Multiplierless Decimation and Commercial Postfiltering of a Discrete-Time Signal

by

Mark Allan Story

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

Master of Engineering

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

June 1997

Author .....
Department of Electrical Engineering and Computer Science
March 14, 1997

Certified by .....
Anantha Chandrakasan
Assistant Professor
Thesis Supervisor

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

Multiplierless Decimation and Commercial Postfiltering of a Discrete-Time Signal

by

Mark Allan Story

Submitted to the Department of Electrical Engineering and Computer Science
on March 14, 1997, in partial fulfillment of the
requirements for the degree of
Master of Engineering

Abstract

This thesis, presents the design of a digital filtering system for an analog microchip. A digital decimator is placed directly on the microchip with the analog circuitry. This decimator is a modification of the standard Cascaded Integrator Comb decimator. Further digital filtering is accomplished off-chip using Altera FLEX8000 PLDs. A technique is developed to cope with delays in first-order highpass and lowpass IIR filters. An FIR lowpass filter is modified to correct for passband attenuation caused by the decimator.

Thesis Supervisor: Anantha Chandrakasan
Title: Assistant Professor

Acknowledgments

First I would like to thank the people of Tektronix, who made this thesis project possible. Special thanks to Dan Knierim, who was my direct mentor. His continual help and direction were invaluable to me throughout the project. His efforts during my untimely completion of this thesis are especially appreciated. Thanks to Dan Wolaver, for entrusting this portion of design work to me as a thesis project, and for helping me during it.

Also at Tektronix, Thanks to Tim Bennington-Davis and Bob Woolhiser, for finding the project and making my place in it. Thanks to Ray Veith, for his ready ear and sound advice. Thanks to Tom Lane, for his help with troublesome software from outside Tektronix. Thanks to Phil Schniter, for his help and direction on signal processing issues. Thanks to Steve Blazo, for helping me see the context of the project. And thanks to the rest of the people of Tektronix, for making me comfortable at Tektronix and helping me to enjoy my time there.

Thanks to Professor Anantha Chandrakasan at MIT, for his help, patience, and hard work as my thesis advisor. Thanks also to Anne Hunter, for her patience and assistance.

Thanks to Caleb Chrome and Jeff Richardson of Altera Corporation, for their help finding information about Altera hardware.

Thanks to my parents for making it possible for me to come to MIT, as well as to Gramma, for her special financial help. Thanks to my whole family, for their support throughout my stay here. Thanks to my friends in Oregon and in Boston for all of the fun times we shared.

Contents

1 Introduction **12**

2 Decimator Analysis **15**

2.1 Cascaded Integrator Comb Decimators 16

2.2 Modification of Cascaded Integrator Comb Decimator 20

2.3 Our Specific Decimator 24

2.4 Signal "Wrapping" 27

3 Decimator Implementation **32**

3.1 Chip Boundary 33

3.2 Unwrapping Bit Extender 33

3.3 Accumulators 36

3.4 Intermediate Stage and Decimation 38

3.5 Output 40

3.6 Carry-Chain Pipeline Slant 43

3.7 Control 44

3.8 Conclusion 45

4 Recursive Filters **50**

4.1 Structure for IIR Filters 50

4.2 Pipelining Recursive Loops 52

4.3 Signal "Wrapping" Effects 56

4.4 Multipliers 59

5 Implementation of Off-chip Filters **63**

5.1 Summary of Altera FLEX8000 Capabilities 66

5.2 Directly Implemented Filters 67

5.3 Filters with Loop Delays 77

5.4 FIR Lowpass Filters 86

5.5 Troublesome Filters 86

5.6 Input Functions 90

6 Conclusion **92**

A Unwrapping Bit Extender Optimization **94**

B FIR Filter Plots **96**

C MATLAB Code for FIR Filters **101**

C.1 firs.m 101

C.2 remezfit.m 103

C.3 quantize.m 104

List of Figures

2-1 Structure of a K-stage CIC decimator with decimation factor M. (Scaling factor $(\frac{1}{M})^K$ ignored.) 16

2-2 Frequency response magnitude of an 6-long rectangular filter $R'_6(\Omega)$. . 17

2-3 Frequency responses of (a) $Num_B(\Omega)$ (b) $\frac{1}{Den_B(\Omega)}$ (c) $R'_M(\Omega)$ 18

2-4 $A_{2,1}(M,\omega)$, the ratio of second-stopband aliasing to first-stopband aliasing, as a function of M, in the limit as $\omega \to 0$. 19

2-5 Decomposition of a 6-long rectangular filter, Frequency domain: (a) dashed line: $R'_3(\Omega)$, (b) dash-dotted line: $R'_2(3\Omega)$, (c) solid line: $R'_6(\Omega) = R'_3(\Omega) \cdot R'_2(3\Omega)$ 21

2-6 Decomposition of a 6-long rectangular filter, Time domain. (a) $r'_3[n]$, (b) $r'_2[\frac{n}{3}]$, (c) $r'_6[n] = r'_3[n] * r'_2[\frac{n}{3}]$. 22

2-7 Structure of an improved CIC decimator with K CIC stages and L intermediate stages. Decimation factor M. 22

2-8 Comparison of typical CIC decimators with a typical improved-CIC decimator. In all cases, decimation factor $M = 6$, and frequency responses are normalized to 1 at $\Omega = 0$. (a) Solid line: An improved CIC decimator with $K = 2$ CIC stages and $L = 1$ intermediate stage. (b) Dashed line: A CIC decimator with $K = 2$ stages. (c) Dash-dotted line: A CIC decimator with $K = 3$ stages. 23

2-9 Frequency response (detail of first few stopbands), versus input referenced frequency Ω; and normalized stopband aliasing, versus output referenced frequency ω, of three candidate decimators. First stopbands are highlighted with dash-dotted lines, and the second stopbands are highlighted with dashed lines. Other stopbands are plotted with dotted lines, and the passband is plotted with a solid line. Notice in the stopband aliasing plots that the stopbands have been normalized by dividing by the passband response, to account for later filters that correct passband attenuation. 25

3-1 Partition of the decimator between the analog chip and PLDs. 33
3-2 Straightforward unwrap extender implementation. X0..X13 are data input signals, UES8..UES13 are control signals, and Y0..Y26 are data output signals. 35
3-3 Improved unwrap extender implementation. X0..X13 are data input, UES8..UES13 are control signals, and Y0..Y26 are data output. . . . 37
3-4 Implementation of the three accumulators. 38
3-5 Implementation of the intermediate stage and the decimation. 39
3-6 Timing diagram for intermediate stage control signals, with $M = 12$. Both signals are shown asserted high, that is, a high signal enables the register or makes the latch transparent. 39
3-7 Multiplexor circuit. 42
3-8 Overall circuit. 47
3-9 Timing diagram for control signals when $M = 16$. 48
3-10 Timing diagram for control signals when $M = 12$. 48
3-11 Timing diagram for control signals when $M = 4$. 48
3-12 Timing diagram for control signals when $M = 2$. 49

4-1 Direct Form structures to implement equation 4.1 51
4-2 Alternate structures to implement equation 4.2 51
4-3 IIR filter structure, without extra delays in the large loop. 53

4-4 Modified IIR filter structure, with extra delays in the large loop. . . . 54
4-5 Sum of powers of two multiplier forms using two adders. (a) multiplies by $2^{-s}(1 \pm 2^{-m} \pm 2^{-n})$. (b) multiplies by $2^{-s}(1 \pm 2^{-m})(1 \pm 2^{-n})$. . . . 62

5-1 Filtering System. . . . 64
5-2 Simplified timing diagram for Altera FLEX8000. . . . 67
5-3 IIR filter structures, without pipelined loops, including coefficient multipliers. "Zero pad" modules add zeros onto the LSP, "sign extend" modules add bits identical to the sign bit onto the MSP, and "truncate" modules truncate the LSP. . . . 69
5-4 Maximum Clock Frequency, as a function of A_b, using the circuit with no extra loop delay elements. The solid line represents an A-2 speed grade FLEX81188, and the dashed line represents an A-3 speed grade FLEX8820. . . . 76
5-5 IIR filter structures with pipelined loops, including coefficient multipliers. "Zero pad" modules add zeros onto the LSP, "sign extend" modules add bits identical to the sign bit onto the MSP, and "truncate" modules truncate the LSP. . . . 78
5-6 Maximum Clock Frequency, as a function of A_b, for an adder with input from the same row. The solid line represents an A-2 speed grade FLEX81188, and the dashed line represents an A-3 speed grade FLEX8820. . . . 80
5-7 IIR filter structures with pipelined loops and accumulators, including coefficient multipliers. "Zero pad" modules add zeros onto the LSP, "sign extend" modules add bits identical to the sign bit onto the MSP, and "truncate" modules truncate the LSBs. . . . 82
5-8 Maximum Clock Frequency as a function of A_c, for an adder with an input from a different row. The solid line represents an A-2 speed grade FLEX81188, and the dashed line represents an A-3 speed grade FLEX8820. . . . 84

5-9 Filter to correct ripple of the 400kHz lowpass filter for the 44.736 MHz data rate. This implements the impulse response $h[n] = \delta[n-5] - \epsilon_m \sum_{m=1}^{k} \delta[n-m-5] \approx \frac{\epsilon'}{\epsilon} h_{fix}[n-5]$. Here, $\epsilon_m = (2^{-5})(1+2^{-1}+2^{-4}) = 0.0489$. The portion of the circuit below the dashed line should be placed on the same row. 89

5-10 Input functions: Three first-differencers complete the decimation, and a first-differencer followed by an accumulator "unwraps" the input by r bits. 90

A-1 A single bit of a first-differencer followed by an accumulator. 94

B-1 Frequency response and passband detail of the FIR filter for the 2.048 MHz data rate. 97

B-2 Frequency response and passband detail of the FIR filter for the 8.448 MHz data rate. Includes the effect of decimator passband attenuation. 97

B-3 Frequency response and passband detail of the FIR filter for the 34.368 MHz data rate. Includes the effect of decimator passband attenuation. 98

B-4 Frequency response and passband detail of the FIR filter for the 51.84 MHz data rate. Includes the effect of decimator passband attenuation. 98

B-5 Frequency response and passband detail of the FIR filter for the 139.264 MHz data rate. Includes the effect of decimator passband attenuation. 99

B-6 Frequency response and passband detail of the FIR filter for the 155.52 MHz data rate. Includes the effect of decimator passband attenuation. 99

B-7 Frequency response and passband detail of the FIR filter for the 622.08 MHz data rate. Includes the effect of decimator passband attenuation. 100

B-8 Frequency response and passband detail of the FIR filter to be used with the 250 kHz highpass filter in the 622.08 MHz data rate, cascaded with that filter. Includes the effect of decimator passband attenuation and the 250 kHz IIR filter. 100

List of Tables

2.1 Chosen decimation rates. 27

2.2 Minimum B_y required by each data rate. Because of implementation details to be explained in section 3.5, values for V_{MSB} in parentheses, where given, should be used instead of the ones given by the formula. 30

3.1 Output pins. Note from the 51.84 MHz rate that no more than 21 pins are needed for any data rate. 41

3.2 Control signals which do not vary with time. 45

4.1 All possible coefficients which may be realized with two adders, while increasing the data path by no more than 8 bits. The maximum percentage error from such a coefficient to the number halfway between it and the previous coefficient is also given. Thus the maximum possible error between any desired coefficient and its implementation is 1.734%. 61

5.1 Filters to be implemented. 64

5.2 Timing parameter values for FLEX8820 and FLEX81188. 68

5.3 Filters which may be implemented without adding delay elements in the loop, using FLEX8820s in the A-3 speed grade. The data path width b is 16 bits. 74

5.4 Filters which may be implemented without adding delay elements in the loop, using FLEX81188s in the A-2 speed grade. The data path width b is 24 bits. Coefficient Implementation, frequency error, and C_{iir} are the same as in the previous table. 75

5.5 Filters which may be implemented with $k = 3$ delay elements in the loop, using FLEX8820s in the A-3 speed grade. The data path width b is 16 bits. Boldface indicates a missed specification. 80

5.6 Filters which may be implemented with $k = 3$ delay elements in the loop, using FLEX81188s in the A-2 speed grade. The data path width b is 24 bits. Coefficient Implementation, frequency error, ripple, and C_{iir} are the same as in the previous table. Boldface indicates a missed specification. 81

5.7 Filters which may be implemented with $k = 4$ delay elements in the loop, using FLEX8820s in the A-3 speed grade. The data path width b is 16 bits. Boldface indicates a missed specification. 85

5.8 Filters which may be implemented with $k = 4$ delay elements in the loop, using FLEX81188 chips in the A-2 speed grade. The data path width b is 24 bits. Several coefficient implementations, and therefore frequency error and ripple, have changed from the previous table. C_{iir} is the same as in the previous table, however. Boldface indicates a missed specification. 85

5.9 Symmetric FIR filter coefficients. $C_{fir} = \sum_{j=0}^{15} a_j$ gives the scale constant for each filter. 87

5.10 Filters implemented with rounded coefficients. 88

5.11 FIR filter coefficients for use with the 250 kHz highpass filter for the 622.08 MHz base data rate. 89

Chapter 1

Introduction

This thesis presents the design of a digital filtering system for the output signal of a microchip being designed by Tektronix, Inc. This signal may have any of ten sample rates, ranging up to more than 600 MHz. The signal must be filtered to isolate two frequency bands: a "bandpass" frequency band, and a "lowpass" frequency band at very low frequencies to measure long-term drift. The user may wish to examine both bands simultaneously. Part of the system will be implemented on the chip itself, and part of it will be implemented in Altera FLEX8000 Programmable Logic Devices.

This signal has the unusual property that it is unbounded. However, its first difference is bounded, i.e. the difference between one sample and the previous sample is limited. Since we know that a sample value may not differ by more than a certain limit from the previous sample value, we may represent the signal with a finite number of bits. The filters must be able to filter the unbounded signal, given its finite representation.

A sample rate above 600 MHz is much too fast for reasonable digital filter implementations, including Altera FLEX8000s as planned. However, since both frequency bands of interest are well below the sample rate, we may decimate the signal before the final filtering. Because of the high speed, this will have to be done on the chip. For satisfactory performance, any aliased energy should be attenuated at least 60dB, by the combination of the decimating filter and the lowpass filters specified in table 5.1. Furthermore, the decimator must implement multiple decimation rates, over a

wide enough range to accommodate all sample rates. The decimator must use as few of the chip resources as possible, so that the main functions of the chip can use more chip area and more power. We will use a modified Cascaded Integrator-Comb (CIC) decimator, as developed by Hogenauer[5].

The CIC decimator will attenuate the passband slightly, and leave some aliased energy at frequencies above the frequencies of interest in the two bands. For both bands, the filters will remove aliased energy not in the passband. The CIC attenuation will not be a problem for the "lowpass" band, because the band is narrow enough that the attenuation is insignificant. However, the attenuation of the CIC filters is significant in the "bandpass" band. For this band, an FIR filter will be used to both compensate for the attenuation of the CIC filter in the passband and to simulate the third-order butterworth lowpass portion of the "bandpass" frequency response.

Chapter 2 lays an analytical foundation for the decimator. I will examine the Cascaded Integrator Comb decimator, and propose a modification to reduce computation. The number of integrators and combs required for this application will be identified. Finally, the requirements imposed by the unboundedness of the signal will be examined.

Chapter 3 outlines the circuitry necessary for the portions of the decimator which will be implemented on the microchip. A portion of the decimator which may be implemented in the FLEX8000 PLD hardware will be identified. Special circuitry for dealing with the unboundedness of the signal will be developed. The amount of hardware which will be required by the decimator will be quantified, and requirements of control circuitry will be given.

Chapter 4 lays an analytical foundation for problems encountered with the first-order recursive filters. The chosen structure is presented, and methods are developed for dealing with delay elements in recursive loops and unbounded signals. The implementation of coefficient multiplication will be presented.

Chapter 5 outlines the implementation of the filters in the Altera FLEX8000 architecture. The capabilities of FLEX8000s are first summarized. Circuits for direct implementation of the recursive filters are presented, and filters for which the sample

rates are too high for direct implementation are identified. Circuits for implementation of the remaining filters are developed, using delay elements in the recursive loops. Filters which are not satisfactorily implemented by these circuits are dealt with individually. FIR filter coefficients for emulating the third-order lowpass filters are given. Also, the portions of the decimator which were not placed on the microchip are implemented.

Chapter 2

Decimator Analysis

The on-chip decimating prefilter required must have very low complexity so that it may be implemented on the same chip as the analog functions. At least 60dB of stopband attenuation is required. However, because a lowpass filter with a cutoff frequency no higher than about $\omega_c \approx \frac{\pi}{4}$ will be required on the output, we have a very wide "don't-care" band of about $\frac{\pi}{4} < \omega < \pi$ in which aliasing is acceptable on the output of the decimator. Thus the decimator filter may have a series of wide "don't-care" bands of $\frac{2\pi N}{M} + \frac{\pi}{4M} < \Omega < \frac{2\pi N}{M} + \frac{7\pi}{4M}$, where M is the decimation rate. (Throughout this paper, Ω will denote discrete-time frequencies of the input signal, and ω will denote discrete-time frequencies of the decimated output signal.) Furthermore, FIR filters must be used for higher cutoff frequencies, so the filters may be designed to compensate for modest passband attenuation which may be caused by the decimator. The IIR lowpass filters used (with a few exceptions which will be dealt with later) have much lower cutoff frequencies. So some passband attenuation is acceptable, except at very low frequencies. Lastly, the decimator should work for a wide range of decimation rates, specifically $2 \leq M \leq 16$.

The Cascaded Integrator Comb (CIC) decimator introduced by Hogenauer [5] fits these specifications very well. Only a handful of adders and registers are required to implement such a decimator. Furthermore, the decimation rate may be changed by simply changing the downsampling rate: the structure of the filter does not need to be changed. The trade-off for the simplicity is a narrow stopband (which requires

wide "don't-care" bands) and significant passband attenuation, both of which are acceptable. Thus the CIC decimator is a good starting point for this decimating prefilter. However, careful analysis reveals a way to improve the performance of a CIC decimator.

2.1 Cascaded Integrator Comb Decimators

A K-stage CIC decimator with decimation rate M consists of K "integrator" stages operating at the high input sampling rate f_s, followed by an M-sample downsampler, followed by K "comb" stages operating at the low output sampling rate $\frac{f_s}{M}$. See figure 2-1.

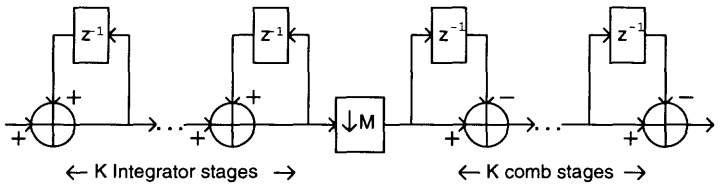

Figure 2-1: Structure of a K-stage CIC decimator with decimation factor M. (Scaling factor $(\frac{1}{M})^K$ ignored.)

This has the same frequency response, referenced to the high sample rate f_s, as K cascaded M-long rectangular filters. The M-length rectangular filter, which will be referred to as $r_M[n]$, is:

$$r_M[n] = (u[n] - u[n-M]) = \sum_{k=0}^{M-1} \delta(n-k)$$

The frequency response, which will be referred to as $R_M(\Omega)$, is:

$$R_M(\Omega) = \frac{\sin(\frac{1}{2}M\Omega)}{\sin(\frac{1}{2}\Omega)} e^{-j\Omega(\frac{M-1}{2})}$$

Because it is more convenient for the analysis of the decimator to normalize frequency responses to unit magnitude at $\Omega = 0$, we will also define $r'_M[n] = \frac{1}{M} r_M[n]$ and

$R'_M(\Omega) = \frac{1}{M} R_M(\Omega)$.

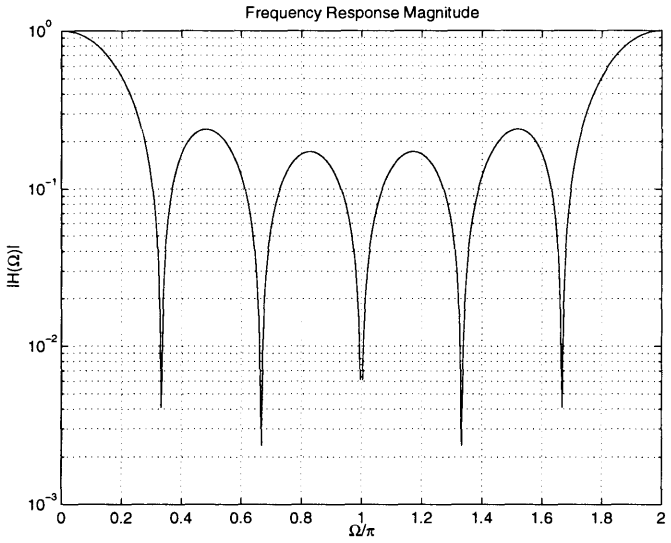

Figure 2-2: Frequency response magnitude of an 6-long rectangular filter $R'_6(\Omega)$.

Figure 2-2 illustrates the frequency response magnitude of the 6-long rectangular filter $R'_6(\Omega)$. Note that the zeros fall at $\Omega = \frac{2\pi N}{M}$ $(1 \leq N \leq M-1)$, the centers of the stopbands of the decimator. We have some output passband ω_p, which is small with respect to π for practical CIC decimators. Thus the passband of $R'_M(\Omega)$ is $|\Omega_p| \leq \frac{\omega_p}{M}$, and the stopbands are $\frac{2\pi N - \omega_p}{M} \leq \Omega_s \leq \frac{2\pi N + \omega_p}{M}$. If one rectangular filter provides insufficient stopband attenuation, we can cascade K rectangular filters. This K-stage CIC filter frequency response magnitude is simply $|(R'_M(\Omega))^K|$.

We may decompose $R'_M(\Omega)$ into the response due to the numerator,

$$Num_B(\Omega) = \sin(\frac{1}{2}M\Omega)e^{-j\Omega(\frac{M-1}{2})}$$

the response due to the denominator,

$$\frac{1}{Den_B(\Omega)} = \frac{1}{M\sin(\frac{1}{2}\Omega)}$$

and a linear phase term with magnitude 1 at all frequencies.

Figure 2-3 illustrates this for a 6-long rectangular filter. To examine the aliasing at the decimated frequency ω due to the N^{th} stopband, we evaluate $|R'_M(\Omega)|$ at

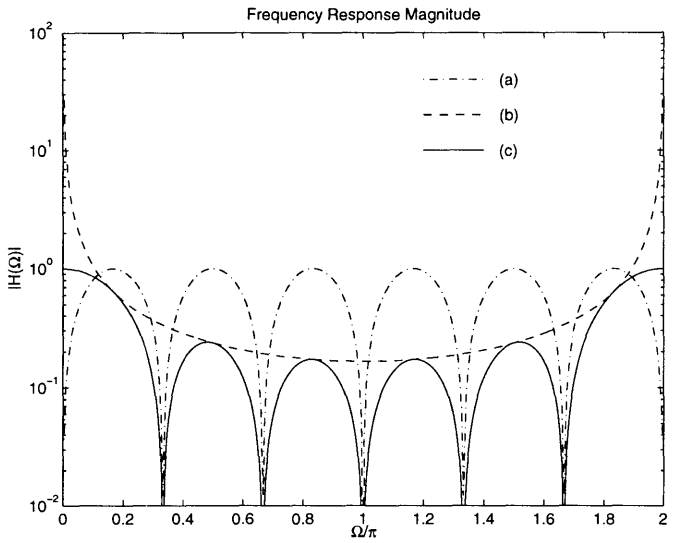

Figure 2-3: Frequency responses of (a) $Num_B(\Omega)$ (b) $\frac{1}{Den_B(\Omega)}$ (c) $R'_M(\Omega)$

$\Omega = \frac{2\pi N}{M} + \frac{\omega}{M}$, with $-\pi < \omega < \pi$. Note that the magnitude response at a given frequency ω due to the numerator, $|Num_B(\Omega)| = |\sin(\pi N + \frac{\omega}{2})|$, is the same for all N. Therefore, The ratio of aliasing at ω from the N_1^{th} stopband to the aliasing at ω from the N_2^{th} stopband is the ratio of the responses to due to the denominators. Therefore,

$$|R'_M(\Omega = \frac{2\pi N_1}{M} + \frac{\omega}{M})| > |R'_M(\Omega = \frac{2\pi N_2}{M} + \frac{\omega}{M})|$$

exactly when

$$|\sin(\frac{\pi N_1}{M} + \frac{\omega}{2M})| < |\sin(\frac{\pi N_2}{M} + \frac{\omega}{2M})|$$

But since $|Den_B(\Omega)| = M \sin\frac{1}{2}\Omega$ is monotonically increasing for $0 \leq \Omega < \pi$, this implies that $N_1 < N_2$ for positive frequency stopbands (i.e. $N_{1,2} < \frac{1}{2}M$ and $-\pi < \omega < \pi$, or $N_{1,2} = \frac{1}{2}M$ and $-\pi < \omega \leq 0$). This tells us that for a given ω, the greatest aliasing comes from the lower frequency stopbands (i.e. lower N). So the aliasing from the first stopband ($N = 1$) is always the greatest. This is the main result of this section. (The frequency response with N=0 is of course greater, but this is the passband.) Furthermore, the aliasing from the second positive-frequency stopband (when it exists, i.e. for $M \geq 4$) is always the next greatest. So we only need to compare the first stopband to the second stopband to see how much worse the first

stopband is.

The ratio of the first-stopband aliasing to the second-stopband aliasing is

$$A_{2,1}(M,\omega) = \frac{\sin(\frac{2\pi}{M} - \frac{\omega}{2M})}{\sin(\frac{\pi}{M} - \frac{\omega}{2M})}$$

For $M = 4$, this reduces to

$$\frac{\sin(\frac{\pi}{2} - \frac{\omega}{8})}{\sin(\frac{\pi}{4} - \frac{\omega}{8})} = \frac{2\cos\frac{\omega}{8}}{\sqrt{2}(\cos\frac{\omega}{8} - \sin\frac{\omega}{8})} = \frac{2}{\sqrt{2}(1 - \tan\frac{\omega}{8})}$$

For small ω, $A_{2,1}(M,\omega) \approx \sqrt{2}$, and specifically $A_{2,1}(M,\omega) \geq \sqrt{2}$. As $M \to \infty$,

$$A_{2,1}(M,\omega) \to \frac{\frac{2\pi}{M} - \frac{\omega}{2M}}{\frac{\pi}{M} - \frac{\omega}{2M}} = \frac{4\pi - \omega}{2\pi - \omega}$$

We see that for small ω, $A_{2,1}(M,\omega) \approx 2$, and specifically $A_{2,1}(M,\omega) \geq 2$.

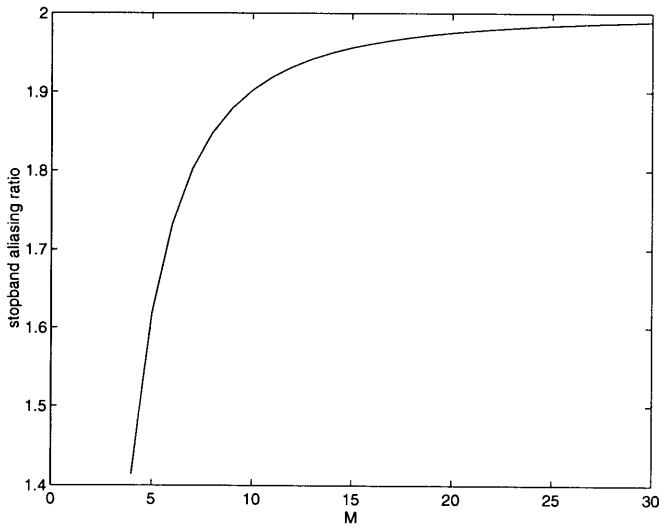

Figure 2-4: $A_{2,1}(M,\omega)$, the ratio of second-stopband aliasing to first-stopband aliasing, as a function of M, in the limit as $\omega \to 0$.

Figure 2-4 shows that $A_{2,1}(M,\omega)$ increases monotonically with increasing M for small ω. It is easy to show that $A_{2,1}(M,\omega)$ increases with $|\omega|$ (for all M and $|\omega| < \pi$). So one stage of a CIC decimator always attenuates other stopbands by at least a factor of $\sqrt{2}$ more than the first stopband. For larger M, other stopbands are attenuated by about a factor of 2 more than the first stopband. For a K-stage CIC decimator,

the other stopbands are attenuated by a factor of at least $(\sqrt{2})^K$ more than the first stopband, and by about 2^K for larger M. Thus the first stopband will often be significantly worse than the other stopbands. This suggests that the performance of a CIC decimator could be enhanced by attenuating the first stopband.

2.2 Modification of Cascaded Integrator Comb Decimator

The previous section showed that the first stopband could be significantly worse than the others. This section presents a way to improve the performance of the first stopband using less hardware than would be required to improve the CIC performance by simply using a higher order CIC decimator. This method requires that the decimation rate M be divisible by 2.

Suppose then that M is divisible by two. Then

$$R'_M(\Omega) = \frac{\sin(\frac{1}{2}M\Omega)}{M\sin(\frac{1}{2}\Omega)}e^{-j\Omega(\frac{M-1}{2})} = \frac{\sin(\frac{1}{2}\frac{M}{2}\Omega)}{\frac{M}{2}\sin(\frac{1}{2}\Omega)}e^{-j\Omega(\frac{M}{4}-\frac{1}{2})} \cdot \frac{\sin(\frac{1}{2}M\Omega)}{2\sin(\frac{1}{2}\frac{M}{2}\Omega)}e^{-j\Omega(\frac{M}{4})}$$

$$R'_M(\Omega) = R'_{\frac{M}{2}}(\Omega) \cdot R'_2(\frac{M}{2}\Omega)$$

or, in the time domain,

$$r'_M[n] = \frac{1}{M}\sum_{k=0}^{M-1}\delta[n-k] = (\frac{2}{M}\sum_{k=0}^{\frac{M}{2}-1}\delta[n-k]) * r'_2[\frac{2n}{M}]$$

$$r'_M[n] = r'_{\frac{M}{2}}[n] * r'_2[\frac{2n}{M}]$$

where $r'_2[\frac{2n}{M}]$ is sloppy notation for

$$r'_2[\frac{2n}{M}] = \frac{1}{2}(\delta[n] + \delta[n-\frac{M}{2}])$$

to emphasize that $r'_2[\frac{2n}{M}]$ is a 2-long rectangular filter with $\frac{M}{2}-1$ zeros inserted between the taps. Figures 2-6 and 2-5 illustrate this in the frequency domain and the time

domain, respectively, for $M = 6$. Notice that $R'_2(\frac{M}{2}\Omega)$ puts zeros at odd-numbered stopbands, while $R'_{\frac{M}{2}}(\Omega)$ puts zeros at even-numbered stopbands. Therefore, $R'_2(\frac{M}{2}\Omega)$ is responsible for almost all of the attenuation in the first stopband. Since we wish to attenuate the first stopband, we will look at $R'_2(\frac{M}{2}\Omega)$. For notational convenience, I will define $g'_M[n] = r'_2[\frac{2n}{M}]$, $g_M[n] = 2g'_M[n]$, $G'_M(\Omega) = R'_2(\frac{M}{2}\Omega)$, and $G_M(\Omega) = 2G'_M(\Omega)$.

