lfloor \frac{l}{R} \right\rfloor . \end{aligned} \tag{4.26}$$

The accumulator of (4.6) then becomes

$$\begin{aligned} \mu_{l+1} &= \mu_l + \frac{T_{\text{out}}}{T_{\text{in}}} - \left\lfloor \mu_l + \frac{T_{\text{out}}}{T_{\text{in}}} \right\rfloor \\ &= \mu_l + \frac{1}{R} - \left\lfloor \mu_l + \frac{1}{R} \right\rfloor . \end{aligned} \tag{4.27}$$

Likewise, from (4.15) the fractional interval, in the case of the Transposed Farrow Structure, can be written in terms of the SRC factor (1.5) as

$$\begin{aligned} \mu_k &= \left\lceil \frac{kT_{\text{in}}}{T_{\text{out}}} \right\rceil - \frac{kT_{\text{in}}}{T_{\text{out}}} \\ &= \lceil kR \rceil - kR \end{aligned} \tag{4.28}$$

The accumulator of (4.20) then becomes

$$\begin{aligned} \mu_{k+1} &= \mu_k - \frac{T_{\text{in}}}{T_{\text{out}}} - \left\lfloor \mu_k - \frac{T_{\text{in}}}{T_{\text{out}}} \right\rfloor \\ &= \mu_k - R - \lfloor \mu_k - R \rfloor \end{aligned} \tag{4.29}$$

Thus the generation of the fractional interval is accomplished via an overflowing accumulator with input $1/R$ or R for the Farrow or Transposed Farrow Structure, respectively.

As mentioned in the previous section, in order to compute μ, it must be quantized. This means that the input to the μ generator, or the SRC factor, must be quantized. The precision used to represent μ determines the precision of the SRC factor, and thus the precision of the SRC itself.

For the remainder of this section, we focus on the Transposed Farrow Structure. Similar derivations can be done for the Farrow Structure. Define the quantized SRC factor as

$$\begin{aligned} R_q = Q[R] &= \frac{k_q}{2^{B_{\text{gen}}}}, \quad \text{for} \quad 0 \leq k_q \leq 2^{B_{\text{gen}}-1}, \\ &= \frac{L}{K}, \end{aligned} \tag{4.30}$$

where L and K are relatively prime integers with $L < K$ (decimation). Note that if $R = P/Q$ is already a rational fraction, it can be represented exactly with

$$B_{\text{gen}} = \lceil \log_2(Q) \rceil \tag{4.31}$$

bits [18]. If R is irrational, or to be continuously variable, then, for a fixed input sample rate the output sample rate precision or resolution can be written in terms of the input sample rate as

$$\Delta F_{\text{out}} = \frac{F_{\text{in}}}{2^{B_{\text{gen}}}}, \tag{4.32}$$

such that, given a desired resolution the number of bits required in the fractional interval accumulator to achieve this resolution can be calculated as

$$B_{\text{gen}} = \left\lceil \log_2 \left(\frac{F_{\text{in}}}{\Delta F_{\text{out}}} \right) \right\rceil \tag{4.33}$$

4.3.3 Combining the Quantization Requirements for Implementation

From the results of the previous two sections, the system given in Fig. 25 can be used to implement the μ generator.

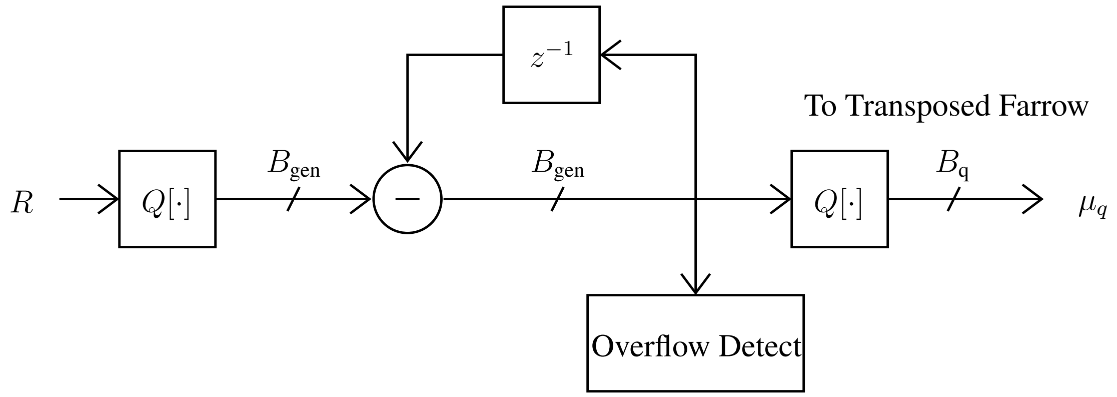

Figure 25: Fractional Interval Generator

Here, two quantizers are employed such that a different number of bits may be used for controlling SRC resolution and ZOH image distortion. This is of major advantage as compared to using one wordlength that satisfies both requirements. From the figure, we see that controlling the SRC resolution only affects the number of bits used in a single accumulator, while the number of bits used for polynomial evaluation are passed to the filter which performs all the shifts, additions, and multiplications. In practice, for continuously variable SRC, a large number of bits may be used in the μ generator to give very fine resolution control over the SRC. Then, these bits may be truncated before being passed to the Transposed Farrow Structure.

To give a specific example, consider the following. Given

1. $F_{\text{in}} = 100$ MHz
2. F_{out} is to be continuously variable with resolution of .001 Hz
3. The PBF has normalized passband edge of $f_p = .2$
4. ZOH image distortion is to be kept at -80dB

From (4.33),

$$B_{\text{gen}} = \left\lceil \log_2 \left(\frac{100 \text{ MHz}}{.001 \text{ Hz}} \right) \right\rceil = 37. \tag{4.34}$$

But, from (4.25)

$$B_q \geq (20 \log_{10}(0.2) - (-80))/6.02 \geq 10.97 \tag{4.35}$$

Therefore,

$$B_q = 11 \tag{4.36}$$

bits are sufficient to keep ZOH images at ≈ -80dB. This example points out the dramatic savings that can be achieved by employing two quantizers instead of one, and this without giving up any of the desired performance.

4.4 The Discrete-Time Model

It has been shown [48], that when implementing SRC by a rational factor $R = L/K$ using PBFs, there exists an exactly equivalent DT model. In fact, in practical implementation R is quantized and thus is always given exactly by a rational factor as shown in (4.30). The DT model for SRC is given in Fig. 26.

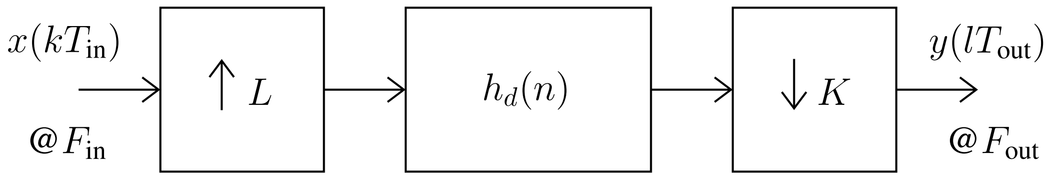

Figure 26: Discrete-Time Model for SRC

In Fig. 26, the input signal is first upsampled by L followed by filtering with a DT FIR filter, $h_d(n)$, whose output is then downsampled by K to obtain the final output. The relationship between the CT designed PBF and the DT filter is given by

$$h_d(n) = h_a(t)\Big|_{t=nT_s} = h_a(nT_s), \tag{4.37}$$

where,

$$T_s = \begin{cases} T_{\text{in}}/L, & \text{for the Farrow Structure} \\ T_{\text{out}}/K, & \text{for the Transposed Farrow Structure} \end{cases} \tag{4.38}$$

This sampling of the CT impulse response results in aliasing. Thus, since the CT filter is not perfectly bandlimited, the DT filter will not match the designed CT filter in the frequency domain.

For example, if $R = 2/3$ for the Transposed Farrow Structure, the impulse response has only 3 samples per polynomial piece, or

$$h_d(n) = h_a\left(n\frac{T_{\text{out}}}{3}\right). \tag{4.39}$$

When this sampling effect occurs during implementation the CT frequency response aliases upon itself yielding the DT filter. This sampling is applied to the example filter from Fig. 22 illustrating this phenomenon.

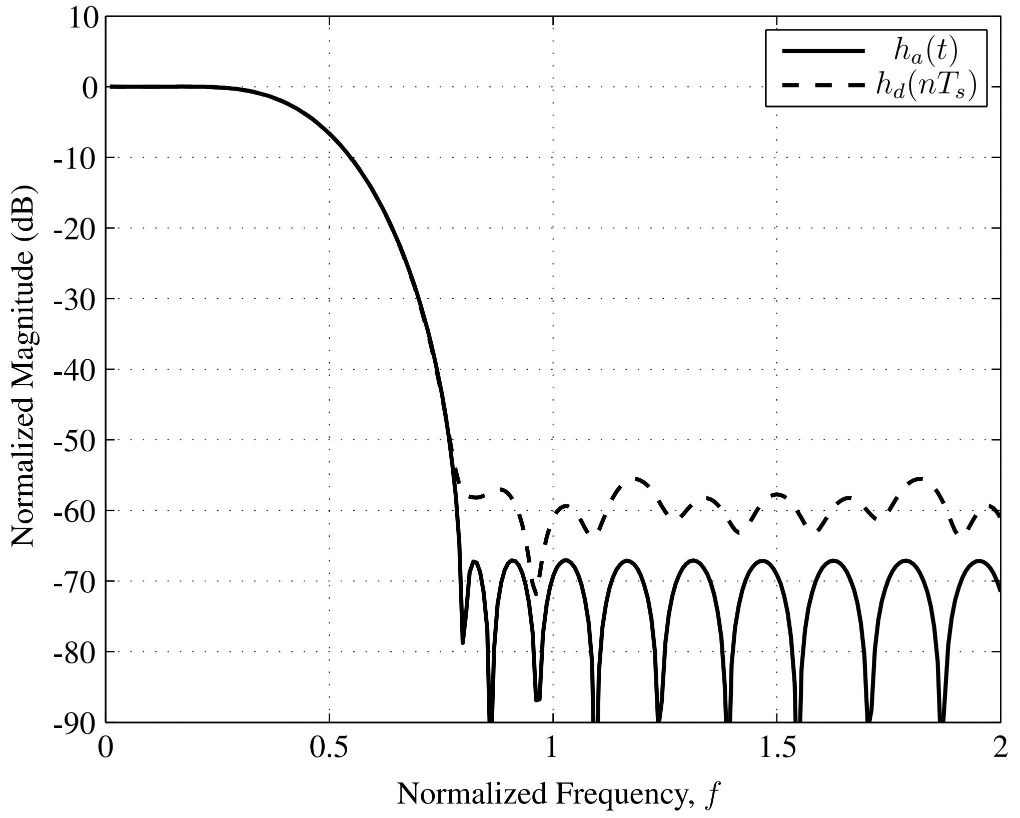

Figure 27: Frequency Response of CT Filter

As can be seen from Fig. 22, the stopband decays as frequency increases. Because of this, as the sample rate of the DT filter increases, the effect of aliasing decreases. This observation leads to the following question: Can the filter be made to decay faster such that the effect of aliasing is not as severe? The answer is in fact yes. Through the design techniques presented in section 3.4.3, the impulse response of a frequency domain optimized PBF can be made to have the γ^{th} derivative continuous. This means the filter will exhibit a decay rate of (see [56, 36])

$$\text{D.R.} = \frac{1}{f^{\gamma+1}}. \tag{4.40}$$

Figure 28 shows the CT frequency response of a multi-band PBF ($M = 5$, $N = 6$) with no continuity constraints as well as a filter of the same order and length whose zeroth derivative is made continuous. The dramatic difference in decay rate is obvious from the figure. It is also worth pointing out that the stopband attenuation starts out at virtually the same level. Figure 29 shows a zoomed in view of the passband where it can be seen that the continuity constraint increases the ripple level by an insignificant amount.

Figure 30 shows the decrease in stopband attenuation versus the normalized sample rate (K for the Transposed Farrow Structure) of the filters of Fig. 28, as well as a filter of the same order and length with its first derivative made continuous. Also, given in the legend is the worst case stopband attenuation the CT Filter achieved from optimization. The plots show how each of the filters deviate from their design as a function of the sample rate. As can be seen, the filters with continuity constraints vary the least. In fact, the filter with its first derivative continuous actually improves in performance when sampled. This comes at the expense of a small initial decrease in the CT designed stopband attenuation due to the degrees of freedom lost in imposing more restrictive time domain constraints.

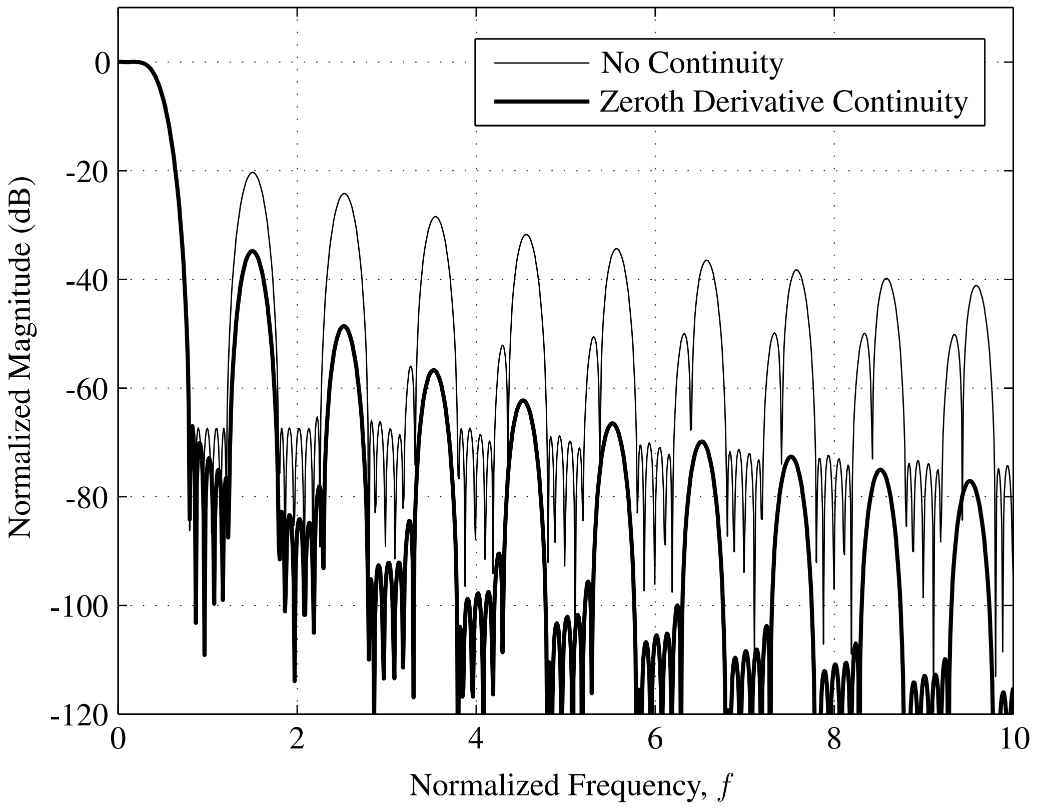

Figure 28: Difference in Filter Decay Rate

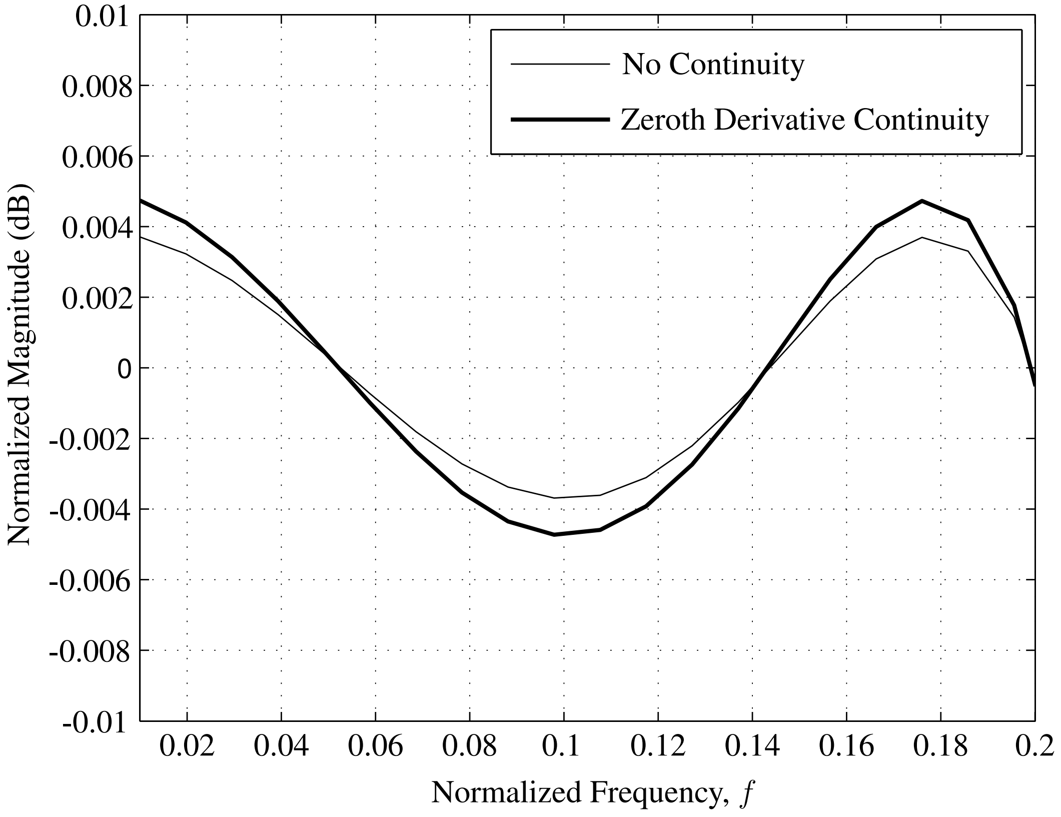

Figure 29: Difference in Filter Ripple Level

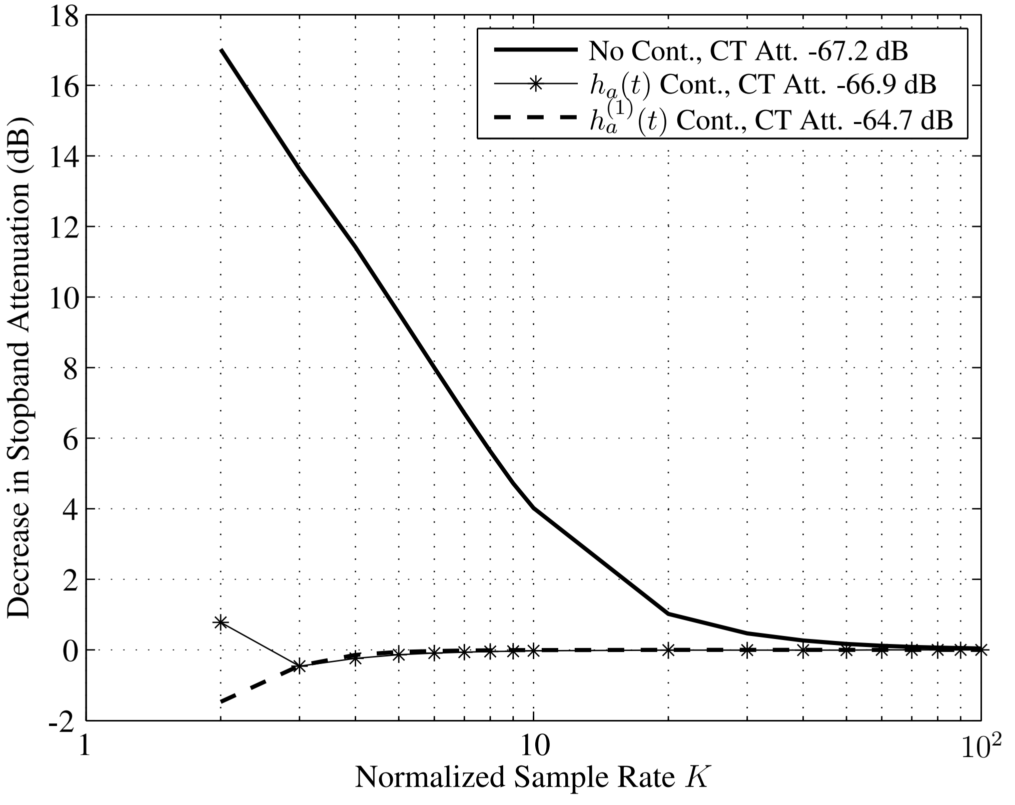

Figure 30: Decrease in Stopband Attenuation Due to Aliasing

The decay rate of the filter also plays an important role in decreasing alias build up in the passband. When the filter is used for decimation, each of the frequency bands centered at multiples of F_{out} aliases into the passband. This phenomenon can be quantified in terms of the Integrated Sidelobe Level (ISL) [36]. The ISL represents the amount of power that will be concentrated in the passband as a result of the summation of all of the aliasing bands. As an illustration, consider the application of decimation by 12. This gives $K = 12$ and $L = 1$. Figure 31 shows the filters from Fig. 28 after being sampled according to (4.37). Also shown are the aliasing bands that will fold over into the passband as a result of decimation. As can be seen from Fig. 31, the filter with