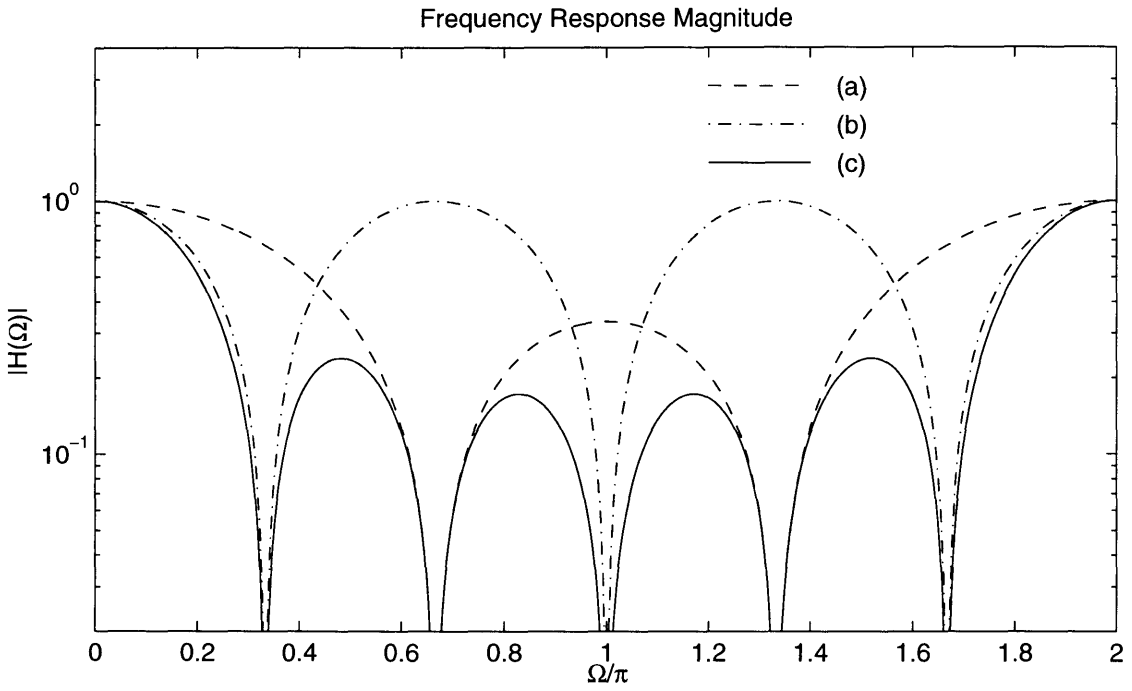

Figure 2-5: Decomposition of a 6-long rectangular filter, Frequency domain: (a) dashed line: $R'_3(\Omega)$, (b) dash-dotted line: $R'_2(3\Omega)$, (c) solid line: $R'_6(\Omega) = R'_3(\Omega) \cdot R'_2(3\Omega)$

Notice in figure 2-6 that the filter $g'_M[n]$ has only two non-zero coefficients, both of which are $\frac{1}{2}$. Therefore only one adder is required to implement this filter. Furthermore, because the nonzero coefficients are $\frac{M}{2}$ samples apart, we may decimate by $\frac{M}{2}$ ahead of the filter. Therefore only one register is needed to implement this filter, which becomes a 2-long rectangular filter.

Recall that each CIC stage requires two adders and two registers. Therefore, $g_M[n]$ provides almost as much first-stopband attenuation as one extra CIC stage (i.e. one integrator stage and one comb stage), while using about half as much hardware. This

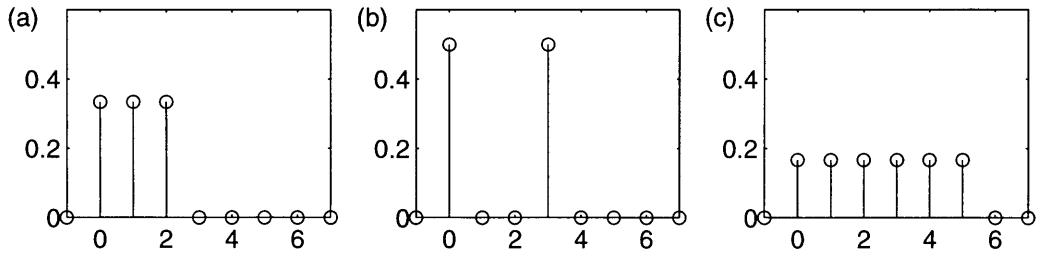

Figure 2-6: Decomposition of a 6-long rectangular filter, Time domain. (a) $r_3'[n]$, (b) $r_2'[\frac{n}{3}]$, (c) $r_6'[n] = r_3'[n] * r_2'[\frac{n}{3}]$.

suggests a design procedure for improving CIC decimators. The improved decimator should use the minimum number of CIC stages, K, to attenuate the second stopband to specifications. This CIC filter should then be cascaded with the minimum number of intermediate $G_M(\Omega)$ stages, L, to attenuate the first stopband to specifications. The improved decimator is shown in figure 2-7. Figure 2-8 compares an improved CIC decimator with $K = 2$ CIC stages and $L = 1$ intermediate stage to CIC decimators with $K = 2$ stages and $K = 3$ stages.

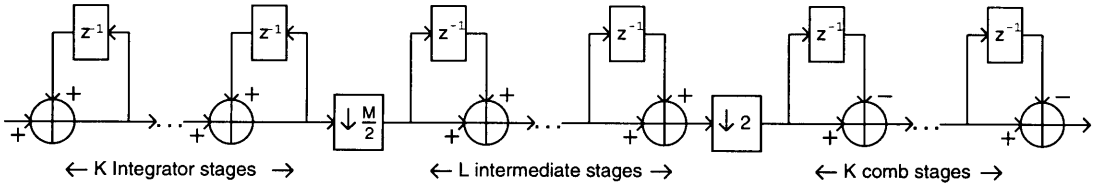

Figure 2-7: Structure of an improved CIC decimator with K CIC stages and L intermediate stages. Decimation factor M.

Notice that this is equivalent to a multistage decimation technique. If we use K cascaded $R_{\frac{M}{2}}(\Omega)$ filters, decimate by $\frac{M}{2}$, use $K + L$ $G_M(\Omega)$ filters, and decimate by 2, then this decimator filter will have the same frequency response as our improved CIC decimator. We used more filtering in the final decimation stage than in the others because the final decimation stage always has the tightest requirements for filters [3]. However, because of the CIC filter structure, we were able to combine K $R_{\frac{M}{2}}(\Omega)$ filters and K $G_M(\Omega)$ filters into K $R_M(\Omega)$ filters when implementing them. These observations suggest that we decimate using more than 2 stages, splitting

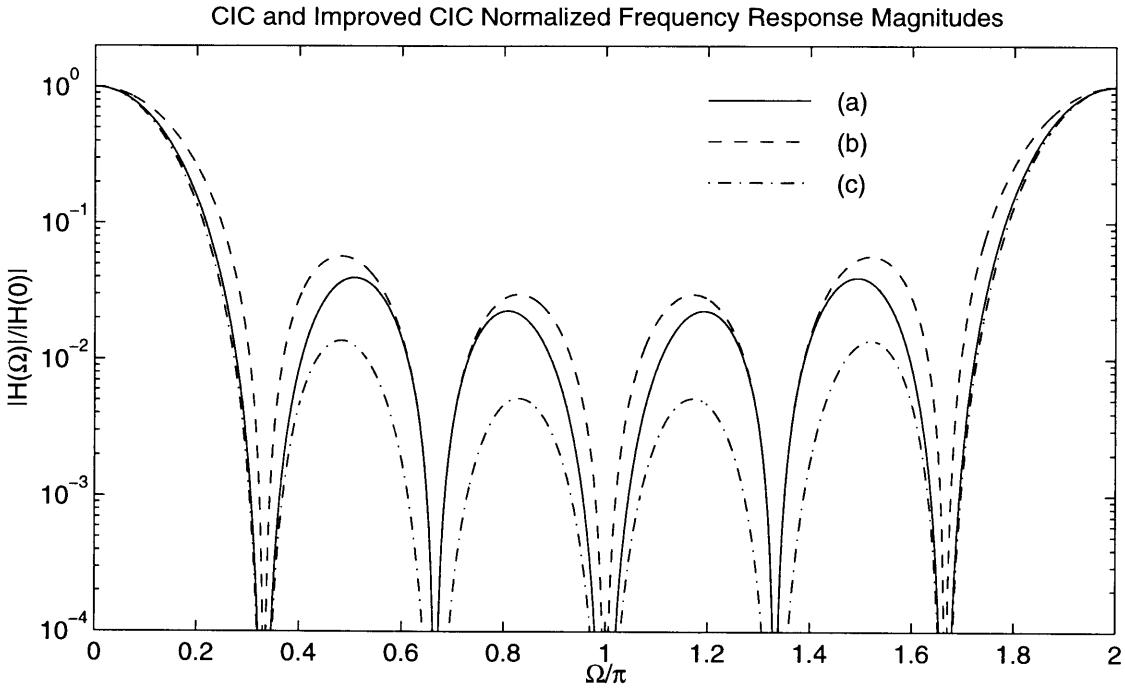

Figure 2-8: Comparison of typical CIC decimators with a typical improved-CIC decimator. In all cases, decimation factor $M = 6$, and frequency responses are normalized to 1 at $\Omega = 0$. (a) Solid line: An improved CIC decimator with $K = 2$ CIC stages and $L = 1$ intermediate stage. (b) Dashed line: A CIC decimator with $K = 2$ stages. (c) Dash-dotted line: A CIC decimator with $K = 3$ stages.

$R_{\frac{M}{2}}(\Omega)$ into $R_{\frac{M}{4}}(\Omega)$ and $G_{\frac{M}{2}}(\Omega)$. We would then use K cascaded $R_{\frac{M}{4}}(\Omega)$ stages, $K + J$ $G_{\frac{M}{2}}(\Omega)$ stages, and $K + J + L$ $G_M(\Omega)$ stages. However, J stages of $G_{\frac{M}{2}}(\Omega)$ and $G_M(\Omega)$ require 2 adders each, while $R_M(\Omega)$ implements $G_{\frac{M}{2}}(\Omega)$ and $G_M(\Omega)$ as well as $R_{\frac{M}{4}}(\Omega)$ simultaneously using only 2 adders. So it is best to use only 2-stage decimation.

Another idea would be to use a more complex filter in place of L stages of $G_M(\Omega)$. However, because of the simplicity of $R_M(\Omega)$ that implements the other stages, the second stopband will not be attenuated sufficiently to require the use of a more complex filter. Furthermore, because the filter would be implemented between the integrator and comb stages, complex filters would cause register growth problems.

2.3 Our Specific Decimator

The practical limit for the speed of the IIR filters in Altera FLEX8000 hardware was found to be near 45 MHz. Therefore all signals need to be decimated below 45 MHz by the on-chip decimator. We will see in chapter 5 (see table 5.1) that the highest-frequency passband cutoff will be the 5 MHz lowpass filter for the 622.08 MHz data rate. Decimating by 16 gives an output frequency of 38.88 MHz, which is acceptable for the Altera hardware. This leaves the 5 MHz lowpass filter cutoff at $\omega = 2\pi\frac{5MHz}{38.88MHz} = .257 \cdot \pi$. Therefore, the decimator must be able to decimate by 16, while keeping aliasing 60 dB below the passband for a stopband width of $.257 \cdot \pi$.

Figure 2-9 demonstrates the performance of three candidate decimators: A CIC decimator with $K = 4$, a CIC decimator with $K = 3$, and an improved CIC decimator with $K = 3$, $L = 1$. Each has decimation factor $M = 16$. On the left is a frequency response plot for each decimator showing detail of the first several stopbands. The frequency response is plotted versus Ω, the radian frequency with respect to the high input data rate. The passband is represented by a solid line, while the first and second stopbands are highlighted with dash-dotted and dashed lines, respectively. Other stopbands are shown by dotted lines.

On the right is a stopband aliasing plot for each decimator, depicting the proportion of aliased response to frequencies in the stopbands to the response of the passband. The output of the decimator will be passed through an FIR filter which will correct stopband attenuation thus amplifying signals in the stopband aliased to the same frequency. Therefore, it is insufficient simply to guarantee that the decimator frequency response attenuates the stopband by 60 dB. Instead, we must divide the stopband response at the decimated radian frequency ω by the passband response at that frequency. So in these plots, each stopband aliasing response is normalized by dividing by the passband response. Again, the first and second stopbands are highlighted with dash-dotted and dashed lines, respectively. Note that there are two lines for each stopband in the plot. The greater one corresponds to the response of the left side of the positive stopband, and therefore, by symmetry, the response of

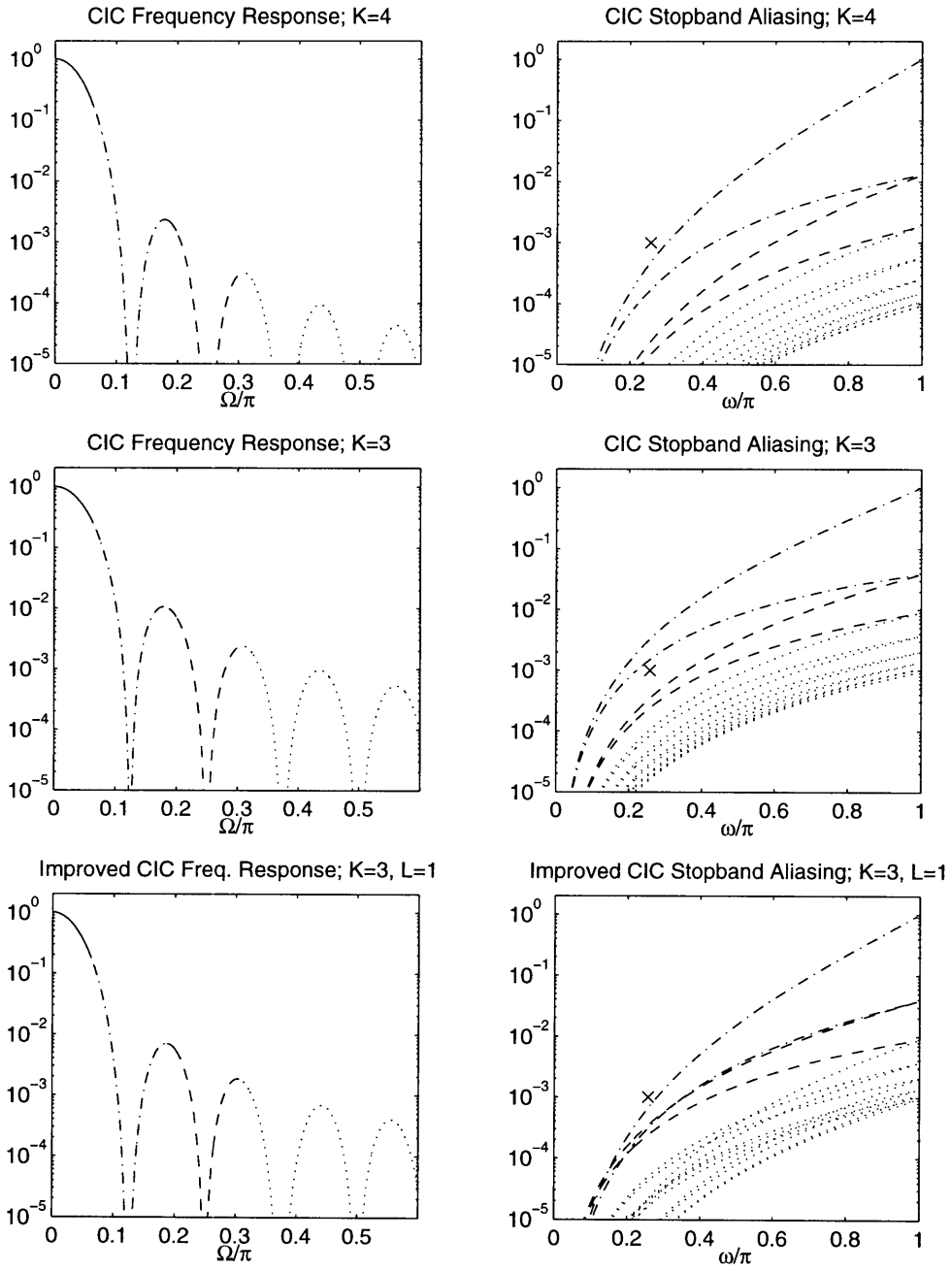

Figure 2-9: Frequency response (detail of first few stopbands), versus input referenced frequency Ω; and normalized stopband aliasing, versus output referenced frequency ω, of three candidate decimators. First stopbands are highlighted with dash-dotted lines, and the second stopbands are highlighted with dashed lines. Other stopbands are plotted with dotted lines, and the passband is plotted with a solid line. Notice in the stopband aliasing plots that the stopbands have been normalized by dividing by the passband response, to account for later filters that correct passband attenuation.

the right side of the corresponding negative stopband. The lesser line for a given stopband corresponds to the right side of the positive stopband and the left side of the corresponding negative stopband.

For convenience in analyzing the decimators, I have placed an "×" to mark the specified limit: that the stopbands should be attenuated 60 dB below the stopband at $\omega = .257 \cdot \pi$. The CIC decimator meets this specification with $K = 4$, but not with $K = 3$. However, note that the second stopband does meet the specification with $K = 3$. So we try using an improved CIC decimator with $K = 3$, $L = 1$. This does meet the specification. We will choose the improved CIC decimator with $K = 3$, $L = 1$, since it requires less hardware.

To insure that other filters don't require more Altera FLEX8000 hardware than is necessary to implement the 622.08 MHz data rate, we will want to use no more FIR filter taps for the other filters than we use for the 622.08 MHz data rate. The number of filter taps required tends to increase with decreasing transition band width [3]. The transition band width is proportional to M:

$$\Delta\omega = \frac{\Delta f}{f_r} \cdot 2\pi M$$

where $\Delta\omega$ is the decimated discrete-time transition band width, Δf is the continuous-time transition band width resulting from the third-order lowpass filter, and f_r is the data rate before decimation. Therefore, too small a value for M may increase the number of filter taps required.

However, ω_c must also remain low enough so that no aliasing occurs, and ω_c is also proportional to M.

$$\omega_c = \frac{f_c}{f_r} \cdot 2\pi M$$

Therefore, too large a value for M will result in excessive aliasing in the decimator.

Thus we have a tradeoff for M. We must keep M large enough that ω_c is not much less than $.257 \cdot \pi$ for other data rates, so that FIR filters with a similar number of filter taps may be implemented for other data rates. But we must keep M small enough that an improved CIC decimator with $K = 3$ and $L = 1$ will be adequate.

We expect the decimator to have similar stopband attenuation at other decimation rates, so if ω_c is not much greater than $.257 \cdot \pi$, the decimator should be adequate. Table 2.1 shows decimation rates that were found to work well. All ω_c are within the passband, so a stronger decimator is not required, and ω_c is high enough such that no more FIR filter taps are required. Notice that the decimation rates were chosen primarily to place ω_c in the right place. This often resulted in decimated data rates much lower than the practical limit of about 45MHz for the Altera hardware.

Input Data Rate (MHz)	Lowpass Cutoff f_c (kHz)	Decimation Rate M	Output Data Rate (MHz)	Passband Width (π)	Lowpass Cutoff ω_c (π)
1.544	N/A	1	1.544	N/A	N/A
2.048	200	1	2.048	N/A	.195
6.312	N/A	1	6.312	N/A	N/A
8.448	400	2	4.224	.224	.189
34.368	800	4	8.592	.254	.186
44.736	N/A	1	44.736	N/A	N/A
51.84	400	12	4.32	.281	.185
139.264	3500	4	34.816	.254	.201
155.52	1300	12	12.96	.281	.201
622.08	5000	16	38.88	.281	.257

Table 2.1: Chosen decimation rates.

2.4 Signal "Wrapping"

The input signal $x[n]$ which we want to decimate and filter is a type of random walk process. That is, as n grows, $x[n]$ will drift far from $x[0]$. Thus $x[n]$ is unbounded, and may not be represented directly as a twos-complement integer using a finite number of bits. Instead, only the least significant B_x bits of $x[n]$ are available. We will refer to this as $x_{lsp}[n]$. However, it will be shown that if

$$|x[n] - x[n-1]| < 2^{B_x - 1} \tag{2.1}$$

$x[n]$ may be reconstructed unambiguously from $x_{lsp}[n]$, except for a constant offset k. This offset is unimportant for the final output. Intuitively, what is happening is that $x_{lsp}[n]$ "wraps around" from positive to negative values whenever $x[n]$ grows by 2^{B_x-1}, or from negative to positive values whenever $x[n]$ decreases by 2^{B_x-1}. As long as $|x[n] - x[n-1]| < 2^{B_x-1}$, we can determine whether it "wrapped" upwards or downwards.

Let $x'[n] = x[n] - x[n-1]$. Then we have

$$x[n] = x[0] + \sum_{k=1}^{n} x'[k]$$

so we may reconstruct $x[n]$ from $x'[n]$ (to within a constant offset) using an accumulator. Furthermore, since $x[n]$ and therefore $x'[n]$ are integers, $x'[n]$ may be represented in twos-complement form using B_x bits, if $|x'[n]| < 2^{B_x-1}$. Under twos-complement addition and subtraction, only the least significant B_x bits of x[n] are needed to calculate the least significant B_x bits of $x'[n]$. Therefore $x_{lsp}[n] - x_{lsp}[n-1] = x'[n] = x[n] - x[n-1]$. Thus we may reconstruct $x[n]$ from $x_{lsp}[n]$ (to within a constant offset) by taking the first difference and accumulating the result; i.e.

$$x[n] = x[0] + \sum_{k=1}^{n} (x_{lsp}[k] - x_{lsp}[k-1])$$

Therefore if 2.1 holds, $x_{lsp}[n]$ specifies $x[n]$ to within a constant offset $x[0]$.

Now suppose we want to decimate $x[n]$ by M. Let $y_f[n]$ be the filtered signal $x[n] * h[n]$ with $h[n]$ being an FIR filter with integer coefficients. Then $y[n] = y_f[Mn]$ is the filtered and decimated signal. However, $x[n]$, $y_f[n]$, and $y[n]$ may not be represented using a finite number of bits. So we wish to filter and decimate $x_{lsp}[n]$ in such a way that yields $y_{lsp}[n]$, the least significant B_y bits of $y[n]$. As above, this may be used to reconstruct $y[n]$. Thus we must find B_y such that $|y[n] - y[n-1]| < 2^{B_y-1}$. We have that

$$|y[n] - y[n-1]| = |\sum_{k=-\infty}^{\infty} (x[k] - x[k-M])h[n-k]|$$

So

$$|y[n] - y[n-1]| \leq \sum_{k=-\infty}^{\infty} |(x[k] - x[k-M])| \cdot |h[n-k]|$$

with equality when the sign of $x[k] - x[k-M]$ is the same as the sign of $h[n-k]$. From 2.1 we have

$$|y[n] - y[n-1]| < M \cdot 2^{B_x - 1} \sum_{k=-\infty}^{\infty} |h[k]|$$

Therefore,

$$B_y = \lceil B_x + \log_2(M) + \log_2(\sum_{k=-\infty}^{\infty} |h[k]|) \rceil \tag{2.2}$$

The first term is the number of input bits, and the third term is due to the filter gain. The second term is required for resolving "wraps" when we are decimating a signal, since the input signal might "wrap" more than once.

This tells us that the following method may be used for generating $y_{lsp}[n]$. First, generate $x[n]$ from $x_{lsp}[n]$ by taking the first difference and accumulating the result. Then, conceptually filter and decimate $x[n]$ with ideal hardware capable of handling arbitrarily large values of $x[n]$ to produce $y[n]$. Finally, discard all but the least significant B_y bits of $y[n]$, yielding $y_{lsp}[n]$. Also discard all hardware used to calculate discarded bits of $y[n]$. Since our decimator uses only adders and registers, only the least significant B_y bits of the adders and registers are necessary to calculate $y_{lsp}[n]$. Therefore, the total filter we need to implement consists of a B_x-bit first-differencer, a B_y-bit accumulator, and an improved-CIC decimator with B_y-bit adders and registers.

Now we must determine B_y in terms of B_x, K, L, and M. The Z-transform of $r_M[n]$ is $R_M(z) = \sum_{n=0}^{M-1} z^{-n}$, while the Z-transform of $g_M[n]$ is $G_M(z) = 1 + z^{-\frac{M}{2}}$. Therefore the Z-transform of the whole filter is

$$H(z) = (R_M(z))^K (G_M(z))^L = (\sum_{n=0}^{M-1} z^{-n})^K (1 + z^{-\frac{M}{2}})^L$$

Then we have that

$$\sum_{k=-\infty}^{\infty} h[k] = H(z)|_{z=1} = M^K \cdot 2^L \tag{2.3}$$

Since both $r_M[n]$ and $g_M[n]$ are nonnegative, $h[n]$ is nonnegative and $h[n] = |h[n]|$ for

all n. Plugging into equation 2.2 gives

$$B_y = \lceil B_x + (K+1)\log_2(M) + L \rceil$$

For our decimator, $K = 3$ and $L = 1$, so

$$B_y = \lceil B_x + 4\log_2(M) + 1 \rceil$$

Table 2.2 gives B_x and B_y for each data rate. Note that the highest B_y is 27 bits, for the 51.84 MHz data rate. Therefore the decimator must use a 27-bit high data path.

Input rate (MHz)	B_x	M	B_y	V_{MSB}
1.544	16	1	16	1
2.048	16	1	16	1
6.312	14	1	14	1
8.448	14	2	19	2 (8)
34.368	12	4	21	4
44.736	11	1	11	1
51.84	11	12	27	18.963
139.264	10	4	19	4 (16)
155.52	10	12	26	18.963
622.08	8	16	25	16

Table 2.2: Minimum B_y required by each data rate. Because of implementation details to be explained in section 3.5, values for V_{MSB} in parentheses, where given, should be used instead of the ones given by the formula.

The gain of our filter, from equation 2.3, is $M^3 \cdot 2$. We have also added $B_y - B_x = \lceil 4\log_2(M) + 1 \rceil$ bits above the MSB of the input. Therefore, calling the value of the MSB of the decimator input to be 1, the value of the MSB of the decimator output is

$$V_{MSB} = \frac{2^{B_y - B_x}}{M^3 \cdot 2} = \frac{2^{\lceil 4\log_2(M) \rceil}}{M^3}$$

This factor is listed in table 2.2. For the 8.448 MHz and 139.264 MHz data rates,

implementation details to be described in section 3.5 will require a different value for V_{MSB} than given by the above formula. These values are placed in parentheses in table 2.2. The values in parentheses should be used.

Chapter 3

Decimator Implementation

As noted previously, the digital signal to be filtered has a very high sample rate, which may be up to 622.08 MHz. This is too fast for most digital technologies. However, the analog system producing the signal is designed for a very high-speed fabrication process, which allows high-speed digital circuitry also. Therefore the decimator may be placed on the same chip as the analog system.

To lower switching noise, the decimator will use differential current-mode logic cells designed for use on primarily analog chips. Unlike common CMOS or TTL logic families, differential current-mode logic does not support AND or OR gates with high numbers of inputs particularly well. Instead, the best way to estimate complexity is to count the number of logic "trees" used. One differential current-mode tree may implement a 2:1, 3:1, or 4:1 multiplexor. A 2:1 multiplexor may be wired in a feedback configuration to form a latch, and two of these may be cascaded to form a flip-flop. Standard techniques use 3:1 or 4:1 multiplexors to form flip-flops which have features such as a "clear" or "enable" input, or multiple inputs, while still using only two trees. Alternately, one tree may implement any single-output 3-input logic function. Since a one-bit full adder has 3 inputs and 2 outputs, a full adder may be implemented with two trees—one to generate the "sum" output and one to generate the "carry" output. Subtracters also require two trees per bit. Logic inverters require no hardware in differential logic, since switching the differential signals inverts the logic signal.

3.1 Chip Boundary

We see from figure 2-7 that the comb stages operate at the lower sample rate, which is the rate the Altera FLEX8000 PLDs will use to implement the filters. Therefore the comb stages may also be implemented on the PLDs. This will reduce the amount of expensive on-chip area consumed by the decimator, and reduce the amount of power the chip will have to dissipate. Note also that the intermediate stage will only need to perform one addition for each clock cycle at the low sample rate, since the decimate-by-two will ignore every second output of the intermediate stage. However, implementing the intermediate stage on the PLDs would require the output pins to operate at a faster rate, consuming more power. The output pin drivers consume much more power than a small amount of logic. Therefore, the intermediate stage will be implemented on the analog chip, even though the calculations could be done at a slower rate. The three integrator stages and the unwrapping bit extender will also be implemented on the analog chip. Figure 3-1 shows how the decimator will be partitioned between the analog chip and the PLDs. The portion of the decimator to be implemented on the analog chip will be covered in this chapter, and the remainder of the decimator will be implemented in section 5.6.

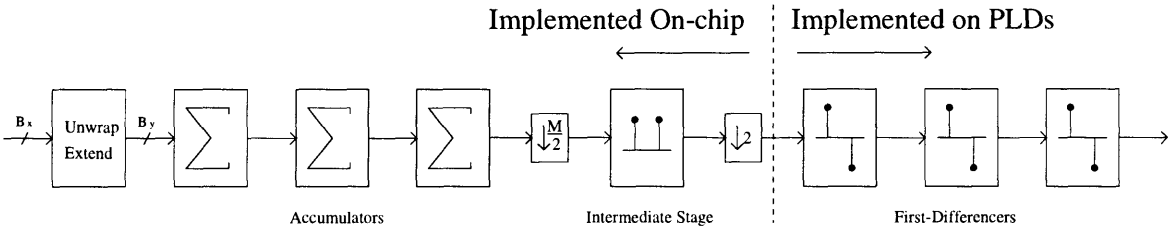

Figure 3-1: Partition of the decimator between the analog chip and PLDs.

3.2 Unwrapping Bit Extender

We determined in section 2.4 that "unwrapping" the signal could be done by taking the first difference, and bit extending and accumulating the result, truncating at the

higher number of bits. We also determined that we need to unwrap the input to a new bit width B_y of 27 bits. However, recall from table 2.2 that different input data rates which need to be decimated have different values for B_x, ranging from 8 to 14. (Two data rates have $B_x = 16$, but these data rates do not need to be decimated, and therefore they do not need to be unwrapped.) Therefore the first differencer must be 14 bits high, to allow for all possible B_x. However, those bits which are not used will contain incorrect values. Therefore, we need to put a 2:1 multiplexor on input bits 8 though 13, to select whether the output of the first-differencer or the bit-extend bit will be used as the input to the accumulator. The 6 bit signal comprising the select inputs to the multiplexors will be called "UNWRAP EXTEND SELECT." It selects which bits should be taken from the input and which ones should be calculated by the unwrapping method, using a thermometer code. The resulting logic diagram is shown in figure 3-2.

However, several trees may be saved over the diagram in figure 3-2. Notice that the least significant B_x bits of the output of the accumulator are the same as the inputs to the first differencer, since the output is just a bit-unwrapped version of the input. Furthermore, we will show in appendix A that the B_x^{th} carry bit of the accumulator is the same as the inverted B_x^{th} borrow bit of the first-differencer. Therefore the trees which calculate the first B_x sum and carry bits of the accumulator are not needed, and neither are the accumulator register trees for the least significant B_x bits. The first $B_x - 1$ output bits of the first-differencer are not needed either, since they are only used to calculate the least significant part of the accumulator, which we are not using, and the carry signal into the most significant part, which is the same as the inverted borrow bit. (The B_x^{th} output bit of the first-differencer is used to sign-extend the first-difference.) Since B_x varies, the most efficient way to take advantage of this fact is to simply eliminate the sum, carry, and borrow trees we do not need when $B_x = 8$, the minimum.