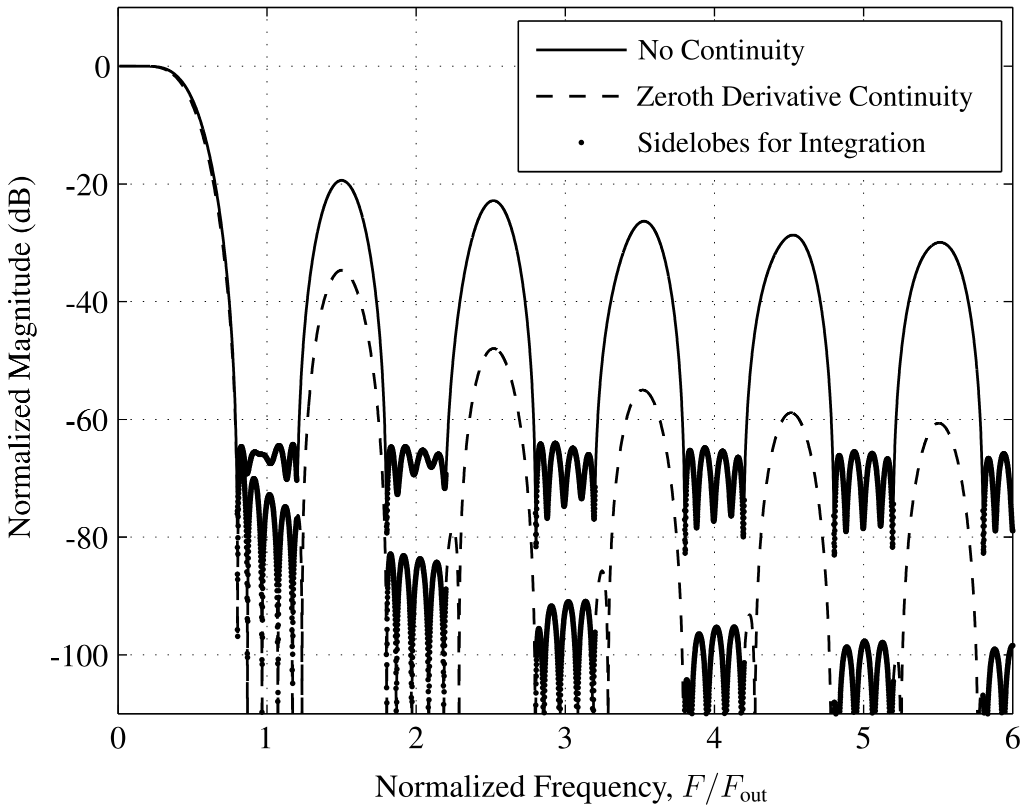

Figure 31: Aliasing in Decimation

continuity constraints retains the CT decay rate. This results in a major performance improvement over the filter with no continuity constraints. The ISL for each filter is given in Table 1.

Table 1: ISL Comparison

Filter Continuity	ISL
None	-56.2 dB
Zeroth Derivative	$\mathbf{-69.7}$ **dB**

Thus, the filter with continuity constraints exhibits an effective alias suppression improvement of 13.5 dB!

CHAPTER 5: APPLICATIONS

5.1 Wideband Digital Downconverters

The flexibility and high speed operation of Field Programmable Gate Arrays (FPGA) have made them ideal targets for real-time DSP algorithms in SIs. Of these algorithms, the Digital Down Converter (DDC) plays a major role. The DDC is the digital signal processing front end of modern Vector Signal Analyzers (VSA) and Spectrum Analyzers (SA) [5]. In an SI platform, the DDC would most likely be found in the digitizer component. The term DDC has come to refer to the process of spectral translation, filtering, and Sample Rate Conversion (SRC). DDC's first tune to a signal's center frequency using a complex valued Numerically Controlled Oscillator (NCO). The NCO is then mixed with the input signal, producing the signal's baseband Inphase (I) and Quadrature (Q) components. SRC is then used to zoom-in to the bandwidth of interest while rejecting signals outside this bandwidth. Because the SRC is required to produce continuously variable output sample rates from a fixed input sample rate over a large range, it becomes the complexity driver of the system.

The SRC subsystem takes input signals sampled at a rate of F_{in} samples per second (sps) and produces output signals sampled at $F_{\text{out}} = F_{\text{in}}/D$ sps. The overall decimation factor is defined as

$$D = F_{\text{in}}/F_{\text{out}} \tag{5.1}$$

where $D \in \mathbb{R}$ which is continuously variable between 1 and some maximum value. Many current systems implement only integer factor SRC in real-time hardware, leaving the non-integer remainder to software for off-line computation [60]. In order to increase measurement speed, other systems implement the complete SRC in real-time [61, 62]. These real-time systems perform the SRC in three steps. First, the input signal is passed through an Anti-Aliasing Filter (AAF). Then, it is resampled by a small, non-integer factor. Finally, the signal is decimated by the remaining integer factor to accomplish the overall SRC. These systems are quite complex [63], and can require multi-chip Application Specific Integrated Circuit (ASIC) implementations [61].

In this section, the SRC of the WDDC is simplified reducing the computational requirements by a factor of three or more. Reducing computational cost translates into hardware implementation savings and the use of FPGAs as the main implementation platform. This is of great advantage in future-proofing SI platforms because FPGAs provide the flexibility to make algorithm upgrades and add new functions simply by reprogramming.

5.1.1 Wideband Digital Downconverters

First, the term wideband must be defined. By wideband, we mean that the maximum bandwidth supported is on the order of the Analog-to-Digital Converter (ADC) sample rate. This is in contrast to narrowband where the signal bandwidth is much less than the sample rate. The wideband nature of SI digitizers precludes the exclusive use of the power/resource efficient techniques used in narrowband systems. A block diagram of a WDDC illustrating the "Tune & Zoom" effect is given in Fig. 32.

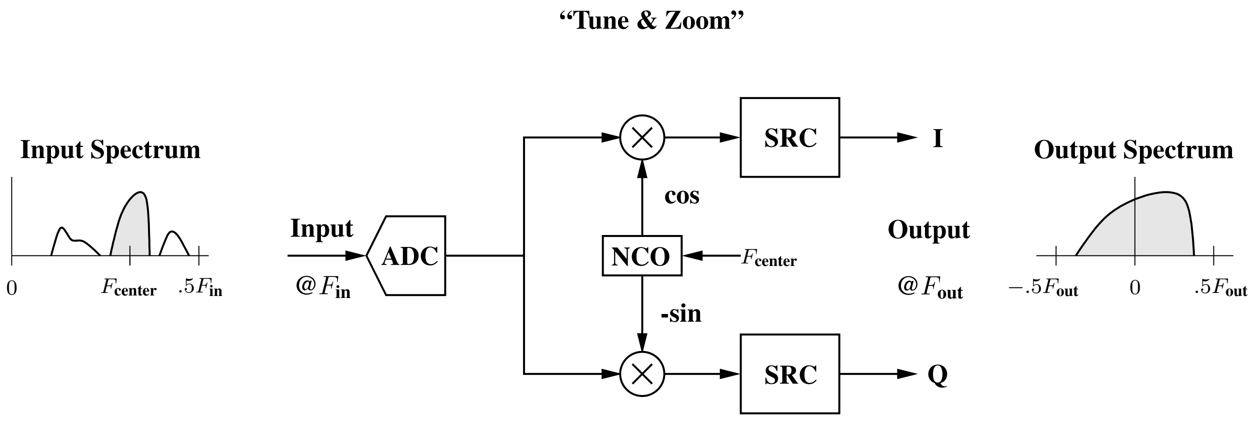

Figure 32: Wideband Digital Downconverter

The system complexity is driven by the SRC. The SRC must be able to produce continuously variable output sample rates from a fixed input sample rate while protecting the bandwidth of interest from aliasing. The output sample rate determines the available output or zoom bandwidth. The zoom bandwidth is a constant fraction of the output sample rate, a typical choice being $.8F_{\text{out}}$.

5.1.1.1 Current Implementations

Current Implementations of the real-time WDDC are based on the block diagram given in Fig. 33, examples of which are found in [61,62].

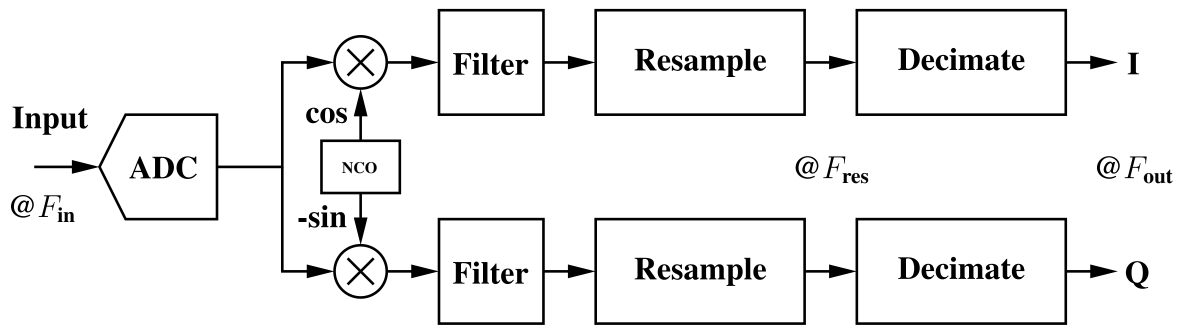

Figure 33: Signal Processing in Current DDCs

The Filter block is a band-limiting filter that reduces the input bandwidth to prevent the resampler from aliasing. The Resample block then reduces or increases the sample rate by a non-integer factor between 1 and 2. The Decimation block completes the SRC by reducing the sample rate by an integer factor.

The resampler has a frequency response normalized to the input sample rate. This means that if the resampler is used for variable sample rate reduction, the filter preceding it must band-limit the input signal by a variable amount to prevent aliasing, as in [61]. This is because resamplers based on the input sample rate are anti-imaging filters, not anti-aliasing filters [5]. The resampling process causes the input spectrum to replicate at integer multiples of F_{res}, where F_{res} is the output sample rate of the resampler. In order to protect a zoom bandwidth $\in (-.4F_{\text{res}}, .4F_{\text{res}})$ from aliasing, the input must be band-limited to $\pm .6F_{\text{res}}$ before resampling. This will keep the spectral replicas located at multiples of F_{res} from overlapping in the zoom bandwidth.

The Decimation block typically performs sample rate reduction by powers of two [64]. This can be accomplished by cascading N_{filts} stages of identical decimate-by-two filters. This provides output sample rates of $F_{\text{out}} = F_{\text{res}}/k$, where k ranges from 2^0 to $2^{\text{N}_{\text{filts}}}$.

A typical set of design specifications for the SRC allow no more than .1 dB of ripple in the zoom bandwidth, attenuate aliasing components by at least 80 dB, and provide a linear phase response. The variable band-limiting filter can be implemented with the same structure as a fixed Finite Impulse Response (FIR) filter by computing the coefficients at setup time [42]. This is done by first designing two optimal prototype filters. The coefficients of the prototype filters can be used to compute the coefficients of the tunable filter. The coefficients of the tunable filter are a simple function of the cutoff frequency and allow fast computation at setup. As shown in Fig. 34, a filter of length 51 is sufficient to bandlimit the input signal by a variable amount, while still meeting the attenuation and ripple specifications.

Ignoring the resampler for now, we move to the Decimation block. Halfband FIR filters are particularly suited to decimation by two [24, 36]. When the filter is decomposed into its two polyphase subfilters for implementation, one subfilter reduces to a delay and scale. This reduces the multiplications required by two. A 47 tap Halfband filter meets the specifications given above.

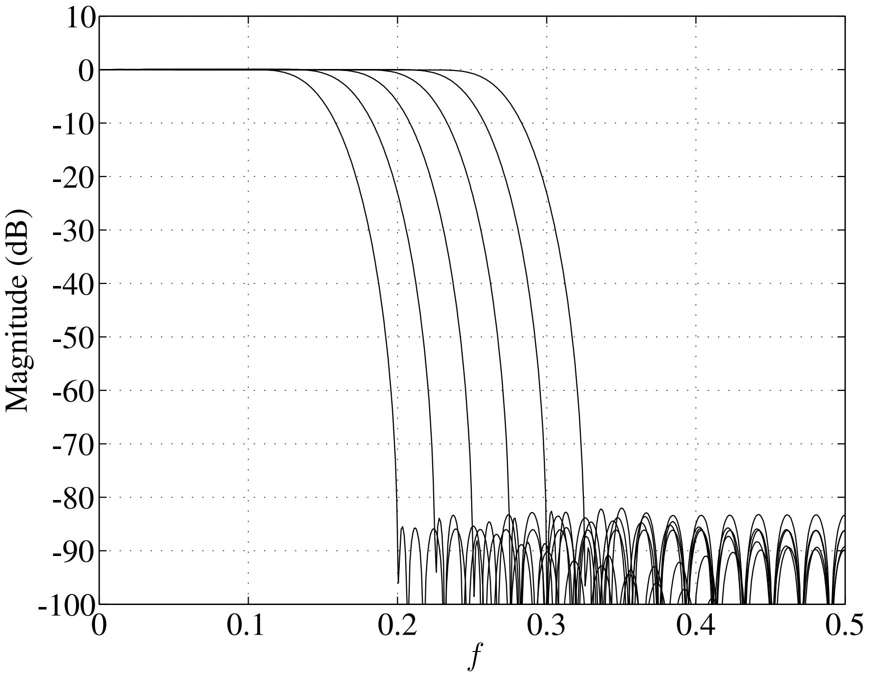

Figure 34: Tunable FIR filter

5.1.1.2 High Performance, Computationally Efficient WDDCs

In this section a new high performance, computationally efficient WDDC is presented. The optimal sample rate converter for lowering the sample rate should have a frequency response normalized to its output sample rate. This is illustrated in Fig. 35.

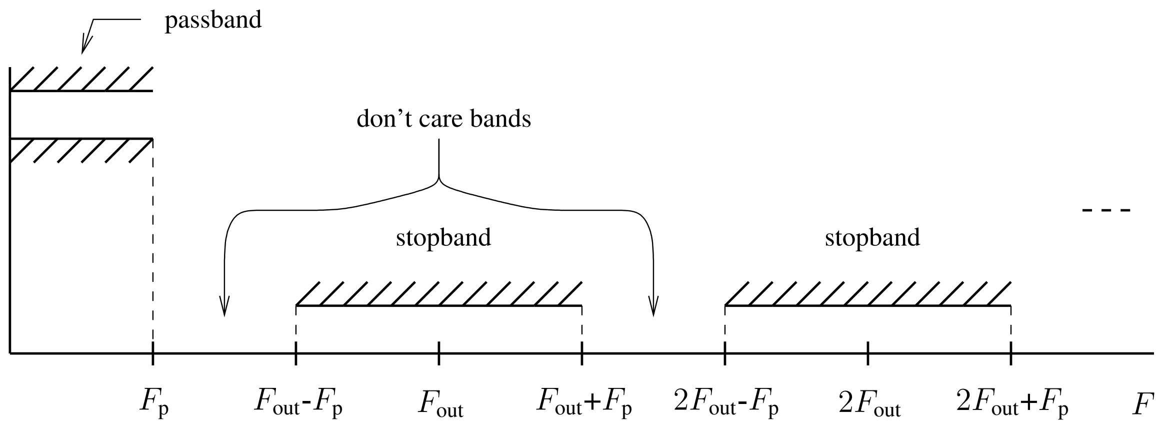
passband
don't care bands
stopband
stopband
- - -
F_p
F_out-F_p
F_out
F_out+F_p
$2F_\text{out}$-F_p
$2F_\text{out}$
$2F_\text{out}$+F_p
F

Figure 35: SRC Passband/Stopband Requirements

Here, $F_\text{p} = .4F_\text{out}$ is the passband edge of the zoom bandwidth. A SRC system with this type of frequency response protects the zoom bandwidth from aliasing regardless of changes in the output sample rate.

A resampler with frequency response normalized to its output sample rate, F_res, can be designed using a Polynomial-Based Filter (PBF). The PBF implementation structure can be derived from the Continuous-Time (CT) model for resampling given in Fig. 4. As shown section 4.2 this results in the so-called Transposed Farrow Structure. The derivation is summarized as follows.

First, inserting (3.5) into (5.2) with $T = T_{\text{res}}$.

$$y(lT_{\text{res}}) = \sum_{k=-\infty}^{\infty} x(kT_{\text{in}})h_a(lT_{\text{res}} - kT_{\text{in}}). \tag{5.2}$$

where, with $t = lT_{\text{res}} - kT_{\text{in}}$

$$h_a((n+\mu_k)T_{\text{res}}) = \sum_{m=0}^{M-1} c_m(n)\,(a\mu_k + b)^m \tag{5.3}$$

for

$$nT_{\text{res}} \le lT_{\text{res}} - kT_{\text{in}} < (n+1)T_{\text{res}}, \tag{5.4}$$

and zero otherwise. From section 4.2, μ_k is the distance between the desired output sample and the current input sample as a fraction of the output period and is termed the fractional interval. It is given by

$$\mu_k = \left\lceil \frac{kT_{\text{in}}}{T_{\text{res}}} \right\rceil - \frac{kT_{\text{in}}}{T_{\text{res}}}. \tag{5.5}$$

In the above, $\lfloor\cdot\rfloor$ and $\lceil\cdot\rceil$ are the floor and ceiling operators, respectively. The relationship between μ_k and the input and output samples is illustrated in Fig. 36.