To assure that the output of the decimator starts at the correct value, the accumulator register and first-differencer register should be equipped with a "clear" signal, which should be used to clear the registers when the chip is powered up or when the

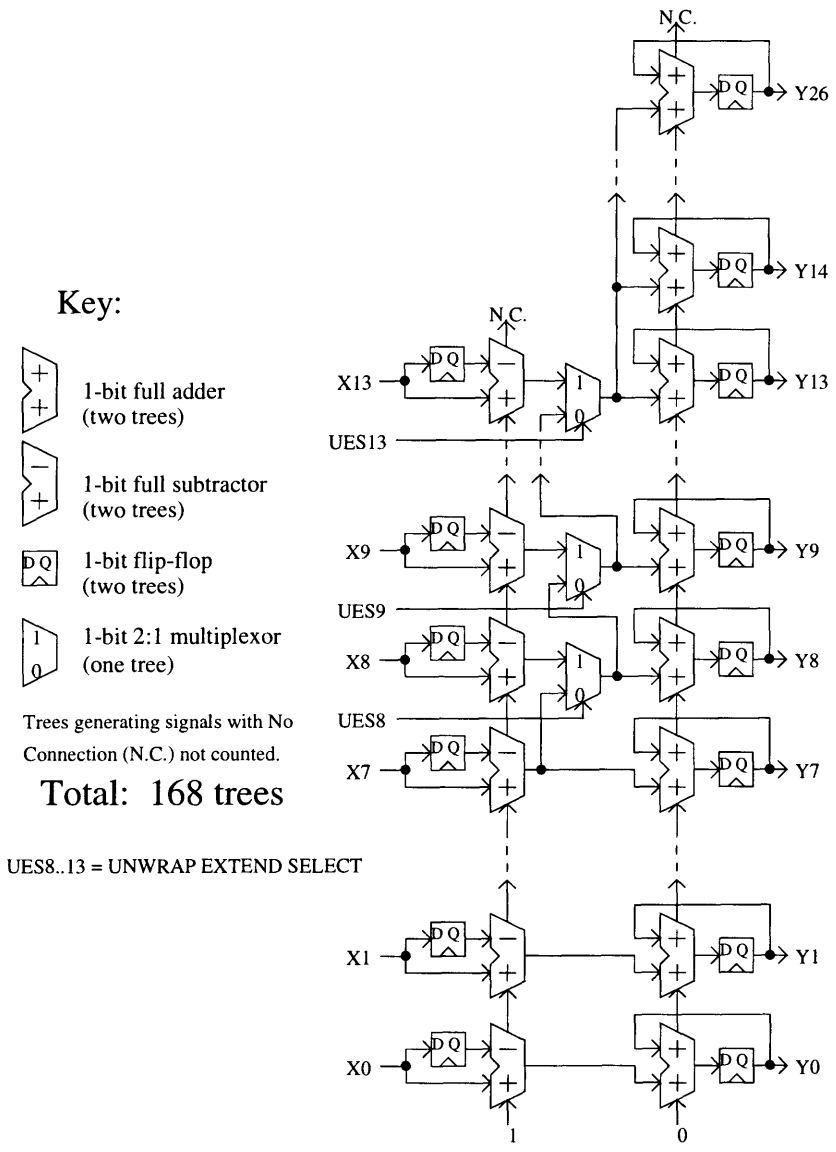

Figure 3-2: Straightforward unwrap extender implementation. X0..X13 are data input signals, UES8..UES13 are control signals, and Y0..Y26 are data output signals.

chip changes from one sample rate to another. This signal will be called "REGISTER CLEAR." Without this signal, all output values would be offset by the initial value in the accumulator register minus the value in the most significant 6 bits of the initial value in the first-differencer register.

To pipeline the decimator, the output of the unwrapping bit extender will be taken from the accumulator registers, rather than directly from the sum outputs of the accumulator adders. Therefore we do not need extra registers to pipeline this stage. The bottom 8 bits of the unwrapping bit extender do not have accumulator hardware; instead the output is the same as the input. Therefore, the pipelined output will be taken from the register already used for the first-differencer. Figure 3-3 shows the resulting circuit diagram for the first-differencer.

The 14-bit first-differencer uses 48 trees—two for each of 14 flip-flops in the register, one for each of 13 carry signals, and one for each of the 7 most significant bits in the difference, recalling that the 7 least significant difference bits are not needed. The 27-bit accumulator uses 75 trees—two for each of 19 flip-flops in the register, one for each of 18 carry signals, and one for each of 19 sum signals, recalling that no accumulator is used for the 8 least significant bits. 6 trees are used for the 2:1 multiplexors which allow multiple values of B_x. So the unwrapping bit extender uses 129 logic trees.

3.3 Accumulators

Following the unwrapping bit extender, the decimator uses three accumulators, with bit width $B_y = 27$. To pipeline the decimators, the output of the accumulators will be taken from the accumulator registers, rather than directly from the sum outputs of the accumulator adders. Because the decimator contains three first-differencers off-chip at the decimated rate, initial values in these three accumulators will be subtracted out of the output. Therefore, no "clear" signal needs to be placed on these accumulators for normal chip operation. (For testing purposes, a "clear" signal may be desirable for these registers anyway. This would not require any addtional trees.) Each 27-

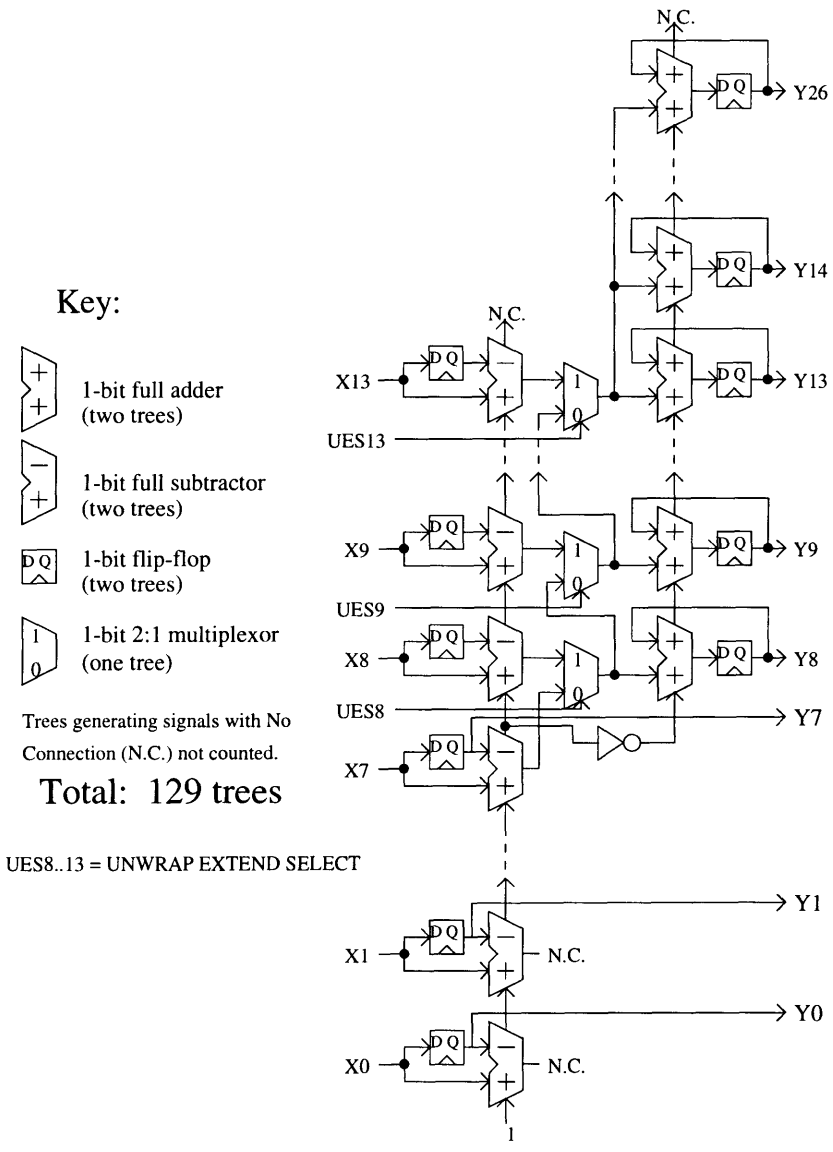

Figure 3-3: Improved unwrap extender implementation. X0..X13 are data input, UES8..UES13 are control signals, and Y0..Y26 are data output.

bit accumulator uses 107 trees—two for each flip-flop in the register and two for each adder bit except the top one, which does not need to generate a carry signal. The three accumulators together use 321 trees. Figure 3-4 illustrates the three accumulators.

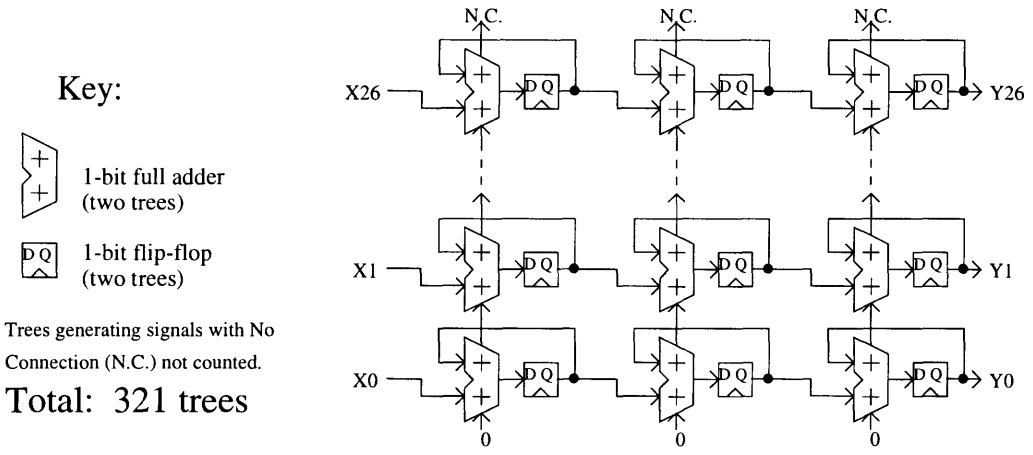

Figure 3-4: Implementation of the three accumulators.

3.4 Intermediate Stage and Decimation

The final on-chip section of the decimator implements the intermediate stage while decimating the signal down to the output frequency. For development in earlier sections, we decimated by $\frac{M}{2}$, implemented a filter with impulse response $\delta[n]+\delta[n-1]$, and decimated by 2. However, this is the same as implementing a filter with impulse response $\delta[n]+\delta[n-\frac{M}{2}]$, and then decimating by M. Also note that since we decimate immediately following the filter, we only need to calculate one output value every M clock cycles in an output cycle. Therefore there is no reason why the intermediate stage should calculate "correct" values for the output of the filter during the remaining clock cycles. For minimum hardware complexity, the remaining clock cycles will calculate meaningless values.

Figure 3-5 shows a circuit for implementing this stage. First, we will use a transparent latch on one input of an adder to record one of the input samples $v[n]$ at time n_0. The output of the intermediate stage for $n_0 \leq n < n_0 + M$ is $v[n_0] + v[n]$ So

when $n = n_0 + \frac{M}{2}$, the output is $v[n_0] + v[n_0 + \frac{M}{2}]$, which is what we want. So during each M clock cycles, the latch should be made transparent during the first half only of clock cycle 0, latching the first needed sample when it is available. The decimating register should be enabled during clock cycle $\frac{M}{2}$, when the second needed sample is available. Figure 3-6 presents the sample timing diagram for these control signals for the case of $M = 12$. All devices are 27 bits wide. The latch uses 27 trees, the adder uses 53 trees, and the output register uses 54 trees, for a total of 134 trees.

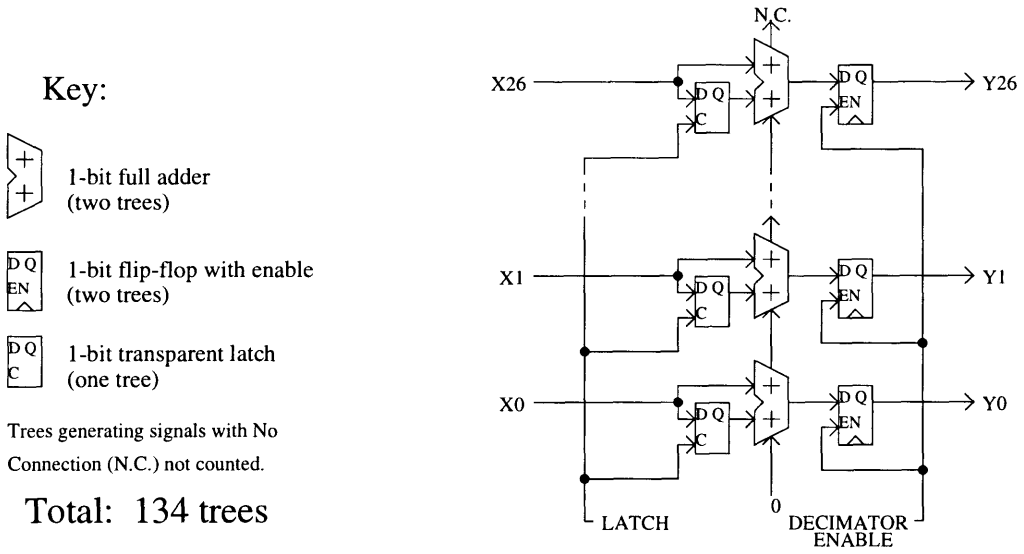

Figure 3-5: Implementation of the intermediate stage and the decimation.

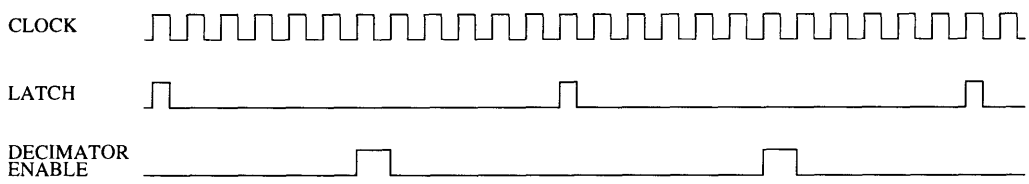

Figure 3-6: Timing diagram for intermediate stage control signals, with $M = 12$. Both signals are shown asserted high, that is, a high signal enables the register or makes the latch transparent.

3.5 Output

Because of the gain of the decimation filter, for high decimation rates M, the output bit width is greater than the signal-to-noise ratio justifies. Therefore we wish to lower the width of the data path by truncating several of the least significant bits. We will model this truncation as adding uncorrelated white noise with mean $\mu = 2^{B_t-1}$ and variance $\sigma^2 = \frac{1}{12}2^{2B_t}$, where B_t is the number of bits truncated. We will allow truncation as long as the standard deviation of the noise added to the final output is no more than half of the standard deviation of the noise at the output due to the roundoff inherent in digitizing the signal we are decimating.

Output pins and output drivers are costly compared to both off-chip and on-chip circuitry. Therefore, we will perform truncation in such a way to minimize the data path width going from the chip to the Altera PLDs. The transfer function from the output pins to the output of the decimator is the convolution of the three comb filters, which is

$$h_{combs}[n] = \delta[n] - 3\delta[n-M] + 3\delta[n-2M] - \delta[n-3M]$$

The variance gain of this transfer function is

$$\sum_{n=-\infty}^{\infty} h_{combs}^2[n] = 20$$

Therefore the noise variance at the output caused by truncation at the output pins is

$$\sigma_{out}^2 = 20 \cdot \frac{1}{12}2^{2B_t}$$

Zero bits are truncated at the input, so the noise variance at the output caused by inherent noise at the input is due to the variance gain of the total transfer function. This gives

$$\sigma_{out}^2 = \frac{1}{12}2^0 \cdot \sum_{n=-\infty}^{\infty} h^2[n]$$

For the standard deviation of the noise added by the truncation to be no more than

half of the standard deviation of the inherent noise at the input,

$$20 \cdot \frac{1}{12} 2^{2B_t} \leq \frac{1}{4} \cdot \frac{1}{12} 2^0 \cdot \sum_{n=-\infty}^{\infty} h^2[n]$$

which gives us

$$B_t \leq \frac{1}{2} \log_2 (\frac{1}{80} \sum_{n=-\infty}^{\infty} h^2[n])$$

The number of output pins required by any given data rate is $B_y - B_t$, the most significant bit required minus the number of LSBs truncated at the output pins. Table 3.1 shows that 21 output pins are needed for the 51.84 MHz data rate.

Input rate (MHz)	B_x	M	B_y	B_t	$B_y - B_t$
1.544	16	1	16	0	16
2.048	16	1	16	0	16
6.312	14	1	14	0	14
8.448	14	2	19	0	19
34.368	12	4	21	2	19
44.736	11	1	11	0	11
51.84	11	12	27	6	21
139.264	10	4	19	2	17
155.52	10	12	26	6	20
622.08	8	16	25	7	18

Table 3.1: Output pins. Note from the 51.84 MHz rate that no more than 21 pins are needed for any data rate.

Also note that the mean of the output caused by truncation at the output pins is

$$\mu_{out} = 2^{B_t - 1} \cdot \sum_{n=-\infty}^{\infty} h_{combs}[n] = 0$$

since the transfer function of the combs puts three zeros at $\Omega = 0$. Therefore, as with CIC decimators [5], using rounding does not have an advantage over truncation.

Since we have 21 output pins, we will use all of them for all of the data rates (unless $B_y < 21$), and some data rates will therefore have less noise due to output pin truncation than we decided to allow. Multiplexors will be required to shift the

output of the intermediate stage such that the most significant bit B_y of the output of the intermediate stage corresponds to the most significant bit of the output pins. Furthermore, since some data rates do not require decimation, the multiplexors will allow the user to bypass the decimator circuitry. Note from table 3.1 that there are 4 different values of B_y which will require that those data rates be multiplexed such that bit B_y matches the MSB of the output pins and as few LSB's as possible are truncated.

The 8.448 MHz and 139.264 MHz data rates have $B_y = 19$, which is less than the number of output pins. For these data rates, we will make the LSB of the decimator output correspond to the LSB of the output pins by multiplexing as if $B_y = 21$. This gives two additional properly unwrapped MSBs at the output. Because of this, V_{MSB} in table 2.2 is raised by a factor of four. This new value is placed in parentheses in table 2.2.

We therefore need a 5:1 multiplexor for each output pin, so that the control circuitry may choose any of the 4 values of $B_y \geq 21$, or the undecimated data when $M = 1$. A 5:1 multiplexor requires 2 trees, and we need one for each output pin, so the multiplexors use 42 trees. Figure 3-7 illustrates the multiplexors. A three-bit control signal, OUTPUT MULTIPLEXOR SELECT, is connected to the select inputs of the multiplexor to select which output bits to use.

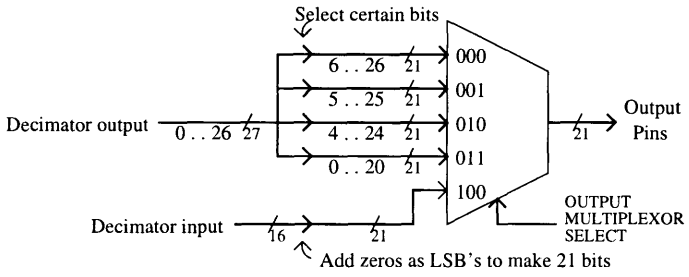

Figure 3-7: Multiplexor circuit.

3.6 Carry-Chain Pipeline Slant

Because of the high speeds required of the decimator, it is necessary to pipeline the carry chains in the decimator into 4 stages. Figure 3-8 illustrates the pipelining of the entire circuit. The first stage calculates the least significant part, bits 0 through 6. The second stage similarly calculates bits 7 through 13, the third stage calculates bits 14 through 20, and the fourth stage calculates the most significant part, bits 21 through 26. A seven-bit register is needed to initiate the pipeline "slant," and another 6 seven-bit registers are needed to realign the bits at the end of the calculation. 15 more one-bit registers are used for pipelining the carry chains. In the unwrapping bit extender, an additional sign-extend signal is required, so 2 more registers pipeline this signal. These 66 registers require 132 trees.

Note that the Intermediate Stage and Decimation block, described in section 3.4, has been partitioned. An output register which is enabled only when a new value is calculated will provide the cleanest output signal. Therefore, the decimation register will be the last part of the decimator before the output. The pipeline "slant" will be re-aligned before reaching the decimation register, and the output multiplexor will also be implemented before the decimation register. Since the output multiplexor reduces the bit width by 6 bits, 12 fewer trees are needed for the decimation register.

For one pipeline stage of the accumulators, the critical signal path begins in the LSB (for that pipeline stage) of the register for the input or the accumulator register, and follows the carry chain through 7 trees (6 for the fourth pipeline stage) to the MSB (for that pipeline stage) of the accumulator register, or to the carry chain register. The critical path of the first, second, or third pipeline stage of the intermediate stage begins in the LSB of the input register, and follows the carry chain through 7 trees to the MSB of the output register, or to the carry chain register. For the fourth pipeline stage, the carry chain passes through 6 trees, and then the sum passes through 2 multiplexor trees, for a total of 8 trees. The first, third, and fourth pipeline stages of the unwrapping bit extender are similar to the accumulators and the intermediate stages, with critical paths passing through 7 or 6 trees in the carry chain. However,

note from figure 3-3 that the longest signal path from input bit 7 to output bit 13 of the unwrapping extender (the second pipeline stage) passes through 9 trees, including the multiplexor, the adder, and the subtracter.

In most cases, the critical signal path passes through 7 trees between registers. But for the fourth pipeline stage of the accumulator and the second pipeline stage of the unwrapping bit extender, the critical path passes through 8 or 9 trees. Fortunately, it is possible to use faster gates for those critical signal paths than for the others. This would be more efficient than adding extra pipeline stages.

3.7 Control

There will be some additional digital circuitry on the chip which is not part of the decimation. It is likely that the control circuitry for the decimator may be smoothly integrated with the other control circuitry, and therefore the control circuitry itself will not be designed. However, this section will describe the control signals collectively.

One control signal, REGISTER CLEAR, is used to initialize the registers in the unwrapping bit extender to zero, so that there is not an offset accumulated into the input during the unwrap-extend. This signal should be asserted for at least one clock cycle when the chip is powered up, or when the data rate mode is changed. Note that the registers used for pipelining the unwrapping-extender should also be equipped with clear inputs, which should be connected to REGISTER CLEAR.

Two control signals, UNWRAP EXTEND SELECT and OUTPUT MULTIPLEXOR SELECT, depend on which data rate mode is chosen, and do not vary with time. UNWRAP EXTEND SELECT is a 6-bit signal which depends on B_x, the input bit width. OUTPUT MULTIPLEXOR SELECT is a 3-bit signal which depends on B_y, the output bit width. Table 3.2 shows the appropriate logic signals for each data rate.

The other control signals are responsible for control of the intermediate stage and decimation. Figure 3-6 showed the proper timing for the intermediate stage without pipelining of the carry chain, with $M = 12$. With pipelining, a separate latch signal is required for each stage. Note that each latch signal is only asserted for the first

Input rate (MHz)	M	B_x	UNWRAP EXTEND SELECT	B_y	OUTPUT MULTIPLEXOR SELECT
1.544	1	16	don't care	16	100
2.048	1	16	don't care	16	100
6.312	1	14	don't care	14	100
8.448	2	14	111111	19	011
34.368	4	12	111100	21	011
44.736	1	11	don't care	11	100
51.84	12	11	111000	27	000
139.264	4	10	110000	19	011
155.52	12	10	110000	26	001
622.08	16	8	000000	25	010

Table 3.2: Control signals which do not vary with time.

half of the clock cycle. The DECIMATOR ENABLE signal should be asserted $\frac{M}{2}$ cycles after the latch signal for the fourth stage is asserted, since no registers were inserted between the fourth stage and the decimation register when the carry chains were pipelined. Figures 3-9, 3-10, 3-11, and 3-12 show the proper timing for $M = 16$, $M = 12$, $M = 4$, and $M = 2$, respectively. When $M = 1$, the decimator is not used and the input is multiplexed around to the decimation register. Therefore, the DECIMATOR ENABLE signal should be permanently asserted, and the latch signals may do anything.

3.8 Conclusion

On the chip, then, we will implement the unwrapping bit extender, the three accumulators, and the intermediate stage, as shown in figure 3-8. The unwrapping bit extender requires 129 trees. The accumulators require 321 trees total. The intermediate stage and decimation require 122 trees, including the savings of 12 trees noted in section 3.6. The output multiplexor used 42 trees. 132 trees more are required to pipeline the carry chains into four stages. Therefore 746 trees are required to implement the on-chip portion of the decimator. This does not include hardware which

will implement the control signals described in section 3.7.

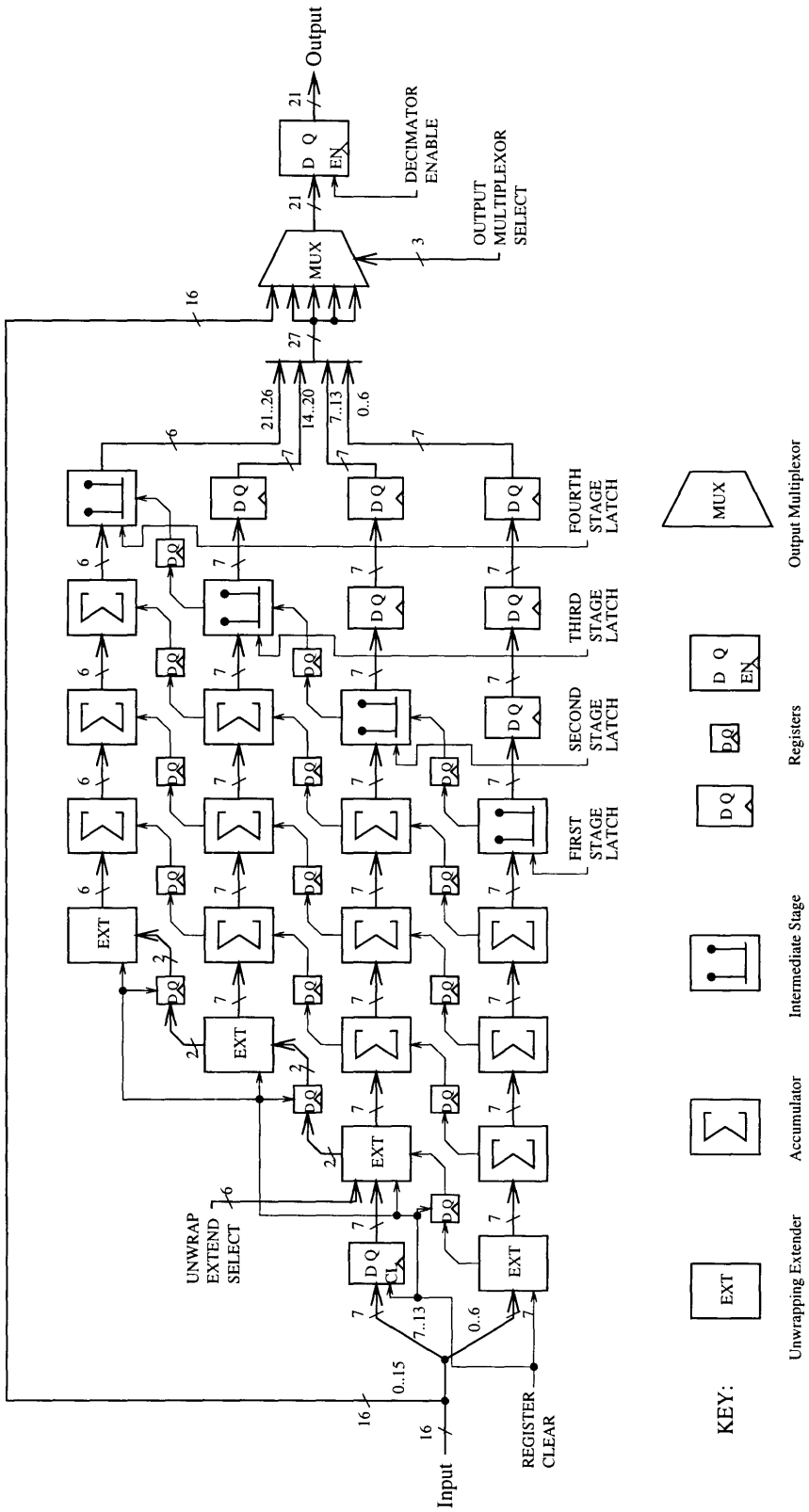

Figure 3-8: Overall circuit.

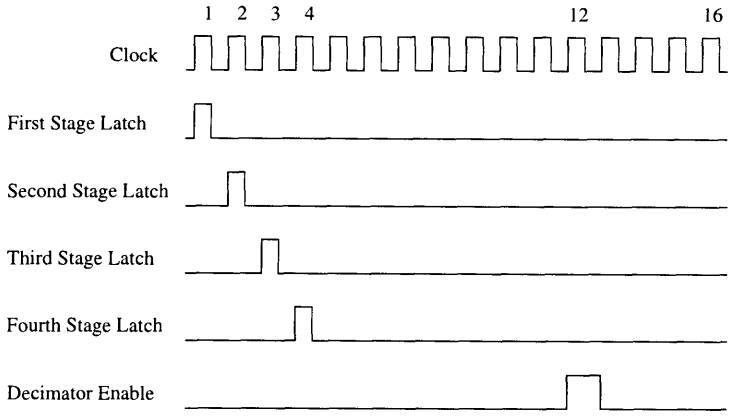

Figure 3-9: Timing diagram for control signals when $M = 16$.

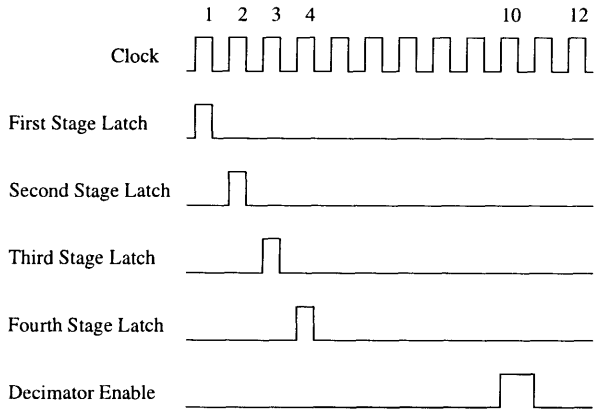

Figure 3-10: Timing diagram for control signals when $M = 12$.

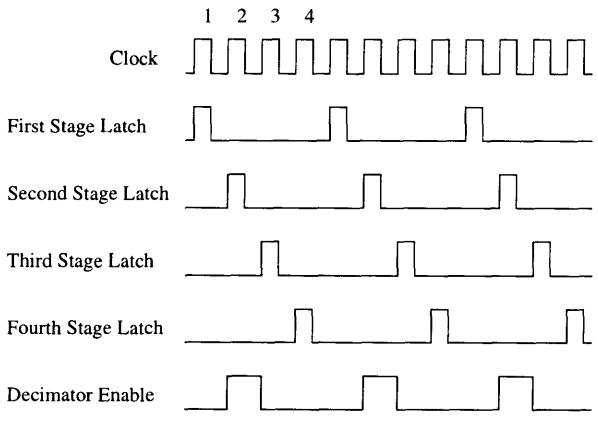

Figure 3-11: Timing diagram for control signals when $M = 4$.