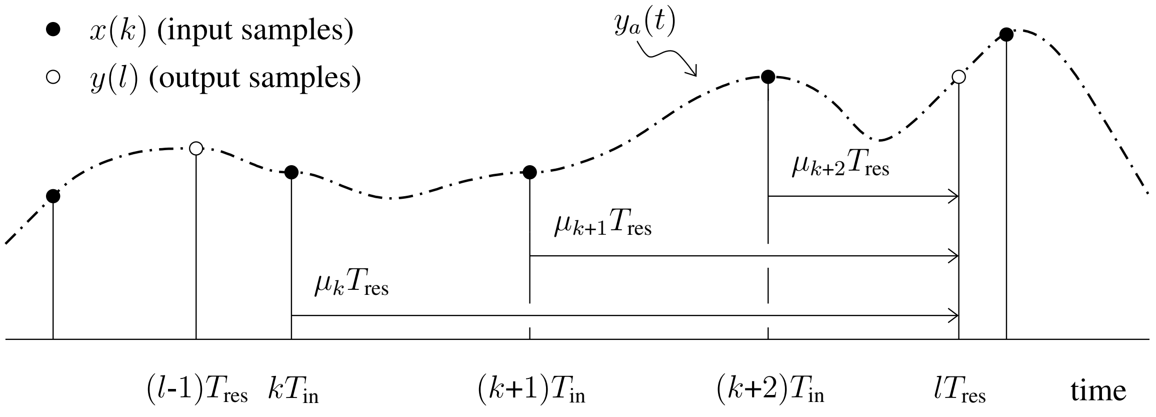
● $x(k)$ (input samples)
○ $y(l)$ (output samples)
$y_a(t)$
$\mu_{k+2}T_{\text{res}}$
$\mu_{k+1}T_{\text{res}}$
$\mu_k T_{\text{res}}$
$(l\text{-}1)T_{\text{res}}$
kT_{in}
$(k\text{+}1)T_{\text{in}}$
$(k\text{+}2)T_{\text{in}}$
lT_{res}
time

Figure 36: Resampler Input / Output Sample Time Relationship

Substituting (5.3) into (5.2) gives the final form

$$\begin{aligned} y(lT_{\text{res}}) &= \sum_{k=k_1}^{k_2} \sum_{n=0}^{N-1} x(kT_{\text{in}}) h_a((n+\mu_k)T_{\text{res}}) \\ &= \sum_{n=0}^{N-1} \sum_{m=0}^{M} \left(\sum_{k=k_1}^{k_2} x(kT_{\text{in}}) \left(a\mu_k + b\right)^m \right) c_m(n). \end{aligned} \tag{5.6}$$

An implementation of (5.6) is given in Fig. 37. The I&D (Integrate-and-Dump) blocks perform the summation over k. Multiple input samples can be involved in the computation of each output sample. One way to sum over the correct input samples is to implement the computation of μ_k as a down-counter/subtracting accumulator and use the inherent wrap around to issue a "dump" command to the I&D blocks. In this scenario μ_k is computed upon the arrival of each new input sample as

$$\mu_{k+1} = \mu_k - \frac{T_{\text{in}}}{T_{\text{res}}} - \left\lfloor \mu_k - \frac{T_{\text{in}}}{T_{\text{res}}} \right\rfloor . \tag{5.7}$$

Since this sequence is strictly decreasing over each T_{res} interval, the accumulator output can be monitored for the condition $\mu_{k+1} \geq \mu_k$ indicating wrap around. When this condition is met, the outputs of the I&D blocks are passed to the Farrow coefficient network which computes a new output sample by performing the remaining summations over m and n.

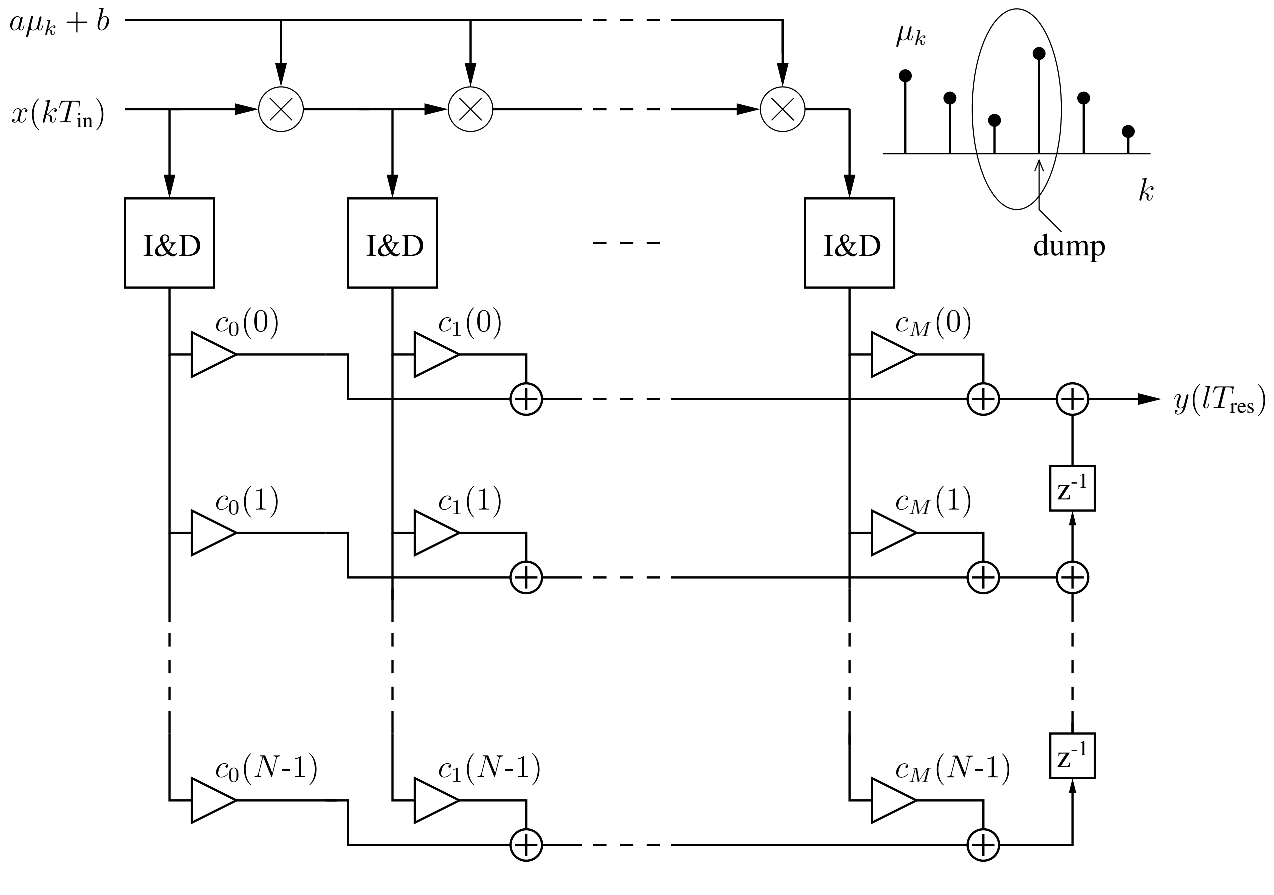

Figure 37: Transposed Farrow Structure

Further simplifications can be made by restricting $h_a(t)$ to be symmetric. This yields the linear phase property we desire and has the added benefit of reducing the number of fixed coefficients by two (see [44, 59], and section 3.3.1). For a symmetric $h_a(t)$, the cost of implementation is M multiplications at F_{in} and $(M+1)N/2$ multiplications at F_{res}. As an aside, it is interesting to note that given a particular rational SRC factor, the Transposed Farrow Structure has an exactly equivalent polyphase filter implementation (See [48] for the details.).

Given the same set of specifications as in the previous section, .1 dB ripple and 80 dB of attenuation, a polynomial based filter with $M = 4$ and $N = 6$ in cascade with a single halfband filter meet the specifications. The ripple and attenuation specifications can be met with a halfband filter of 47 taps. The system block diagram is given in Fig. 38, and the frequency response is given in Fig. 39. In Fig. 39, the passband, extending from zero to $.4F_{\text{out}}$, passes the desired zoom bandwidth. The stopbands, located at integer multiples of F_{out}, protect the zoom bandwidth from aliasing by attenuating the aliasing bands by more than 80 dB.

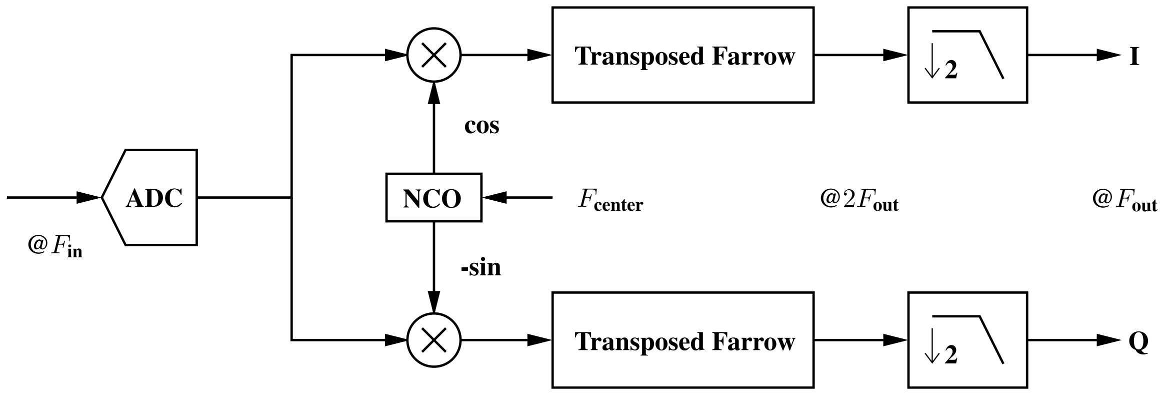

Figure 38: High Performance, Computationally Efficient WDDC

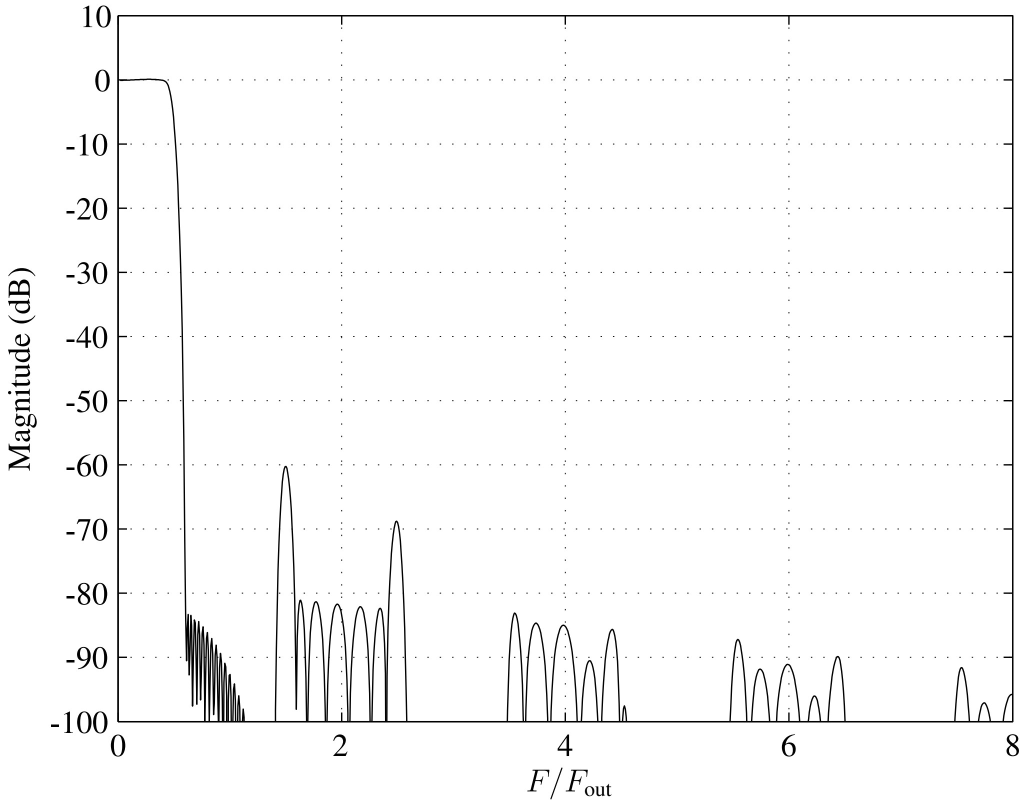

Figure 39: Freq. Response of Proposed WDDC $F_{\text{out}} = .5F_{\text{res}}$

5.1.2 Computational Complexity Comparison

In order to compare the two types of systems, we consider the worst case multiplication rate in terms of Multiplications Per Input Sample (MPIS). The system parameters are:

1. Zoom Bandwidth $= .8F_{\text{out}}$
2. Less than .1 dB of ripple in the Zoom Bandwidth
3. ADC sample rate $F_{\text{in}} = 100$ Msps
4. Output sample rate range 100 Msps to 10 ksps

The number of MPIS of the example current implementation given in section 5.1.1.1 can be computed as,

$$\text{MPIS}_{\text{curr}} = \text{N}_{\text{filter}} + \text{N}_{\text{res}} + \text{N}_{\text{dec}} \tag{5.8}$$

where N_{filter}, N_{res}, and N_{dec} are the worst case MPIS required by the Filter block, the Resample block, and the Decimate block from Fig. 33, respectively. The worst case multiplication rate occurs when $F_{\text{res}} \approx F_{\text{in}}$. As shown in section 5.1.1.1, the Filter block could be implemented with a 51-tap filter. This results in $\text{N}_{\text{filter}} = 51$ MPIS. The number of MPIS from the Decimate block can be computed as,

$$\text{N}_{\text{dec}} = 12.5\left(\sum_{k=0}^{\text{N}_{\text{filts}}-1} \frac{1}{2^k}\right) \tag{5.9}$$

where the number of halfband filters required to perform the largest rate change is $\text{N}_{\text{filts}} = 13$, and each halfband filter is 47-taps. This gives $\text{N}_{\text{dec}} \approx 25$ MPIS. Allowing the current system to require the same number of computations as the new system for the resampler, the total becomes

$$\text{MPIS}_{\text{curr}} = 95. \tag{5.10}$$

Now, let us consider the new system proposed in section 5.1.1.2. The worst case number of MPIS is $\approx M + (M+1)N/2$ for the resampler and 12.5 for the halfband filter. The total for the proposed system is given by

$$\text{MPIS}_{\text{new}} = 19 + 12.5 = 31.5. \tag{5.11}$$

Thus, the proposed system significantly reduces the computational requirements! These results are summarized in Table 2.

Table 2: Computational Complexity in MPIS

	Filter	Resample	Decimate	**Total**
Current	51	19	25	**95**
New	—	19	12.5	**31.5**

5.1.3 Simulation

To illustrate the operation of the proposed system of Fig. 38 as designed in Section 5.1.1.2, a simulation was carried out. A zoom bandwidth of 10.24 MHz centered at $F_{\text{center}} = 21.4$ MHz is to be analyzed. There also exists an unwanted signal that will alias on top of the desired signal due to SRC. This is done to illustrate the effectiveness of the system in providing anti-aliasing. For a zoom bandwidth of 10.24 MHz, the output sample rate is

$$F_{\text{out}} = (10.24 \text{ Msps})/.8 = 12.8 \text{ Msps}, \tag{5.12}$$

giving a decimation factor of

$$D = F_{\text{in}}/F_{\text{out}} = (100 \text{ Msps})/(12.8 \text{ Msps}) = 7.8125. \tag{5.13}$$

Fig. 40 is a plot of the spectrum at 3 locations in the system: the input, the output of the mixer, and the final output of the system. The middle plot shows the desired signal band centered at zero and an unwanted signal at F_{out}. The unwanted signal will fold on top of the desired signal as a result of the resampling process. This is shown in the bottom plot, where it can be seen that the desired signal is intact, while the aliasing band is in the noise floor, having been attenuated by more than 80 dB.

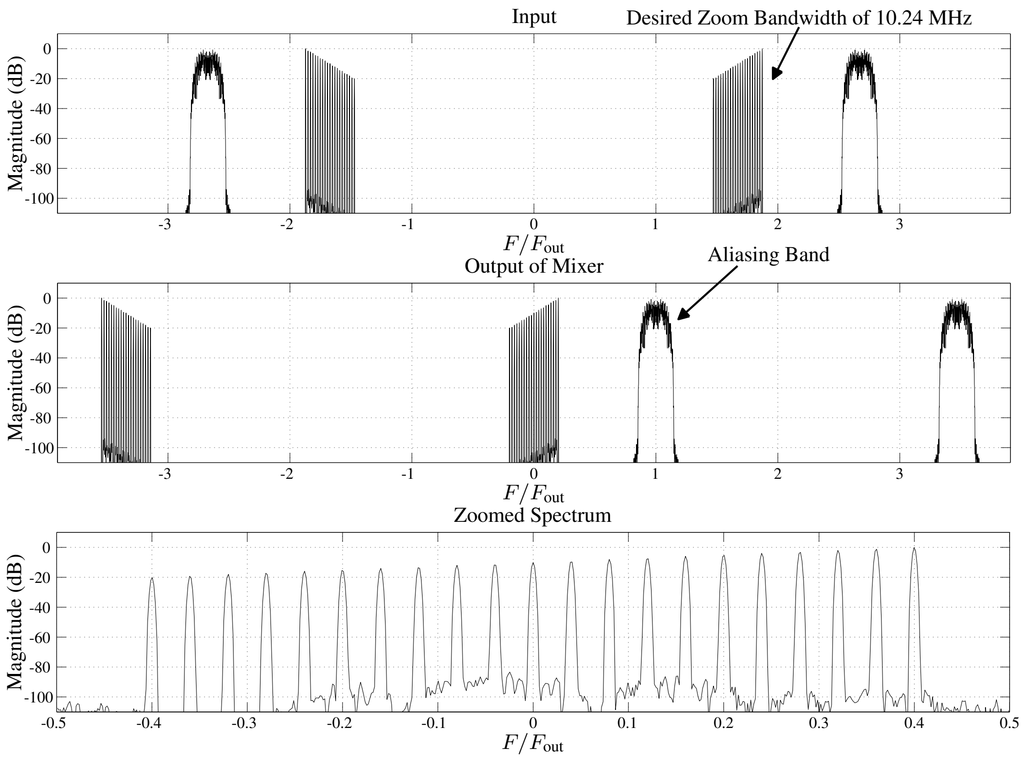

Figure 40: Spectral Zooming with Decimation Factor $D = F_{\text{in}}/F_{\text{out}} = 7.8125$

5.1.4 FFT-Based Spectral Analysis

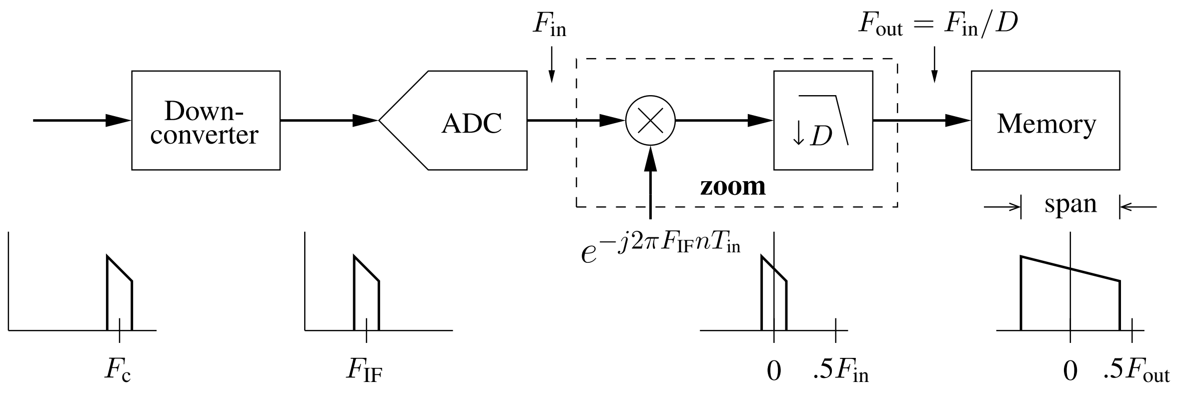

Figure 41: FFT-Based Spectrum Analyzer and Associated Spectra

A major application of the WDDC is FFT-based spectral analysis. A high level block diagram of an FFT-based spectrum analyzer is given in Fig. 41. Some of the basic parameters required to make a measurement are center frequency (F_{c}), span, and frequency resolution (ΔF). The downconverter translates the desired F_{c} to an intermediate frequency (F_{IF}) for digitization. The analog to digital converter(ADC) samples at fixed rate, F_{in}, and typically supports a bandwidth of $0.4F_{\text{in}}$. This, along with the final IF filter, sets an upper bound on the instantaneous bandwidth (or span) of the spectrum analyzer. In order to view larger spans, multiple spans of the maximum instantaneous bandwidth or less are pasted together. After digitization, the FFT could be directly computed and displayed. If the desired span was less than the instantaneous bandwidth, only the desired portion would be displayed. This technique is shown to be inefficient upon observing the relationship between sample rate, span, and frequency resolution.

The relationship between sample rate and frequency resolution is given in (5.14).

$$\Delta F = \frac{F_{\text{in}}}{N_{\text{samp}}} \tag{5.14}$$

N_{samp} is the number of samples collected for FFT computation [25]. From (5.14) we see that in order to increase the frequency resolution(i.e., decrease ΔF) we can lower the sample rate and/or increase the number of samples. Anytime the desired span is less than the maximum instantaneous bandwidth, the sample rate may be lowered while still preserving the span. This means that fewer samples can be collected for a given resolution by lowering the sample rate (decimation), thereby reducing the amount of computations and memory required to perform the measurement. As an example, consider a system with a 30 MHz instantaneous bandwidth and a 65 MegaSample Per Second (Msps) ADC. If it is desired to compute the spectrum over a 10 kHz span with 10 Hz frequency resolution and the sample rate is left unchanged,

$$N_{\text{samp}} = 65 \;\text{ Msps}/10 \;\text{ Hz} = 6,500,000 \;\text{ samples} \tag{5.15}$$

would be required. On the other hand, if the sample rate was lowered to 12.5 kHz, the number of samples required would be

$$N_{\text{samp}} = 12.5 \;\text{ kHz}/10 \;\text{ Hz} = 1250 \;\text{ samples.} \tag{5.16}$$

This clearly merits the use of an adjustable sample rate converter. This insight led to the development of “zoom” spectral analysis [65, 66]. In zoom spectral analysis, a complex digital downconversion is performed followed by decimation by D before FFT computation. This operation is illustrated in Fig. 41 as indicated by the dashed box. A key issue in performing this operation is anti-aliasing, wherein the desired span must be protected from interference. This is readily accomplished by constructing the sample rate converter as a PBF as described in section 5.1.1.2.

5.2 Arbitrary Waveform Generators

The stimulus functionality of a SI is comprised of a digital source, Digital to Analog Conversion (DAC), and an Up Converter (UC) [67]. The DAC can often reside on an Arbitrary Waveform Generator (AWG) providing the baseband / Intermediate Frequency (IF) source to the UC. The digital source provides the flexible modulation functionality while the upconverter provides the frequency translation functionality.

Digital Signal Processing (DSP) plays a central role in the implementation of SIs. Moving as many signal processing tasks as possible from the analog to the digital domain makes for a more flexible, future-proof system. Advances in DSP can also be exploited to reduce the complexity of, or remove completely, the analog components.

Sampling theory provides the foundation for making efficient, high performance AWGs. Samples taken from a Continuous-Time (CT) signal at a periodic rate are sufficient to perfectly reconstruct the CT signal, provided a few conditions are met [25]. This is how AWGs work. CT signals or waveforms are synthesized by reconstructing signal samples stored in memory. The maximum time length of a waveform is given by the product of the time spacing of the samples, or sample period, and the maximum number of samples that can be stored in memory. Thus, for a fixed amount of memory, the maximum signal duration can be increased by increasing the sample period of the waveform, or equivalently by decreasing the sample rate. Therefore, from a storage savings perspective, it is highly desirable to have the sample rate of the waveform as low as possible. This is the point at which sampling theory provides valuable information. Given that the CT signal to be reconstructed is a lowpass strictly bandlimited signal, then the minimum sample rate required to reconstruct the signal is twice the signal bandwidth. For practical reasons the minimum sample rate is usually selected to be slightly higher than the theoretical minimum. Nevertheless, this insight can be used to maximize the storage efficiency and playback time of an AWG.

The majority of AWGs simply clock waveform samples from memory into a Digital-to-Analog Converter (DAC) at the DAC sample rate which is equal to the waveform sample rate [68, 69]. DACs create analog signals by taking input samples and producing an analog voltage proportional to the sample amplitude at the output. These voltage levels change only when a new input sample arrives, thus producing a stair-step approximation to the original CT signal. These stair-steps in the time domain correspond to what are called images in the frequency domain. These images are spectral replicas of the original signal occurring at integer multiples of the sample rate. These images must be filtered out to reconstruct the original signal. Herein lies a problem. If the sample rate changes, the position of the images changes, therefore the filter must change. This conflicts with the previous observation of the benefits of having the minimum sample rate possible. Fixing the sample rate in this type of system requires all waveforms to be stored in memory at that rate. Since it is an *arbitrary* waveform generator, it is desirable to provide the capability to synthesize narrowband as well as wideband signals. Thus, fixing the waveform sample rate to accommodate all signals would require narrowband signals to be highly oversampled. As has been shown, this dramatically increases the memory requirements and consequently decreases waveform playback time. Due to this tradeoff, most currently available AWGs provide a maximum sample rate that can be divided down by an integer factor in conjunction with a few switchable analog filters for CT signal reconstruction [69].

An alternative to this approach is to use Field Programmable Gate Array (FPGA) based Real-Time Signal Processing (RTSP) to increase the sample rate by an integer factor in the digital domain as an intermediate step between the waveform memory and the DAC. In this manner, the waveform can be stored at the minimum rate, but can be clocked into the DAC at a fixed higher rate. In this manner, only one analog reconstruction filter is required at the DAC output. This approach achieves memory and hardware savings, yet is inflexible in the choice of sample rates. Another similar approach uses two hardware sample clocks. One sample clock provides a spectrally pure integer submultiple of the maximum clock rate for high performance, and the other offers the flexibility of high resolution frequency selection at the expense of spectral purity. See [70, 71, 68, 72] for examples of both types.

In this section, advanced Digital Signal Processing (DSP) techniques are used to create a flexible, high performance AWG. The new approach provides added flexibility and performance while simultaneously reducing analog hardware complexity. The proposed system allows waveforms to be stored in memory at a minimum sample rate which does not have to be related to the DAC sample rate. Using real-time arbitrary factor interpolation the waveform sample rate is increased to a fixed DAC sample rate. This technique allows an essentially infinite number of waveform sample rates. Since the DAC sample rate is fixed a single analog filter can be used to reconstruct the CT signal.

5.2.1 New Flexible AWGs with RTSP

In order to provide a flexible, high performance AWG without increasing hardware complexity, we propose the use of real-time FPGA-based interpolation in conjunction with a single, fixed frequency DAC sample clock. Since the DAC sample clock is fixed, the proposed system must be able to increase the sample rate of the waveform stored in memory by an arbitrary factor R. Thus, real-time continuously variable Sample Rate Conversion (SRC) by arbitrary factors is employed. Moreover, to make SRC possible, a procedure is devised to request samples from waveform memory at the proper time instants.

In this work, a PBF as described in Chapter 3 is used as the real time interpolator in the design of the system. Polynomial-based filters are an attractive option for continuously variable sample rate conversion by arbitrary factors. This class of filters finds efficient real-time implementation in the Farrow Structure of Fig. 18 [38,29,30,58] and its variants [5,40,46,44,43,18,73]. They can also be designed directly from a set of frequency domain specifications [31]. The computational cost is $(M+1)N$ multiplications at the lower input sample rate, and only M multiplications at the higher output sample rate. As shown in Chapter 3, constraining the impulse response to be symmetric about $(N/2)T_{\text{in}}$ yields a linear phase filter with symmetric polynomial coefficients. This reduces the number of fixed coefficients required for implementation resulting in $(M+1)N/2$ multiplications at the input sample rate for N even.

A key component in the system is the computation of the fractional interval, μ_l. An overflowing accumulator is employed to compute μ_l as given in (4.6) which runs at the fixed DAC sample rate

$$F_{\text{DAC}} = F_{\text{out}} = \frac{1}{T_{\text{out}}}. \tag{5.17}$$

Since the waveform sample rate is increased, multiple output samples can be generated for each input sample. This is illustrated in Fig. 42, where it can be seen that the μ values are increasing over each input sample period.

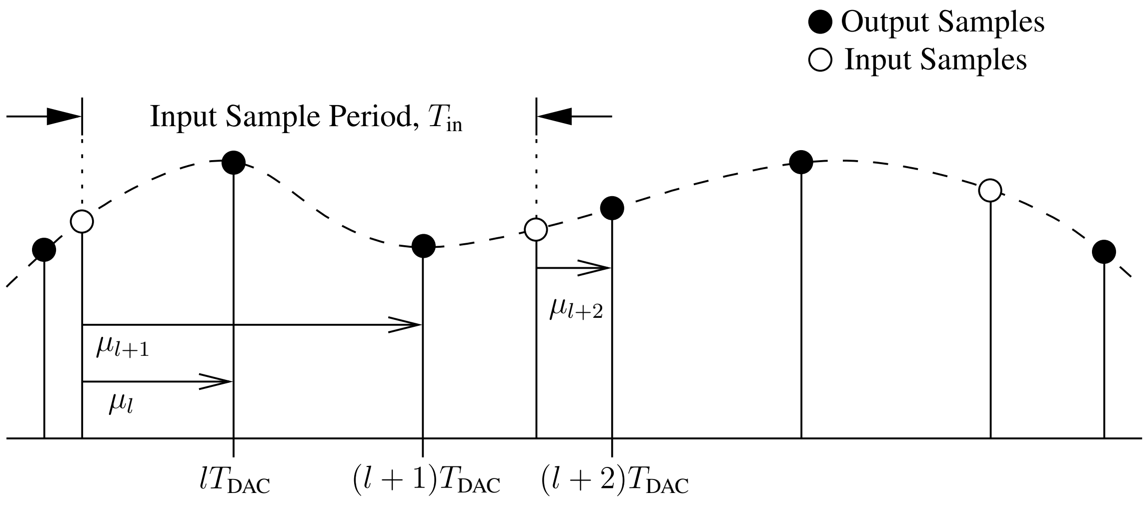

Figure 42: Interpolator Input/Output Timing

When a new input sample is required, the generated μ value is less than or equal to the previous μ value. This condition is indicated when the accumulator overflows signaling a "new input sample request". This technique obviates the need for an adjustable sample clock making possible the use of a fixed frequency oscillator for clocking the DAC. Also, because the DAC sample clock is fixed, a single analog filter can be used to remove DAC images regardless of the original sample rate at which the waveform samples were created.

The PBF and sample requester form the foundation of the AWG. In addition to this base, several more features can be included such as real-time carrier generation. This allows all waveforms to be stored in memory in lowpass form at the minimum sample rate required to reconstruct the signal. The lowpass signal can then be carrier modulated in real-time. This avoids the need to store the higher frequency carrier signal in memory which would require a higher sample rate. Also, to accommodate the class of linear digital modulation schemes, ubiquitous in modern communication systems, a programmable pulse shaping filter can be added to the system just before the PBF. Since pulse shaping is performed in real-time, only the data symbols need to be stored in memory. This is discussed in the next section.

5.2.2 Vector Signals

Vector modulated signals encompass the popular linear modulation schemes including Phase Shift Keying (PSK), Pulse Amplitude Modulation (PAM), and Quadrature Amplitude Modulation (QAM) [74]. The baseband representation of a vector signal is given by

$$\begin{aligned} s(t) &= \sum_{k=-\infty}^{\infty} a_k p(t - kT_{\text{sym}}) \\ &= s_I(t) + js_Q(t). \end{aligned} \tag{5.18}$$

In (5.18), $a_k = a_{Ik} + ja_{Qk}$ are the complex valued data symbols, and $p(t)$ is the impulse response of a pulse shaping filter. The data symbols are coordinates chosen from a signal space constellation in the I/Q plane. Carrier modulation to an Intermediate Frequency (IF), w_{IF}, is performed by mixing the baseband signal with a complex exponential and taking the real part,

$$\begin{aligned} x(t) &= \Re\left\{s(t)e^{j\Omega_{\text{IF}}t}\right\} \\ &= s_I(t)\cos(\Omega_{\text{IF}}t) - s_Q(t)\sin(\Omega_{\text{IF}}t). \end{aligned} \tag{5.19}$$

The signal $s(t)$ is generated by feeding symbol pulses into the shaping filter at the symbol rate $F_{\text{sym}} = 1/T_{\text{sym}}$. This operation is readily accomplished in the sampled data domain using FIR filters having impulse response

$$p(nT_{\text{s}}) = p\left(n\frac{T_{\text{sym}}}{L}\right). \tag{5.20}$$

In (5.20), T_{s} is the sample period of the filter and L is an integer representing the number of samples per symbol contained in the shaped pulse. The filter thus accepts symbol pulses at the symbol rate and produces shaped pulses at the sample rate $F_{\text{s}} = LF_{\text{sym}}$.

AWGs that cannot adjust the waveform sample rate to the symbol rate would require the symbols to be shaped and resampled to one of the available waveform sample rates *offline*. This is a major drawback for two reasons. First, the number of samples required to store the waveform in memory will increase, possibly dramatically, depending on the available sample rates. Second, since the shaping and resampling is done offline, the computation time can be excessive, especially for long waveforms. One way to solve this problem is to provide a high resolution sample clock that can adjust the waveform sample rate to virtually any symbol rate over a given range and perform pulse shaping in real-time. Therefore, only the symbol values need to be stored in memory. This is the approach taken by some systems and will be explored in the next section.

5.2.3 Current Flexible AWGs with RTSP

To achieve improved memory compression and the flexibility of virtually continuously variable waveform sample rates, some recently developed systems provide two sample clocks [70, 72]. As mentioned in the introduction, one sample clock provides a spectrally pure integer submultiple of the maximum clock rate for high performance, and the other offers the flexibility of high resolution frequency selection at the expense of spectral purity degradation. Sample clock spectral purity is essential in producing high fidelity signals with DACs. The reason for this is that any spurs and phase noise present on the sample clock will be convolved with the signal being converted and appear as distortion in the DAC output. This is clearly seen from observing the plots in [72], which compare the use of an external sample clock with the high resolution sample clock. When the high resolution sample clock is used, spectral distortion components appear as high as ≈ 50 dB below the intended signal. The high resolution sample clock in [72] is a Direct Digital Synthesis (DDS) based sample clock. Besides an extra sample clock adding to the hardware complexity and cost, the wideband Spurious Free Dynamic Range (SFDR) of current integrated DDS chips is limited to the 40 to 60 dB range for typical sample rates approaching 100 MHz or more (see for example [75]). These spurious components are what cause the undesirable distortion in the output spectrum of the AWG.

5.2.3.1 Pulse Shaping and Interpolation

A block diagram of the FPGA signal processing for pulse shaping and interpolation in the example current system [72] is given in Fig. 43.

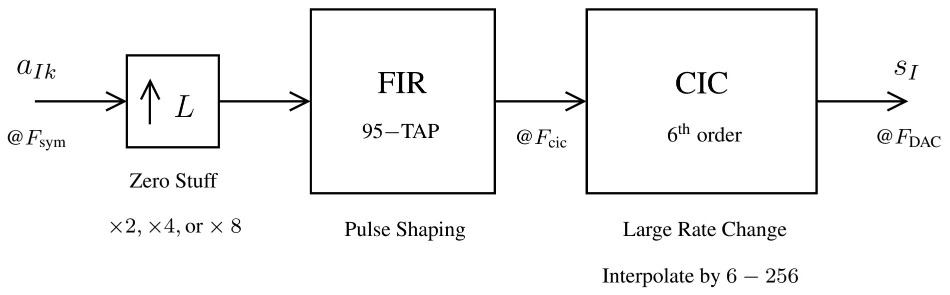

Figure 43: I-path Pulse Shaping and Interpolation. (Q-path is the same)

Here, symbols are fed into an upsampler which stuffs zeros into the symbol stream increasing the sample rate by $L = 2, 4, \text{or } 8$ times. The upsampled symbols are then routed to a 95-TAP FIR filter running at

$$F_{\text{cic}} = LF_{\text{sym}}, \tag{5.21}$$

which corrects for the Cascaded-Integrator-Comb (CIC) filter droop and shapes the symbol pulses. The shaped symbols are then interpolated to the DAC sample rate by a 6th order CIC filter. The overall interpolation ranges supported are 12-512 in steps of 2, 512 to 1024 in steps of 4, and 1024 to 2048 in steps of 8. With the FIR filter providing pulse shaping, the interpolation image suppression is totally provided by the CIC filter.

The worst case image suppression occurs when the input signal bandwidth to the CIC filter is at a maximum. This system supports Nyquist pulse shapes (see [76,74,77] for details on Nyquist pulses) with rolloff factors of $.1 \leq \alpha \leq .9$. The rolloff factor determines the single sided bandwidth of the shaped pulses and is equal to

$$B = (1 + \alpha)F_{\text{sym}}/2. \tag{5.22}$$

This can be mapped to the design space of the CIC filter by rewriting the equation in terms of the CIC input sample rate, where from (5.21),

$$B = (1 + \alpha)\frac{F_{\text{cic}}}{2L}. \tag{5.23}$$

The minimum image attenuation provided by the CIC filter can now be plotted as a function of α, for each L value as shown in Fig. 44.

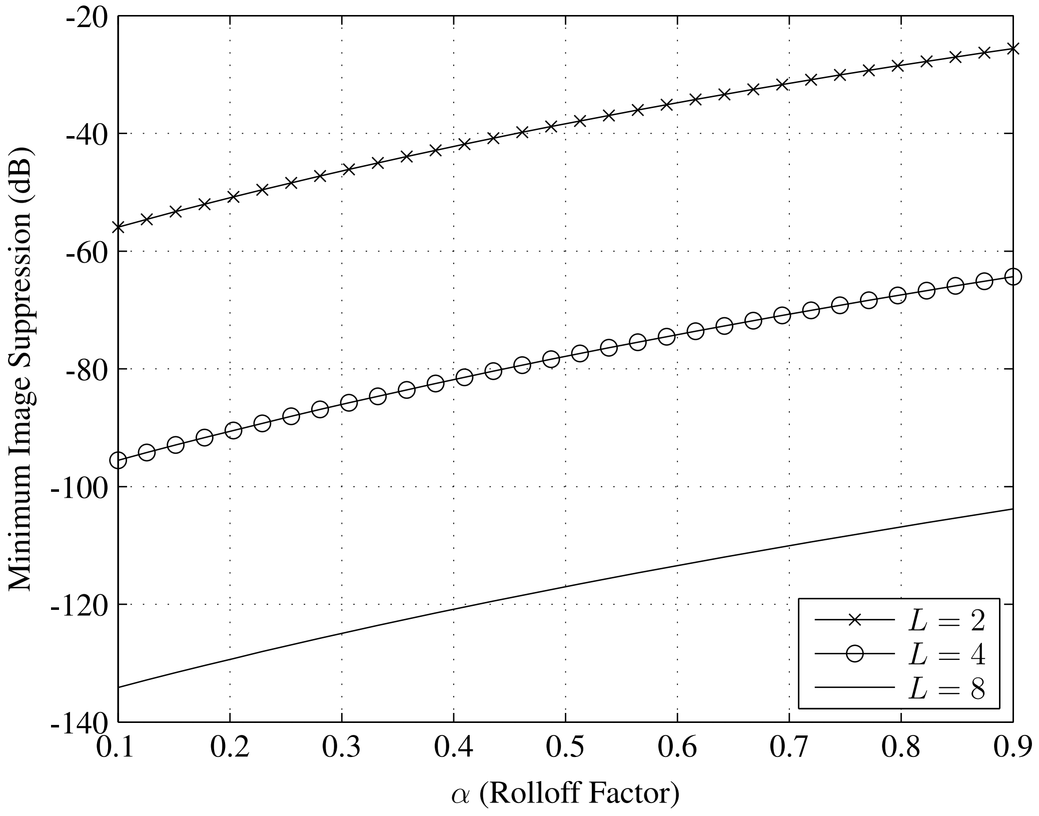

Figure 44: CIC Minimum Image Attenuation

The quantization noise level of an n-bit signal is approximately $6n$ dB below the signal level [25]. Most high performance AWGs use 12-16 bit DACs. Also, modern DACs can achieve SFDRs of 75-100 dB depending on output signal frequency. A reasonable design choice for the amount of image suppression required is to attenuate images to or below the quantization noise level and/or DAC SFDR. As can be observed from Fig. 44, $L = 2$ provides inadequate image suppression for all α's, $L = 4$ is acceptable for lower α values, and $L = 8$ is over designed. The choice of multiple L values provides more selections for the overall interpolation factor,

$$\text{Overall Interpolation} = L \times \text{ (CIC Interpolation)} \,, \tag{5.24}$$

since the CIC Interpolation factor is limited to 6-256. However, performance somewhere in between the $L = 4$ and $L = 8$ curves for a 16-bit AWG would be ideal.

After the data symbols are shaped and interpolated to the DAC sample rate, carrier modulation to an IF is performed based on (5.19). The result of this operation is to produce a single real bandpass signal from the I/Q signals.

5.2.4 The New Approach to Pulse Shaping and Interpolation

As discussed in section 1.2, the design specifications for an interpolation filter can be derived by observing that any time the sample rate of a signal is increased, images are created at multiples of the original sample rate. Consequently, if we want to interpolate lowpass signals of single sided bandwidth B, then the interpolation filter needs to eliminate images residing in the frequency bands $kF_{\text{in}} \pm B$, where $k = 1, 2, \ldots$, and F_{in} is the sample rate of the signal at the interpolator input.