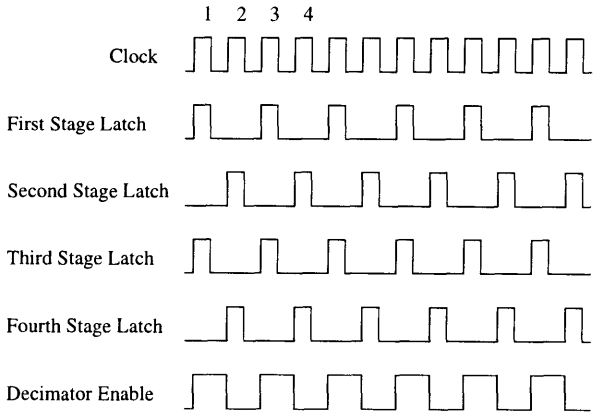

Figure 3-12: Timing diagram for control signals when $M = 2$.

Chapter 4

Recursive Filters

The specifications require a number of lowpass and highpass filters which simulate first-order analog filters. With a few exceptions which will be dealt with separately, these filters have very low cutoff frequencies, relative to the sample rate. This chapter outlines methods for dealing with problems encountered in implementing the filters.

4.1 Structure for IIR Filters

Because of the low cutoff frequencies of most of our filters, the filter structures shown in figure 4-2 offer several advantages over direct form structures such as the ones shown in figure 4-1. The advantages will be mentioned as they are encountered.

Performing an impulse invariance filter transformation on a first-order analog lowpass filter yields a system function of the form

$$H(z) = \frac{1}{1 - \alpha z^{-1}} \tag{4.1}$$

where $\alpha = e^{-2\pi \frac{F_c}{F_s}}$, with analog cutoff frequency F_c and sampling frequency F_s (using F_s as the sampling frequency for the impulse invariance transformation as well). This system function is directly implemented by the direct form lowpass structure in figure

4-1 (a). The lowpass structure in figure 4-2 (a) implements the system function

$$H(z) = \frac{\epsilon z^{-1}}{1 - (1 - \epsilon)z^{-1}} \tag{4.2}$$

which is identical when $\alpha = 1 - \epsilon$, except for a delay term and a multiplicative constant. This delay term does not matter; and a registered output is preferable anyway. The multiplicative constant does not matter either, as the instrument will have to appropriately scale the final output.

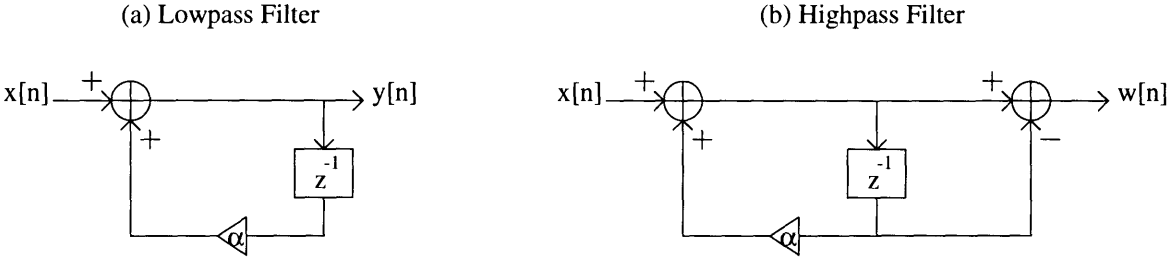

Figure 4-1: Direct Form structures to implement equation 4.1

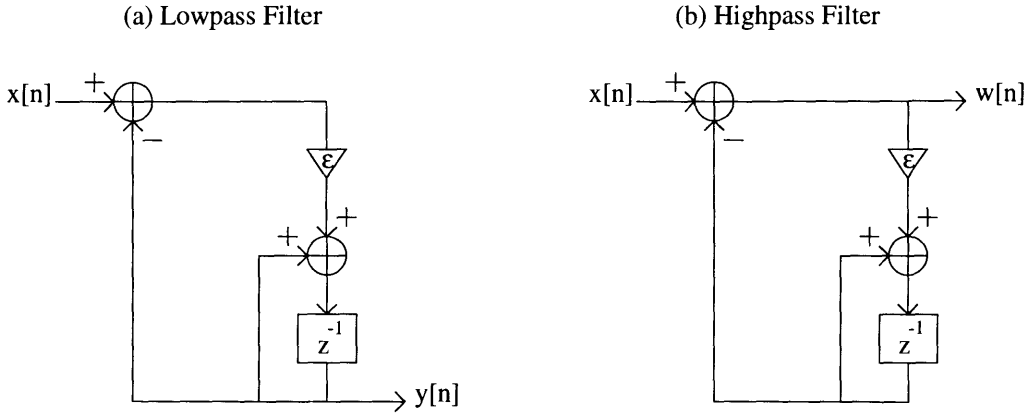

Figure 4-2: Alternate structures to implement equation 4.2

The analogous highpass filter with system function

$$H(z) = \frac{1 - z^{-1}}{1 - \alpha z^{-1}} \tag{4.3}$$

is implemented by the direct form highpass structure in figure 4-1 (b). The highpass

structure in figure 4-2 (b) implements the same system function when $\alpha = 1-\epsilon$. Notice that the highpass structure in figure 4-2 (b) is identical to the lowpass structure, except that a different node is defined as the output. Thus properties developed for this structure apply to both highpass and lowpass forms.

Note that $H(z)|_{z=1} = 1$ for our lowpass structure in figure 4-2(a), so our lowpass filters will have a D.C. gain of 1. However, $H(z)|_{z=-1} = C_{iir} = \frac{2}{2-\epsilon} \approx 1$ for our highpass structure in figure 4-2(b), so we will need to scale the output of our filters appropriately if ϵ is not small enough.

4.2 Pipelining Recursive Loops

For the higher sample rates, we will see in section 5.3 that the propagation delay along the large loop is greater than the sampling interval. Because of this, we will have to insert delay elements into the large loop. The accumulator loop, however, may be pipelined without inserting delays into the accumulator loop.

We will use a technique similar to standard look-ahead computation [6] to compensate for the delay in our structure. However, when the cutoff frequency ω_c is very small, the effect of delay in the large loop is, surprisingly, negligible. To provide intuitive justification for this, suppose that delay is introduced into the ϵ coefficient multiplier in figure 4-3. Because $\epsilon \approx \omega_c$ is very small, $y[n]$ changes very little from sample to sample. Therefore, inserting delay should not change $y[n]$ much. Similarly, $w[n] = x[n] - y[n]$ should not change much either. We will find that our intuition holds, and the frequency response is not greatly affected when delay is inserted into the large loop.

The difference equation for the lowpass output is

$$y[n] = \epsilon x[n-1] + (1-\epsilon)y[n-1]$$

If we rewrite this as

$$y[n] = \epsilon x[n-1] + y[n-1] - \epsilon y[n-1] \tag{4.4}$$

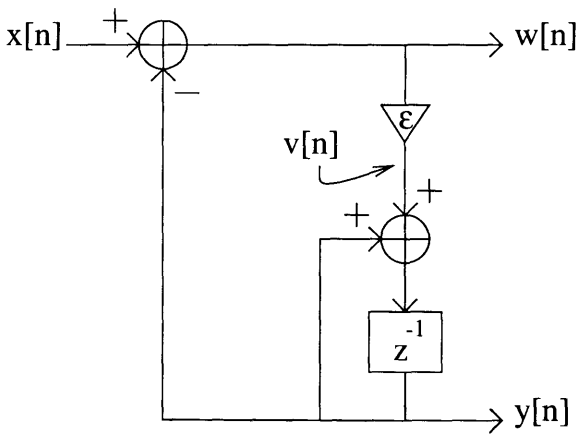

Figure 4-3: IIR filter structure, without extra delays in the large loop.

then we have separated the input term, $\epsilon x[n-1]$, the accumulator loop term, $y[n-1]$, and the large loop term, $-\epsilon y[n-1]$. The large loop contains the input adder and the coefficient multiplier, so we may have to insert delay elements to allow time for propagation delays.

Substituting 4.4 into the large loop term of the same equation (but not into the accumulator loop term) yields

$$y[n] = \epsilon x[n-1] - \epsilon^2 x[n-2] + y[n-1] - \epsilon(1-\epsilon)y[n-2] \tag{4.5}$$

Now, the large loop term $\epsilon(1-\epsilon)y[n-2]$ has an extra delay element in it. Furthermore, the input terms may be expressed as a convolution,

$$\epsilon x[n-1] - \epsilon^2 x[n-2] = \epsilon(\delta[n] - \epsilon\delta[n-1]) * x[n-1]$$

Defining $\epsilon' = \epsilon(1-\epsilon)$ and $h_{fix}[n] = \frac{\epsilon}{\epsilon'}(\delta[n] - \epsilon\delta[n-1])$ gives us

$$y[n] = h_{fix}[n] * \epsilon' x[n-1] + y[n-1] - \epsilon' y[n-2]$$

This difference equation may be implemented as shown in figure 4-4, with k=1, except for an additional latency of 1. The large loop now has an extra delay element in it, so that we can divide the large loop propagation delay into two parts. The accumulator

loop still has one delay. The difference is that the coefficient has changed from ϵ to ϵ', and we filter the input with an FIR filter $h_{fix}[n]$ to correct the effects of the extra delays. But the output of the filter is the same, except for the additional latency of 1.

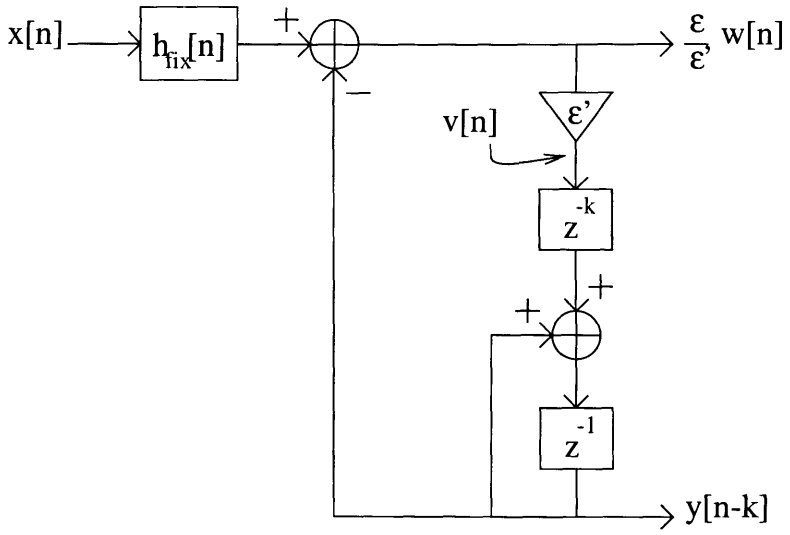

Figure 4-4: Modified IIR filter structure, with extra delays in the large loop.

Now suppose we need k extra delays in the large loop. If we repeat our strategy of substituting equation 4.4 into the large loop term, substituting k times, we derive

$$y[n] = \epsilon x[n-1] - \epsilon^2 \sum_{m=1}^{k} (1-\epsilon)^{m-1} x[n-m-1] + y[n-1] - \epsilon(1-\epsilon)^k y[n-k-1] \quad (4.6)$$

This may be expressed as

$$y[n] = h_{fix}[n] * \epsilon' x[n-1] + y[n-1] - \epsilon' y[n-k-1]$$

where $\epsilon' = \epsilon(1-\epsilon)^k$ and

$$h_{fix}[n] = \frac{\epsilon}{\epsilon'}(\delta[n] - \epsilon \sum_{m=1}^{k} (1-\epsilon)^{m-1} \delta[n-m])$$

Figure 4-4 implements this, except for an additional latency of k.

This strategy will also work for the highpass output. Note that $y[n]$ depends only

on the signal labelled $v[n]$. Specifically, from figure 4-3,

$$y[n] = \sum_{n=0}^{\infty} v[n]$$

Since we showed above that the lowpass output did not change except for an additional latency of k, and since the delay of k lies between $v[n]$ and $y[n-k]$, $v[n]$ did not change either. From figure 4-3, we see that $v[n] = \epsilon w[n]$. But working backwards from $v[n]$, we see that the highpass output is $\frac{v[n]}{\epsilon'}$, not $w[n] = \frac{v[n]}{\epsilon}$. Therefore, the highpass output of a filter with extra delay elements in the large loop and correction filter $h_{fix}[n]$ is $\frac{\epsilon}{\epsilon'}w[n]$. Therefore, this strategy also works for the highpass output, but the highpass output should be multiplied by $\frac{\epsilon'}{\epsilon}$ to yield $w[n]$.

The above methods have no advantage over look-ahead computation with direct form structures, since we still need a correction filter. However, the fact that ϵ is very small gives us an additional option. Note that the frequency response of $h_{fix}[n]$ at $\omega = 0$ is

$$\sum_{n=-\infty}^{\infty} h_{fix}[n] = \frac{\epsilon}{\epsilon'}(1-\epsilon)^k = 1$$

So for lowpass filters, without changing the frequency response of the IIR filter at $\omega = 0$, we may approximate

$$h_{fix}[n] \approx \delta[n]$$

for small enough ϵ. Therefore we can simply eliminate $h_{fix}[n]$ from the filter, without exceeding the maximum allowable ripple. The ability of the IIR filter to retain the shape of its frequency response when delays are added to the coefficient accrues from having no extra delays in the accumulator loop.

For highpass filters, the frequency response at $\omega = 0$ is 0, so normalizing $h_{fix}[n]$ to have unit frequency response at $\omega = 0$ does not help. Instead, it works well to simply eliminate the small terms, and use the approximation

$$h_{fix}[n] \approx \frac{\epsilon}{\epsilon'}\delta[n]$$

Therefore $w[n]$ should be multiplied by $\frac{\epsilon}{\epsilon'}$ when $h_{fix}[n]$ is not used. This cancels the

factor of $\frac{\epsilon'}{\epsilon}$ due to the extra loop delays.

When ϵ is small, but not small enough for the approximation above to be valid, we may make the stronger approximation

$$h_{fix}[n] \approx \frac{\epsilon}{\epsilon'}(\delta[n] - \epsilon_m \sum_{m=1}^{k} \delta[n-m]) \tag{4.7}$$

where $\epsilon_m = \frac{1}{2}(\epsilon + \epsilon(1-\epsilon)^{k-1})$. This allows us to implement $h_{fix}[n]$ using only one coefficient multiplier.

4.3 Signal "Wrapping" Effects

Recall from section 2.4 that our signal may have infinite amplitude, and may "wrap around" from positive to negative values. As long as the input $x[n]$ has the property that $|x[n] - x[n-1]| < 2^{B_x - 1}$, the signal may be reconstructed unambiguously from $x_{lsp}[n]$, its least significant B_x bits.

Consider a filter without extra delay elements. The difference equation for the highpass output is

$$w[n] = x[n] - x[n-1] + (1-\epsilon)w[n-1]$$

Rearranging, and substituting $\alpha = 1 - \epsilon$ gives

$$w[n] - \alpha w[n-1] = x[n] - x[n-1]$$

Therefore,

$$|w[n] - \alpha w[n-1]| < 2^{B_x - 1}$$

Substituting $n-1$ for n and multiplying both sides by α gives

$$\alpha|w[n-1] - \alpha w[n-2]| < \alpha 2^{B_x - 1}$$

Combining these two inequalities gives us

$$|w[n] - \alpha^2 w[n-2]| < 2^{B_x - 1} + \alpha 2^{B_x - 1}$$

Repeating this process N times gives us

$$|w[n] - \alpha^N w[n-2]| < 2^{B_x - 1} \sum_{k=0}^{N} \alpha^k$$

As $N \to \infty$, since $|\alpha| < 1$,

$$|w[n]| < \frac{1}{1-\alpha} 2^{B_x - 1} = \frac{1}{\epsilon} 2^{B_x - 1}$$

Therefore $|w[n]|$ is limited and does not "wrap."

Suppose also that we wish to know $w[n]$ to a precision of $\pm\frac{q}{2}$. Then to represent $w[n]$, we require a data path width of

$$B_w = \lceil \log_2(\frac{1}{\epsilon} 2^{B_x}) - \log_2(V_{MSB}) \rceil - \lfloor \log_2(q) - \log_2(V_{MSB}) \rfloor \tag{4.8}$$

Where V_{MSB} is the value of the MSB of the output of the decimator, given in table 2.2. Note that if we ignore the bit quantization,

$$B_w \approx B_x - \log_2(q\epsilon) \tag{4.9}$$

Our structure may be simply modified to work properly with input $x_{lsp}[n]$ rather than $x[n]$. Since $w[n]$ may be represented using B_w bits, our input adder only needs to be B_w bits wide. Therefore, we do not need bits of $x[n]$ and $y[n]$ more significant than the MSB of $w[n]$ to calculate $w[n]$ correctly. Therefore these highpass filters will work using $x_{lsp}[n]$, "unwrapped" to the level of the MSB of $w[n]$, as described in section 2.4. Specifically, $x_{lsp}[n]$ should be "unwrapped" by r bits, where

$$r = \lceil \log_2(\frac{1}{\epsilon} 2^{B_x}) - \log_2(V_{MSB}) \rceil \tag{4.10}$$

More significant bits would only affect more significant bits of $w[n]$, which we may obtain by sign-extending the MSB of $w[n]$, since $|w[n]| < \frac{1}{\epsilon}2^{B_x-1}$.

Since ϵ is so small, $|w[n]| < \frac{1}{\epsilon}2^{B_x-1}$ may result in a data path that is far too large. In that case, we may be able to guarantee instead that $|w[n]| < \frac{P}{2}$, for some P. Then

$$B_w = \lceil \log_2(P) - \log_2(V_{MSB}) \rceil - \lfloor \log_2(q) - \log_2(V_{MSB}) \rfloor \tag{4.11}$$

and

$$r = \lceil \log_2(P) - \log_2(V_{MSB}) \rceil \tag{4.12}$$

The amplitude of $w[n]$ will be limited whether the filter is actually being used as a highpass filter or a lowpass filter. Furthermore, since $y[n]$ is calculated directly from $w[n]$, we may calculate $y[n]$ from $x_{lsp}[n]$. Therefore lowpass filters with this structure will work with "wrapping" signals also. Note that the lowpass output will not have a limited amplitude. Therefore, we will only calculate $y_{lsp}[n]$. However, it should be noted that the carry-out signal of the accumulator may be used to calculate $y_{lsp}[n]$ to any desired bit height.

Note that the carry-out bit of the input adder should be ignored. When an adder has B_w bits of input for both inputs, the output normally requires $B_w + 1$ bits, with the carry-out as the extra bit. However, this extra bit would be incorrect, since the correct value would depend also on one bit of $x[n]$ which is not part of $x_{lsp}[n]$.

When there are extra delays in the large loop, and no filter $h_{fix}[n]$ is used, we need to guarantee that the highpass output is bounded, so that these filters will work with wrapping signals also. We define the highpass output of a filter with extra delays as $w'[n]$, i.e. $w[n] = \frac{\epsilon'}{\epsilon}w'[n] * h_{fix}[n]$. Then we need to find a bound P' such that $|w'[n]| < \frac{P'}{2}$. For the purpose of finding the bound, therefore, let us consider placing a correction filter $h_{fix}[n]$ after the highpass output $w'[n]$. Since the IIR filter and $h_{fix}[n]$ are linear time-invariant systems, changing the order of the cascade does not change the total system, so the output of $h_{fix}[n]$ will be $\frac{\epsilon}{\epsilon'}w[n]$. The output of a filter cannot be more than the maximum magnitude of the input times the sum of the magnitudes of its coefficients. The input to $h_{fix}[n]$ is $w'[n]$, which is bounded by

$|w'[n]| < \frac{P'}{2}$. Therefore,

$$\frac{\epsilon}{\epsilon'} w[n] \leq \frac{P'}{2} \sum_{n=0}^{\infty} |h_{fix}[n]|$$

However, P is defined as the limit such that $|w[n]| < \frac{P}{2}$, so $\frac{\epsilon}{\epsilon'}|w[n]| < \frac{\epsilon P}{2\epsilon'}$ Therefore, a sufficient bound P' is given by

$$\frac{\epsilon P}{2\epsilon'} = \frac{P'}{2} \sum_{n=0}^{\infty} |h_{fix}[n]|$$

Solving for P' gives

$$P' = \frac{\epsilon P}{\epsilon' \sum_{n=0}^{\infty} |h_{fix}[n]|}$$

Thus we have a bound $|w'[n]| < \frac{P'}{2}$ where P' is given above, and filters with extra delays in the large loop will work with wrapping signals if we can bound the ideal output by $|w[n]| < \frac{P}{2}$. Then, P' rather than P should be used in equation 4.11 to determine B_w, and in equation 4.12 to determine r.

The approximation to the correction filter will only be used in one instance, for a lowpass filter in the "bandpass" band, where highpass filters have already removed the wrapping. However, in the more general case where $h_{fix}[n]$, or its approximation given in section 4.2, may need to be used on a wrapping signal, it is straightforward to apply the techniques of section 2.4 to the correction filter.

4.4 Multipliers

General purpose multipliers can be very complex. However, we only need to design one special-purpose multiplier for each filter, to multiply by the coefficient we need for that filter. For this purpose, we may multiply by a sum of a small number of powers of two by adding the input to itself, shifted. For example, we may multiply a data stream by $10 = 2^1 + 2^3$ using one adder, by shifting the data left by 1 bit, shifting the data left by 3 bits, and then adding the two shifted data streams.

We want to use as few adders as possible for the coefficient multiplication, to fit the system on the smallest amount of Altera hardware possible and to lower the

propagation delay around the large loop. Furthermore, rounding the outputs of the multipliers may cause the coefficient ϵ to effectively change values when the amplitude of the input is small. Because of this, the output of the multipliers will not be rounded except when necessary. Therefore, we need to limit the maximum shift allowable, so that the adders will not become too large. Our constraint is that the cutoff frequency may not vary by more than $\pm 2\%$.

Since $\omega_c = 2\pi \frac{F_c}{F_s}$ is the discrete-time cutoff frequency, $\alpha = e^{-\omega_c}$. Taking the first order term in the Taylor expansion yields the approximation $\alpha \approx 1 - \omega_c$. However, recall that $\alpha = 1 - \epsilon$. Therefore $\epsilon \approx \omega_c$. Since the cutoff frequencies ω_c are very low, this approximation is very good. Therefore, if our coefficient ϵ is within $\pm 2\%$, our cutoff frequency will be within $\pm 2\%$, as required. Since there is only one coefficient, movement of the cutoff frequency is the only effect of coefficient rounding.

Using two adders, there are two independent ways of using shifts to generate the coefficients ϵ. The structure in figure 4-5 (a) will multiply by any coefficient of the form $2^{-s}(1 \pm 2^{-m} \pm 2^{-n})$. The structure in figure 4-5 (b) will multiply by any coefficient of the form $2^{-s}(1 \pm 2^{-m})(1 \pm 2^{-n})$.

We may find such a coefficient in a straightforward manner. First, express ϵ as $\epsilon = \epsilon_n \cdot 2^{-s'}$, where $s' = \lfloor -\log_2 \epsilon \rfloor$. This gives $.5 < \epsilon_n \leq 1$. An exhaustive search of all possible coefficients which are between .5 and 1, which use no more than two adders and which increase the data width by no more than 9 bits, is straightforward and quick using MATLAB. Table 4.1 indicates that there will always be such a coefficient within $\pm 1.734\%$ of any possible coefficient ϵ_n. Furthermore, most coefficients may be approximated much more closely than that. Therefore two adders and a data path width increase of 9 bits is sufficient to keep the cutoff frequency within $\pm 2\%$. Without loss of generality, we may choose m and n such that $m < n$. Notice that all of the values listed in the table for $\epsilon_n < .82$ include a factor of 2^{-1}, and their 2^{-m} term is added into the expression. For these coefficients, $s = s' + 1$. For coefficients with $\epsilon_n > .82$, the 2^{-m} term is subtracted, and $s = s'$.

The way we are implementing coefficient multipliers demonstrates one advantage of the structure we are using, as shown in figure 4-2, when our cutoff frequency is

Coefficient	Two Adder Realization	Max Err.	Coefficient	Two Adder Realization	Max Err.
0.5019531250	$(2^{-1})(1+2^{-8})$	0.195 %	0.7500000000	$(2^{-1})(1+2^{-1})$	0.130 %
0.5039062500	$(2^{-1})(1+2^{-7})$	0.194 %	0.7519531250	$(2^{-1})(1+2^{-1}+2^{-8})$	0.130 %
0.5058593750	$(2^{-1})(1+2^{-7}+2^{-8})$	0.193 %	0.7539062500	$(2^{-1})(1+2^{-1}+2^{-7})$	0.130 %
0.5078125000	$(2^{-1})(1+2^{-6})$	0.193 %	0.7558593750	$(2^{-1})(1+2^{-1})(1+2^{-7})$	0.129 %
0.5097656250	$(2^{-1})(1+2^{-6}+2^{-8})$	0.192 %	0.7578125000	$(2^{-1})(1+2^{-1}+2^{-6})$	0.129 %
0.5117187500	$(2^{-1})(1+2^{-6}+2^{-7})$	0.191 %	0.7617187500	$(2^{-1})(1+2^{-1})(1+2^{-6})$	0.257 %
0.5136718750	$(2^{-1})(1+2^{-5}-2^{-8})$	0.190 %	0.7656250000	$(2^{-1})(1+2^{-1}+2^{-5})$	0.256 %
0.5156250000	$(2^{-1})(1+2^{-5})$	0.190 %	0.7734375000	$(2^{-1})(1+2^{-1})(1+2^{-5})$	0.508 %
0.5175781250	$(2^{-1})(1+2^{-5}+2^{-8})$	0.189 %	0.7812500000	$(2^{-1})(1+2^{-1}+2^{-4})$	0.503 %
0.5195312500	$(2^{-1})(1+2^{-5}+2^{-7})$	0.188 %	0.7968750000	$(2^{-1})(1+2^{-1})(1+2^{-4})$	0.990 %
0.5234375000	$(2^{-1})(1+2^{-5}+2^{-6})$	0.375 %	0.8125000000	$(2^{-1})(1+2^{-1}+2^{-3})$	0.971 %
0.5273437500	$(2^{-1})(1+2^{-4}-2^{-7})$	0.372 %	0.8203125000	$(2^{0})(1-2^{-3})(1-2^{-4})$	0.478 %
0.5292968750	$(2^{-1})(1+2^{-4}-2^{-8})$	0.185 %	0.8437500000	$(2^{0})(1-2^{-3}-2^{-5})$	1.408 %
0.5312500000	$(2^{-1})(1+2^{-4})$	0.184 %	0.8476562500	$(2^{0})(1-2^{-3})(1-2^{-5})$	0.231 %
0.5332031250	$(2^{-1})(1+2^{-4}+2^{-8})$	0.183 %	0.8593750000	$(2^{0})(1-2^{-3}-2^{-6})$	0.686 %
0.5351562500	$(2^{-1})(1+2^{-4}+2^{-7})$	0.183 %	0.8613281250	$(2^{0})(1-2^{-3})(1-2^{-6})$	0.114 %
0.5390625000	$(2^{-1})(1+2^{-4}+2^{-6})$	0.364 %	0.8671875000	$(2^{0})(1-2^{-3}-2^{-7})$	0.339 %
0.5449218750	$(2^{-1})(1+2^{-3})(1-2^{-5})$	0.541 %	0.8710937500	$(2^{0})(1-2^{-3}-2^{-8})$	0.225 %
0.5468750000	$(2^{-1})(1+2^{-4}+2^{-5})$	0.179 %	0.8730468750	$(2^{0})(1-2^{-3}-2^{-9})$	0.112 %
0.5546875000	$(2^{-1})(1+2^{-3}-2^{-6})$	0.709 %	0.8750000000	$(2^{0})(1-2^{-3})$	0.112 %
0.5585937500	$(2^{-1})(1+2^{-3}-2^{-7})$	0.351 %	0.8769531250	$(2^{0})(1-2^{-3}+2^{-9})$	0.111 %
0.5605468750	$(2^{-1})(1+2^{-3}-2^{-8})$	0.175 %	0.8789062500	$(2^{0})(1-2^{-3}+2^{-8})$	0.111 %
0.5625000000	$(2^{-1})(1+2^{-3})$	0.174 %	0.8828125000	$(2^{0})(1-2^{-3}+2^{-7})$	0.222 %
0.5644531250	$(2^{-1})(1+2^{-3}+2^{-8})$	0.173 %	0.8867187500	$(2^{0})(1-2^{-3})(1+2^{-6})$	0.331 %
0.5664062500	$(2^{-1})(1+2^{-3}+2^{-7})$	0.173 %	0.8906250000	$(2^{0})(1-2^{-3}+2^{-6})$	0.110 %
0.5703125000	$(2^{-1})(1+2^{-3}+2^{-6})$	0.344 %	0.9023437500	$(2^{0})(1-2^{-3})(1+2^{-5})$	0.654 %
0.5781250000	$(2^{-1})(1+2^{-3}+2^{-5})$	0.680 %	0.9062500000	$(2^{0})(1-2^{-3}+2^{-5})$	0.216 %
0.5800781250	$(2^{-1})(1+2^{-3})(1+2^{-5})$	0.169 %	0.9082031250	$(2^{0})(1-2^{-4})(1-2^{-5})$	0.108 %
0.5859375000	$(2^{-1})(1+2^{-2})(1-2^{-4})$	0.503 %	0.9218750000	$(2^{0})(1-2^{-4}-2^{-6})$	0.747 %
0.5937500000	$(2^{-1})(1+2^{-3}+2^{-4})$	0.662 %	0.9296875000	$(2^{0})(1-2^{-4}-2^{-7})$	0.422 %
0.5976562500	$(2^{-1})(1+2^{-3})(1+2^{-4})$	0.328 %	0.9335937500	$(2^{0})(1-2^{-4}-2^{-8})$	0.210 %
0.6054687500	$(2^{-1})(1+2^{-2})(1-2^{-5})$	0.649 %	0.9355468750	$(2^{0})(1-2^{-4}-2^{-9})$	0.104 %
0.6093750000	$(2^{-1})(1+2^{-2}-2^{-5})$	0.322 %	0.9375000000	$(2^{0})(1-2^{-4})$	0.104 %
0.6152343750	$(2^{-1})(1+2^{-2})(1-2^{-6})$	0.478 %	0.9394531250	$(2^{0})(1-2^{-4}+2^{-9})$	0.104 %
0.6171875000	$(2^{-1})(1+2^{-2}-2^{-6})$	0.158 %	0.9414062500	$(2^{0})(1-2^{-4}+2^{-8})$	0.104 %
0.6210937500	$(2^{-1})(1+2^{-2}-2^{-7})$	0.315 %	0.9453125000	$(2^{0})(1-2^{-4}+2^{-7})$	0.207 %
0.6230468750	$(2^{-1})(1+2^{-2}-2^{-8})$	0.157 %	0.9531250000	$(2^{0})(1-2^{-4}+2^{-6})$	0.412 %
0.6250000000	$(2^{-1})(1+2^{-2})$	0.156 %	0.9609375000	$(2^{0})(1-2^{-5}-2^{-7})$	0.408 %
0.6269531250	$(2^{-1})(1+2^{-2}+2^{-8})$	0.156 %	0.9648437500	$(2^{0})(1-2^{-5}-2^{-8})$	0.203 %
0.6289062500	$(2^{-1})(1+2^{-2}+2^{-7})$	0.156 %	0.9667968750	$(2^{0})(1-2^{-5}-2^{-9})$	0.101 %
0.6328125000	$(2^{-1})(1+2^{-2}+2^{-6})$	0.310 %	0.9687500000	$(2^{0})(1-2^{-5})$	0.101 %
0.6347656250	$(2^{-1})(1+2^{-2})(1+2^{-6})$	0.154 %	0.9707031250	$(2^{0})(1-2^{-5}+2^{-9})$	0.101 %
0.6406250000	$(2^{-1})(1+2^{-2}+2^{-5})$	0.459 %	0.9726562500	$(2^{0})(1-2^{-5}+2^{-8})$	0.101 %
0.6445312500	$(2^{-1})(1+2^{-2})(1+2^{-5})$	0.304 %	0.9765625000	$(2^{0})(1-2^{-5}+2^{-7})$	0.200 %
0.6562500000	$(2^{-1})(1+2^{-2}+2^{-4})$	0.901 %	0.9804687500	$(2^{0})(1-2^{-6}-2^{-8})$	0.200 %
0.6640625000	$(2^{-1})(1+2^{-2})(1+2^{-4})$	0.592 %	0.9824218750	$(2^{0})(1-2^{-6}-2^{-9})$	0.100 %
0.6875000000	$(2^{-1})(1+2^{-2}+2^{-3})$	1.734 %	0.9843750000	$(2^{0})(1-2^{-6})$	0.099 %
0.7031250000	$(2^{-1})(1+2^{-2})(1+2^{-3})$	1.124 %	0.9863281250	$(2^{0})(1-2^{-6}+2^{-9})$	0.099 %
0.7187500000	$(2^{-1})(1+2^{-1}-2^{-4})$	1.099 %	0.9882812500	$(2^{0})(1-2^{-6}+2^{-8})$	0.099 %
0.7265625000	$(2^{-1})(1+2^{-1})(1-2^{-5})$	0.541 %	0.9902343750	$(2^{0})(1-2^{-7}-2^{-9})$	0.099 %
0.7343750000	$(2^{-1})(1+2^{-1}-2^{-5})$	0.535 %	0.9921875000	$(2^{0})(1-2^{-7})$	0.099 %
0.7382812500	$(2^{-1})(1+2^{-1})(1-2^{-6})$	0.265 %	0.9941406250	$(2^{0})(1-2^{-7}+2^{-9})$	0.098 %
0.7421875000	$(2^{-1})(1+2^{-1}-2^{-6})$	0.264 %	0.9960937500	$(2^{0})(1-2^{-8})$	0.098 %
0.7441406250	$(2^{-1})(1+2^{-1})(1-2^{-7})$	0.131 %	0.9980468750	$(2^{0})(1-2^{-9})$	0.098 %
0.7460937500	$(2^{-1})(1+2^{-1}-2^{-7})$	0.131 %	1.0000000000	1	0.098 %
0.7480468750	$(2^{-1})(1+2^{-1}-2^{-8})$	0.131 %			

Table 4.1: All possible coefficients which may be realized with two adders, while increasing the data path by no more than 8 bits. The maximum percentage error from such a coefficient to the number halfway between it and the previous coefficient is also given. Thus the maximum possible error between any desired coefficient and its implementation is 1.734%.

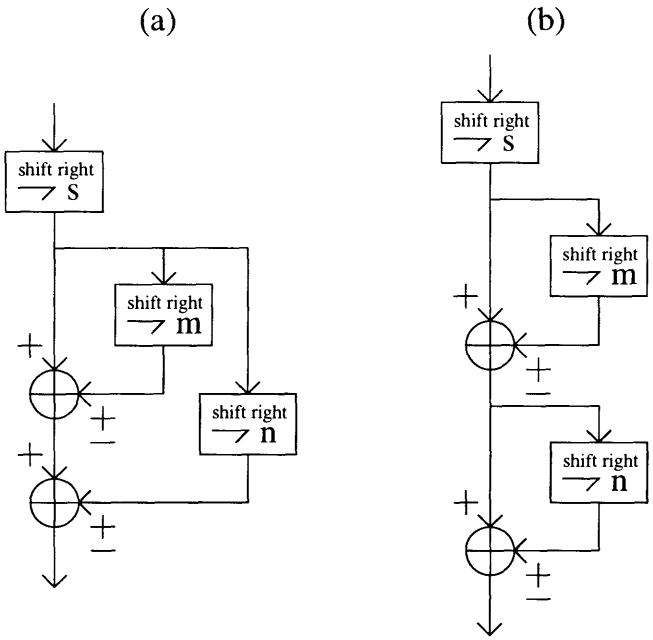

Figure 4-5: Sum of powers of two multiplier forms using two adders. (a) multiplies by $2^{-s}(1 \pm 2^{-m} \pm 2^{-n})$. (b) multiplies by $2^{-s}(1 \pm 2^{-m})(1 \pm 2^{-n})$.

very low. If, instead, we had chosen to use a direct form structure, then we would have had to multiply by α rather than ϵ. Because α is very close to 1, the most efficient way to implement the coefficient multiplication by α, using powers of two again, would have been to multiply by $1-\epsilon$. This would have required one more adder (actually a subtracter), and then ϵ could be implemented as above, using two adders. So though the lowpass direct form structure seems to have more adders, there is not a real savings when the cutoff frequency is very low. Instead, since s in figure 4-5 is very large, the data path would have been very wide using the direct form structure, and all of the adders would have to be very wide. Instead, for our structure, the wide data path is confined to the accumulator loop, and only the accumulator adder needs to be very wide. Consider, for example, the 10 Hz lowpass filter required for the 1.544 MHz sample rate. $\omega_c = .00004641$, and $s = \lceil -\log_2 \omega_c \rceil = 15$. Therefore we would need 15 more bits in the data path if we used a direct form structure. Instead, for our structure, the wide data path is confined to the accumulator loop, and only the accumulator adder needs to be very wide.

Chapter 5

Implementation of Off-chip Filters

This chapter will show how to implement the filtering system, using Altera FLEX8000 Programmable Logic Devices. These devices are based on SRAM technology, so they may be reprogrammed almost instantly by the system when the user selects a new filter configuration. This means that the FLEX8000's will only have to implement one filter configuration at a time. Therefore, we only need enough FLEX8000 programmable hardware to implement the largest configuration.

The filtering system is pictured in figure 5-1. The user may wish to have access to two bands simultaneously. These bands are the "lowpass" band and the "bandpass" band. The filter for the "lowpass" band should consist of a first-order lowpass filter with a filter cutoff at either 10 Hz or 100 Hz, which the user may select. The filter for the "bandpass" band should consist of a third- or first-order (depending on the base data rate) butterworth lowpass filter and a first-order highpass filter. The filter cutoffs for these filters are specified in table 5.1. Furthermore, it is possible that the input to the filter would contain an additive ramp signal, with slope such that our limit on the first difference given in 4.3 is satisfied. Because the first-order highpass filter places only one zero at $\omega = 0$, this can result in an D.C. offset on $w[n]$, and therefore on the output of the "bandpass" band. The user may wish to remove this offset. Because we showed in section 4.3 that $w[n]$ is bounded, one more highpass filter is sufficient to remove this offset. Because of this, the user should be able to use an additional first-order highpass filter with a filter cutoff of 0.2 Hz. Note that since

the 0.2 Hz highpass filter follows another highpass filter, this filter does not have to deal with "wrapping," and the data path width requirement given in equations 4.8 and 4.11 are relaxed. First-order filter frequency cutoffs should deviate no more than 2% from the given values. Third-order filter cutoffs should be within 5% of the given values. For no filter should the passband ripple exceed 2%.

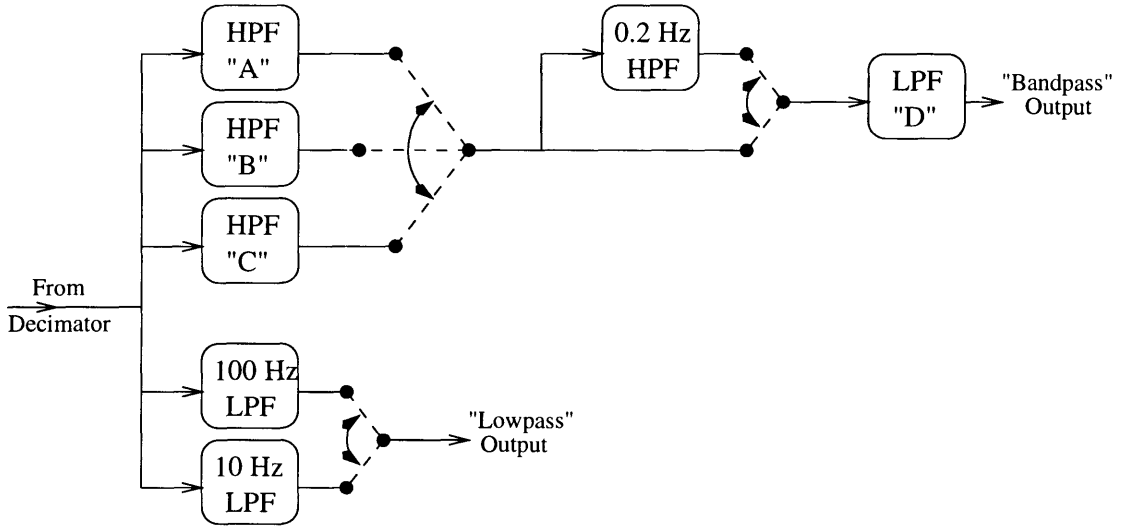

Figure 5-1: Filtering System.

input rate (MHz)	10 Hz LPF	100 Hz LPF	0.2 Hz HPF	HPF "A" (Hz)	HPF "B" (kHz)	HPF "C" (kHz)	LPF "D" (kHz)	LPF "D" order
1.544	10	100	0.2	10	8		40	1
2.048	10	100	0.2	20	0.7	18	200	3
6.312	10	100	0.2	10	3		60	1
8.448	10	100	0.2	20	3	80	400	3
34.368	10	100	0.2	100	10		800	3
44.736	10	100	0.2	10	30		400	1
51.84	10	100	0.2	10	.5	20	400	3
139.264	10	100	0.2	200	10		3500	3
155.52	10	100	0.2	10	1	65	1300	3
622.08	10	100	0.2	10	5	250	5000	3

Table 5.1: Filters to be implemented.

In addition to the filtering system, we also need to place part of the decimator

on the Altera hardware, as well as an unwrapping bit extender. Recall from section 3.1 that 3 first-differencers were required to be implemented off-chip, as part of the decimator. Also, recall from section 4.3 that we will need to "unwrap" the input to extend the MSB to allow proper filtering of a "wrapping" signal. Furthermore, we need to correct passband attenuation from the decimator, for those data rates which required a decimator.

To give flexibility to the product designers, two designs are presented here. One design implements all of the filters with 16-bit data paths. This design requires one FLEX8820 and one FLEX81188, both in the A-3 speed grade. The 10 Hz or 100 Hz lowpass filter, the 0.2 Hz highpass filter, the "bandpass" highpass filter, the unwrapping bit extender, and the remainder of the decimator will be placed on the FLEX8820. A 16-tap FIR filter will be placed on the FLEX81188, for those data rates requiring a third-order butterworth lowpass filter for the "bandpass" band. The FIR filter will emulate the third-order filter, while correcting passband attenuation of the decimator. For those data rates requiring a first-order lowpass filter for the "bandpass" band, this filter will be placed on the FLEX81188. Those data rates using first-order lowpass filters for the bandpass band do not use the decimator, so no correction for passband attenuation is required.

A second design implements the filters with 24-bit data paths. This design requires one FLEX81188 and two FLEX8820s, all in the A-2 speed grade. The functions implemented above on the FLEX8820 are implemented in the FLEX81188, and the functions implemented above on the FLEX81188 are implemented on the two FLEX8820s.

Furthermore, the product designers will be able to choose the number of bits r from section 4.3, by which the signal will be "unwrapped." This will allow the bits to be chosen which make the best compromise between maximum highpass signal amplitude P and precision q, given in section 4.3.

5.1 Summary of Altera FLEX8000 Capabilities

The Altera FLEX8000 architecture is well suited to structures such as ours. Details relevant to our implementation will be summarized here. For a full description of the architecture, refer to the Altera Data Book [2]. We will use FLEX8820s and FLEX81188s.

In the configuration we are interested in, a single Logic Element ("LE") consists of two three-input lookup tables and an output register. These lookup tables may be programmed by the user. The output of one three-input lookup table, which will be called the "sum" lookup table, may go either to the register or directly to the output. The other lookup table, which will be called the "carry" lookup table, generates a carry chain signal, which has a dedicated, fast connection to the next LE. Since a one-bit full adder or subtracter has three inputs and two outputs, with the carry output going to the next bit, one logic element may be used to implement an adder or a subtracter.

Eight LEs are grouped together in a Logic Array Block ("LAB"). Each LAB is connected to other LABs on its row by a row interconnection channel. FLEX8820s and FLEX81188s have 21 LABs per row, and therefore 168 LEs per row. Each LAB is also connected to a column interconnection channel, which connects to the row channels for other rows. FLEX8820s have 4 rows, and FLEX81188s have 6 rows. The output of an LE is directed to the row channel if the signal will be used by an LE in another LAB on the same row. If the output of an LE will be used by an LE in a LAB on a different row, the output is directed to the column channel. A signal in a column channel then may be directed to any row channel, and then to the correct LE. A carry chain also connects the last LE in a LAB to the first LE in the next LAB.

A simplified block diagram in figure 5-2 shows the timing parameters which we will need. Table 5.2 describes the timing parameters and gives the values for A-2 and A-3 speed grades, in the FLEX8820s and FLEX81188s, which we will use. Timing parameters for other FLEX8000 chips are similar or identical, except that $t_{row} = 6.2$ns

for the FLEX81500, the largest FLEX8000 chip. Unfortunately, at the time this work was done, Altera software which would have been needed to design, simulate, and program the chips was not working properly. Because of this, no simulations or experimental data are available. The timing parameters in table 5.2 will be used for all calculations and plots. However, the actual values are guaranteed to be no worse than these parameters.

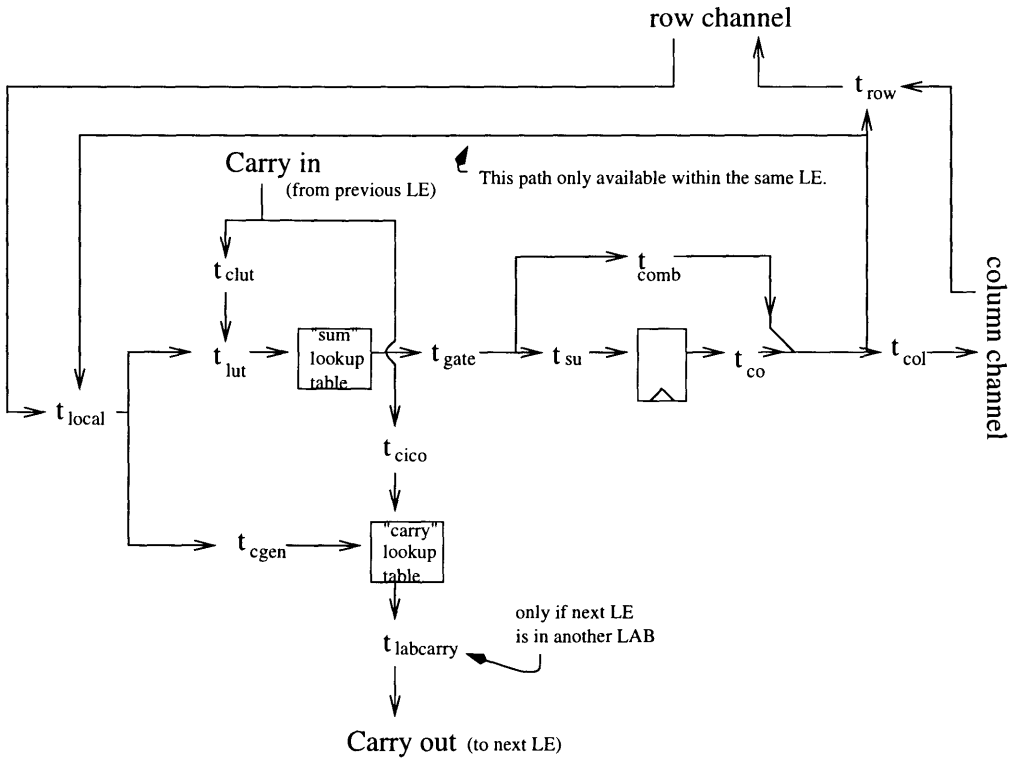

Figure 5-2: Simplified timing diagram for Altera FLEX8000.

5.2 Directly Implemented Filters

The circuit in figure 5-3 shows our filter structure, including the coefficient multipliers. In 5-3(a), The input adder is b bits wide, so it uses b LEs. The first coefficient adder is $b+m+1$ bits wide, including the carry-out signal. No carry-out signal is required for the second coefficient adder, since $0 < m < n$. Therefore, the second coefficient adder is $b+n+1$ bits wide. The accumulator is $b+n+s$ bits wide, and the highpass

Parameter	Description	A-2 grade value (ns)	A-3 grade value (ns)
t_{lut}	"Sum" look-up table delay.	2.0	2.5
t_{clut}	Carry chain-to-lookup table delay.	0.0	0.0
t_{gate}	Gate delay.	0.0	0.0
t_{cico}	Carry-in, carry-out delay.	0.4	0.5
t_{cgen}	Carry-out generation delay.	0.4	0.5
t_{co}	LE register clock-to-output delay.	0.4	0.5
t_{comb}	Combinatorial output delay.	0.4	0.5
t_{su}	Register setup time.	0.8	1.1
$t_{labcarry}$	Additional carry chain delay incurred by carrying to the next LAB.	0.3	0.3
t_{local}	Local interconnect delay.	0.5	0.6
t_{row}	Row channel delay.	5.0	5.0
t_{col}	Column channel delay.	3.0	3.0

Table 5.2: Timing parameter values for FLEX8820 and FLEX81188.

output register, which we need for highpass filters only, is b bits wide. It should be noted that truncating, rather than rounding, the $y[n]$ term in the input adder does not hurt the output. This is equivalent to adding an offset equal to half the value of the LSB from the input. However, this offset will be removed from the highpass output $w[n]$.

Furthermore, some extra LEs will be used as buffers to lower the fanout of some signals. The value listed for t_{row} in the Altera Data Book is valid for a fanout of no more than 4 [2]. Most outputs of the input adder drive 4 signals—one bit each of the output register, the unshifted input of the first coefficient adder, the shifted input of the first coefficient adder, and the shifted input of the second coefficient adder. However, because of the sign extending necessary for the shifted inputs to the coefficient adders, the MSB of the input adder also drives m extra sign extend bits of the first coefficient adder and n extra sign extend bits of the second coefficient adder. Therefore, the MSB has a fanout of $4 + m + n$. The carry chain may be used to repeat the MSB in the next LE. In this way, we may use $\lceil \frac{4+m+n}{4} \rceil - 1$ LEs to buffer the output so that no LE output drives more than 4 LE inputs, as required. Similarly, $\lceil \frac{s}{4} \rceil - 1$ LEs are needed to buffer the MSB of the second coefficient adder,

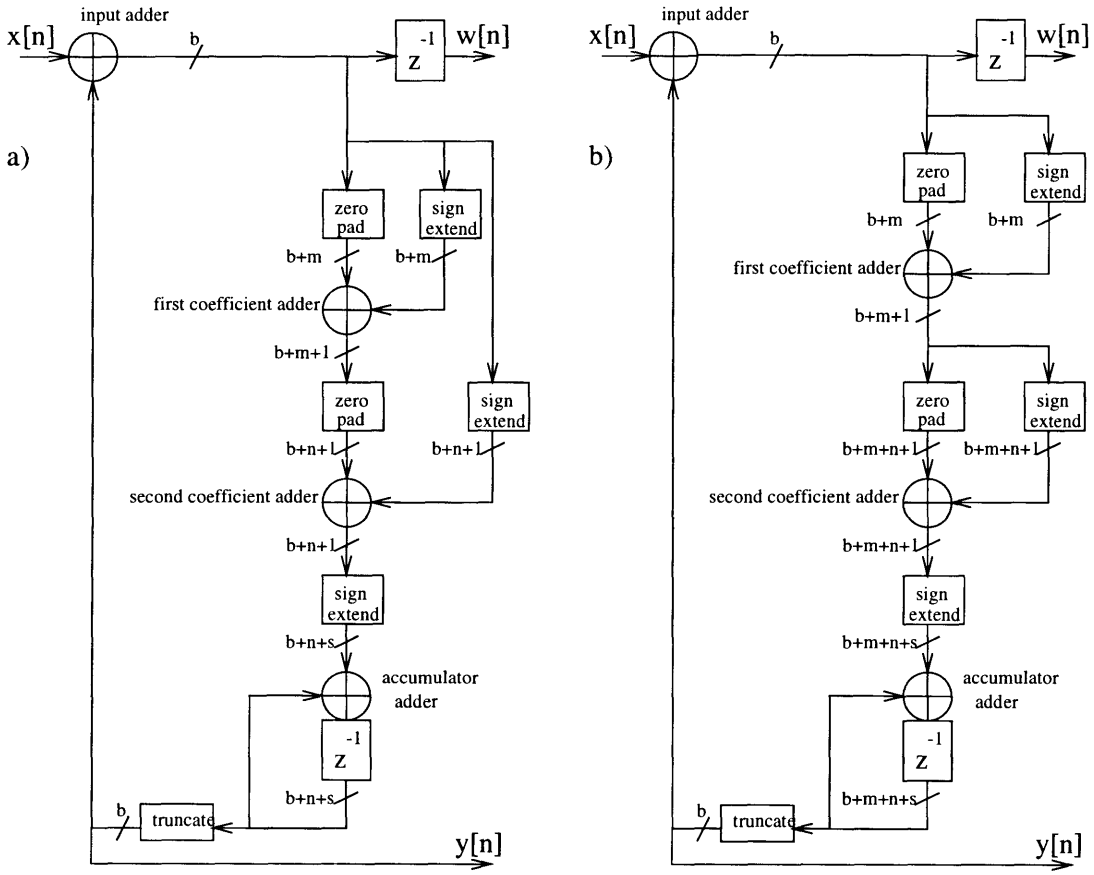

Figure 5-3: IIR filter structures, without pipelined loops, including coefficient multipliers. "Zero pad" modules add zeros onto the LSP, "sign extend" modules add bits identical to the sign bit onto the MSP, and "truncate" modules truncate the LSP.

which is sign-extended $s-1$ bits. The buffering requires

$$q = \lceil \frac{4+m+n}{4} \rceil + \lceil \frac{s}{4} \rceil - 2$$

LEs. The total number of LEs for a highpass filter in the form of figure 5-3(a) is

$$5b + s + m + 2n + 2 + q$$

A lowpass filter will not require the output register for $w[n]$, since $y[n]$ is the output. Therefore

$$4b + s + m + 2n + 2 + q$$

LEs are required for a lowpass filter of the form of figure 5-3(a).

For a highpass filter of the form of figure 5-3(b), the second coefficient adder requires $b+m+n+1$ LEs and the accumulator requires $b+m+n+s$ LEs. Furthermore, because the shifted input to the second coefficient adder comes from the first coefficient adder rather than the input adder, the buffering is different. We must use $\lceil \frac{3+m}{4} \rceil - 1$ LEs to buffer the MSB of the input adder, $\lceil \frac{1+n}{4} \rceil - 1$ LEs to buffer the MSB of the first coefficient adder, and $\lceil \frac{s}{4} \rceil - 1$ LEs to buffer the MSB of the second coefficient adder. Therefore the buffering requires

$$q = \lceil \frac{3+m}{4} \rceil + \lceil \frac{1+n}{4} \rceil + \lceil \frac{s}{4} \rceil - 3$$

LEs. The total number of LEs for a highpass filter in the form of figure 5-3(b) is

$$5b + s + 3m + 2n + 2 + q$$

A lowpass filter will not require the output register for $w[n]$, since $y[n]$ is the output. Therefore

$$4b + s + 3m + 2n + 2 + q$$

LEs are required for a lowpass filter of the form of figure 5-3(b).

The longest delay in the circuit will be the time for the LSB of the input signal to propagate to the MSB of the accumulator register. First, let us consider one particular signal path from the LSB of the input to the MSB of the output register. The path begins in a register on a different row than our filter. The signal incurs delay t_{co} in the register, t_{col} in the column channel, t_{row} in the row channel, and t_{local} when it reaches the LAB where the LSB of the input adder is located. So the time into the first LAB is

$$t_{in} = t_{co} + t_{col} + t_{row} + t_{local}$$

The path then follows the carry chain, incurring delay t_{cgen} in the carry lookup table, generating the carry signal. If the data path width is b, then we incur $b-2$ delays of t_{cico} to get to the top of the carry chain. Recall from section 4.2 that the carry-out signal from the MSB of the input adder is ignored, so we do not calculate it. Depending on where in a LAB the LSB of the input adder is located, we could incur up to $\lceil \frac{b-1}{8} \rceil$ delays of $t_{labcarry}$. At the end of the carry chain, the path incurs delay t_{clut} before reaching the lookup table. The time required for the carry chain is a function of b:

$$T_{chain}(b) = t_{cgen} + (b-2) \cdot t_{cico} + \lceil \frac{b-1}{8} \rceil \cdot t_{labcarry} + t_{clut}$$

We then incur a delay of t_{lut} calculating the sum bit of the input adder. After delays t_{gate} and t_{comb}, our signal reaches the output of the MSB of the input adder. Each sum output of the adder goes to two inputs of the first coefficient adder: one shifted right, and one unshifted. The signal path that is shifted m bits to the right is the longer path. From the MSB of the input adder, the signal incurs delays t_{row} and t_{local} to get to the input of the lookup tables of the second coefficient adder. Note that we did not incur a delay from the column channel, since we are able to put the whole filter on one row. So the time required for a signal to propagate through a sum lookup table to the input of another adder is

$$t_{next} = t_{lut} + t_{gate} + t_{comb} + t_{row} + t_{local}$$

The signal then incurs a delay of $T_{chain}(m+1)$ in reaching the carry-out bit of the first coefficient adder.

A similar path runs through the second coefficient adder, incurring delays of t_{next} and $T_{chain}(n)$. To reach the accumulator adder, the signal again incurs a delay of t_{next}. The signal reaches the end of the carry chain after a further delay of $T_{chain}(s-1)$. Calculating the final sum bit for the of the accumulator incurs a delay of t_{lut}. Finally, after a delays of t_{gate} and t_{su}, the signal has reached the end of the signal path, ready for another clock edge. The delay from the end of the accumulator carry chain to the accumulator register is

$$t_{out} = t_{lut} + t_{gate} + t_{su}$$

The total propagation delay for the whole path is

$$t_{in} + 3 \cdot t_{next} + t_{out} + T_{chain}(b) + T_{chain}(m+1) + T_{chain}(n) + T_{chain}(s-1)$$

Many signal paths similar to the one above lead from the LSB of the input to the MSB of the accumulator register. These paths follow a_1 bits of the carry chain in the input adder before following a sum bit to the first coefficient adder, where they follow a_2 bits of that carry chain, follow a sum bit to the second coefficient adder, follow a_3 bits of that carry chain, follow a sum bit to the accumulator, and follow a_4 bits of the carry chain to the MSB. However, in all cases $a_1 + a_2 + a_3 + a_4 = A_b$, where $A_b = b+m+n+s$. Therefore, the number of carry delays does not change. However, some paths might incur more $t_{labcarry}$ delays than others. The propagation delay for one of these paths is

$$t_{in} + 3 \cdot t_{next} + t_{out} + \sum_{k=1}^{4} T_{chain}(a_k)$$

Now suppose that $a_1 = a_2 = a_3 = 1$, and $a_4 = b + m + n + s - 3$. Then $\lceil \frac{1}{8} \rceil = 1$, so $\sum_{k=1}^{4} T_{chain}(a_k)$ becomes

$$4 \cdot t_{cgen} + (A_b - 8) \cdot t_{cico} + (3 + \lceil A_b - 7 \rceil) \cdot t_{labcarry} + 4 \cdot t_{clut}$$

Therefore, the worst possible propagation delay of the whole circuit is

$$t_{in} + 3 \cdot t_{next} + t_{out} + 4 \cdot t_{cgen} + (A_b - 8) \cdot t_{cico} + (3 + \lceil A_b - 7 \rceil) \cdot t_{labcarry} + 4 \cdot t_{clut} \quad (5.1)$$

Note that the buffers do not affect timing, if we use them intelligently. The first MSB output should be used to drive the most critical signals. In this case, the most critical signal is the non-sign-extended input of the first coefficient adder. The first few sign-extended bits are also critical. Because every t_{cico} delay incurred by the carry chain buffering yields 4 output signals, the buffering will not affect our timing.

The propagation delay of the circuit in figure 5-3(b) is slightly different. Notice that the shifted input to the second adder comes from the input adder. For all speed grades and all coefficients with a maximum shift of no more than 9, $t_{next} > T_{chain}(n)$. Therefore, a shifted input bit to the second coefficient adder will already be calculated when the output of the first coefficient adder becomes valid. Therefore, if $A_b = b + m + s$, the worst possible propagation delay of the whole circuit is given by 5.1.

Figure 5-4 plots the maximum clock frequency resulting from expression 5.1 against A_b. A filter may be implemented without resorting to extra delay tactics developed in section 4.2 if the maximum clock frequency is greater than the decimated data rate. Table 5.3 lists the filters which may be implemented with 16 bit data paths in A-3 speed grade FLEX8820s, while table 5.4 lists the filters which may be implemented with 24 bit data paths in A-2 speed grade FLEX81188s. In both cases, all of the filters were implemented for the base data rates 1.544 MHz, 2.048 MHz, 6.312 MHz, 8.448 MHz, 34.368 MHz, 51.840 MHz, and 155.520 MHz. For each of these base data rates, the decimated data rate was below 13 MHz. The actual frequency cutoff error was always below 2%, as required. Also, each filter required no more than 168 LEs, so each filter may be implemented on one row.