In this section, we show how the limited image suppression and narrow range and resolution of overall interpolation factors provided by the system of Fig. 43 can be overcome using the new approach. A block diagram of the new pulse shaping and interpolation is given in Fig. 45.

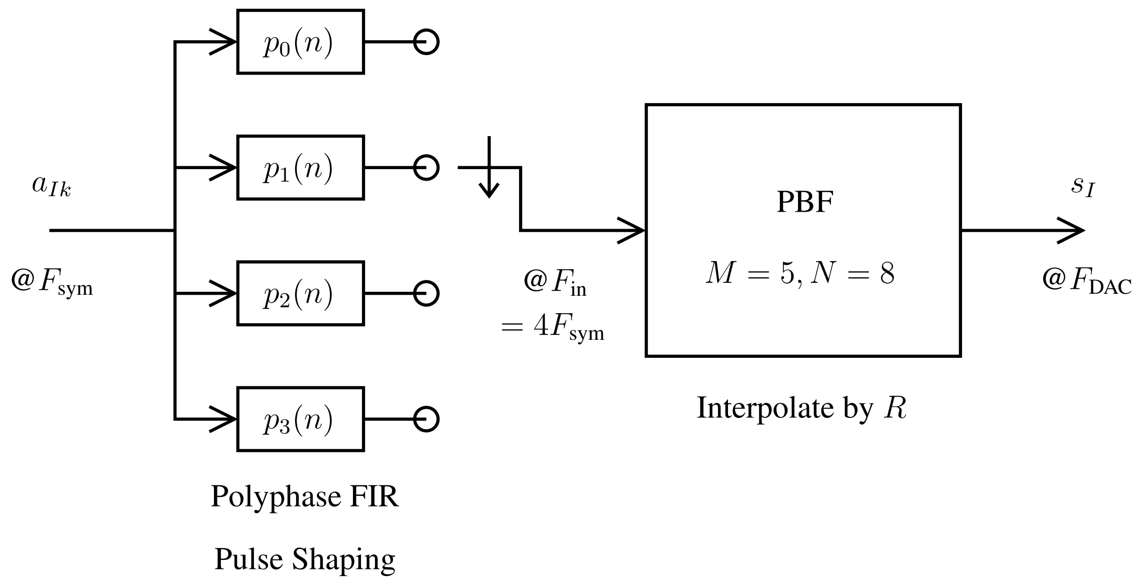

Figure 45: New I-path Pulse Shaping and Interpolation. (Q-path is the same.)

In this system, the pulse shaping filter interpolates by a fixed factor of four. However, this does not restrict the choice of the overall interpolation factor. The overall interpolation factor is given by

$$I = 4R, \tag{5.25}$$

where R is an arbitrary factor, not necessarily an integer. To reduce computational complexity, the pulse shaping filter is implemented in its polyphase form [24]. The same 95-tap FIR filter from Fig. 43 reduces to 4 subfilters of length 24, each operating at F_{sym}. The objective of the PBF is to eliminate interpolation images centered at multiples of its input sample rate for all possible input signal bandwidths, i.e. for the full range of supported α's. The filter was designed to have stopbands that would attenuate the worst case input signal by at least 80 dB. The worst case input signal corresponds to the signal having the maximum bandwidth, or equivalently, the largest α value.

For this case, $\alpha = .9$, thus from (5.22),

$$B_{\text{max}} = (1 + .9)F_{\text{sym}}/2 = .95F_{\text{sym}}. \tag{5.26}$$

Rewriting (5.26) in terms of the PBF input sample rate, we obtain

$$B_{\text{max}} = .95F_{\text{in}}/4 = .2375F_{\text{in}}. \tag{5.27}$$

Equation (5.27) is the worst case input single-sided bandwidth. Therefore, to provide guaranteed image rejection, the filter must have stopbands of $k\text{F}_{\text{in}} \pm B$, where $k = 1, 2, \ldots$. Normalizing to the input sample rate gives,

$$\text{Filter Stopbands} = k \pm .2375, \text{ for } k = 1, 2, \ldots. \tag{5.28}$$

The frequency response of the PBF is given in Fig. 46, where the observable stopbands are centered at normalized frequencies of 1,2, and 3. As can be seen from the figure, the stopbands are attenuated by at least 80 dB.

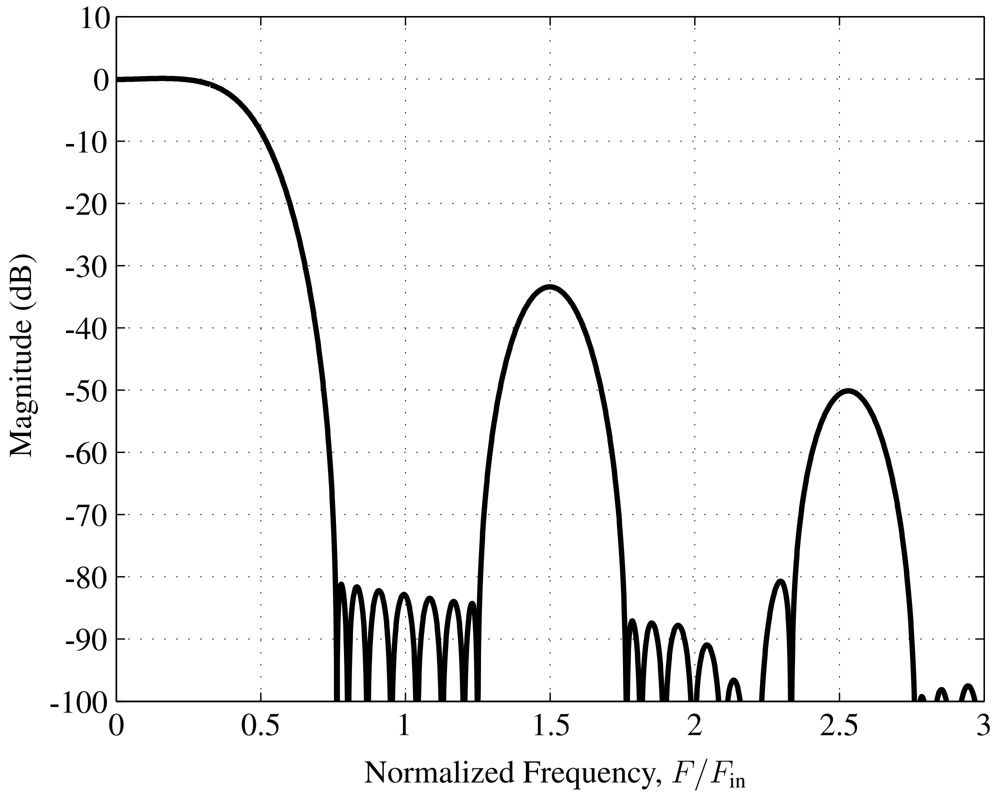

Figure 46: PBF Frequency Response $M = 5, N = 8$

After the data symbols are shaped and interpolated to the DAC sample rate, carrier modulation to an IF is performed. Due to the availability of DACs which accept I/Q inputs, perform integer interpolation, and coarse frequency shifting, we can eliminate taking the real part in (5.19) and let the DAC perform that operation. Thus, a *complex* bandpass signal is produced for input to the DAC. Since the DAC can increase the sample rate of the signal by an additional integer factor, a higher carrier frequency can be produced. By leaving the signal complex, there is no negative frequency image which generates an additional sideband due to the final frequency shifting performed by the DAC.

5.2.5 Interpolation Comparison

In order to compare the computational cost of the two systems, the worst case number of Multiplications Per Output Sample (MPOS) is examined for the case where the pulse shaping filter output sample rate is $4F_{\text{sym}}$. The required number of MPOS of the current system of Fig. 43 result from the FIR filter. The filter is symmetric and thus requires

$$\text{MPOS}_{\text{curr}} = ((95-1)/2+1)/6 = 8. \tag{5.29}$$

The additional factor of 6 in (5.29) is due to the fact that the filter operates at a maximum of 6 times less than the overall output sample rate. This is because the CIC filter interpolates by a minimum of 6.

The new system of Fig. 45 requires MPOS from both of its subsystems. The pulse shaping filter consists of four subfilters of length 24, operating at F_{sym}. This gives

$$\text{MPOS}_{\text{new\_PS}} = 24(4)/(4(6)) = 4. \tag{5.30}$$

As in (5.29), the factor of 6 comes from the minimum interpolation by 6 from the following stage, in this case the PBF. From Section 5.2.4, the MPOS for the PBF is given by

$$\begin{aligned}\text{MPOS}_{\text{new\_PBF}} &= M + ((M+1)N/2)/6 \\ &= 5 + ((5+1)8/2)/6 = 9.\end{aligned} \tag{5.31}$$

Hence, the total MPOS for the new system is given by the sum of (5.31) and (5.30),

$$\text{MPOS}_{\text{new\_total}} = 4+9 = 13 \tag{5.32}$$

Table 3: System Comparison Summary

	Min. Image Att.	Interpolation Factor	MPOS
Curr. System	$\approx$ 64 dB	24-2048 (Integers)	8
New System	$\approx$ **80 dB**	**24-32768** (Arb. Factors)	**13**

Table 3 summarizes the findings.

The maximum interpolation factor of the proposed system was chosen arbitrarily to be 32,768, but may be chosen higher or lower. As shown in section 4.3.2, the precision of the rate conversion depends only on the precision used in the μ generator and may be adjusted by increasing the number of bits used to represent μ. For comparison purposes, we limited the example design of the new system of Fig. 45 to the minimum interpolation factor of the current system of Fig. 43 when the pulse shaping filter outputs four samples per symbol. This gives a total minimum interpolation factor of 24. However, the proposed approach is fully capable of becoming a more wideband system supporting lower minimum overall interpolation, at the expense of more MPOS. In fact, the system can be designed to meet virtually any image suppression specifications required. This is in contrast to the use of a CIC filter, which is traditionally meant for narrowband interpolation [35]. Another benefit of the proposed approach is that the pulse shaping filter need not correct for the inherent passband droop of the CIC filter. Therefore a user may define their own pulse shaping coefficients without having to adjust them to compensate for the CIC filter.

5.2.6 Simulation

In order to show the effectiveness of the proposed approach in performing pulse shaping and interpolation by arbitrary factors, a simulation of the system of Fig. 45 was carried out. The simulation consists of generating a 3.2MHz Quaternary Phase Shift Keying (QPSK) signal from only its symbols. The simulation parameters are as follows.

1. $F_{\text{sym}} = 3.2$ MHz
2. $F_{\text{DAC}} = 100$ MHz
3. Raised Cosine Pulse Shaping, $\alpha = .75$
4. Overall Interpolation Factor $= F_{\text{DAC}}/F_{\text{sym}} = 31.25$
5. PBF interpolation factor $R = 31.25/4 = 7.8125$

Fig. 47 shows the spectrum of the output of the shaping filter and the output of the PBF. As can be seen from the figure, interpolation images are held below the designed level of 80 dB.

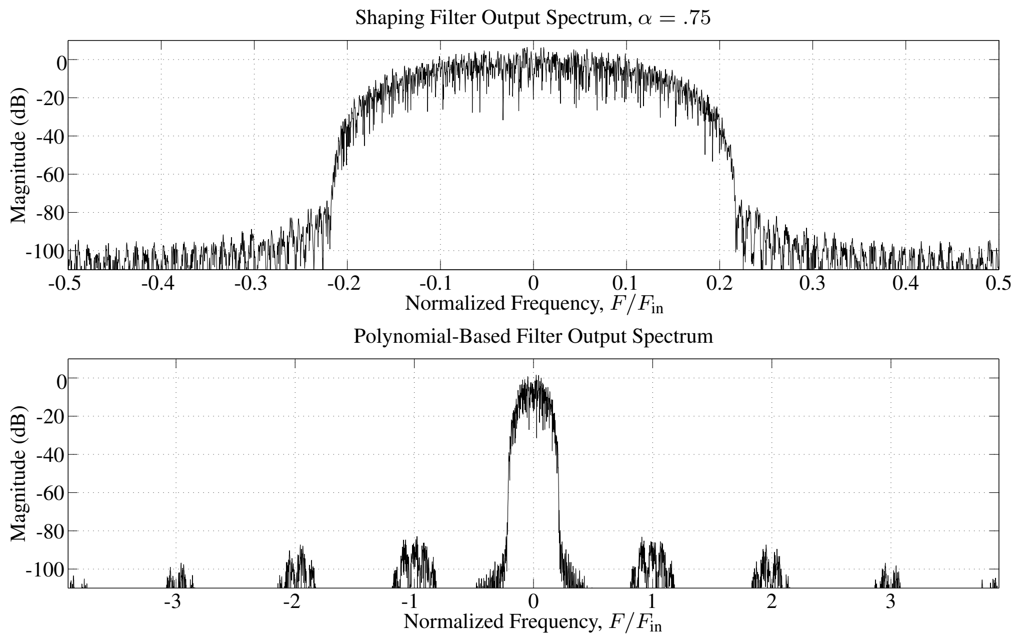

Figure 47: Results of Simulation

5.3 Hardware Implementation

In this section, it is shown how the combination of a new FPGA-based signal synthesis technique developed in section 5.2 with high speed interpolating DAC creates a flexible, high performance source for an UC in a SI.

The signal synthesis technique uses a fixed frequency sample clock in combination with a real-time, arbitrary factor interpolator. The interpolator converts waveform samples having any arbitrary sample rate to the DAC sample rate. Following interpolation, fine center frequency tuning and gain adjustment is performed. Then, the DAC optionally interpolates the signal by an additional integer factor and performs coarse frequency shifting to center the signal's spectrum at the desired IF.

The new system has the following advantages over current FPGA-based approaches using real-time signal processing (see e.g. [70]). First, unlike systems with variable DAC sample rates, the new approach fixes the DAC sample rate. This allows the use of a single analog filter to guarantee removal of distortion images at the DAC output regardless of the waveform sample rate. Second, similar to the system found in [72], because the waveform is interpolated, it can be stored at virtually the minimum sample rate required to reconstruct the signal. This yields dramatic memory savings of up to several orders of magnitude. Unlike the system in [72], the interpolation factor can easily be extended beyond 2048 making possible even more memory savings for signals sampled at slow rates. Additionally, using a DAC with coarse frequency shifting ability and interpolation allows higher IFs to be generated. Consequently, the filtering burden of the UC is relaxed by an increased separation between the signal and its mirror spectrum.

A real implementation of the proposed system is presented showing the feasibility and attainable performance of such an approach. Measured results are given confirming expectations from design and simulation.

A block diagram of the new AWG is given in Fig. 48.

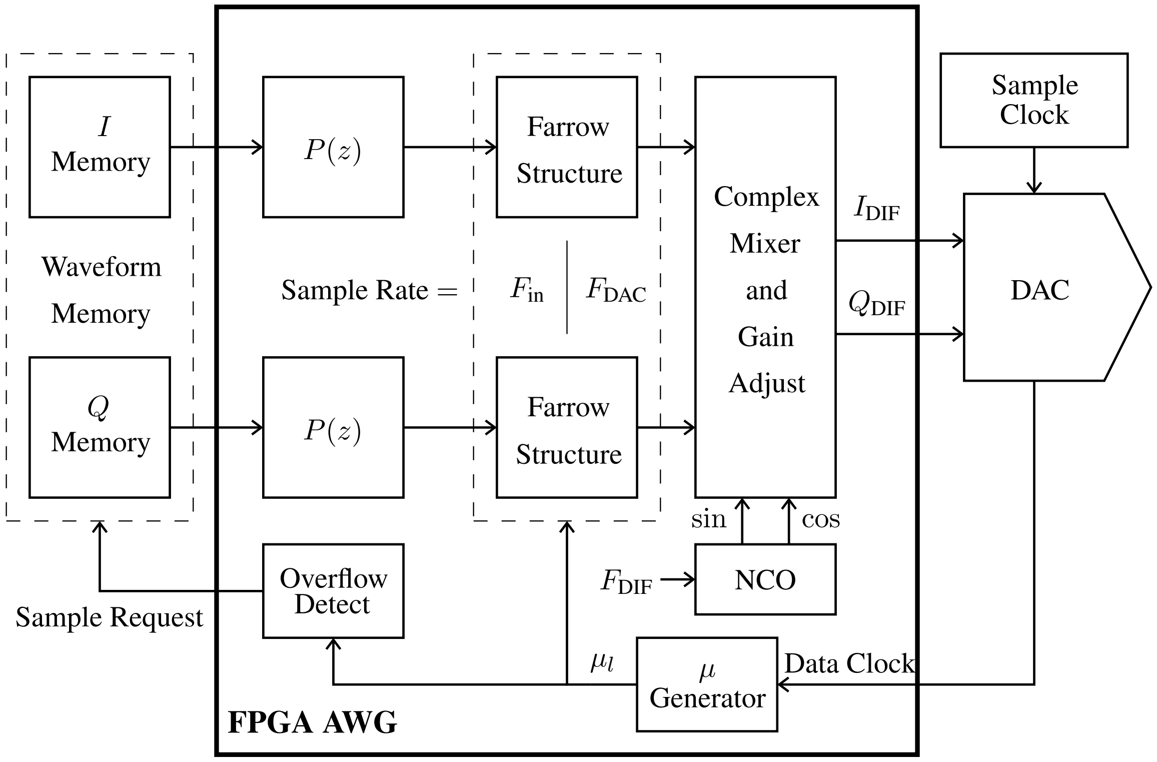

Figure 48: Flexible, High Performance, Fixed Sample Clock AWG

In Fig. 48, the DAC is given a sample clock of fixed frequency F_{DAC} Hz. Data Clock is a clock derived from the sample clock for the digital source driving the DAC, in this case the FPGA AWG. Data Clock is also of frequency F_{DAC} Hz. The μ generator computes the normalized fractional time distance between the next interpolated output sample and the current input sample. The μ values are used by the Farrow Structure [38] to compute interpolated output samples. New input sample requests are generated by an overflow detector which monitors the μ values for the condition

$$\mu_{l+1} \geq \mu_l. \tag{5.33}$$

This indicates that a new input sample is required to generate the next interpolated output sample. In this configuration, the Farrow Structure takes as input a Discrete-Time signal sampled at F_{in} Hz. The Continuous-Time (CT) signal underlying the DT input signal is then reconstructed and resampled at a rate of $F_{\text{out}} = F_{\text{DAC}}$ Hz. This is illustrated in Fig. 49.

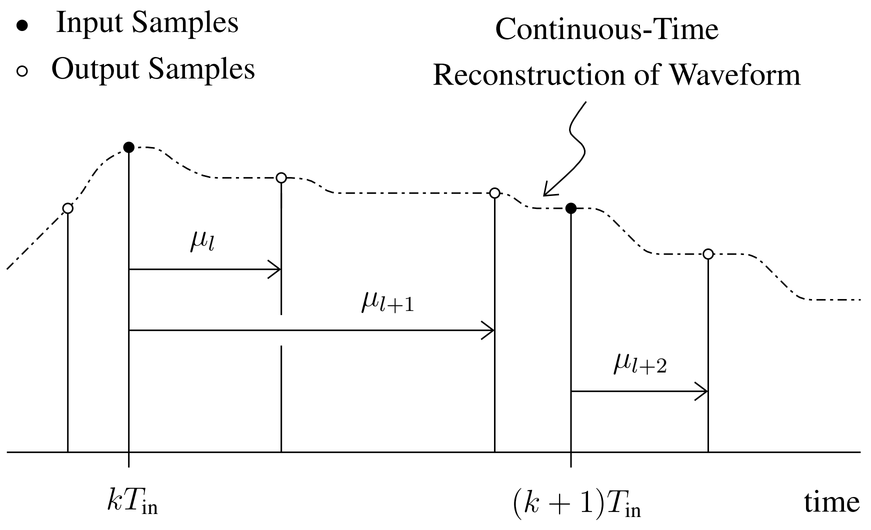

Figure 49: Farrow Structure Input / Output Timing

The output sample rate is given by $F_{\text{out}} = RF_{\text{in}}$. R is the Sample Rate Conversion (SRC) factor from section 1.2, defined as

$$R = \frac{F_{\text{out}}}{F_{\text{in}}}. \tag{5.34}$$

Upon a new input sample request, waveform samples are supplied from Waveform Memory to the user-defined digital filter(s) $P(z)$. If $P(z)$ is interpolating, then the Sample Requests derived from the overflow detector request samples from $P(z)$. $P(z)$, in turn, requests samples from memory at a slower rate. $P(z)$ serves two purposes. First, it can provide real-time pulse shaping for digital modulation schemes [76]. This means that only the symbol values need to be stored in memory rather than a highly oversampled waveform. The details on implementing $P(z)$ as a pulse shaping filter can be found in [1]. Second, $P(z)$ can simply interpolate the input waveform by a small integer factor. This can reduce the complexity of the Farrow Structure significantly.