If we had chosen the closest value of ϵ_n from table 4.1 as described in section 5.4, the 0.2 Hz highpass filters for the 8.448 MHz and 34.368 base data rates with data path width $b = 24$ would have consumed more than 168 LEs, and would therefore

Filter	Decimated Data Rate (MHz)	Coefficient Implementation	Percent Frequency Error	LEs	A_b	Maximum Clock Rate (MHz)	C_{iir}
1.544, 10 Hz LPF	1.544	$(2^{-15})(1+2^{-2})(1+2^{-4})$	0.4 %	100	37	17.513	1.000
1.544, 100 Hz LPF	1.544	$(2^{-11})(1-2^{-3}-2^{-5})$	1.3 %	94	30	18.762	1.000
1.544, 0.2 Hz HPF	1.544	$(2^{-20})(1-2^{-3})(1-2^{-5})$	0.7 %	127	44	16.420	1.000
1.544, 10 Hz HPF	1.544	$(2^{-15})(1+2^{-2})(1+2^{-4})$	0.4 %	116	37	17.513	1.000
1.544, 8 kHz HPF	1.544	$(2^{-5})(1+2^{-6}+2^{-7})$	0.2 %	112	27	19.305	1.016
1.544, 40 kHz LPF	1.544	$(2^{-3})(1+2^{-3})(1+2^{-4})$	0.6 %	88	26	19.493	1.000
2.048, 10 Hz LPF	2.048	$(2^{-15})(1+2^{-8})$	0.1 %	86	39	20.202	1.000
2.048, 100 Hz LPF	2.048	$(2^{-12})(1+2^{-2}+2^{-7})$	0.1 %	99	30	18.762	1.000
2.048, 0.2 Hz HPF	2.048	$(2^{-21})(1+2^{-2})(1+2^{-5})$	0.2 %	126	44	16.420	1.000
2.048, 20 Hz HPF	2.048	$(2^{-14})(1+2^{-8})$	0.1 %	101	38	20.408	1.000
2.048, 700 Hz HPF	2.048	$(2^{-9})(1+2^{-4}+2^{-5})$	0.4 %	110	29	18.939	1.001
2.048, 18 kHz HPF	2.048	$(2^{-4})(1-2^{-3}-2^{-6})$	0.0 %	104	23	20.202	1.028
6.312, 10 Hz LPF	6.312	$(2^{-17})(1+2^{-2}+2^{-4})$	0.6 %	99	35	17.825	1.000
6.312, 100 Hz LPF	6.312	$(2^{-14})(1+2^{-1}+2^{-3})$	0.4 %	91	31	18.587	1.000
6.312, 0.2 Hz HPF	6.312	$(2^{-22})(1-2^{-3}-2^{-5})$	1.0 %	124	41	16.835	1.000
6.312, 10 Hz HPF	6.312	$(2^{-17})(1+2^{-2}+2^{-4})$	0.6 %	115	35	17.825	1.000
6.312, 3 kHz HPF	6.312	$(2^{-9})(1+2^{-1})(1+2^{-6})$	0.2 %	109	32	18.315	1.001
6.312, 60 kHz LPF	6.312	$(2^{-4})(1-2^{-4}-2^{-7})$	0.2 %	91	24	19.881	1.000
8.448, 10 Hz LPF	4.224	$(2^{-16})(1-2^{-5}+2^{-7})$	0.2 %	107	37	17.513	1.000
8.448, 100 Hz LPF	4.224	$(2^{-13})(1+2^{-2}-2^{-5})$	0.0 %	96	31	18.587	1.000
8.448, 0.2 Hz HPF	4.224	$(2^{-22})(1+2^{-2}-2^{-8})$	0.1 %	130	40	16.978	1.000
8.448, 20 Hz HPF	4.224	$(2^{-15})(1-2^{-5}+2^{-7})$	0.2 %	122	36	17.668	1.000
8.448, 3 kHz HPF	4.224	$(2^{-8})(1+2^{-3}+2^{-6})$	0.1 %	109	27	19.305	1.002
8.448, 80 kHz HPF	4.224	$(2^{-3})(1-2^{-3})(1+2^{-5})$	0.6 %	106	27	19.305	1.060
34.368, 10 Hz LPF	8.592	$(2^{-17})(1-2^{-5}-2^{-7})$	0.3 %	109	38	17.361	1.000
34.368, 100 Hz LPF	8.592	$(2^{-14})(1+2^{-3})(1+2^{-4})$	0.2 %	102	37	17.513	1.000
34.368, 0.2 Hz HPF	8.592	$(2^{-23})(1+2^{-2})(1-2^{-6})$	0.3 %	130	47	16.026	1.000
34.368, 100 Hz HPF	8.592	$(2^{-14})(1+2^{-3})(1+2^{-4})$	0.2 %	118	37	17.513	1.000
34.368, 10 kHz HPF	8.592	$(2^{-7})(1-2^{-4}-2^{-8})$	0.1 %	113	27	19.305	1.004
51.840, 10 Hz LPF	4.320	$(2^{-16})(1-2^{-4}+2^{-6})$	0.0 %	104	36	17.668	1.000
51.840, 100 Hz LPF	4.320	$(2^{-13})(1+2^{-3}+2^{-4})$	0.3 %	95	32	18.315	1.000
51.840, 0.2 Hz HPF	4.320	$(2^{-22})(1+2^{-2}-2^{-5})$	0.1 %	123	40	16.978	1.000
51.840, 10 Hz HPF	4.320	$(2^{-16})(1-2^{-4}+2^{-6})$	0.0 %	120	36	17.668	1.000
51.840, 500 Hz HPF	4.320	$(2^{-11})(1+2^{-1})(1-2^{-7})$	0.0 %	113	35	17.825	1.000
51.840, 20 kHz HPF	4.320	$(2^{-5})(1-2^{-4}-2^{-6})$	0.5 %	107	25	19.685	1.015
155.520, 10 Hz LPF	12.960	$(2^{-18})(1+2^{-2})(1+2^{-6})$	0.1 %	108	42	16.694	1.000
155.520, 100 Hz LPF	12.960	$(2^{-15})(1+2^{-1})(1+2^{-4})$	0.3 %	96	36	17.668	1.000
155.520, 0.2 Hz HPF	12.960	$(2^{-24})(1+2^{-1}+2^{-3})$	0.1 %	119	41	16.835	1.000
155.520, 10 Hz HPF	12.960	$(2^{-18})(1+2^{-2})(1+2^{-6})$	0.1 %	124	42	16.694	1.000
155.520, 1 kHz HPF	12.960	$(2^{-11})(1-2^{-7})$	0.0 %	94	34	21.277	1.000
155.520, 65 kHz HPF	12.960	$(2^{-5})(1-2^{-7})$	0.1 %	87	28	22.883	1.016

Table 5.3: Filters which may be implemented without adding delay elements in the loop, using FLEX8820s in the A-3 speed grade. The data path width b is 16 bits.

Filter	Decimated Data Rate (MHz)	Coefficient Implementation	Percent Frequency Error	LEs	A_b	Maximum Clock Rate (MHz)	C_{iir}
1.544, 10 Hz LPF	1.544	$(2^{-15})(1+2^{-2})(1+2^{-4})$	0.4 %	132	45	18.450	1.000
1.544, 100 Hz LPF	1.544	$(2^{-11})(1-2^{-3}-2^{-5})$	1.3 %	126	38	19.569	1.000
1.544, 0.2 Hz HPF	1.544	$(2^{-20})(1-2^{-3})(1-2^{-5})$	0.7 %	167	52	17.452	1.000
1.544, 10 Hz HPF	1.544	$(2^{-15})(1+2^{-2})(1+2^{-4})$	0.4 %	156	45	18.450	1.000
1.544, 8 kHz HPF	1.544	$(2^{-5})(1+2^{-6}+2^{-7})$	0.2 %	152	35	20.040	1.016
1.544, 40 kHz LPF	1.544	$(2^{-3})(1+2^{-3})(1+2^{-4})$	0.6 %	120	34	20.202	1.000
2.048, 10 Hz LPF	2.048	$(2^{-15})(1+2^{-8})$	0.1 %	110	47	21.231	1.000
2.048, 100 Hz LPF	2.048	$(2^{-12})(1+2^{-2}+2^{-7})$	0.1 %	131	38	19.569	1.000
2.048, 0.2 Hz HPF	2.048	$(2^{-21})(1+2^{-2})(1+2^{-5})$	0.2 %	166	52	17.452	1.000
2.048, 20 Hz HPF	2.048	$(2^{-14})(1+2^{-8})$	0.1 %	133	46	21.413	1.000
2.048, 700 Hz HPF	2.048	$(2^{-9})(1+2^{-4}+2^{-5})$	0.4 %	150	37	19.724	1.001
2.048, 18 kHz HPF	2.048	$(2^{-4})(1-2^{-3}-2^{-6})$	0.0 %	144	31	20.833	1.028
6.312, 10 Hz LPF	6.312	$(2^{-17})(1+2^{-2}+2^{-4})$	0.6 %	131	43	18.727	1.000
6.312, 100 Hz LPF	6.312	$(2^{-14})(1+2^{-1}+2^{-3})$	0.4 %	123	39	19.417	1.000
6.312, 0.2 Hz HPF	6.312	$(2^{-22})(1-2^{-3}-2^{-5})$	1.0 %	164	49	17.825	1.000
6.312, 10 Hz HPF	6.312	$(2^{-17})(1+2^{-2}+2^{-4})$	0.6 %	155	43	18.727	1.000
6.312, 3 kHz HPF	6.312	$(2^{-9})(1+2^{-1})(1+2^{-6})$	0.2 %	149	40	19.157	1.001
6.312, 60 kHz LPF	6.312	$(2^{-4})(1-2^{-4}-2^{-7})$	0.2 %	123	32	20.534	1.000
8.448, 10 Hz LPF	4.224	$(2^{-16})(1-2^{-5}+2^{-7})$	0.2 %	139	45	18.450	1.000
8.448, 100 Hz LPF	4.224	$(2^{-13})(1+2^{-2}-2^{-5})$	0.0 %	128	39	19.417	1.000
8.448, 0.2 Hz HPF	4.224	$(2^{-22})(1+2^{-2})$	0.2 %	129	48	20.921	1.000
8.448, 20 Hz HPF	4.224	$(2^{-15})(1-2^{-5}+2^{-7})$	0.2 %	162	44	18.587	1.000
8.448, 3 kHz HPF	4.224	$(2^{-8})(1+2^{-3}+2^{-6})$	0.1 %	149	35	20.040	1.002
8.448, 80 kHz HPF	4.224	$(2^{-3})(1-2^{-3})(1+2^{-5})$	0.6 %	146	35	20.040	1.060
34.368, 10 Hz LPF	8.592	$(2^{-17})(1-2^{-5}-2^{-7})$	0.3 %	141	46	18.315	1.000
34.368, 100 Hz LPF	8.592	$(2^{-14})(1+2^{-3})(1+2^{-4})$	0.2 %	134	45	18.450	1.000
34.368, 0.2 Hz HPF	8.592	$(2^{-23})(1+2^{-2}-2^{-6})$	0.6 %	166	49	17.825	1.000
34.368, 100 Hz HPF	8.592	$(2^{-14})(1+2^{-3})(1+2^{-4})$	0.2 %	158	45	18.450	1.000
34.368, 10 kHz HPF	8.592	$(2^{-7})(1-2^{-4}-2^{-8})$	0.1 %	153	35	20.040	1.004
51.840, 10 Hz LPF	4.320	$(2^{-16})(1-2^{-4}+2^{-6})$	0.0 %	136	44	18.587	1.000
51.840, 100 Hz LPF	4.320	$(2^{-13})(1+2^{-3}+2^{-4})$	0.3 %	127	40	19.157	1.000
51.840, 0.2 Hz HPF	4.320	$(2^{-22})(1+2^{-2}-2^{-5})$	0.1 %	163	48	17.953	1.000
51.840, 10 Hz HPF	4.320	$(2^{-16})(1-2^{-4}+2^{-6})$	0.0 %	160	44	18.587	1.000
51.840, 500 Hz HPF	4.320	$(2^{-11})(1+2^{-1})(1-2^{-7})$	0.0 %	153	43	18.727	1.000
51.840, 20 kHz HPF	4.320	$(2^{-5})(1-2^{-4}-2^{-6})$	0.5 %	147	33	20.367	1.015
155.520, 10 Hz LPF	12.960	$(2^{-18})(1+2^{-2})(1+2^{-6})$	0.1 %	140	50	17.699	1.000
155.520, 100 Hz LPF	12.960	$(2^{-15})(1+2^{-1})(1+2^{-4})$	0.3 %	128	44	18.587	1.000
155.520, 0.2 Hz HPF	12.960	$(2^{-24})(1+2^{-1}+2^{-3})$	0.1 %	159	49	17.825	1.000
155.520, 10 Hz HPF	12.960	$(2^{-18})(1+2^{-2})(1+2^{-6})$	0.1 %	164	50	17.699	1.000
155.520, 1 kHz HPF	12.960	$(2^{-11})(1-2^{-7})$	0.0 %	126	42	22.173	1.000
155.520, 65 kHz HPF	12.960	$(2^{-5})(1-2^{-7})$	0.1 %	119	36	23.585	1.016

Table 5.4: Filters which may be implemented without adding delay elements in the loop, using FLEX81188s in the A-2 speed grade. The data path width b is 24 bits. Coefficient Implementation, frequency error, and C_{iir} are the same as in the previous table.

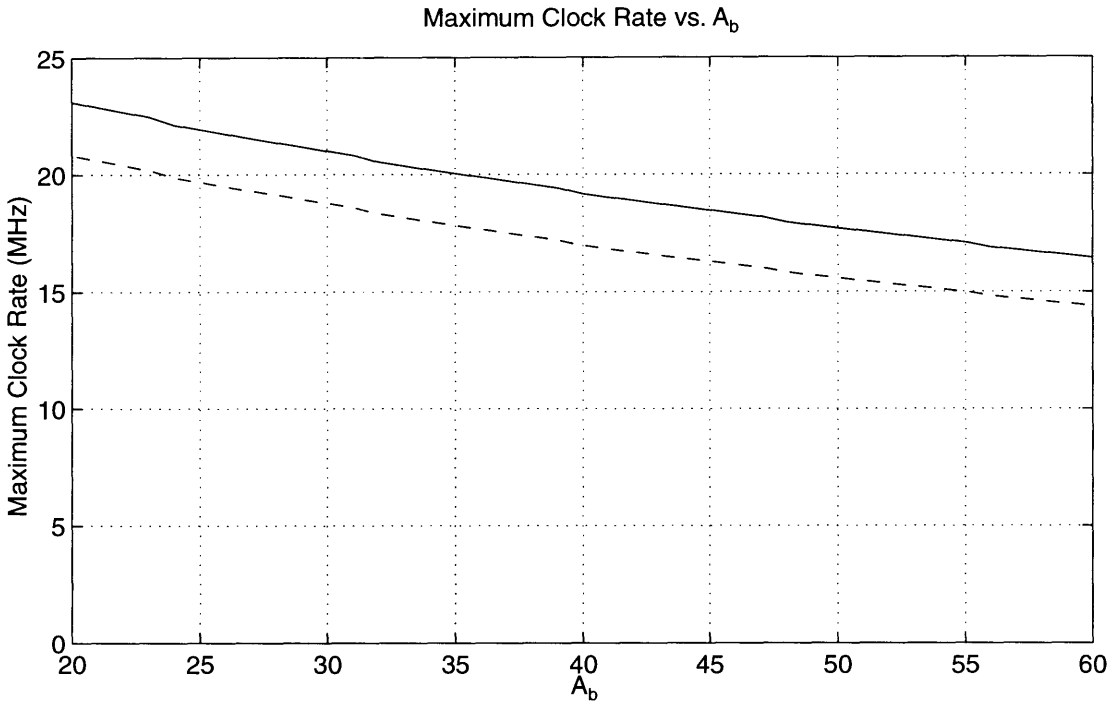

Figure 5-4: Maximum Clock Frequency, as a function of A_b, using the circuit with no extra loop delay elements. The solid line represents an A-2 speed grade FLEX81188, and the dashed line represents an A-3 speed grade FLEX8820.

not fit on one row of a FLEX8820. Instead, we chose other close coefficients ϵ_n which required fewer LEs to implement, without raising the frequency cutoff error above 2%.

Some of the filters in tables 5.3 and 5.4 used coefficient multipliers which only needed 1 adder. The propagation delays for these filters were lower by t_{next} than given by 5.1. The number of LEs required was also lower.

Recall from section 4.1 that highpass filters have an unwanted scaling factor of $C_{iir} = \frac{2}{2-\epsilon}$. This factor is given in the tables. The instrument should use this constant as a scaling factor to interpret the output. Specifically, the instrument should divide the answer by C_{iir} to correct for the extra gain in the highpass filters.

5.3 Filters with Loop Delays

The remaining filters will have to be implemented with extra delays in the loop, so that the incoming data rate will not violate the propagation delays. In section 4.2, we showed that we could correct the effects of extra loop delays by cascading the input with an FIR correction filter $h_{fix}[n]$. We then showed that for ϵ very small, the effects of this filter were negligible, so it could be eliminated. In this section, we will implement the remaining filters using extra loop delays of $k = 3$ or $k = 4$, without implementing correction filters $h_{fix}[n]$. (Extra loop delays of $k = 1$ or $k = 2$ were not useful, because of the large gap between the decimated data rates below 13 MHz and the ones above 34 MHz.) For all but a few of these filters, the resulting ripple will be below our ripple limit of 2%.

In figure 5-5(a), the input adder, first coefficient adder, second coefficient adder, and accumulator adder are b, $b+m+1$, $b+n+1$, and $b+s+n$ bits wide, respectively. Recall that each LE includes a register, so a register placed directly on the output of an adder indicates that these registers are used. An additional b bit wide register is required on the data path from the first coefficient adder to the the shifted input of the second coefficient adder, because the output of the first coefficient adder is registered. This additional register is implemented in separate LEs.

To prevent fanout of more than 4, we need $c = \lceil \frac{n+2}{4} \rceil - 1$ additional LEs buffering the MSB of the register storing the output of the input adder for the shifted input of the second coefficient adder, $\lceil \frac{4+m+c}{4} \rceil - 1$ LEs buffering the MSB of the input adder, and $\lceil \frac{s}{4} \rceil - 1$ LEs buffering the MSB of the second coefficient adder. This requires

$$q = \lceil \frac{4+m+c}{4} \rceil + c + \lceil \frac{s}{4} \rceil - 2$$

LEs. The total number of LEs for a filter of the form of figure 5-5(a) is

$$5b + s + m + 2n + 2 + q$$

Because the shifted input to the second coefficient adder comes from the first

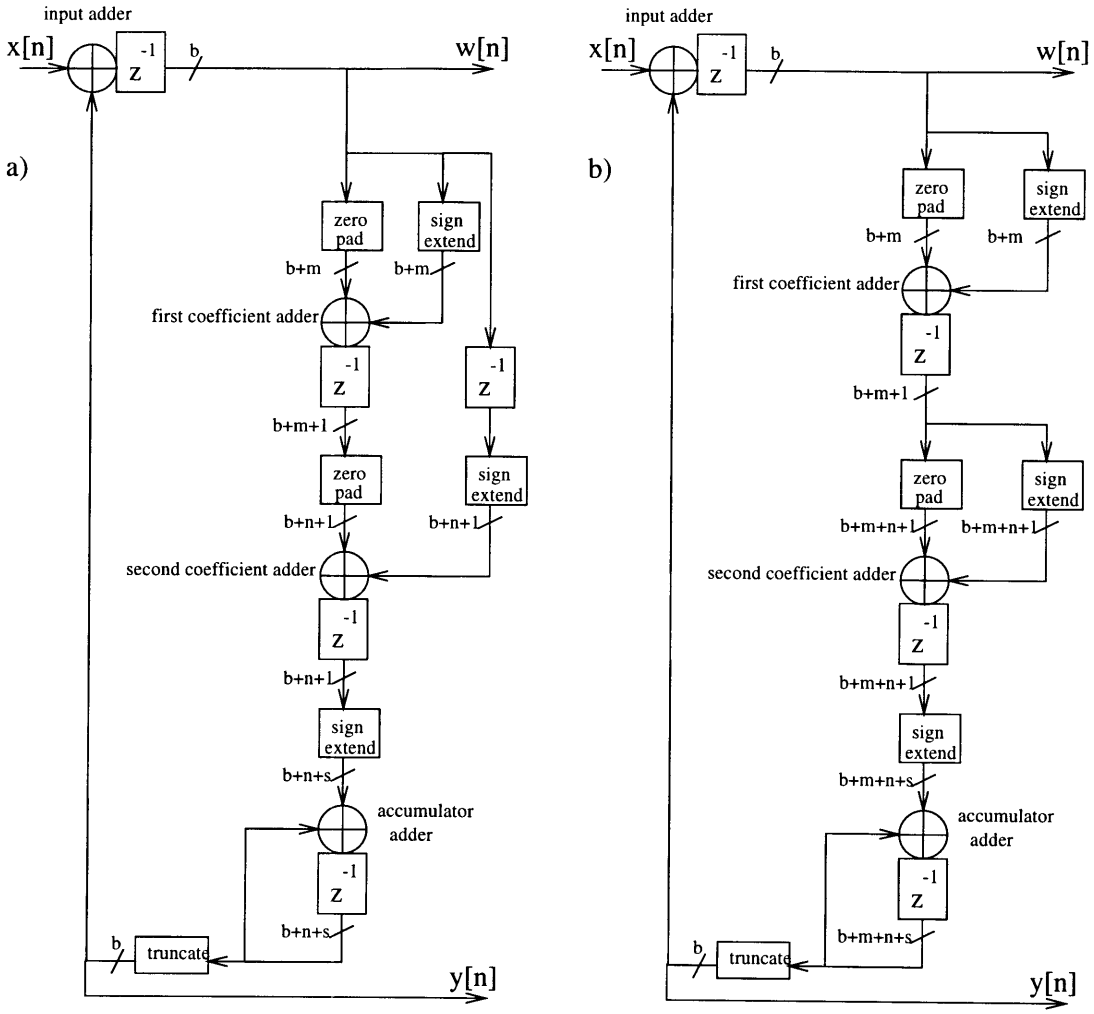

Figure 5-5: IIR filter structures with pipelined loops, including coefficient multipliers. "Zero pad" modules add zeros onto the LSP, "sign extend" modules add bits identical to the sign bit onto the MSP, and "truncate" modules truncate the LSP.

coefficient adder for the circuit in figure 5-5(b), this circuit does not need a register for the shifted input to the second coefficient adder. For this circuit,

$$q = \lceil \frac{3+m}{4} \rceil + \lceil \frac{1+n}{4} \rceil + \lceil \frac{s}{4} \rceil - 3$$

buffer LEs are needed to prevent excessive fanout. The total number of LEs for a filter of the form of figure 5-5(b) is

$$4b + s + 3m + 2n + 2 + q$$

The critical path of this circuit is the path from the LSB of the output of the second coefficient adder register to the MSB of the accumulator. The path incurs delay $t_{in} - t_{col}$ between the LSB of the second coefficient adder and the LSB of the accumulator. (t_{col} is not required since the whole filter will be located on one row.) If we define $A_b = b + s + n$ for the circuit in figure 5-5(a), or $A_b = b + s + n + m$ for the circuit in figure 5-5(b), then the carry chain incurs delay $T_{chain}(A_b)$. Finally, delay of t_{out} is incurred in reaching the register. If s, m, and n are small, it is possible for the input adder carry chain to be the critical path, since an additional delay of t_{col} is incurred by the input signal, which comes from another row. However, this does not occur for any of our filters. Therefore, the maximum propagation delay for this circuit is the delay for an adder with an input from the same row, which is

$$t_{in} - t_{col} + T_{chain}(A_b) + t_{out} \tag{5.2}$$

Figure 5-6 plots the maximum clock frequency resulting from expression 5.2 against A_b. Tables 5.5 and 5.6 show that a handful of filters may be implemented with $k = 3$, for $b = 16$ bit data paths in A-3 speed grade FLEX8820s, and $b = 24$ bit data paths in A-2 speed grade FLEX8820s, respectively. The ripple was calculated by MATLAB, by taking the maximum value of the magnitude of difference between the ideal discrete-time filter response without extra delay elements and the filter response with extra delay elements, and dividing by the magnitude of the ideal filter. Note, how-

ever, that the 250 kHz highpass filter for the 622.08 MHz base data rate has more than 2% ripple. This will be dealt with in section 5.5.

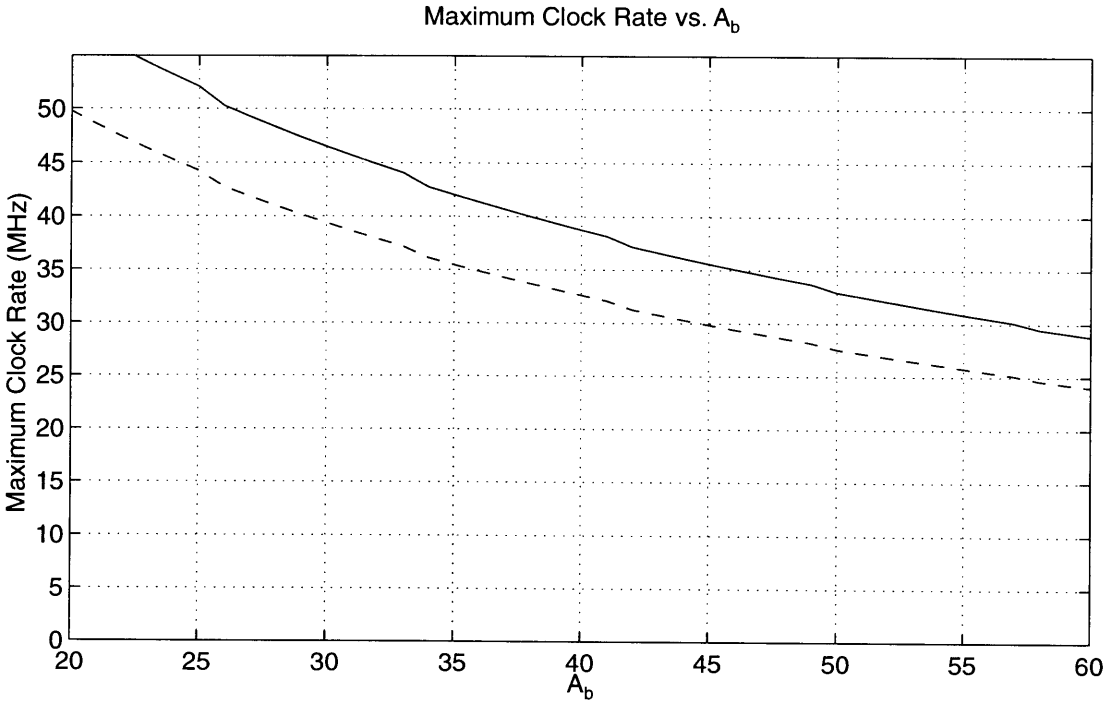

Figure 5-6: Maximum Clock Frequency, as a function of A_b, for an adder with input from the same row. The solid line represents an A-2 speed grade FLEX81188, and the dashed line represents an A-3 speed grade FLEX8820.

Filter	Dec. Data Rate (MHz)	Coefficient Implementation	Freq. Error	LEs	A_b	Max. Clock Rate (MHz)	Peak Ripple	C_{iir}
139.264, 100 Hz LPF	34.816	$(2^{-16})(1+2^{-3}+2^{-4})$	0.4 %	114	36	34.843	0.01%	1.000
139.264, 200 Hz HPF	34.816	$(2^{-15})(1+2^{-3}+2^{-4})$	0.4 %	113	35	35.461	0.01%	1.000
139.264, 10 kHz HPF	34.816	$(2^{-9})(1-2^{-4}-2^{-6})$	0.4 %	112	31	38.610	0.55%	1.001
622.080, 250 kHz HPF	38.880	$(2^{-5})(1+2^{-3}-2^{-8})$	0.1 %	111	29	40.161	**12.87%**	1.020

Table 5.5: Filters which may be implemented with $k = 3$ delay elements in the loop, using FLEX8820s in the A-3 speed grade. The data path width b is 16 bits. Boldface indicates a missed specification.

Pipelining the carry chains of the accumulators in figure 5-5 results in the circuits shown in figure 5-7. In 5-7(a), the input adder, first coefficient adder, and second coefficient adder are b, $b+m+1$, and $b+n+1$ bits wide, respectively. A separate b bit wide register stores the output of the input adder, which is needed by the shifted

Filter	Dec. Data Rate (MHz)	Coefficient Implementation	Freq. Error	LEs	A_b	Max. Clock Rate (MHz)	Peak Ripple	C_{iir}
139.264, 100 Hz LPF	34.816	$(2^{-16})(1+2^{-3}+2^{-4})$	0.4 %	154	44	36.101	0.01%	1.000
139.264, 200 Hz HPF	34.816	$(2^{-15})(1+2^{-3}+2^{-4})$	0.4 %	153	43	36.630	0.01%	1.000
139.264, 10 kHz HPF	34.816	$(2^{-9})(1-2^{-4}-2^{-6})$	0.4 %	152	39	39.370	0.55%	1.001
622.080, 250 kHz HPF	38.880	$(2^{-5})(1+2^{-3}-2^{-8})$	0.1 %	151	37	40.650	**12.87%**	1.020

Table 5.6: Filters which may be implemented with $k = 3$ delay elements in the loop, using FLEX81188s in the A-2 speed grade. The data path width b is 24 bits. Coefficient Implementation, frequency error, ripple, and C_{iir} are the same as in the previous table. Boldface indicates a missed specification.

input of the second coefficient adder. The first stage accumulator is $b+n+1$ bits wide, but an additional LE is required for a register for the carry-out signal. Another register stores the MSB of the second coefficient adder, for sign extension in the second stage accumulator. The second stage accumulator is $s-1$ bits wide. If $b > s-1$, a separate $b-s+1$ bit wide register is used to store bits from the first stage accumulator which are used by the input adder.

To prevent fanout of more than 4, we need $c = \lceil \frac{n+2}{4} \rceil - 1$ additional LEs buffering the MSB of the register storing the output of the input adder for the shifted input of the second coefficient adder, $\lceil \frac{4+m+c}{4} \rceil - 1$ LEs buffering the MSB of the input adder, $d = \lceil \frac{s-1}{4} \rceil - 1$ additional LEs in the register storing the MSB of the second coefficient adder for the second stage accumulator, and $\lceil \frac{1+d}{4} \rceil - 1$ LEs buffering the MSB of the second coefficient adder. This requires

$$q = c + \lceil \frac{4+m+c}{4} \rceil + d + \lceil \frac{1+d}{4} \rceil - 2$$

LEs. The total number of LEs for a filter of the form of figure 5-7(a) is

$$5b + s + m + 2n + \max(b-s+1, 0) + 4 + q$$

Because the shifted input to the second coefficient adder comes from the first coefficient adder for the circuit in figure 5-7(b), this circuit does not need a register

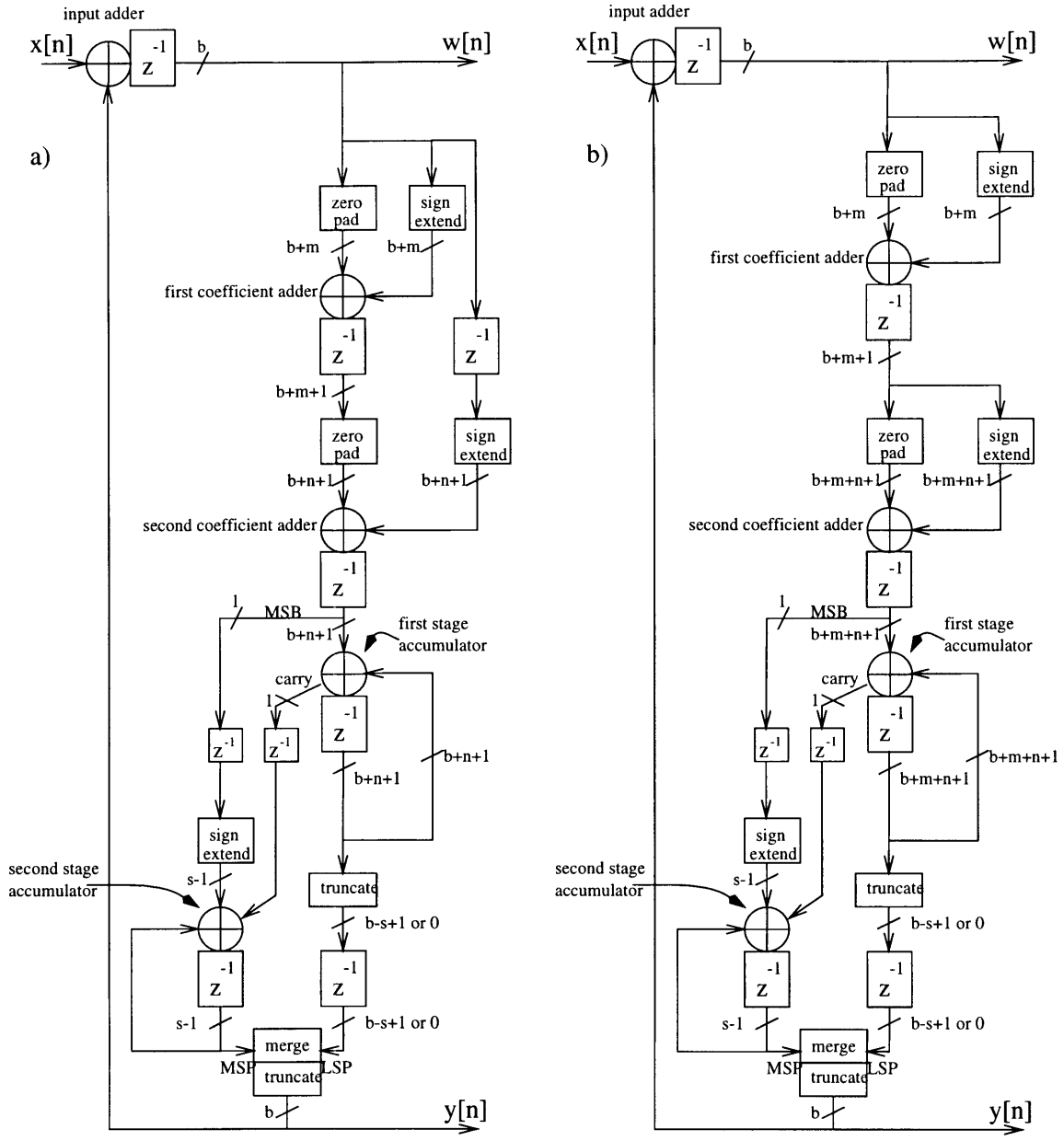

Figure 5-7: IIR filter structures with pipelined loops and accumulators, including coefficient multipliers. "Zero pad" modules add zeros onto the LSP, "sign extend" modules add bits identical to the sign bit onto the MSP, and "truncate" modules truncate the LSBs.

for the shifted input to the second coefficient adder. For this circuit,

$$q = \lceil \frac{3+m}{4} \rceil + d + \lceil \frac{1+d}{4} \rceil - 2$$

buffer LEs are needed to prevent excessive fanout. The total number of LEs for a filter of the form of figure 5-7(b) is

$$4b + s + 3m + 2n + \max(b - s + 1, 0) + 4 + q$$

There are three possible critical paths in these circuits, depending on the values of b, s, m, and n. If $s - 1 > b + n + 2$ for 5-7(a), or if $s - 1 > b + m + n + 2$ for 5-7(b), then the critical path may follow the carry chain of the second stage accumulator. If not, then the critical path may follow the carry chain of the first stage accumulator, which includes an extra carry chain delay for the carry-out signal. In either case, because the input to the circuit comes from a different row, the carry chain of the input adder may be the critical path, even though it will always be shorter than the carry chains of all of the other adders.

We therefore define $A_b = \max(s - 1, b + n + 2)$ for 5-7(a), or $A_b = \max(s - 1, b + m + n + 2)$ for 5-7(b). The maximum propagation delay time is then the propagation delay for an adder with an input from the same row, given by expression 5.2, which we used for $k = 3$. Likewise, the maximum clock frequency allowed by the accumulator may be found using the new A_b in the plot of figure 5-6, as for $k = 3$.

For the input adder, since the output of the MSB may need to be buffered (before the register) to satisfy the fanout conditions of the shifted input to the second coefficient adder, we define $A_c = b + \lceil \frac{4+m+c}{4} \rceil - 1$. Because the input signal comes from a different row, the maximum propagation delay allowed by the input adder is the delay for an adder with an input from a different row, which is

$$t_{in} + T_{chain}(A_c) + t_{out} \tag{5.3}$$

Figure 5-8 plots the maximum frequency resulting from expression 5.3 against A_c.

The maximum frequency allowed by the circuit is the lesser of the frequency allowed by the input adder, and the frequency allowed by the accumulator adders.

Tables 5.7 and 5.8 list the $b = 16$ and $b = 24$ implementations, respectively. For the 44.736, 0.2 Hz HPF filter and the 44.736, 20kHz HPF filter in table 5.7, I chose alternate coefficients ϵ_n other than the closest value, to reduce A_b and raise the maximum clock rate above the decimated data rate. For 4 filters in table 5.8 (44.736, 100 Hz LPF; 44.736, 0.2 Hz HPF; 139.264, 10 Hz LPF; and 622.08, 100 Hz LPF), I also chose an alternate coefficient ϵ_n, to fit the filter within the 168 LEs available on a single row. Unfortunately, several filters in the table are still unacceptable. These filters will be corrected in section 5.5. Note that some of these filters use a coefficient requiring only one adder. For these filters, the second coefficient adder in figure 5-7(b) should be replaced with a register, to keep the number of extra delays the same. Reducing the number of delays would change the value of the coefficient required.

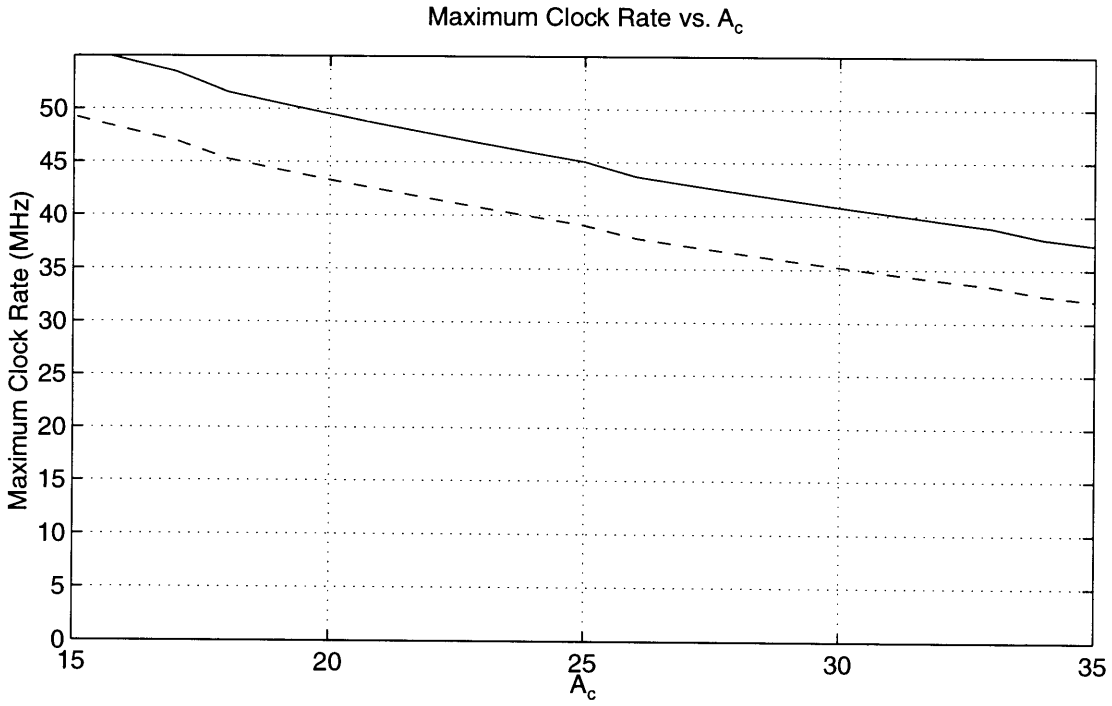

Figure 5-8: Maximum Clock Frequency as a function of A_c, for an adder with an input from a different row. The solid line represents an A-2 speed grade FLEX81188, and the dashed line represents an A-3 speed grade FLEX8820.

Filter	Dec. Data Rate (MHz)	Coefficient Implementation	Freq. Error	LEs	A_b	Max. Clock Rate (MHz)	Peak Ripple	C_{iir}
44.736, 10 Hz LPF	44.736	$(2^{-20})(1+2^{-1})(1-2^{-6})$	0.3 %	109	25	**44.248**	0.00%	1.000
44.736, 100 Hz LPF	44.736	$(2^{-16})(1-2^{-4}-2^{-6})$	0.2 %	125	24	45.249	0.01%	1.000
44.736, 0.2 Hz HPF	44.736	$(2^{-25})(1-2^{-4})$	0.5 %	106	24	45.249	0.00%	1.000
44.736, 10 Hz HPF	44.736	$(2^{-20})(1+2^{-1})(1-2^{-6})$	0.3 %	109	25	**44.248**	0.00%	1.000
44.736, 30 kHz HPF	44.736	$(2^{-8})(1+2^{-4})$	0.4 %	84	22	47.393	1.71%	1.002
44.736, 400 kHz LPF	44.736	$(2^{-5})(1+2^{-2})(1+2^{-3})$	1.0 %	99	23	46.296	**25.72%**	1.000
139.264, 10 Hz LPF	34.816	$(2^{-19})(1-2^{-4}+2^{-7})$	0.1 %	131	25	44.248	0.00%	1.000
139.264, 0.2 Hz HPF	34.816	$(2^{-25})(1+2^{-2})(1-2^{-5})$	0.0 %	117	25	44.248	0.00%	1.000
622.080, 10 Hz LPF	38.880	$(2^{-19})(1-2^{-3})(1-2^{-5})$	0.0 %	113	26	42.735	0.00%	1.000
622.080, 100 Hz LPF	38.880	$(2^{-16})(1+2^{-4}-2^{-8})$	0.0 %	130	26	42.735	0.01%	1.000
622.080, 0.2 Hz HPF	38.880	$(2^{-25})(1+2^{-3})(1-2^{-5})$	0.5 %	120	26	42.735	0.00%	1.000
622.080, 10 Hz HPF	38.880	$(2^{-19})(1-2^{-3})(1-2^{-5})$	0.0 %	113	26	42.735	0.00%	1.000
622.080, 5 kHz HPF	38.880	$(2^{-10})(1-2^{-3})(1-2^{-4})$	0.5 %	106	25	44.248	0.32%	1.000

Table 5.7: Filters which may be implemented with $k = 4$ delay elements in the loop, using FLEX8820s in the A-3 speed grade. The data path width b is 16 bits. Boldface indicates a missed specification.

Filter	Dec. Data Rate (MHz)	Coefficient Implementation	Freq. Error	LEs	A_b	Max. Clock Rate (MHz)	Peak Ripple	C_{iir}
44.736, 10 Hz LPF	44.736	$(2^{-20})(1+2^{-1})(1-2^{-6})$	0.3 %	146	33	**44.053**	0.00%	1.000
44.736, 100 Hz LPF	44.736	$(2^{-16})(1-2^{-4})(1-2^{-5})$	1.3 %	153	35	**42.017**	0.01%	1.000
44.736, 0.2 Hz HPF	44.736	$(2^{-25})(1-2^{-4})$	0.5 %	138	30	45.045	0.00%	1.000
44.736, 10 Hz HPF	44.736	$(2^{-20})(1+2^{-1})(1-2^{-6})$	0.3 %	146	33	**44.053**	0.00%	1.000
44.736, 30 kHz HPF	44.736	$(2^{-8})(1+2^{-4})$	0.4 %	116	30	45.872	1.71%	1.002
44.736, 400 kHz LPF	44.736	$(2^{-5})(1+2^{-2})(1+2^{-3})$	1.0 %	139	31	45.045	**25.72%**	1.000
139.264, 10 Hz LPF	34.816	$(2^{-19})(1-2^{-4})$	0.9 %	130	30	45.045	0.00%	1.000
139.264, 0.2 Hz HPF	34.816	$(2^{-25})(1+2^{-2})(1-2^{-5})$	0.0 %	149	33	44.053	0.00%	1.000
622.080, 10 Hz LPF	38.880	$(2^{-19})(1-2^{-3})(1-2^{-5})$	0.0 %	151	34	42.735	0.00%	1.000
622.080, 100 Hz LPF	38.880	$(2^{-16})(1+2^{-4})$	0.3 %	126	30	45.045	0.01%	1.000
622.080, 0.2 Hz HPF	38.880	$(2^{-25})(1+2^{-3})(1-2^{-5})$	0.5 %	152	34	42.735	0.00%	1.000
622.080, 10 Hz HPF	38.880	$(2^{-19})(1-2^{-3})(1-2^{-5})$	0.0 %	151	34	42.735	0.00%	1.000
622.080, 5 kHz HPF	38.880	$(2^{-10})(1-2^{-3})(1-2^{-4})$	0.5 %	146	33	44.053	0.32%	1.000

Table 5.8: Filters which may be implemented with $k = 4$ delay elements in the loop, using FLEX81188 chips in the A-2 speed grade. The data path width b is 24 bits. Several coefficient implementations, and therefore frequency error and ripple, have changed from the previous table. C_{iir} is the same as in the previous table, however. Boldface indicates a missed specification.

5.4 FIR Lowpass Filters

For most of the lowpass filters for the "bandpass" bands, we require third-order butterworth filters. However, emulating the magnitude rolloff of a third-order butterworth filter in the passband and attenuating more heavily in the stopband is also sufficient. For those data rates which used the decimator, we also need to correct for passband attenuation. 16-tap symmetric FIR filters with 8 bit coefficients were found to be adequate. Altera offers programs which will implement symmetric FIR filters on FLEX8000s [1, 7]. With a 16 bit data path, such a filter will fit on a FLEX81188, or with a 24 bit data path, such a filter may be split between two FLEX8820s [4].

Because MATLAB functions provide a simple way to implement optimal equiripple filters with multiple bands, we may use a "brute-force" method to emulate the third-order butterworth passband rolloff. A piecewise linear ideal response is made to follow a path emulating the butterworth filter, while correcting the effects of the stopband attenuation of the decimator. The corners between the the line segments are made to be arbitrarily narrow "don't-care" bands. A transition band separates the piecewise linear passband from the stopband. By varying the transition bandwidth, the passband edge of the final piecewise linear passband segment, and the weighting functions, we may find an FIR filter which is satisfactory. The MATLAB code used to implement this may be found in appendix C, and the resulting frequency responses may be found in appendix B. Table 5.9 gives the coefficients for the filter. Because of uncertainty about the capabilities of the Altera software, the largest coefficient is normalized to the largest 8-bit twos-complement number, 127/128, rather than to 1. $C_{fir} = \sum_{j=0}^{15} a_j$ gives a scaling constant which should be used to interpret the output.

5.5 Troublesome Filters

We may raise the maximum clock rate of several of the filters in tables 5.7 and 5.8 by rounding a few LSBs from the second coefficient adder. We prefer not to use truncation here, because that would introduce an offset equal to half the value of

Base Data Rate	a_0,a_{15}	a_1,a_{14}	a_2,a_{13}	a_3,a_{12}	a_4,a_{11}	a_5,a_{10}	a_6,a_9	a_7,a_8	C_{fir}
2.048 MHz	-4/128	-10/128	-13/128	-5/128	19/128	58/128	100/128	127/128	4.250
8.448 MHz	-4/128	-9/128	-12/128	-5/128	19/128	58/128	100/128	127/128	4.281
34.368 MHz	-5/128	-10/128	-12/128	-4/128	22/128	60/128	101/128	127/128	4.359
51.84 MHz	-5/128	-10/128	-12/128	-4/128	22/128	61/128	101/128	127/128	4.375
139.264 MHz	-2/128	-7/128	-12/128	-9/128	11/128	50/128	96/128	127/128	3.969
155.52 MHz	-2/128	-7/128	-12/128	-9/128	11/128	50/128	96/128	127/128	3.969
622.08 MHz	1/128	-1/128	-7/128	-14/128	-7/128	28/128	83/128	127/128	3.281

Table 5.9: Symmetric FIR filter coefficients. $C_{fir} = \sum_{j=0}^{15} a_j$ gives the scale constant for each filter.

the LSB of the part not discarded by truncation. This in turn will produce a small offset in $w[n]$. For the lowpass filters, this does not matter; however rounding will not require extra hardware as long as the number of bits to be removed by rounding is less than n.

These filters would be implemented using the circuits in figure 5-7, with minor modifications. As long as the number of bits to be removed by rounding is less than n, then the rounded part of the adder output is due only to the shifted input. The MSB of the discarded part may be used in place of the carry-in signal to the LSB of the remaining part, if we are adding the shifted term. If we are subtracting the shifted term, the MSB of the discarded part should be inverted, and used as the borrow-in signal to the LSB of the remaining part.

For each LSB rounded, the second coefficient adder and the first stage accumulator are shortened by one carry chain. Table 5.10 shows the number of bits which should be rounded to adequately raise the maximum clock rate. Rounding a multiplier can cause the coefficient ϵ to effectively change values when the amplitude of the input is very small. However, since we are only rounding 1 or 3 bits, this should not be a problem. Note that for all cases, n is greater than the number of bits discarded.

For the 44.736 MHz data rate, the lowpass filter in the "bandpass" band is required to be a first-order lowpass filter with a cutoff of 400 kHz, rather than a third-order filter. Because the cutoff of this filter is not as small, relative to the data rate, as the other filters, the approximations we made relying on ϵ being very small are not as good for this filter. But because of the high data rate, we needed to insert a few loop delays. This caused a horrible ripple of 25.72%, far above the maximum acceptable

Filter	b	Speed Grade	Coefficient Implementation	Bits Rounded	LEs	A_b	Maximum Clock Rate (MHz)
44.736, 10 Hz LPF	16	A-3	$(2^{-20})(1+2^{-1})(1-2^{-6})$	1	106	24	45.249
44.736, 10 Hz HPF	16	A-3	$(2^{-20})(1+2^{-1})(1-2^{-6})$	1	106	24	45.249
44.736, 10 Hz LPF	24	A-2	$(2^{-20})(1+2^{-1})(1-2^{-6})$	1	143	32	44.843
44.736, 100 Hz LPF	24	A-2	$(2^{-16})(1-2^{-4})(1-2^{-5})$	3	144	32	44.843
44.736, 10 Hz HPF	24	A-2	$(2^{-20})(1+2^{-1})(1-2^{-6})$	1	143	32	44.843

Table 5.10: Filters implemented with rounded coefficients.

ripple of 2%.

Recall from section 4.2 that we may cascade our IIR filter with $h_{fix}[n]$ to correct the effects of the loop delay. Furthermore, since we do not need to implement a third-order filter and since we do not need to correct for passband attenuation of the decimator, we do not use an FIR filter for this data rate. The IIR lowpass filter for the "bandpass" band only consumes one row. Therefore, we have 5 unused rows on the FLEX81188, if we are using 16-bit data paths, or 7 unused rows on the two FLEX8820s, if we are using 24-bit data paths. So we may implement the approximation to $h_{fix}[n]$ given in equation 4.7 on the chip(s) normally used for the FIR filter. Figure 5-9 shows a circuit which will accomplish this, using 24 bit data paths. Note, however, that we did not include the factor of $\frac{\epsilon}{\epsilon'}$. Besides that, the factor ϵ_m cannot be implemented exactly, so $\frac{\epsilon}{\epsilon'}$ is not quite right anyway. Therefore, the instrument will have to divide by $C_{iir} = .8467 \approx \frac{\epsilon'}{\epsilon}$ in this case. This number was determined experimentally. The resulting peak ripple is 1.536%.

The lower portion of the circuit uses 168 LEs, including fanout buffering, so it exactly fits on one row. The longest carry chain is $A_b = 31$, so figure 5-6 shows that the lower portion of the circuit will function at the required 44.736 MHz data rate. The carry chains in the upper portion of the circuit are pipelined, since the 34 bit carry chain was too long even for signals from the same row. With a maximum carry chain length of $A_c = 18$, the upper portion will easily function at 44.736 MHz with inputs from the other row, as seen by figure 5-8. The upper portion of the circuit consumes an additional 153 LEs—most of another row.

For the 16 bit case, our data path is 8 bits less wide. This means the longest carry chain in the lower portion is $A_b = 23$, and in the upper portion is $A_c = 14$. Figures

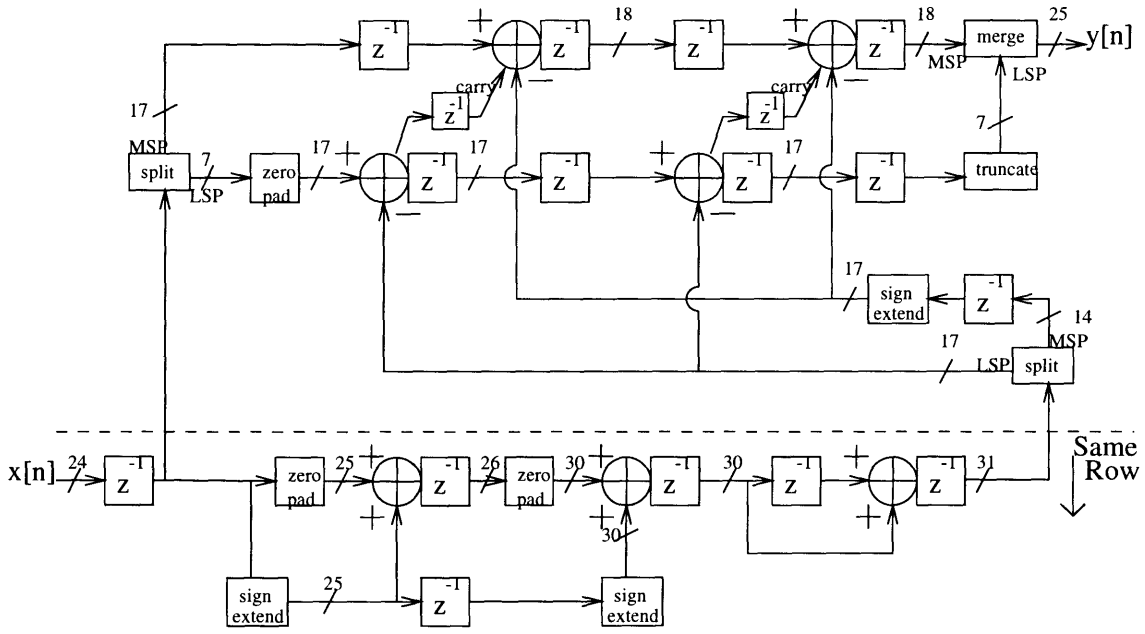

Figure 5-9: Filter to correct ripple of the 400kHz lowpass filter for the 44.736 MHz data rate. This implements the impulse response $h[n] = \delta[n-5] - \epsilon_m \sum_{m=1}^{k} \delta[n-m-5] \approx \frac{\epsilon'}{\epsilon} h_{fix}[n-5]$. Here, $\epsilon_m = (2^{-5})(1+2^{-1}+2^{-4}) = 0.0489$. The portion of the circuit below the dashed line should be placed on the same row.

5-6 and 5-8 show that this circuit will function at 44.736 MHz.

Because the cutoff frequency of the 250 kHz highpass filter for the 622.08 MHz base data rate is also not as small relative to the decimated sample rate of 38.88 MHz as other filters, this filter also had excessive ripple. Since we are already using the FIR filter to correct the effects of passband attenuation, we may include the shape of the ripple in this algorithm, and correct the ripple. Thus we get a separate FIR filter which should only be used with the 250 kHz highpass filter. Figure B-8 plots the total frequency response resulting from using the FIR filter coefficients given by table 5.11. An additional scaling factor was also used, bringing the correct scaling constant for the IIR and FIR filters to $C = 3.213$.

Base Data Rate	a_0,a_{15}	a_1,a_{14}	a_2,a_{13}	a_3,a_{12}	a_4,a_{11}	a_5,a_{10}	a_6,a_9	a_7,a_8	C
622.08 MHz	1/128	-1/128	-8/128	-16/128	-11/128	24/128	82/128	127/128	3.213

Table 5.11: FIR filter coefficients for use with the 250 kHz highpass filter for the 622.08 MHz base data rate.

5.6 Input Functions

Recall from section 3.1 that for data rates using the decimator, three first-differencers were to be implemented in Altera hardware, to complete the decimator. There are 21 bits of output from the decimator, so the first-differencers will be 21 bits wide. To handle signal "wrapping" as described in section 4.3, another first-differencer and an accumulator will be required to "unwrap" the input by r bits. (Because r may be small, and because of the Altera architecture, the hardware-saving technique developed in Appendix A and used in section 3.2 gives little or no savings in complexity here.) We will be unable to fit all of the input functions on one row; therefore the maximum frequency of the adders except for the first first-differencer is given in table 5-8. If, for example, r may be up to 12, we will need to pipeline the carry chain of the accumulator, which will be $21 + r$ bits wide. We will also, then, pipeline the carry chains of all of the adders implementing the input functions. Pipelining with 16 bits in the LSP, as in figure 5-10, satisfies all timing constraints.

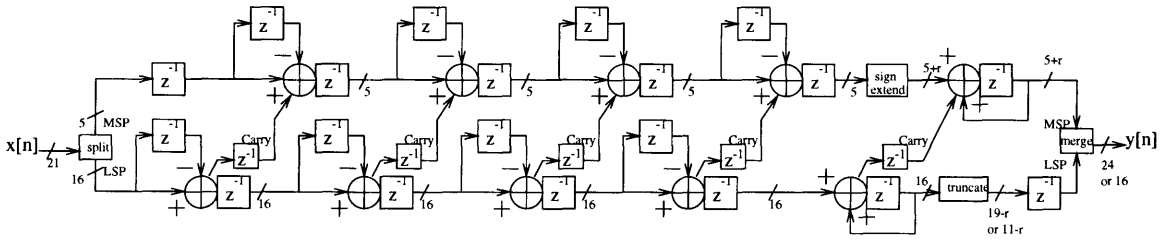

Figure 5-10: Input functions: Three first-differencers complete the decimation, and a first-differencer followed by an accumulator "unwraps" the input by r bits.

Each first-differencer requires one subtracter and one additional register. Each first-differencer consumes 21 LEs for the subtracter, 21 LEs for the additional register, and 1 more LE to register the carry chain, for a total of 43 LEs. The final first differencer, used for "unwrapping" the signal, may need 3 additional LEs to buffer the MSB, which will be sign-extended by r bits. Together, the four first-differencers consume 175 LEs. An additional 5 LEs are required for a register to initiate the pipelining offset. The accumulator requires $21 + r$ LEs, plus 1 additional LE to register the carry chain. Finally, $19 - r$ or $11 - r$ LEs are required for a register

to undo the offset, for $b = 24$ bits or $b = 16$ bits, respectively. Therefore, the input functions require a total of 221 LEs or 213 LEs, for $b = 24$ or $b = 16$ bits, respectively.

The pipeline offset register, the first 3 first-differencers, and the LSP of the fourth first-differencer may fit on one row, together consuming 167 of the 168 LEs on a row. For $b = 16$ bits, there is plenty of room on the rows with the three IIR filters for the remaining 46 LEs. Therefore, all of our functions will fit in the four rows of a FLEX8820, with $b = 16$. However, with $b = 24$, our other IIR filters consumed most of the rows they were on. There are not enough LEs to implement the remaining 54 LEs. Therefore, we will need to use a FLEX81188, the next larger size, with 6 rows.

For those data rates not using the decimator, we only need to "unwrap" the input by r bits. This requires only the final first-differencer and the accumulator from figure 5-10. Furthermore, we will only have B_x bits of input from the chip, where B_x is given in table 2.2, so the input data path will be B_x bits wide, rather than 21 bits wide.

Chapter 6

Conclusion

This thesis presented the design of a filtering system for an analog microchip. The filtering system included a decimator to lower the sample rate of the signal coming from the chip, and filters to isolate desired frequency bands for the user.

Chapter 2 laid the analytical foundation for the decimator. A close look at the Cascaded Integrator Comb decimator showed that the computation could be reduced by factoring out an intermediate stage which attenuated the first stopband (as well as the other odd-numbered stopbands). We chose to use $K = 3$ integrator-comb stages and $L = 1$ intermediate stage, to attenuate our stopband by at least 60dB. The unboundedness of the signal was retained without loss of information, by partially "unwrapping" the input signal, and using a wide enough data path.

Chapter 3 outlined the circuitry necessary for the portion of the decimator which was implemented on the microchip. The comb, or first-differencer, stages, were placed in the FLEX8000 hardware, saving on-chip resources. An unwrapping bit extender was developed and optimized. The decimator required 746 logic trees.

Chapter 4 laid the analytical foundation for the first-order recursive filters. A filter was developed to correct for delay elements in the large recursive loop. This filter was then shown to be negligible for filters with low cutoffs relative to the sample rate, such as most of the required filters. Criteria for dealing with "wrapping" effects were developed. Coefficients which could be implemented using no more than two adders were listed, and shown to be adequate.

Chapter 5 outlined the implementation of the filters in the Altera FLEX 8000 architecture. Most of the filters were implemented directly, without extra delays in the large recursive loop. However, the high decimated sample rates of the 44.736 MHz, 139.264 MHz, and 622.08 MHz base data rates necessitated $k = 3$ or $k = 4$ extra delay elements in the loop. A few filters still had too much delay, so coefficient outputs were rounded by 1 or 3 bits to shorten carry chains. One filter at the 44.736 MHz base data rate had too much ripple, due to a cutoff frequency that was not low enough to justify approximations developed in chapter 4. A stronger approximation, utilizing an approximate correction filter, corrected the ripple. Another filter at the 622.08 MHz data rate also had too much ripple. The FIR filter was modified to correct this ripple, as well as the passband attenuation of the decimator. For all data rates utilizing the decimator, FIR filters satisfactorily correcting the decimator passband attenuation were presented. Finally, the portion of the decimator reserved for the FLEX8000 architecture was implemented. Using 24 bit data paths, the entire system fit on one FLEX81188 and two FLEX8820s, using the A-2 speed grade. Using 16 bit data paths, the system fit on one FLEX8820 and one FLEX81188, using the A-3 speed grade.

Appendix A

Unwrapping Bit Extender Optimization

In section 3.2, we were able to save circuitry by removing the LSP of the accumulator used to "unwrap" the input signal. Here, we show that this was valid.

Figure A-1 shows a single bit of a first-differencer followed by an accumulator. Mixing logic notation with signal processing notation, we refer to the stored value of A and S as Az^{-1} and Sz^{-1}, respectively.

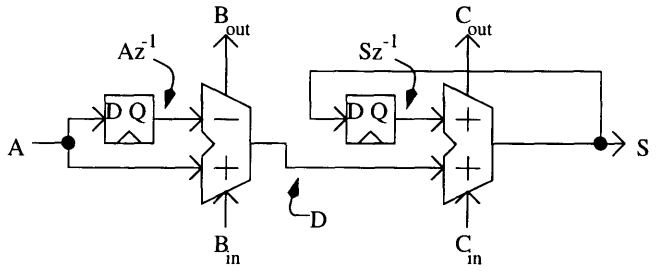

Figure A-1: A single bit of a first-differencer followed by an accumulator.