After the DT signal is filtered and interpolated, gain adjustment can be performed to fine tune the signal output level. It is then mixed with a complex exponential of frequency F_{DIF} Hz whose real and imaginary parts are generated by a Numerically Controlled Oscillator (NCO). Assuming the input spectrum is centered at DC, this operation produces a complex bandpass signal centered at F_{DIF} Hz (A review of complex signal processing is given in [78]). This is illustrated in Fig. 50. If the AWG is used as the baseband/IF source in a SI, it provides the input signal to the UC component. The FPGA AWG can be combined with a DAC that provides interpolation by integer factors and complex, coarse frequency shifting [79]. This combination simplifies the filtering burden of the UC. This is because the additional frequency shift increases the separation between the signal and its mirror spectrum. As can be seen from Fig. 50, there is no mirror spectrum while the signal is in complex form. The mirror spectrum only appears when the complex bandpass signal is converted to a real bandpass signal as illustrated in Fig. 51. This is the final step the signal undergoes before it is converted to an analog voltage or current waveform by the DAC.

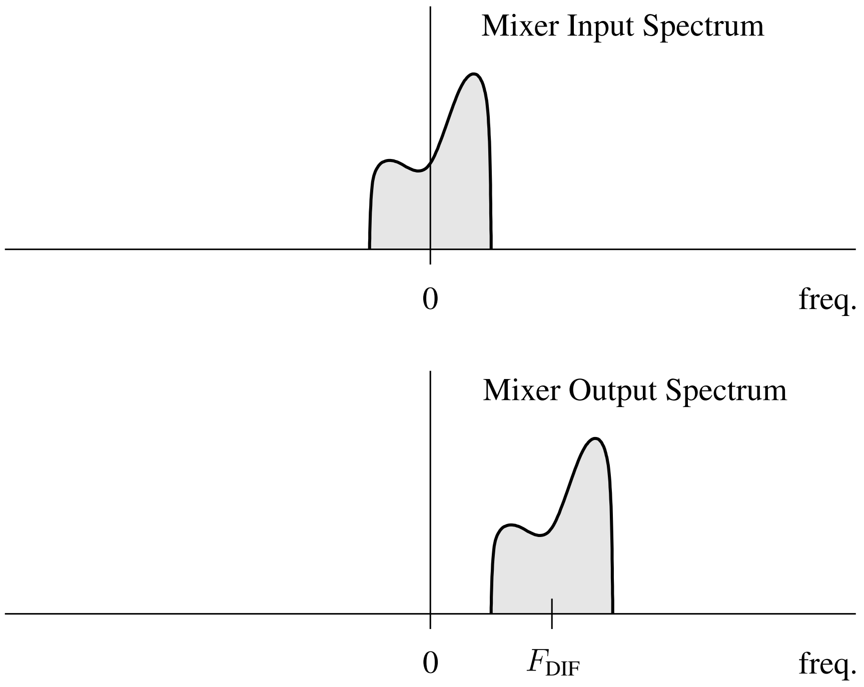

Figure 50: FPGA AWG Complex Mixer Input / Output Spectra

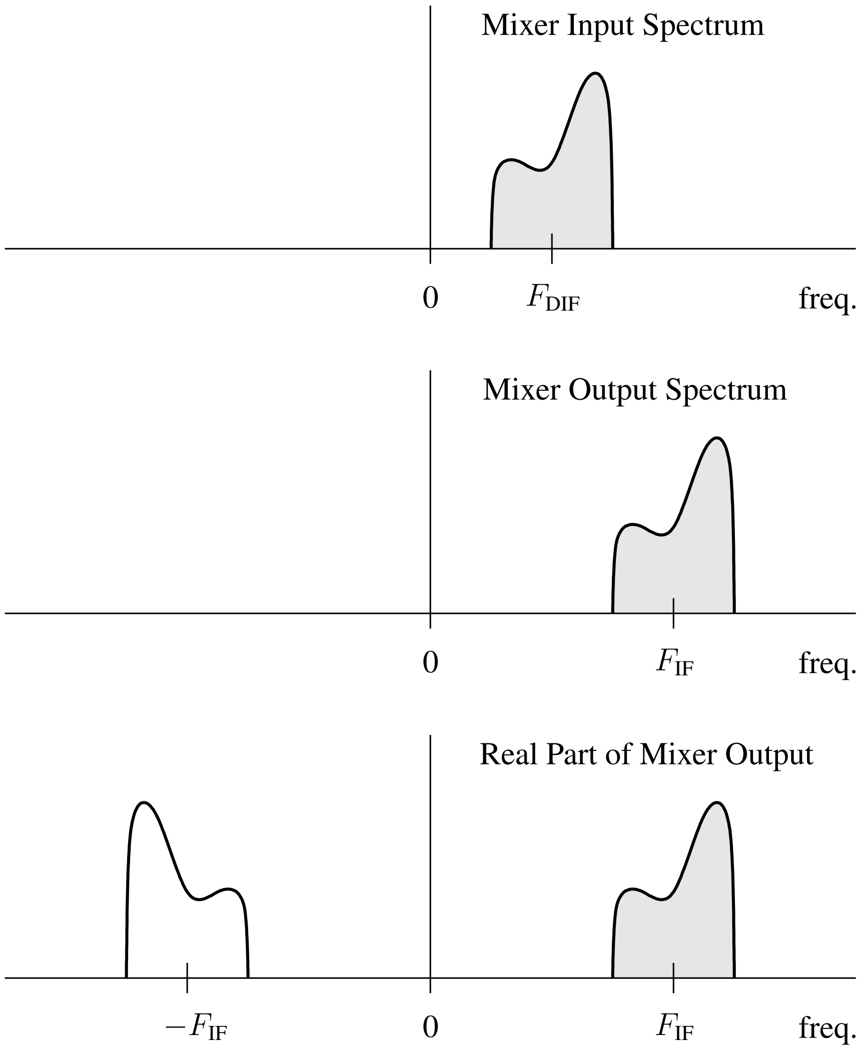

Figure 51: DAC Complex Mixer Input / Output Spectra

To see why the UC filtering requirement is relaxed by a higher IF frequency, consider the real upconversion shown in Fig. 52 (see [80] for transmitter/UC filtering and architectures).

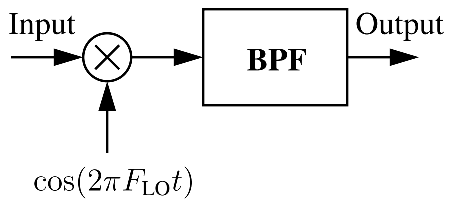

Figure 52: UC First Mixer and BandPass Filter (BPF)

The mixer translates the entire input spectrum to F_{LO} Hz. Since this is a real upconversion, both the desired signal and its mirror spectrum are translated. The objective of the BandPass Filter (BPF) is to eliminate the translated mirror spectrum located $2F_{\mathrm{IF}}$ from the desired spectrum. Obviously as F_{IF} is increased, the separation between the mirror spectrum and the desired spectrum is also increased. This in turn allows a more relaxed filtering requirement as illustrated in Fig. 53.

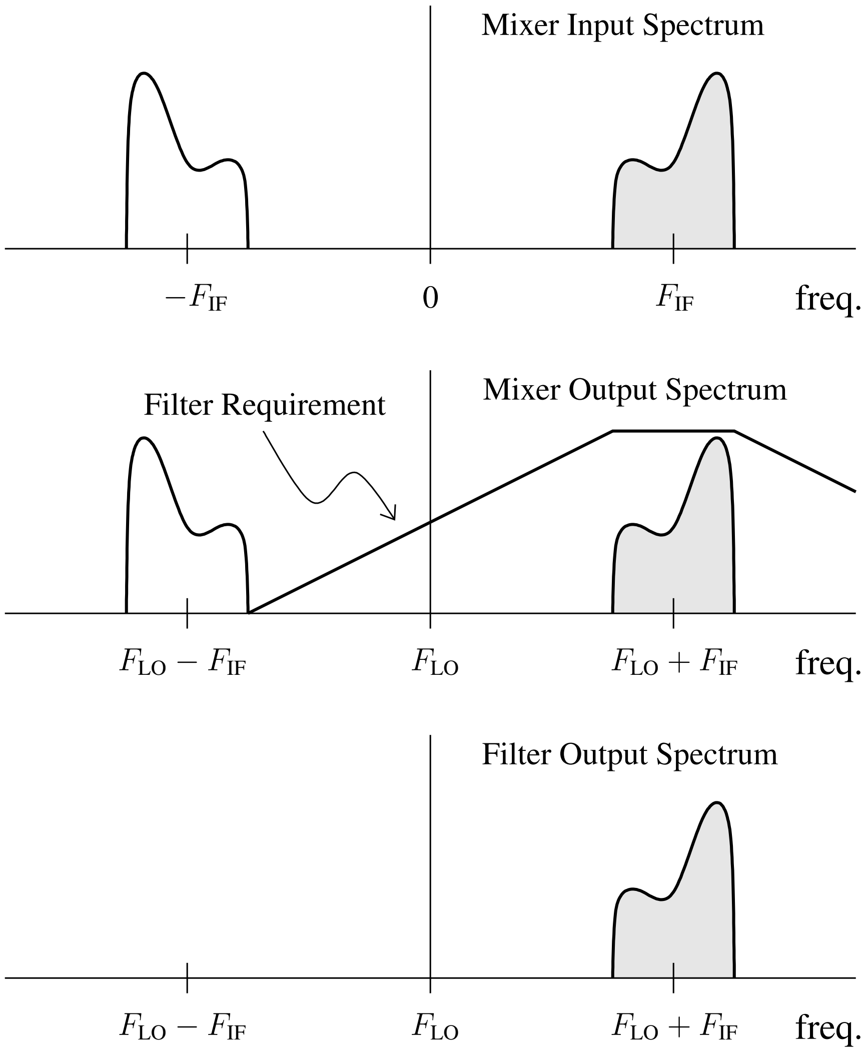

Figure 53: UC Mixer/BPF Input / Output Spectra

5.3.2 Hardware Implementation

To demonstrate the feasibility of the new approach, a high performance system is implemented on a Xilinx xc3s5000-4fg900 Spartan-3 FPGA [81] in conjunction with a 16-bit Analog Devices interpolating DAC [79]. The DAC is configured to interpolate the complex input signal by a factor of 8. The DAC sample clock is $F_{\text{DAC}} = 311.04$ MHz. Since the DAC interpolates the output of the FPGA AWG by a factor of 8, it provides a Data Clock of $F_{\text{DAC}}/8 = 38.88$ MHz rather than F_{DAC} MHz. $P(z)$ is a programmable, 95-tap, interpolate by 4 polyphase Finite Impulse Response (FIR) filter. The Farrow Structure implements a Polynomial-Based Filter (PBF) having $N = 8$ polynomial pieces of order $M = 5$ (See [1,3] for the details on the PBF design). The filter is designed to provide more than 87 dB of interpolation image suppression for single-sided input bandwidths up to $.2F_{\text{in}}$. The frequency response of the PBF is given in Fig. 54.

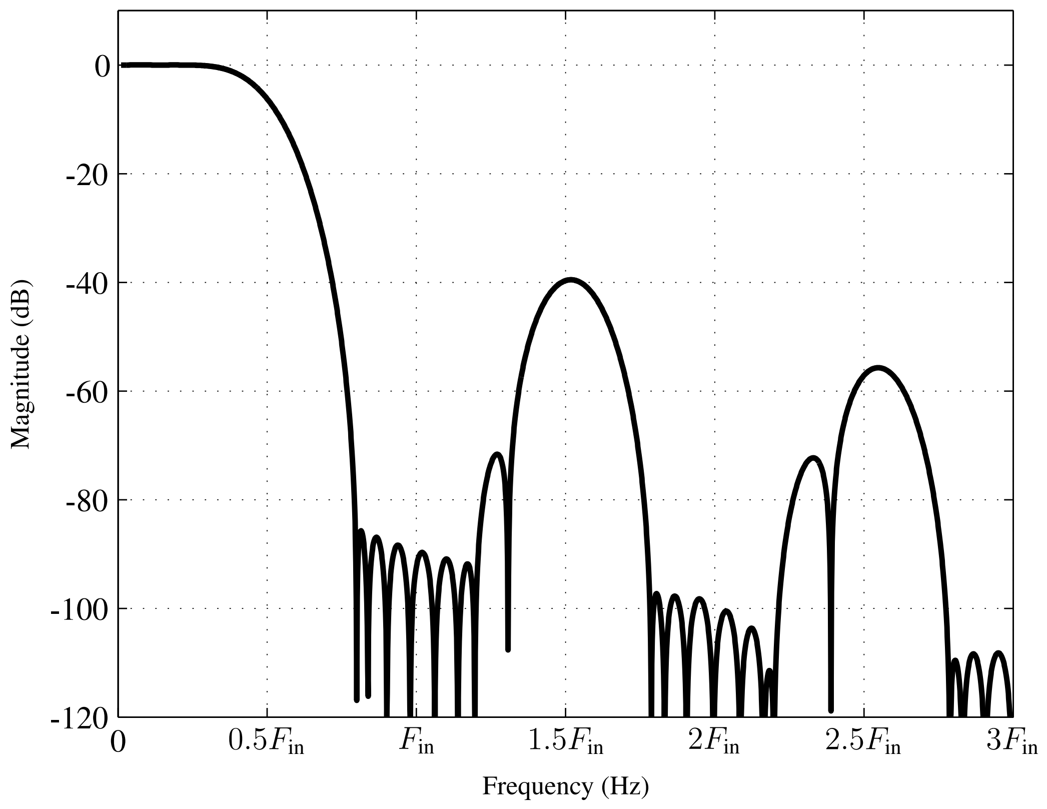

Figure 54: PBF Frequency Response $M = 5, N = 8$

The Farrow Structure input sample rate is not restricted, and thus can be equal to the output sample rate. In practice, the system implementation cost could be reduced by lowering the maximum input sample rate to the Farrow Structure. However, this comes at the cost of reduced maximum signal bandwidth. The device utilization summary is given in Table 4.

Table 4: Device Utilization for Xilinx Part # xc3s5000-4fg900

Logic Utilization	Used	Available	Utilization
Number of Slice Flip Flops	20,482	66,560	30%
Number of 4 input LUTs	11,061	66,560	16%
Total Equivalent Gate Count	1,007,485	-	-

5.3.3 Measured Results

To demonstrate the performance of the system, the AWG is set up to generate a 1 Msymbol/sec, raised cosine pulse-shaped, Quaternary Phase Shift Keying (QPSK) signal from only its symbols. The 1 Msymbol/sec QPSK symbol stream is shaped and interpolated by four, in real-time, by $P(z)$ before entering the Farrow Structure. This interpolation by four results in $F_{\text{in}} = 4$ MHz. The corresponding SRC factor is given by

$$R = \frac{F_{\text{out}}}{F_{\text{in}}} = \frac{38.88 \text{ MHz}}{4 \text{ MHz}} = 9.72. \tag{5.35}$$

The test setup is given in Fig. 55. In Fig. 55, the analog output of the new AWG is connected to an Agilent MXA Signal Analyzer [82] for spectral analysis.

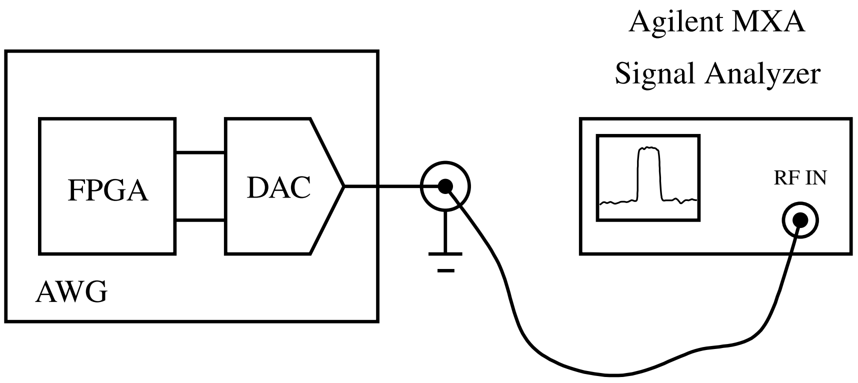

Figure 55: Measurement Test Setup

As a first step, the implementation output is captured and stored in memory before exiting the FPGA. Fig. 56 presents the AWG output spectrum generated from bit-true simulation as well as the actual output spectrum captured from the FPGA implementation.

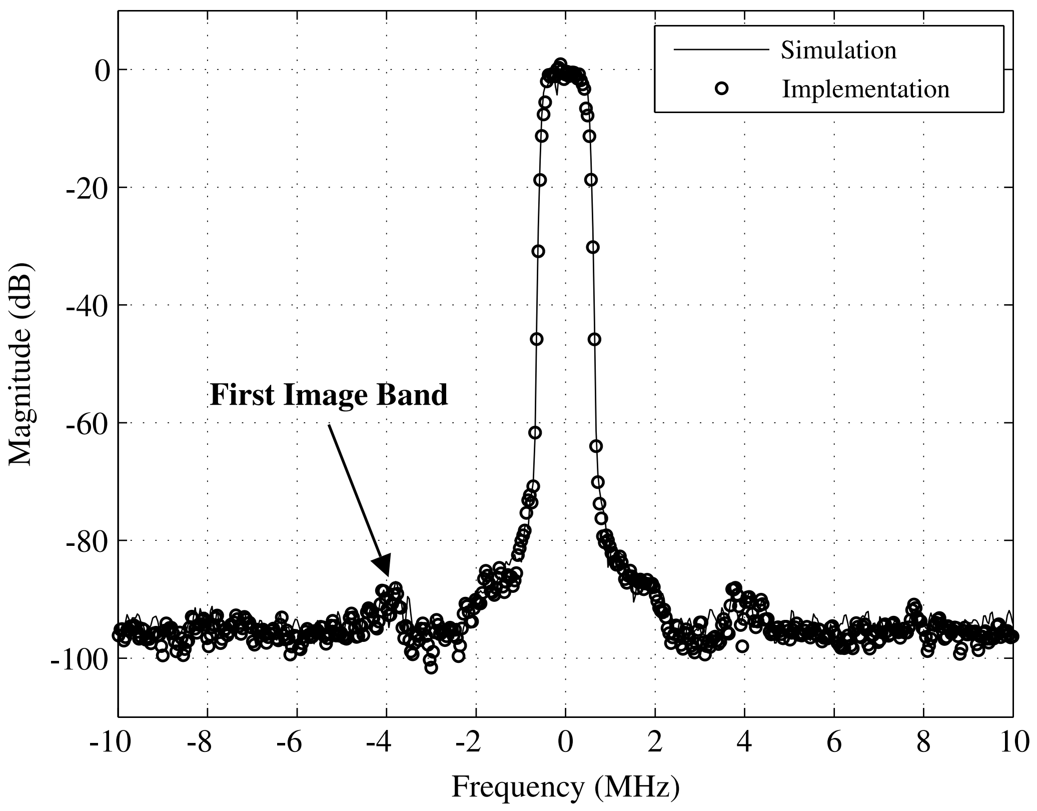

Figure 56: QPSK signal Simulation Versus FPGA Implementation

As can be seen from Fig. 56, the two closely match. In addition, the designed interpolation image suppression, Fig. 54, matches the actual interpolation image suppression.

The AWG analog output results are given next. Figures 57, 58, and 59 are plots of the spectrum trace data saved from the Agilent MXA. In Fig. 57, the DAC does not shift the AWG output spectrum. The AWG NCO is tuned to $F_{\text{DIF}} = 10$ MHz. Thus, the DAC output spectrum is centered at 10 MHz. As can be seen from Fig. 57, there is no visible image distortion or sample clock spurious, which is in contrast with the plots in [70], where sample clock spurious and/or image distortion appear at ≈ 50 dBc. This confirms the high performance expected from the theoretical design of the proposed approach. Figure 58 shows the DAC output spectrum when the DAC is configured to shift the center of the spectrum by $F_{\text{DAC}}/8 = 38.88$ MHz. In this scenario, the AWG tunes the signal's spectrum to $F_{\text{DIF}} = -8.88$ MHz resulting in a DAC IF output of $F_{\text{IF}} = 38.88 + (-8.88) = 30$ MHz. Finally, Fig. 59 gives the spectrum of the signal when the DAC is configured to shift the center of the spectrum by $F_{\text{DAC}}/2 = 155.52$ MHz. The feasibility of high IF signal synthesis is thus demonstrated, even when the waveform is generated at a much lower frequency. As previously shown, the filtering requirement of the UC is significantly relaxed.

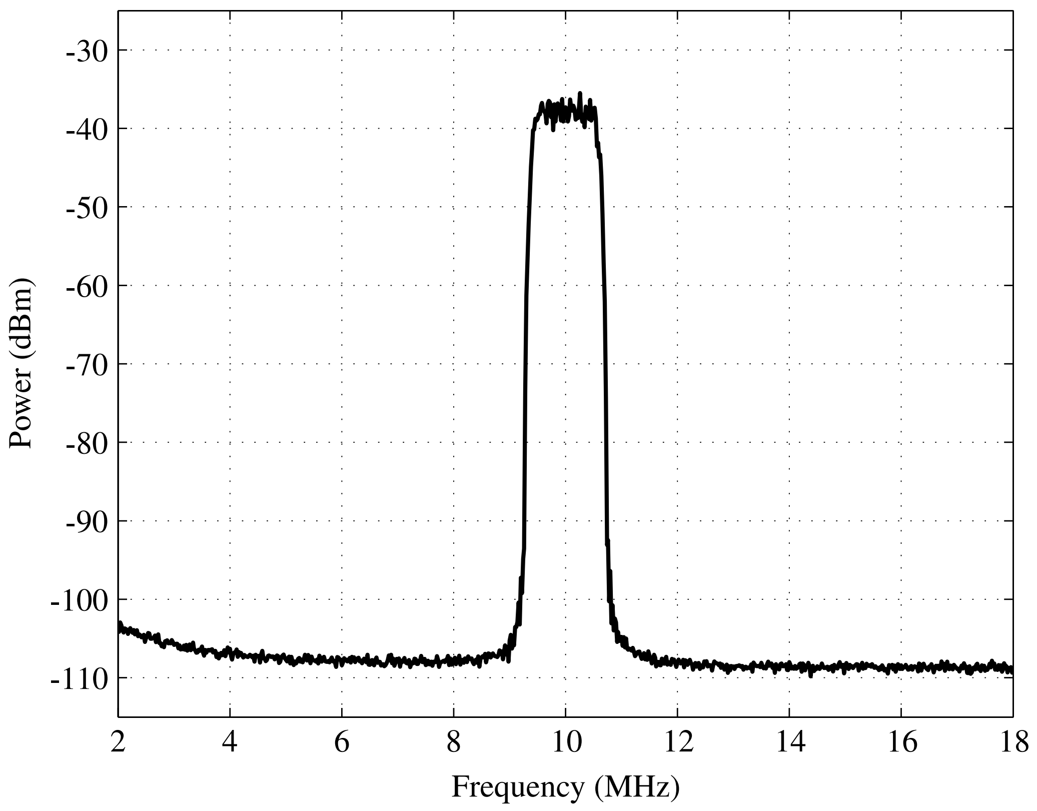

Figure 57: QPSK signal centered at 10 MHz

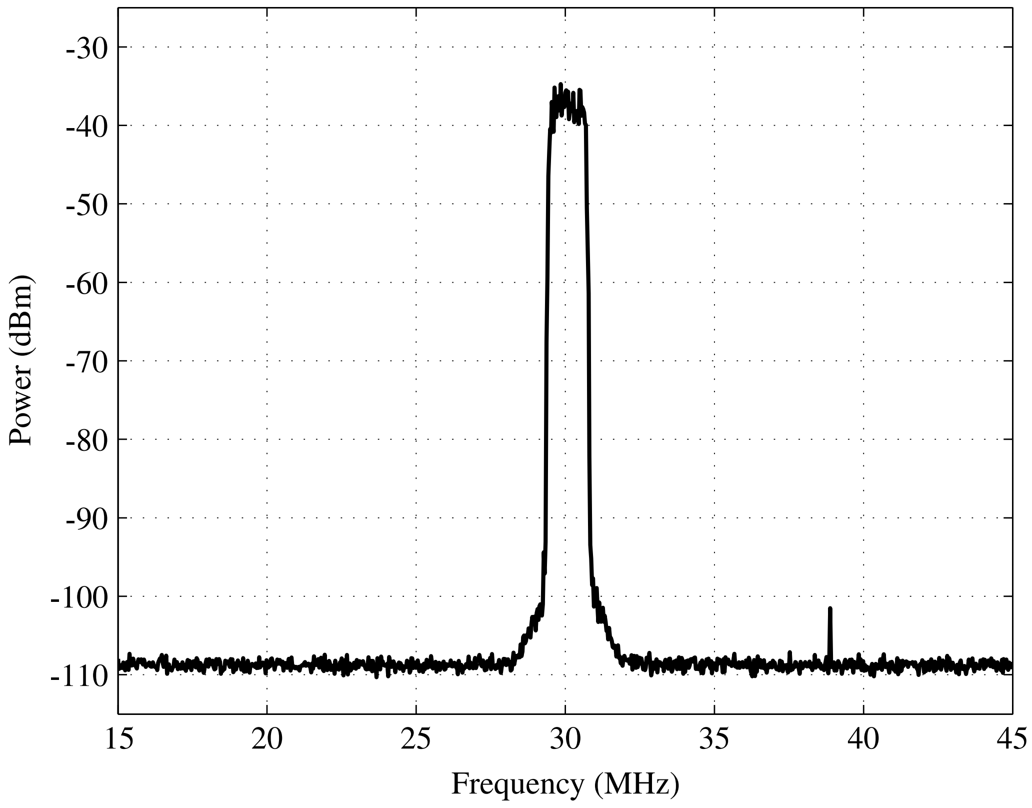

Figure 58: QPSK signal centered at 30 MHz

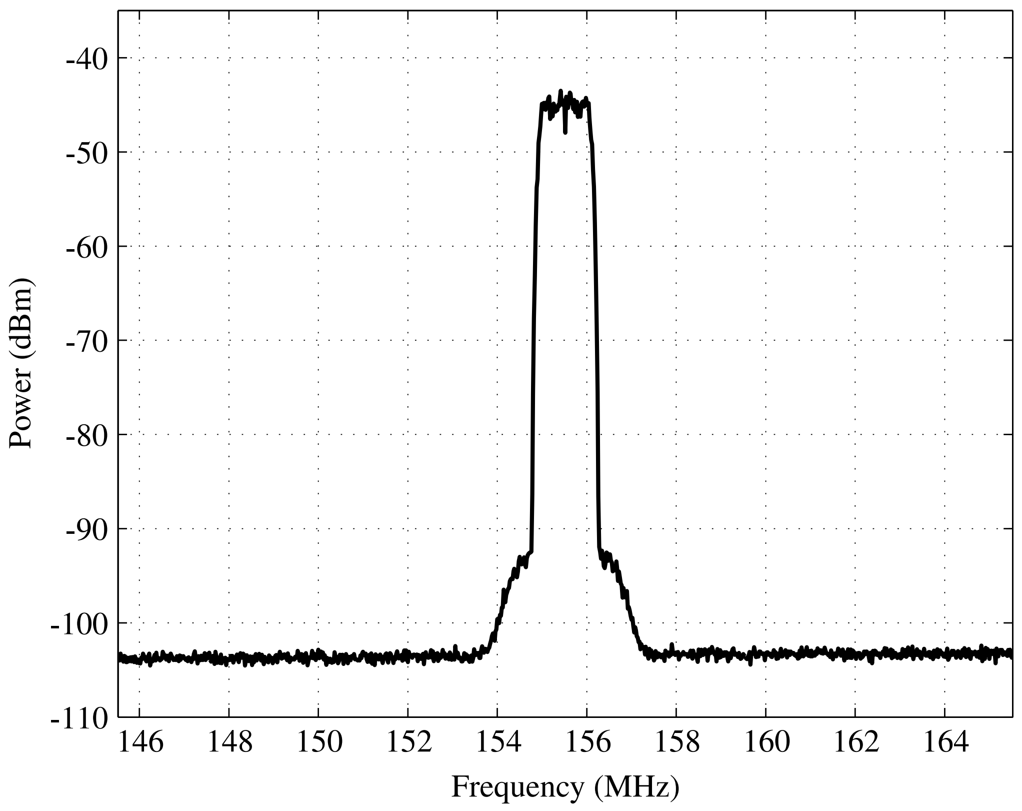

Figure 59: QPSK signal centered at 155.52 MHz

CHAPTER 6: FUTURE WORK

Future work in the design, implementation, and application of Polynomial-Based Filters includes the following.

First under design, new optimization techniques are to be investigated and compared with the linear programming technique presented in this work. New types of PBFs will be studied including complex filters and filters with non-linear phase.

Second, the real implementation of PBFs will be further analyzed. The effects of quantization and finite arithmetic will be researched. Implementation structure modification will also be investigated.

Finally, new applications are to be studied. These include power-efficient techniques for SDR, the use of PBFs in SI applications not covered in this work, and the application of PBFs in biomedical engineering.

APPENDIX: MATLAB CODE

Several Matlab functions are given in this appendix. Some of the functions call some of the other functions. For ease of use, copy and save all of the functions into a single directory. Per Matlab protocol, each function should be saved with the same name as the function itself.

PBF Optimization Example 1

This function will return the coefficient matrix (3.36) of the optimized PBF and display the minimized maximum error, δ, and $\mathbf{c}_E$. It will also plot the frequency and impulse responses.

```
function C = PBF_optim_examp


% PBF parameters

N = 6; % Number of polynomial pieces

M = 3; % Order of the polynomial pieces

a = 1; b = -1/2; % Basis function, psi_m(n,T,t), constants

bsave = b; % b will get used, so save it for later


% Passband / Stopband definition and weighting

omega_pass = 2*pi*.2; % passband edge

omega_stop = 2*pi*.8; % stopband edge

passband   = linspace( 2*pi*.01, omega_pass, 100 ); % passband

stopband   = linspace( omega_stop, 2*pi*4, 500 ); % stopband

omega_p    = [passband stopband]; % frequency points to optimize

P          = length(omega_p);

K_pass     = 10; % passband weight

K_stop     = 1; % stopband weight
```

```
W          = K_pass*(omega_p<=omega_pass)+...

             K_stop*(omega_p>=omega_stop); % Total weighting function

w_hat      = 1./W'; % Invert the weighting to fit the linear program

d          = (omega_p <= omega_pass)'; % Desired response --> 1's in the

                                       % passband and 0's in the stopband


% Construct matrices for optimization

Psi_hat_E = Psi_hat_even( M, N, omega_p, a, b );

A       = [ Psi_hat_E, -w_hat;

           -Psi_hat_E, -w_hat ];

b       = [ d ; -d ];

g       = [ zeros( N/2*(M+1), 1 ) ; 1 ];


% Solve linear program

x     = linprog(g,A,b);

cE    = x(1:end-1)

delta = x(end)


% Build the C matrix

C = cE_to_C(cE,N,M);
```

```
% Compute the frequency response for verification

b        = bsave; % we want the basis constant

omega_p  = linspace( 2*pi*.01, 2*pi*12, 1024 );

ha_omega = PBF_freq_resp(C, omega_p, a, b);


% Plot versus f

f_p = omega_p / (2*pi);

plot( f_p, 20*log10(abs(ha_omega)),'k','linewidth',2);

ylim([-80 10]); grid on;

xlabel('Normalized Frequency, f')

ylabel('Normalized Magnitude (dB)');


% Compute the impulse response

i_der           = 0; % don't compute any derivatives

mu              = [0:.01:.99]'; % mimic continuous time

[n_mu, ha_n_mu] = PBF_imp_resp(C,mu,a,b,i_der);

figure; plot(n_mu, ha_n_mu, 'k'); grid on; hold on;

mu              = .25; % sample the impulse response

[n_mu, ha_n_mu] = PBF_imp_resp(C,mu,a,b,i_der);

stem(n_mu, ha_n_mu, 'k', 'filled');

legend('continuous-time','sampled response');
```

Convert Even PBF Coefficient Vector to Full Matrix

This function transforms $\mathbf{c}_E$ of (3.68) to the PBF coefficient matrix $\mathbf{C}$ of (3.36).

```
% Translate cE to C

function C = cE_to_C(cE,N,M)



mscale = repmat( ((-1).^(0:M)), N/2, 1 ); % cm(n) = (-1)^m * cm(N-n-1)

C      = [ reshape(cE,M+1,N/2)';

            mscale .* flipud(reshape(cE,M+1,N/2)') ];

end
```

Compute Even PBF Basis Frequency Response Vector

Given a length P frequency vector, $\boldsymbol{\omega}_p$, M, N, a, and b, this function returns $\hat{\boldsymbol{\Psi}}_E(\omega_p)$ of (3.70).

```
function Psi_hat_E = Psi_hat_even( M, N, omega_p, a, b )


% Generate Basis Frequency Response Matrix
w = omega_p; P = length( w ); temp_k = 0; mn = 1;
Psi_hat = zeros( P, (M+1)*(N/2-1) );
for p = 1:P
    for n = 0:N/2-1
        for m = 0 : M
            for k = 0 : m
                temp_k = temp_k + ...
                         exp(-j*w(p)*n) .*...
                      ( a^k * ( factorial(m) / factorial(m-k) ) .* ...
                         ( 1 ./ (j*w(p)) ).^(k+1) .* ...
                         ( b^(m-k) - exp(-j*w(p)) .* (a+b).^(m-k) ) );
            end
            Psi_hat_E( p, mn ) = 2 * real( exp(j*w(p)*N/2) * temp_k );
            temp_k           = 0; mn = mn + 1;
        end

    end; mn = 1;

end;
```

Convert Full Matrix to PBF Coefficient Vector

This function transforms the PBF coefficient matrix $\mathbf{C}$ of (3.36) to the vector $\mathbf{c}$ of (3.58).

```
function c = C_to_c(C)


[N,MM] = size(C);

M      = MM - 1;

c      = reshape(C',N*(M+1),1);

end
```

Compute PBF Basis Frequency Response Vector

Given a length P frequency vector, $\boldsymbol{\omega}_p$, M, N, a, and b, this function returns $\hat{\boldsymbol{\Psi}}(\omega_p)$ of (3.60).

```
function Psi_hat = Psi_hat_omega( M, N, omega_p, a, b )


% Generate Basis Frequency Response Matrix

w = omega_p; P = length( w ); temp_k  = 0; mn = 1;

Psi_hat = zeros( P, (M+1)*(N-1) );

for p = 1:P

    for n = 0:N-1

        for m = 0 : M

            for k = 0 : m

                temp_k = temp_k + ...

                    exp(-j*w(p)*n) .*...

                    ( a^k * ( factorial(m) / factorial(m-k) ) .* ...

                    ( 1 ./ (j*w(p)) ).^(k+1) .* ...

                    ( b^(m-k) - exp(-j*w(p)) .* (a+b).^(m-k) ) );

            end

                Psi_hat( p, mn ) = temp_k;

                temp_k           = 0; mn = mn + 1;
        end

    end; mn = 1;

end
```

Compute PBF Frequency Response

Given a length P frequency vector, a, b, and $\mathbf{C}$ of (3.36), this function returns $\mathbf{h}_a(\omega_p)$ by computing (3.59).

```
function ha_omega = PBF_freq_resp(C, omega_p, a, b)


[N,MM]   = size(C);

M        = MM - 1;

c        = C_to_c(C);

Psi_hat  = Psi_hat_omega( M, N, omega_p, a, b );

ha_omega = Psi_hat*c;


end
```

Compute PBF Impulse Response

Given a length P vector of μ values, a, b, and $\mathbf{C}$ of (3.36), this function returns the i^{th} derivative of the impulse response of (3.45), given in (3.49,3.50) and the normalized time axis $(n+\mu)T$, where $T=1$.

```
function [n_mu, ha_n_mu] = PBF_imp_resp( C, mu_p, a, b, i );


[N,MM]   = size(C);

M        = MM-1;

muab     = a*mu_p+b;

P        = length(muab);

mu_mat   = repmat(muab',M+1,1);

Delta_i = zeros(M+1,M+1);


% Create derivative transformation matrix

Delta_i = der_trans_mat(M,i,a,b);


% Construct matrix of mu vectors

for p = 1:P

    mu_mat(:,p) = ( ( mu_mat(:,p)'.^(0:M)' ) );

end
```

```
% Compute Impulse response matrix

Hn = C * Delta_i * mu_mat;


% Reshape matrix to vector of impulse response samples

ha_n_mu = reshape( Hn’, N*P, 1 ); % The impulse response

n_mu = reshape( ( repmat( mu_p’, N, 1 ) +...

       repmat([0:N-1]’,1,P))’, N*P, 1 ); % The normalized time

                                         % axis
```

Compute Derivative Transformation Matrix

Given M, a, b, and i from (3.51), this function returns the $(M+1) \times (M+1)$ matrix $\mathbf{\Delta}^{(i)}$

```
function Delta_i = der_trans_mat(M,i,a,b)


Delta_i = circshift( diag([ zeros(1,i), a^i * factorial(i:M)./...
                            factorial(0:M-i) ]), [0 -i] );
```

Time Domain Equivalence Matrices

Given N, M, a, b, and num_der, this function will return the equivalence matrices $\mathbf{A}_{\text{eq}}$ and $\mathbf{b}_{\text{eq}}$ for constraining a PBF impulse response to have num_der continuous derivatives. A value of num_der$= -1$ does nothing, num_der$= 0$ makes the impulse response itself continuous, and num_der$= k$ makes the impulse response and its derivatives up to the kth derivative continuous.

```
function [Aeq,beq] = cont_eq_matrices(N,M,a,b,num_der)


if num_der < 0
    % If no continuity is required, do nothing!
    Aeq = [];
    beq = [];
else


    % Create the P matrix
    P          = eye(N/2);      % [ e_1 e_2 ... e_N/2 ]
    P(:,end) = zeros(N/2,1); % zero out the last column


    % Create the Q matrix
    Q          = circshift(P,[0 1]);
```

```
% Create the b1_tilde, b2_tilde, and b3_tilde vectors
b1_tilde = zeros(N/2,1);
b2_tilde = zeros(N/2,1);
b3_tilde = zeros(N/2,1);

% Create the R matrix
R          = zeros(N/2); % [ 0vec 0vec ... 0vec ]
R(1,1)     = 1;          % [ e_1 0vec ... 0vec ]

% Each continuous derivative requires its own matrix set
for i = 0:num_der

    % Create A1_tilde matrix
    M_muE0   = mu_matrix_even(0,M,N,a,b,i);
    M_muE1   = mu_matrix_even(1,M,N,a,b,i);
    A1_tilde = P*M_muE1 - Q*M_muE0;

    % Create A2_tilde matrix
    A2_tilde = R*M_muE0;

    % Create the Aeq and beq matrices
```

```
A_tilde = [A1_tilde ; A2_tilde];

Aeqi    = [A_tilde , zeros(N,1)];

beqi    = [b1_tilde ; b2_tilde];


% If ith derivative is odd, make center of impulse response = 0 to avoid a

% point in the (i-1)th derivative

if mod(i,2)~=0

    S          = zeros(N/2); % [ 0vec 0vec ... 0vec ]

    S(end,end) = 1;          % [ 0vec 0vec ... e_N/2 ]


    % Create A3_tilde matrix

    A3_tilde = S*M_muE1;


    % Create the Aeq and beq matrices

    A_tilde = [A_tilde ; A3_tilde];

    Aeqi    = [A_tilde , zeros(3*N/2,1)];

    beqi    = [beq ; b3_tilde];

end


if i>0

    Aeq = [Aeq;Aeqi];
```

```
            beq = [beq;beqi];

        else

            Aeq = Aeqi;

            beq = beqi;

        end


    end

end
```