The logical expressions for adders and subtracters give us:

$$B_{out} = A \cdot A\bar{z}^{-1} + A \cdot B_{in} + A\bar{z}^{-1} \cdot B_{in}$$

$$D = A \cdot Az^{-1} \cdot \bar{B_{in}} + \bar{A} \cdot A\bar{z}^{-1} \cdot \bar{B_{in}} + \bar{A} \cdot Az^{-1} \cdot B_{in} + A \cdot A\bar{z}^{-1} \cdot B_{in}$$

$$C_{out} = D \cdot Sz^{-1} + D \cdot C_{in} + Sz^{-1} \cdot C_{in}$$

$$S = D \cdot S\bar{z}^{-1} \cdot \bar{C_{in}} + \bar{D} \cdot Sz^{-1} \cdot \bar{C_{in}} + \bar{D} \cdot S\bar{z}^{-1} \cdot C_{in} + D \cdot Sz^{-1} \cdot C_{in}$$

Suppose that $Sz^{-1} = Az^{-1}$, i.e. the previous output of the accumulator is the same as the previous input to the first-differencer. Also suppose that $C_{in} = \bar{B_{in}}$. Then we may eliminate Sz^{-1} and C_{in} from the expressions, yielding

$$C_{out} = D \cdot Az^{-1} + D \cdot \bar{B_{in}} + Az^{-1} \cdot \bar{B_{in}}$$

$$S = D \cdot A\bar{z}^{-1} \cdot B_{in} + \bar{D} \cdot Az^{-1} \cdot B_{in} + \bar{D} \cdot A\bar{z}^{-1} \cdot \bar{B_{in}} + D \cdot Az^{-1} \cdot \bar{B_{in}}$$

Substituting D into the expression for C_{out} yields

$$C_{out} = Az^{-1} \cdot B_{in} + \bar{A} \cdot \bar{B_{in}} + \bar{A} \cdot Az^{-1} = \bar{B_{out}}$$

Substituting D into the expression for S yields

$$S = A$$

When this circuit begins operation, then, we may set $Az^{-1} = Sz^{-1} = 0$ for every bit in the unwrapping bit-extender. For the LSB of the first differencer, $B_{in} = 1$ and $C_{in} = 0 = \bar{B_{in}}$. Therefore $C_{out} = \bar{B_{out}}$, and we must have $C_{out} = \bar{B_{out}}$ for the next significant bit. Recursively, this shows that $C_{in} = \bar{B_{in}}$ for every bit. Now, $S = A$ for every bit also. Therefore, $Sz^{-1} = Az^{-1}$ for the next clock cycle, for every bit in the unwrapping bit-extender.

The only outputs of one bit of an accumulator are S and C_{out}. But $C_{out} = \bar{B_{out}}$ and $S = A$. Therefore, for every bit in the unwrapping bit-extender for which we have A and B_{out}, we do not need an accumulator.

Appendix B

FIR Filter Plots

This appendix contains plots of the FIR filters shown in table 5.9. The responses shown include the effects of passband attenuation of the decimator (if a decimator is used for that data rate), so these filters have successfully corrected the passband attenuation. Ideal third-order butterworth lowpass filter responses are shown with dashed lines. Dotted lines in the passband plots at right show the performance limits. The inside pair of dotted lines plots frequency error of $\pm 2\%$ with $\pm 1\%$ ripple. The outside pair of dotted lines plots frequency error of $\pm 5\%$ with $\pm 2\%$ ripple.

For the 250 kHz highpass filter for the 622.08 MHz data rate, a separate FIR filter is used. The coefficients for this filter are given in table 5.11. This filter corrects excessive ripple in the IIR 250 kHz filter, as well as passband attenuation caused by the decimator. Figure B-8 plots the magnitude response of the total filter, including the IIR 250 kHz filter, the decimator attenuation, and the FIR filter.

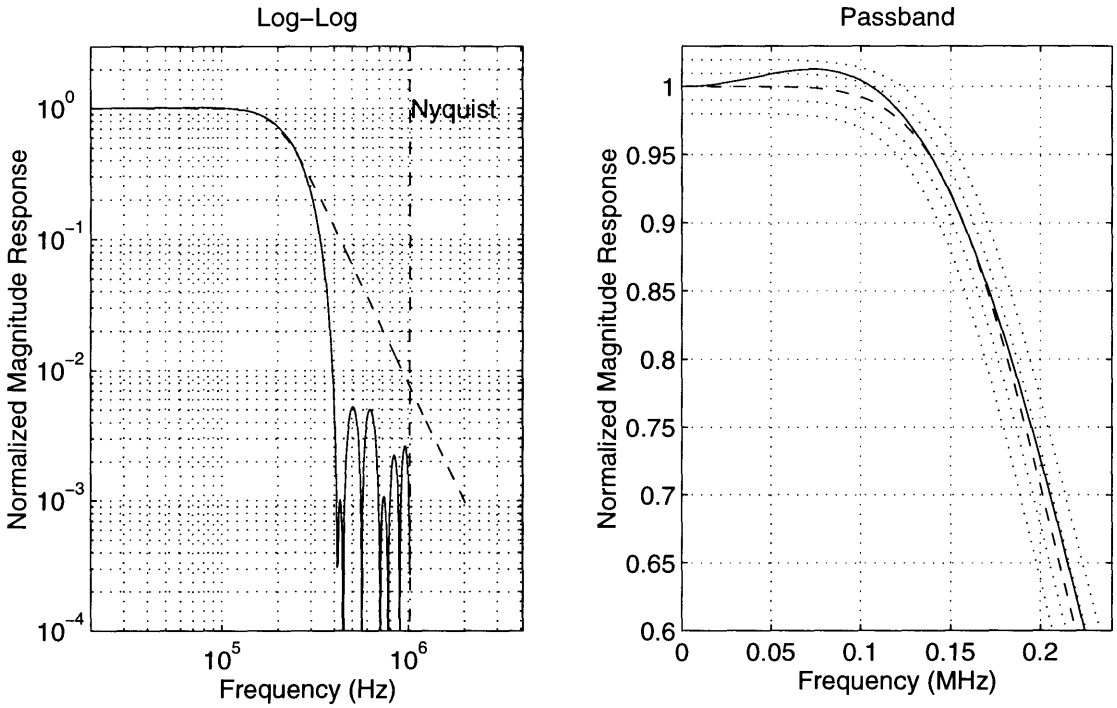

Figure B-1: Frequency response and passband detail of the FIR filter for the 2.048 MHz data rate.

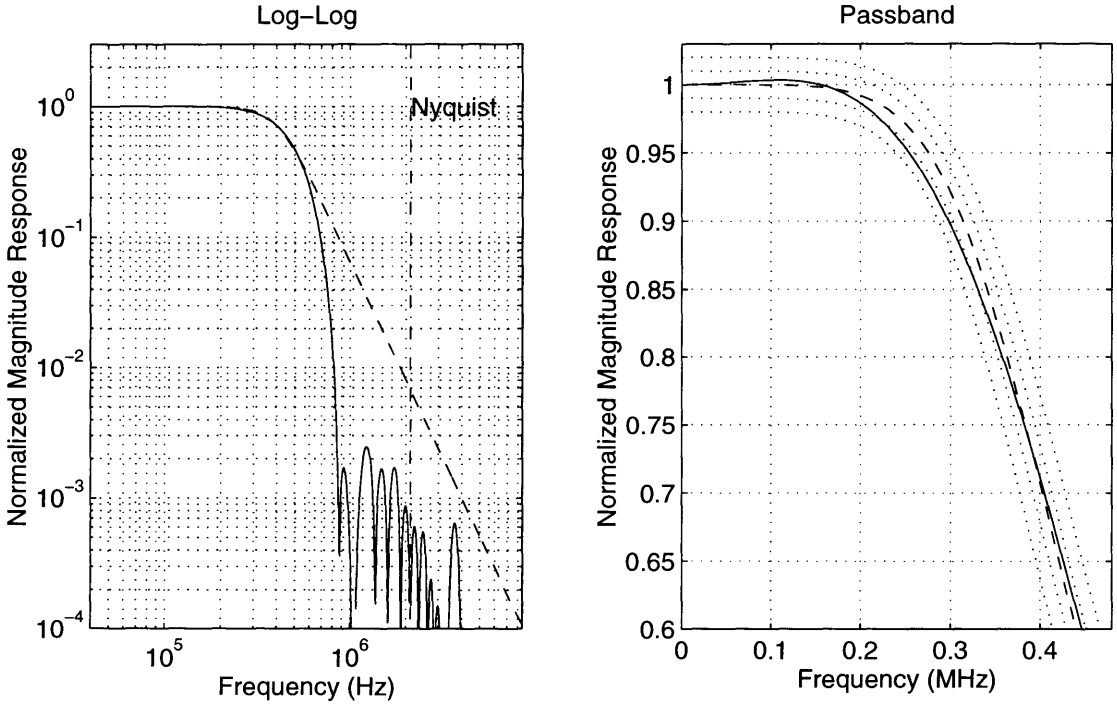

Figure B-2: Frequency response and passband detail of the FIR filter for the 8.448 MHz data rate. Includes the effect of decimator passband attenuation.

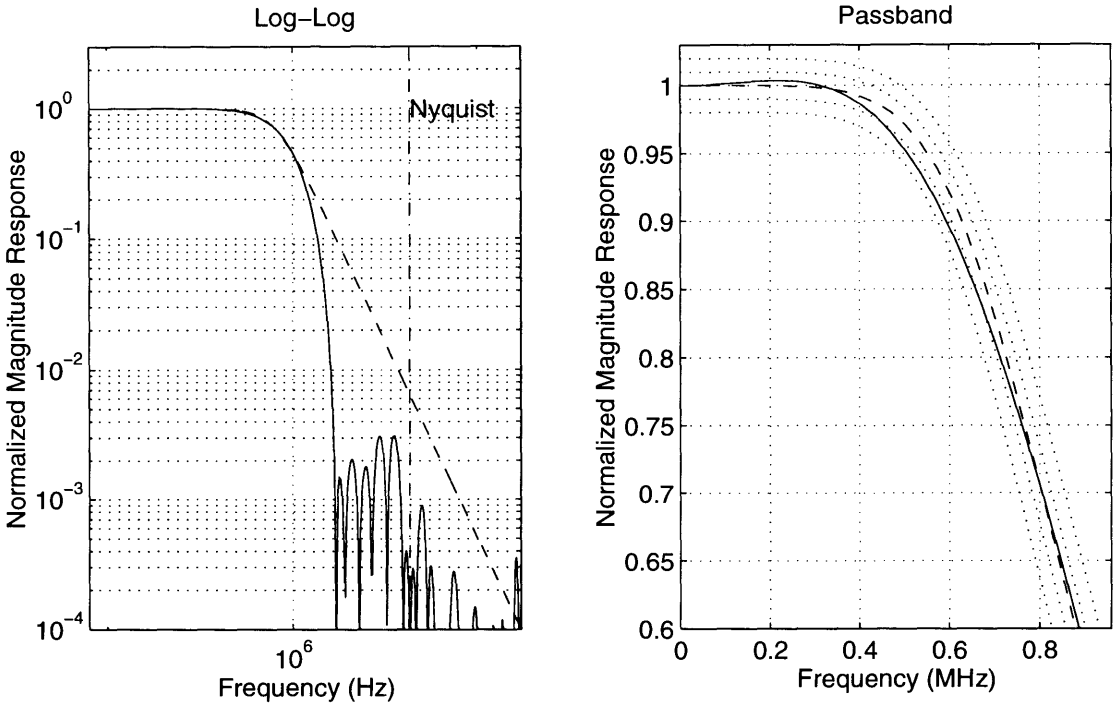

Figure B-3: Frequency response and passband detail of the FIR filter for the 34.368 MHz data rate. Includes the effect of decimator passband attenuation.

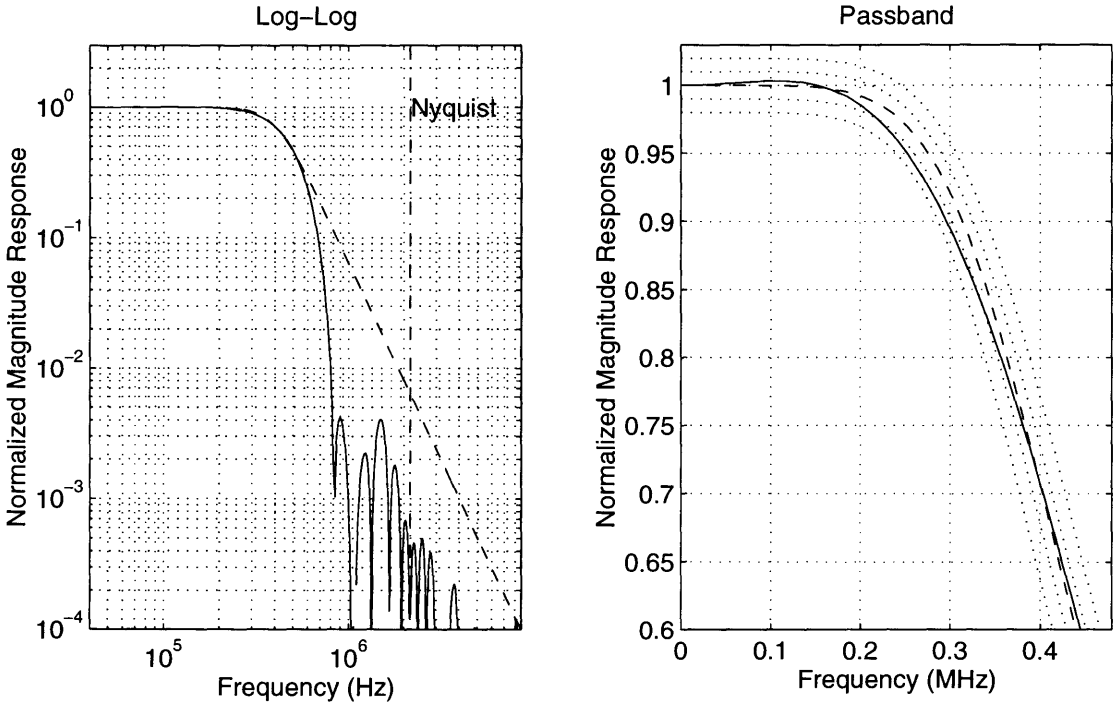

Figure B-4: Frequency response and passband detail of the FIR filter for the 51.84 MHz data rate. Includes the effect of decimator passband attenuation.

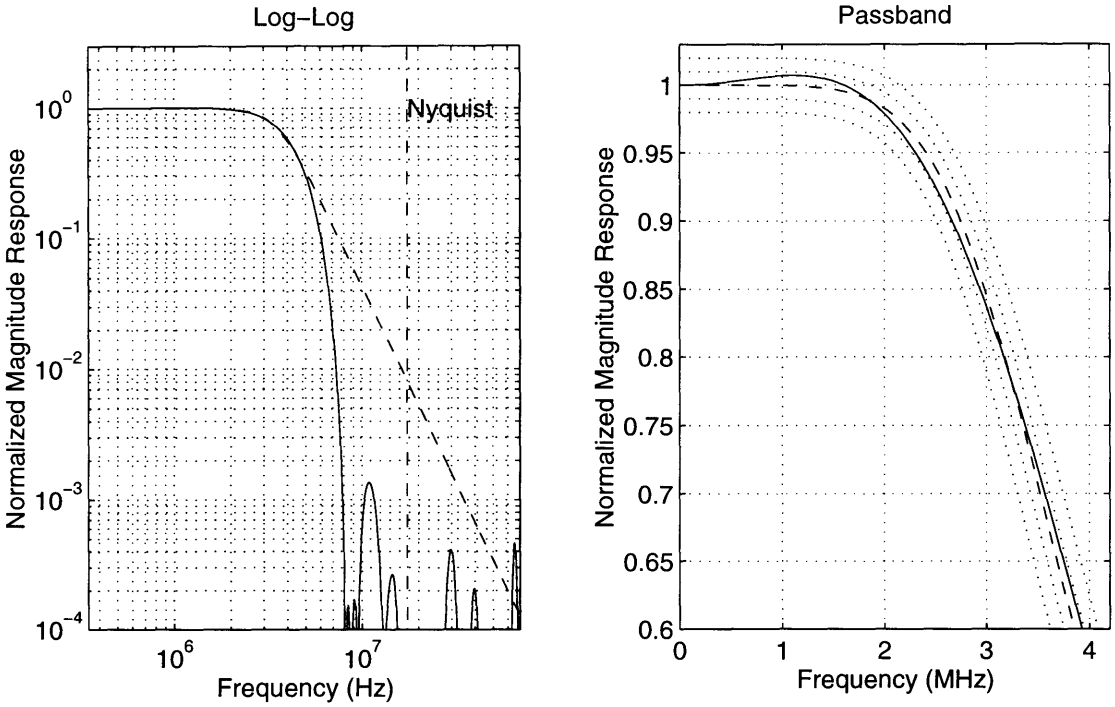

Figure B-5: Frequency response and passband detail of the FIR filter for the 139.264 MHz data rate. Includes the effect of decimator passband attenuation.

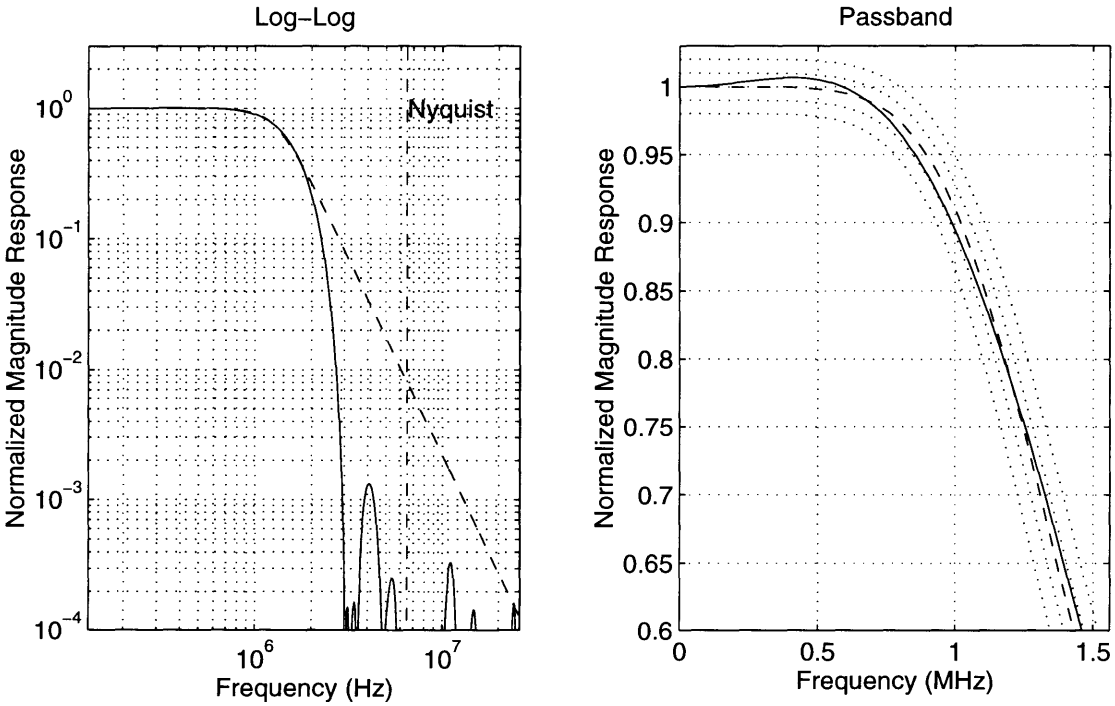

Figure B-6: Frequency response and passband detail of the FIR filter for the 155.52 MHz data rate. Includes the effect of decimator passband attenuation.

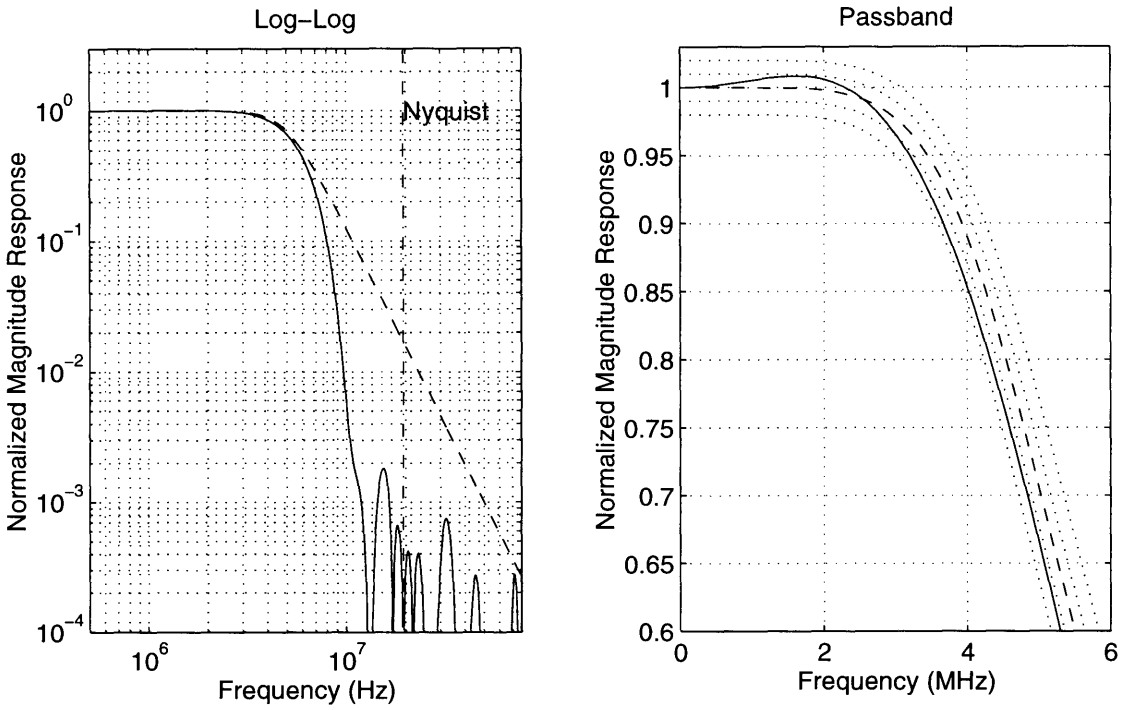

Figure B-7: Frequency response and passband detail of the FIR filter for the 622.08 MHz data rate. Includes the effect of decimator passband attenuation.

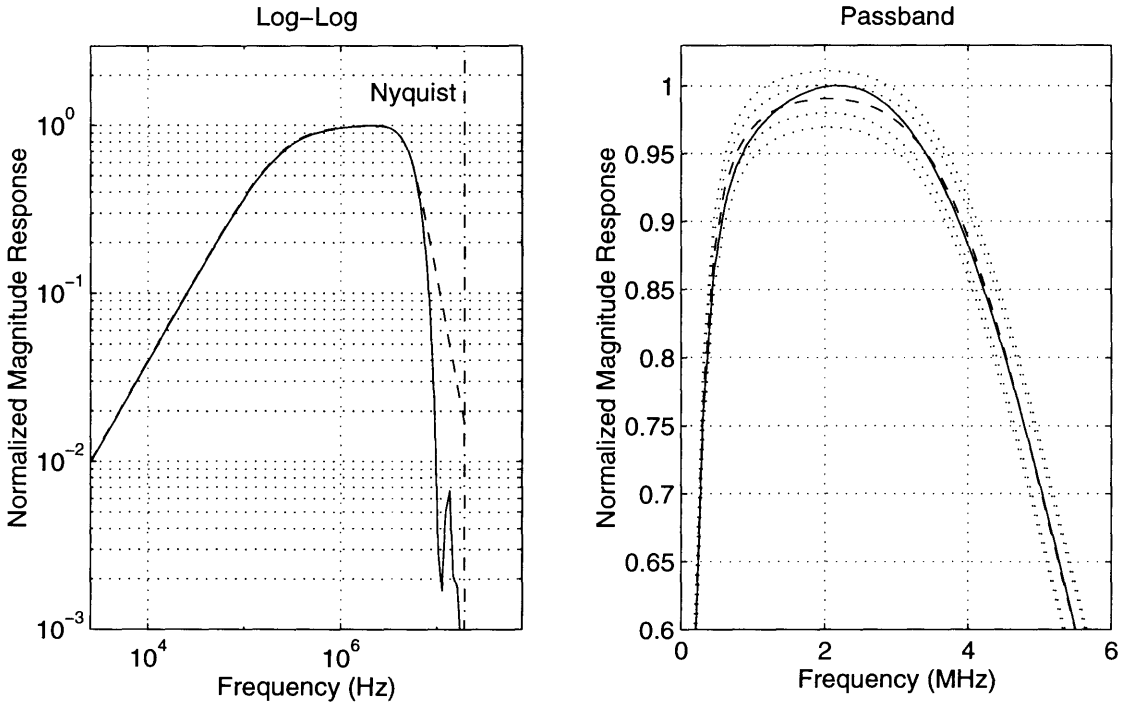

Figure B-8: Frequency response and passband detail of the FIR filter to be used with the 250 kHz highpass filter in the 622.08 MHz data rate, cascaded with that filter. Includes the effect of decimator passband attenuation and the 250 kHz IIR filter.

Appendix C

MATLAB Code for FIR Filters

C.1 firs.m

```
% firs.m:  Generates FIR filters to compensate for CIC passband attenuation
% and simulate 3rd order lowpass rolloffs.

ffilters=[2.048e6 1 200e3 .85 .23 28;
8.448e6 2 400e3 .62 .28 7;
34.368e6 4 800e3 .71 .25 7;
51.84e6 12 400e3 .71 .25 7;
139.264e6 4 3500e3 .53 .35 7;
155.52e6 12 1300e3 .53 .35 7;
622.08e6 16 5000e3 .5 .4 5];

fbitrates=ffilters(:,1);
fdecimation=ffilters(:,2);
fcutoffs=ffilters(:,3);
wpassfacts=ffilters(:,4); % passband cutoff related to 3-pole cutoff
dcband=ffilters(:,5); % don't-care band width
stopweights=ffilters(:,6); % weight of stopband

fsamplerates=fbitrates./fdecimation;
```

```
normcutoffs=2*fcutoffs./fsamplerates;
wpass=normcutoffs.*wpassfacts;
wstop=wpass+dcband;

coefbits=8; % Coefficient precision.
taps=16; % Number of taps in the filters.
specpoints=128; % Number of points nyquist divided into for
% piecewise compensation.

specfreqs=(fsamplerates/2)*((0:specpoints-1)/specpoints);

for x=1:length(fcutoffs)

  if fdecimation(x)>1

    tweak=[1 1;zeros((fdecimation(x)/2)-1,2)];
    tweak=tweak(:);
    cic1=boxcar(fdecimation(x));
    decimator=conv(conv(cic1, cic1),conv(cic1,tweak));
    decimatorft=abs(fft(decimator,specpoints*2*fdecimation(x))).';
    decimatorfudge(x)=1/sum(decimator); %normalize DC attenuation to 1.
    qdecfudge(x)=sum(decimator)/(2^(ceil(log2(sum(decimator)))));
           % ^- use to compensate for decimators.

    compensator=1./decimatorft(1:specpoints);
    firlpf1=remezfit(taps,compensator,wstop(x),wpass(x),stopweights(x));
    firlpf(x,:)=quantize(firlpf1/max(firlpf1),coefbits).';
    firlpffta=abs(fft(firlpf(x,:),specpoints*2)); %raw fft.
    firfudge(x)=1/sum(firlpf(x,:)); %normalize DC attenuation to 1.
    firlpfft(x,:)=firlpffta*firfudge(x); %normalized frequency response.

    decresponse(x,:)=decimatorft(1:4*specpoints)*decimatorfudge(x);
```

```
    wholeresponse(x,:)=decresponse(x,:).*[firlpfft(x,:) firlpfft(x,:)];
    plotfreqs(x,:)=(0:4*specpoints-1)*2*fsamplerates(x)/(4*specpoints);
    modelresponse(x,:)=sqrt(1./(((plotfreqs(x,:)/fcutoffs(x)).^6)+1));

  else % no decimator to compensate for

    firlpf1=remez(taps-1,[0 wpass(x) wstop(x) 1],[1 1 0 0],[1 stopweights(x)]);
    firlpf(x,:)=quantize(firlpf1/max(firlpf1),coefbits);
    firlpffta=abs(fft(firlpf(x,:),specpoints*2)); %raw fft.
    firfudge(x)=1/sum(firlpf(x,:)); %normalize DC attenuation to 1.
    firlpfft(x,:)=firlpffta*firfudge(x); %normalized frequency response.

    decresponse(x,:)=ones(1,4*specpoints); %filler to keep matrices even.
    decimatorfudge(x)=1;
    wholeresponse(x,:)=(abs(fft(firlpf(x,:),specpoints*4))*firfudge(x));
    plotfreqs(x,:)=(0:4*specpoints-1)*fsamplerates(x)/(4*specpoints);
    modelresponse(x,:)=sqrt(1./(((plotfreqs(x,:)/fcutoffs(x)).^6)+1));

  end; %if

end; %for
```

C.2 remezfit.m

```
function y=remezfit(taps, curve, ws, wp, wsweight)

%function y=remezfit(taps, curve, ws, wp, wsweight)

curvepoints = length(curve);
delta=1/(curvepoints*50);
wpindex = floor(wp*curvepoints);
```

```
bandedges = (0:wpindex-1)/curvepoints.';
bandedgem=[bandedges(1:wpindex-1)+delta;bandedges(2:wpindex)-delta];
bandedgepairs = [bandedgem(:); ws; 1];

curve = curve(:);
curve = curve(1:wpindex);
magsm = [curve(1:wpindex-1), curve(2:wpindex)].';
mags = [magsm(:); 0; 0];

weights = [ones(wpindex-1,1); wsweight];

y = remez(taps-1, bandedgepairs, mags, weights).';
```

C.3 quantize.m

```
function y = quantize(infilter, bits)

% quantizes a filter to a certain number of bits such that the largest tap has
% a value of 2^(bits-1)-1.

infilter = infilter/max(infilter);
posbits = bits-1; % half the numbers are negative
y = round((pow2(posbits)-1)*infilter);
```

Bibliography

[1] Application note 73: Implementing FIR filters in FLEX devices. Altera Corporation, San Jose, CA, January 1996.

[2] Altera, San Jose, CA. *1995 Data Book*, 1995.

[3] Ronald E. Crochiere and Lawrence R. Rabiner. *Multirate Digital Signal Processing.* Prentice-Hall, Englewood Cliffs, NJ, 1983.

[4] Caleb Crome. Senior Megacore Development Engineer, Altera Corporation. Personal communication, February 1997.

[5] Eugene B. Hogenauer. An economical class of digital filters for decimation and interpolation. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 29(2):155+, April 1981.

[6] Keshab K. Parhi and David G. Messerschmitt. Look-ahead computation: Improving iteration bound in linear recursions. In *Proc. IEEE Int. Conf. Acoust., Speed, Signal Processing Symposium on the Theory of Computing*, Dallas, TX, April 1987.

[7] Leo Petropoulos. Replace digital signal processors with HCPLDs. *Electronic Design*, pages 99+, September 1995.