Compute The Mu Matrix for Even Length PBFs

Given N, M, a, b, μ and i, this function will return the matrix $\mathbf{M}_E^{(i)}(\mu)$ of (3.74).

```
function M_muE = mu_matrix_even(mu,M,N,a,b,i)



Delta_i = der_trans_mat(M,i,a,b);

mu_T    = [ (Delta_i*(a*mu+b).^(0:M)')' zeros(1,(N/2-1)*(M+1)) ];

M_muE   = repmat(mu_T,N/2,1);

for n = 0:N/2-1;

    M_muE(n+1,:) = circshift(M_muE(n+1,:)', n*(M+1))';

end
```

PBF Optimization Example 2

This function will return the coefficient matrix (3.36) of a PBF optimized with time domain impulse response continuity constraints and display the minimized maximum error, δ, and $\mathbf{c}_E$. It will also plot the frequency and impulse responses.

```
function C = PBF_optim_examp_eq


% PBF parameters

N = 6; % Number of polynomial pieces

M = 3; % Order of the polynomial pieces

a = 1; b = -1/2; % Basis function, psi_m(n,T,t), constants

bsave   = b; % b will get used, so save it for later

num_der = 0; % number of continuous derivatives, -1

             % if the impulse response itself is not

             % continuous


% Passband / Stopband definition and weighting

omega_pass = 2*pi*.2; % passband edge

omega_stop = 2*pi*.8; % stopband edge

passband   = linspace( 2*pi*.01, omega_pass, 40 ); % passband

stopband   = linspace( omega_stop, 2*pi*4, 200 );  % stopband
```

```
omega_p    = [passband stopband]; % frequency points to optimize

P          = length(omega_p);

K_pass     = 10;  % passband weight

K_stop     = 1;   % stopband weight

W          = K_pass*(omega_p<=omega_pass)+...

             K_stop*(omega_p>=omega_stop); % Total weighting function

w_hat      = 1./W'; % Invert the weighting to fit the linear program

d          = (omega_p <= omega_pass)'; % Desired response --> 1's in the

                                       % passband and 0's in the stopband


% Construct matrices for optimization

Psi_hat_E = Psi_hat_even( M, N, omega_p, a, b );

A         = [ Psi_hat_E, -w_hat;

             -Psi_hat_E, -w_hat ];

b         = [ d ; -d ];

g         = [ zeros( N/2*(M+1), 1 ) ; 1 ];


% Include Time-Domain Continuity Constraints

[Aeq,beq] = cont_eq_matrices(N,M,a,bsave,num_der);
```

```
% Solve linear program

x       = linprog(g,A,b,Aeq,beq);

cE      = x(1:end-1)

delta   = x(end)


% Build the C matrix

C = cE_to_C(cE,N,M);


% Compute the frequency response for verification

b        = bsave; % we want the basis constant

omega_p  = linspace( 2*pi*.01, 2*pi*12, 1024 );

ha_omega = PBF_freq_resp(C, omega_p, a, b);


% Plot versus f

f_p = omega_p / (2*pi);

plot( f_p, 20*log10(abs(ha_omega)),'k','linewidth',2);

ylim([-80 10]); grid on;

xlabel('Normalized Frequency, f')

ylabel('Normalized Magnitude (dB)');


% Compute the impulse response
```

```
i_der           = 0;

mu              = [0:.01:.99]'; % mimic continuous time

[n_mu, ha_n_mu] = PBF_imp_resp(C,mu,a,b,i_der);

figure; plot(n_mu, ha_n_mu, 'k'); grid on; hold on;


% Compute the derivative of the impulse response

[n_mu, ha_n_mu] = PBF_imp_resp(C,mu,a,b,i_der+1);

plot(n_mu, ha_n_mu, '--k'); grid on; hold on;

legend('impulse response','derivative response');
```

LIST OF REFERENCES

[1] M. T. Hunter, W. B. Mikhael, and A. G. Kourtellis, “Direct if synthesis of vector modulated signals using real-time signal processing,” in *Proceedings of the 2008 IEEE International Instrumentation and Measurement Technology Conference, 2008. I2MTC 2008.*, Victoria, Vancouver Island, Canada, May 2008, pp. 1853–1858.

[2] ——, “Wideband digital downconverters for synthetic instrumentation,” *IEEE Transactions on Instrumentation and Measurement*, 2008, to be published.

[3] ——, “Flexible, high performance, fixed sample clock signal synthesis,” *IEEE Transactions on Instrumentation and Measurement*, 2009, submitted for publication.

[4] M. T. Hunter, W. B. Mikhael, and T. Tocco, “Arbitrary waveform generators for synthetic instrumentation,” in *Proceedings of the IEEE Systems Readiness Technology Conference, 2008. AUTOTESTCON ’08.*, Salt Lake City, MD, Sep. 2008, to be published.

[5] M. Hunter, “Efficient fft-based spectral analysis using polynomial-based filters for next generation test systems,” in *Proceedings of the IEEE Systems Readiness Technology Conference, 2007. AUTOTESTCON ’07.*, Baltimore, MD, Sep. 2007, pp. 677–686.

[6] M. Hunter and W. Mikhael, “Fixed sample clock fpga-based arbitrary waveform generator,” U.S. patent application, 2008.

[7] R. W. Lowdermilk and F. J. Harris, “Vector signal analyzer implemented as a synthetic instrument,” in *Proceedings of the IEEE Systems Readiness Technology Conference, 2007. AUTOTESTCON 2007.*, Baltimore, MD, Sep. 2007, pp. 157–166.

[8] ——, “Signal conditioning and data collection in synthetic instruments,” in *Proceedings of the IEEE Systems Readiness Technology Conference, 2003. AUTOTESTCON 2003.*, Anaheim, CA, Sep. 2003, pp. 426–431.

[9] M. Sadiku and C. Akujuobi, “Software-defined radio: a brief overview,” *IEEE Potentials*, vol. 23, pp. 14–15, oct-nov 2004.

[10] P. Burns, *Software Defined Radio for 3G*, ser. Mobile Communications. Artech House, 2003.

[11] J. Mitola, *Software Radio Architecture*. Wiley-Interscience, 2000.

[12] P. B. Kenington, *RF and Baseband Techniques for Software Defined Radio*, ser. Mobile Communications. Artech House, 2005.

[13] P. Pragrastis, I. Sihra, and M. N. Granieri, “Synthetic instrumentation: Contemporary architectures and applications (part ii),” *RF DESIGN*, pp. 12–19, Nov. 2004.

[14] F. Sheikh and S. Masud, "Efficient sample rate conversion for multi-standard software defined radios," in *IEEE International Conference on Acoustics, Speech and Signal Processing, 2007. ICASSP 2007.*, vol. 2, Apr. 2007, pp. II–329 – II–332.

[15] W. A. Abu-Al-Saud and G. Stuber, "Efficient sample rate conversion for software radio systems," *IEEE Transactions on Signal Processing*, vol. 54, pp. 932–939, 2006.

[16] U. Mengali and A. N. D'Andrea, *Synchronization Techniques for Digital Receivers*, ser. Applications of Communications Theory. Plenum Press, 1997.

[17] H. Meyr, M. Moenclay, and S. A. Fetchel, *Digital Communication Receivers: Synchronization, Channel Estimation, and Signal Processing*. Wiley, 1998.

[18] D. Babic and M. Renfors, "Power efficient structure for conversion between arbitrary sampling rates," *IEEE Signal Processing Letters*, vol. 12, no. 1, pp. 1–4, Jan. 2005.

[19] S. Andrews and F. Harris, "Polynomial approximations of interpolants," in *Conference Record of the Thirty-Third Asilomar Conference on Signals, Systems, and Computers, 1999.*, vol. 1, Pacific Grove, CA, Oct. 1999, pp. 447–451.

[20] S. Cucchi, F. Desinan, G. Parladori, and G. Sicuranza, "Dsp implementation of arbitrary sampling frequency conversion for high quality sound application," in *1991 International Conference on Acoustics, Speech, and Signal Processing, 1991. ICASSP-91.*, vol. 5, Apr. 1991, pp. 3609–3612.

[21] D. Babic and M. Renfors, "Reconstruction of non-uniformly sampled signal using transposed farrow structure," in *Proceedings of the 2004 International Symposium on Circuits and Systems, 2004. ISCAS '04.*, vol. 3, May 2004, pp. III – 221–4.

[22] D. Babic, "Reconstruction of non-uniformly sampled signal after time-interleaved adcs using transposed farrow structure," in *The International Conference on Computer as a Tool, 2005. EUROCON 2005.*, vol. 2, 2005, pp. 1622–1625.

[23] J. Vesma, R. Hamila, M. Renfors, and T. Saramaki, "Continuous-time signal processing based on polynomial approximation," in *Proceedings of the 1998 IEEE International Symposium on Circuits and Systems, 1998. ISCAS '98.*, vol. 5, May 1998, pp. 61–65.

[24] R. E. Crochiere and L. R. Rabiner, *Multirate Digital Signal Processing*. Prentice Hall, 1983.

[25] A. V. Oppenheim, R. W. Schafer, and J. R. Buck, *Discrete-Time Signal Processing*. Prentice Hall, 1999.

[26] A. Colson, R. G. Vaughan, and M. Poletti, "Frequency shifting using bandpass sampling," *IEEE Transactions on Signal Processing*, vol. 42, no. 6, pp. 1556–1559, Jun. 1994.

[27] R. G. Vaughan, N. L. Scott, and D. R. White, "The theory of bandpass sampling," *IEEE Transactions on Signal Processing*, vol. 39, no. 9, pp. 1973–1984, Sep. 1991.

[28] M. Valkama and M. Renfors, "A novel image rejection architecture for quadrature radio receivers," *IEEE Transactions on Circuits and Systems—Part II: Analog and Digital Signal Processing*, vol. 51, no. 2, pp. 61–68, Feb. 2004.

[29] F. M. Gardner, "Interpolation in digital modems - part i: Fundamentals," *IEEE Transactions on Communications*, vol. 41, pp. 501–507, Mar. 1993.

[30] L. Erupa, F. M. Gardner, and R. A. Harris, "Interpolation in digital modems - part ii: Implementation and performance," *IEEE Transactions on Communications*, vol. 41, pp. 998–1008, Jun. 1993.

[31] J. Vesma, "A frequency-domain approach to polynomial-based interpolation and the farrow structure," *IEEE Transactions on Circuits and Systems—Part II: Analog and Digital Signal Processing*, vol. 47, no. 3, pp. 206–209, Mar. 2000.

[32] T. Hentschel and G. Fettweis, "Sample rate conversion in software defined radio," *IEEE Communications Magazine*, vol. 38, pp. 142–150, Aug. 2000.

[33] R. E. Crochiere and L. R. Rabiner, "Interpolation and decimation of digital signals - a tutorial review," in *Proceedings of the IEEE*, vol. 69, Mar. 1981, pp. 300–331.

[34] P. P. Vaidyanathan, *Multirate Systems and Filter Banks*, ser. Prentice Hall Signal Processing Series. Prentice Hall, 1993.

[35] E. B. Hogenauer, "An economical class of digital filters for decimation and interpolation," *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol. 29, no. 2, pp. 155–162, Apr. 1981.

[36] F. J. Harris, *Multirate Signal Processing for Communication Systems*. Prentice Hall, 2006.

[37] T. A. Ramstad, "Digital methods for conversion between arbitrary sampling frequencies," *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol. 32, pp. 577–591, Jun. 1984.

[38] C. W. Farrow, "A continuously variable digital delay element," in *Proc. IEEE Int. Symp. Circuits and Systems*, Espoo, Finland, Jun. 1998, pp. 2641–2645.

[39] T. Laakso, V. Valimaki, M. Karjalainen, and U. Laine, "Splitting the unit delay," *IEEE Signal Processing Magazine*, vol. 13, pp. 30–60, 1996.

[40] J. Vesma and T. Saramki, "Interpolation filters with arbitrary frequency response for all-digital receivers," in *Int. Symp. Circuits and Systems*, Atlanta, GA, May 1996, pp. 568–571.

[41] L. R. Rabiner and B. Gold, *Theory and Application of Digital Signal Processing*. Prentice-Hall, 1975.

[42] S. K. Mitra and J. F. Kaiser, *Handbook for Digital Signal Processing*. Wiley-Interscience, 1993.

[43] T. Hentschel and G. Fettweis, "Continuous-time digital filters for sample-rate conversion in reconfigurable radio terminals," in *the European Wireless 2000*, Dresden, Germany, Sep. 2000, pp. 55–59.

[44] D. Babic, J. Vesma, T. Saramki, and M. Renfors, "Implementation of the transposed farrow structure," in *Int. Symp. Circuits and Systems*, vol. 4, May 2002, pp. 5–8.

[45] D. Babic and M. Renfors, "Polynomial-based filters with polynomial pieces of different lengths for interpolation," in *3rd Annual International Symposium on Image and Signal Processing Circuits and Systems. ISPA 2003.*, vol. 2, Sep. 2003, pp. 740–744.

[46] D. Babic, T. Saramki, and M. Renfors, "Prolonged transposed polynomial based filters for decimation," in *Int. Symp. Circuits and Systems*, vol. 4, May 2003, pp. 317–320.

[47] M. Henker, T. Hentschel, and G. P. Fettweis, "Time-variant cic-filters for sample-rate conversion with arbitrary rational factors," in *The 6th IEEE International Conference on Electronics, Circuits and Systems, 1999. Proceedings of ICECS '99.*, vol. 1, Sep. 1999, pp. 67–70.

[48] D. Babic, V. Lehtinen, and M. Renfors, "Discrete-time modeling of polynomial-based interpolation filters in rational sampling rate conversion," in *Proceedings of the 2003 International Symposium on Circuits and Systems, 2003. ISCAS '03.*, vol. 4, May 2003, pp. 321 – 324.

[49] V. Lehtinen, D. Babic, and M. Renfors, "Comparison of continuous-and discrete-time modelling of polynomial-based interpolation filters," in *Proceedings of the 6th Nordic Signal Processing Symposium, 2004. NORSIG 2004*, 2004, pp. 49 – 54.

[50] F. Lopez, J. Vesma, T. Saramaki, and M. Renfors, "The effects of quantizing the fractional interval in interpolation filters," in *Proceedings IEEE Nordic Signal Processing Conference, NORSIG 2000*, Jun. 2000.

[51] F. Lopez, J. Vesma, and M. Renfors, "Defining the wordlength of the fractional interval in interpolation filters," in *Proceedings of the XI European Signal Processing Conference, EUSIPCO '2002.*, vol. I, Sep. 2002, pp. 679–682.

[52] R. E. Ziemer and W. H. Tranter, *Principles of Communications*. John Wiley and Sons, 2002.

[53] M. Unser, "Splines: A perfect fit for image and signal processing," *IEEE Signal Processing Magazine*, vol. 16, pp. 22–38, Nov. 1999.

[54] M. Unser, A. Aldroubi, and M. Eden, "B-spline signal processing: Part i - theory," *IEEE Transactions on Signal Processing*, vol. 41, pp. 821–833, Feb. 1993.

[55] ——, "B-spline signal processing: Part ii - efficiency, design, and applications," *IEEE Transactions on Signal Processing*, vol. 41, pp. 834–848, Feb. 1993.

[56] G. B. Folland, *Fourier Analysis and it's Applications*, ser. Mathematics. Wadsworth and Brooks/Cole, 1992.

[57] *Optimization Toolbox 4 User's Guide*, The Mathworks, Natick, MA, 2008. [Online]. Available: http://www.mathworks.com/access/helpdesk/help/pdf_doc/optim/optim_tb.pdf

[58] F. Harris, "Performance and design of farrow filter used for arbitrary resampling," in *Proceedings of the 13th International Conference on Digital Signal Processing, DSP 97*, vol. 2, Jul. 1997, pp. 595–599.

[59] M. Henker and G. Fettweis, "Combined filter for sample rate conversion, matched filtering, and symbol synchronization in software radio terminals," in *the European Wireless 2000*, Dresden, Germany, Sep. 2000, pp. 61–66.

[60] "Ni pxi/pci-5142 specifications," National Instruments Corporation, Austin, Texas, Aug. 2007.

[61] H. Howard and F. Cruger, "Re-configurable hardware for synthetic wideband instruments," in *AUTOTESTCON Proceedings, 2001. IEEE Systems Readiness Technology Conference*, Valley Forge, PA, Aug. 2001, pp. 632 – 647.

[62] *R&S FSQ Operating Manual*, Rohde & Schwarz, Munich, Germany, 2005, version 6.

[63] K. Schmidt and M. Freidhof, "Signal processing device," U.S. patent application 11/102,957, Nov. 3, 2005.

[64] "Vector signal analysis basics," Application Note, Agilent Technologies Company, Jul. 2004. [Online]. Available: http://cp.literature.agilent.com/litweb/pdf/5989-1121EN.pdf

[65] E. Hoyer and R. Stork, "The zoom fft using complex modulation," in *IEEE International Conference on Acoustics, Speech, and Signal Processing, ICASSP '77*, vol. 2, May 1977, pp. 78 – 81.

[66] R. A. Witte, *Spectrum and Network Measurements*. Scitech, 2006.

[67] P. Pragrastis and M. N. Granieri, "The upconverter - a critical synthetic instrument technology," in *Proceedings of the IEEE Systems Readiness Technology Conference, 2005. AUTOTESTCON 2005.*, Sep. 2005, pp. 290–296.

[68] "The abcs of arbitrary waveform generation," Application Note, Agilent Technologies, Inc., Nov. 2005. [Online]. Available: http://cp.literature.agilent.com/litweb/pdf/5989-4138EN.pdf

[69] D. Strassberg, "Making waves: Eight years later, details still matter," *EDN*, pp. 46–57, Sep. 2006. [Online]. Available: http://www.edn.com/contents/images/6368440.pdf

[70] "Onboard signal processing (osp) on national instruments signal generators," White Paper, National Instruments, Jan. 2007. [Online]. Available: http://zone.ni.com/devzone/cda/tut/p/id/2859

[71] “Agilent n6030a series arbitrary waveform generators users guide,” Agilent Technologies, Inc., May 2006. [Online]. Available: http://cp.literature.agilent.com/litweb/pdf/N6030-90004.pdf

[72] “Ni pxi-5441 specifications,” National Instruments Corporation, Austin, Texas, Nov. 2007.

[73] D. Babic, A. Ghadam, and M. Renfors, “Polynomial-based filters with odd number of polynomial segments for interpolation,” *IEEE Signal Processing Letters*, vol. 11, no. 2, pp. 171–174, Feb. 2004.

[74] J. G. Proakis, *Digital Communications*, 3rd ed., ser. Electrical and Computer Engineering. McGraw-Hill, 1995.

[75] “Ad9954 data sheet, rev. a,” Analog Devices, Jan. 2007. [Online]. Available: http://www.analog.com/UploadedFiles/Data_Sheets/AD9954.pdf

[76] B. Sklar, *Digital Communications: Fundamentals and Applications*, 2nd ed. Prentice Hall, 2000.

[77] T. S. Rappaport, *Wireless Communications: Principles and Practice*, 2nd ed., ser. Prentice Hall Communications Engineering and Emerging Technology, T. S. Rappaport, Ed. Prentice Hall PTR, 2002.

[78] K. W. Martin, “Complex signal processing is not complex,” *IEEE Transactions on Circuits and Systems*, vol. 51, no. 9, pp. 1823–1836, Sep. 2004.

[79] “Ad9786 data sheet, rev. b,” Analog Devices, 2005. [Online]. Available: http://www.analog.com/UploadedFiles/Data_Sheets/AD9786.pdf

[80] B. Razavi, *RF Microelectronics*, ser. Prentice Hall Communications Engineering and Emerging Technology, T. S. Rappaport, Ed. Artech House, 1998.

[81] “Spartan 3 family fpga data sheet,” Xilinx, Jun. 2008. [Online]. Available: http://www.xilinx.com/support/documentation/data_sheets/ds099.pdf

[82] “Agilent mxa signal analyzer n9020a,” Agilent, Oct. 2007. [Online]. Available: http://cp.literature.agilent.com/litweb/pdf/5989-4942EN.pdf