Design of a Digital Octave Band Filter

Examensarbete utfört i Elektroniksystem vid Tekniska högskolan av

Ludvig Lindblom

LiTH-ISY-EX--12/4581--SE

Linköping 2012
Design of a Digital Octave Band Filter

Examensarbete utfört i Elektroniksystem vid Tekniska högskolan av

Ludvig Lindblom

LiTH-ISY-EX--12/4581--SE

Handledare: Amir Eghbali
            ISY, Linköpings universitet
            Pär Gunnars Risberg
            Actiwave

Examinator: Kent Palmkvist
            ISY, Linköpings universitet

Linköping, 8 May, 2012
**Design av digitalt oktavbandsfilter**

Ludvig Lindblom

This report describes the design and implementation of a fixed audio equalizer based on a scheme where parts of the signal spectrum are downsampled and treated differently for the purpose of reducing the computational complexity and memory requirements. The primary focus has been on finding a way of taking an equalizer based on a simple minimum-phase FIR filter and transform it to the new type of equalizer. To achieve this, a number of undesirable effects such as aliasing distortion and upsampling imaging had to be considered and dealt with. In order to achieve a good amplitude response of the system, optimization procedures were used.

As part of the thesis, a cost-effective implementation of the filter has been made for an FPGA, in order to verify that the scheme is indeed usable for equalizing an audio signal.

**Nyckelord**

digital filters, multirate filtering, audio processing
Abstract

This report describes the design and implementation of a fixed audio equalizer based on a scheme where parts of the signal spectrum are downsampled and treated differently for the purpose of reducing the computational complexity and memory requirements. The primary focus has been on finding a way of taking an equalizer based on a simple minimum-phase FIR filter and transform it to the new type of equalizer. To achieve this, a number of undesirable effects such as aliasing distortion and upsampling imaging had to be considered and dealt with. In order to achieve a good amplitude response of the system, optimization procedures were used.

As part of the thesis, a cost-effective implementation of the filter has been made for an FPGA, in order to verify that the scheme is indeed usable for equalizing an audio signal.
Acknowledgments

I would like to thank my supervisor Amir Eghbali for helping me get a good start on the thesis work, and for all the valuable advice that I have gotten later. Thanks also to my examiner Kent Palmkvist, who has always given fast and clear answers to all questions regarding the practical details that has had to be considered in order to complete the thesis.

Actiwave has provided me with a very interesting thesis work, where I have had the opportunity to train my skills in several areas relating to digital design. I would like to thank my supervisor Pär Gunnars Risberg, who is always enthusiastic, supporting and manages to come up with new and clever ideas on improvements and solutions. Erik Lindahl deserves a big thanks as well for giving me alot of good advice about how to implement the system, and for the support and encouragement that he has provided during the thesis work.

There are many persons who has taken their time to help with practical parts of the project. Jonny Lindgren and Peter Johansson at ISY has more than once searched through their boxes for cables and other hardware that I have needed. Thomas Johansson at ISY support has fixed problems with my workstation, often within minutes of my first message. To all these I am very grateful. Thanks also to my friend Christoffer Peters, who has eagerly helped with troubleshooting hardware-related problems and has provided ideas on how to make the work easier.

On a more personal level, the interest and encouragement from friends and family has been a great motivator. Special thanks goes to my big brother Johannes Lindblom, who has both the technical know-how and the heart to understand most of my problems, and my soon-to-be wife Marija Klajić, who has been an increadible support on the days were my self confidence has been on the bottom, and has shared the joy that comes from good results.
# Contents

1 Introduction .................................................... 1
   1.1 Background ................................................. 1
   1.2 Purpose ..................................................... 2
   1.3 Intended Readers ........................................... 2
   1.4 Methods .................................................... 2
   1.5 Limitations and Scope .................................... 2

2 Multirate Digital Signal Processing ......................... 5
   2.1 Downsampling ................................................. 5
      2.1.1 Decimation ............................................. 6
      2.1.2 Noble Identity of Downsampling .................... 9
   2.2 Upsampling ................................................ 9
      2.2.1 Interpolation .......................................... 9
      2.2.2 Noble Identity of Upsampling ....................... 12

3 Finite Impulse Response Filters ............................. 13
   3.1 Linear Phase FIR Filters ................................. 14
   3.2 Complementary Linear Phase Filters ................... 16
   3.3 Minimum Phase FIR Filters ............................... 18
   3.4 A Note on Interpolation Filters ......................... 18

4 Octave Band Filter ........................................... 21
   4.1 Overview .................................................. 21
   4.2 Considerations .............................................. 21
      4.2.1 Aliasing ............................................... 23
      4.2.2 Imaging ............................................... 23
      4.2.3 Delay ................................................. 24
      4.2.4 Phase Distorsion ..................................... 25
      4.2.5 Complexity and Memory .............................. 25

5 Choice of Filters ............................................. 27
   5.1 Downssampler .............................................. 27
   5.2 Upsampler ................................................ 28
   5.3 Processing Filters ....................................... 28
<table>
<thead>
<tr>
<th>Chapter</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>6</td>
<td>System Analysis</td>
<td>31</td>
</tr>
<tr>
<td>6.1</td>
<td>Transfer Function</td>
<td>31</td>
</tr>
<tr>
<td>6.1.1</td>
<td>Downsampler Stages</td>
<td>31</td>
</tr>
<tr>
<td>6.1.2</td>
<td>Upsampler Stages</td>
<td>34</td>
</tr>
<tr>
<td>6.1.3</td>
<td>Processing Stages</td>
<td>34</td>
</tr>
<tr>
<td>6.1.4</td>
<td>Total Transfer Function</td>
<td>35</td>
</tr>
<tr>
<td>6.2</td>
<td>Time Domain Representation</td>
<td>35</td>
</tr>
<tr>
<td>6.3</td>
<td>Delay</td>
<td>35</td>
</tr>
<tr>
<td>6.4</td>
<td>Computational Complexity</td>
<td>36</td>
</tr>
<tr>
<td>6.5</td>
<td>Memory Requirement</td>
<td>36</td>
</tr>
<tr>
<td>7</td>
<td>Filter Design</td>
<td>39</td>
</tr>
<tr>
<td>7.1</td>
<td>Method</td>
<td>39</td>
</tr>
<tr>
<td>7.2</td>
<td>Analysis and Synthesis Banks</td>
<td>39</td>
</tr>
<tr>
<td>7.3</td>
<td>Processing Filters</td>
<td>40</td>
</tr>
<tr>
<td>8</td>
<td>Implementation</td>
<td>45</td>
</tr>
<tr>
<td>8.1</td>
<td>Hardware Choice</td>
<td>45</td>
</tr>
<tr>
<td>8.2</td>
<td>Block Layout</td>
<td>46</td>
</tr>
<tr>
<td>8.3</td>
<td>Downsampler and Upsampler Implementation</td>
<td>47</td>
</tr>
<tr>
<td>8.4</td>
<td>Control Unit</td>
<td>47</td>
</tr>
<tr>
<td>9</td>
<td>Results</td>
<td>51</td>
</tr>
<tr>
<td>9.1</td>
<td>Design</td>
<td>51</td>
</tr>
<tr>
<td>9.2</td>
<td>Implementation</td>
<td>53</td>
</tr>
<tr>
<td>10</td>
<td>Future Work</td>
<td>57</td>
</tr>
<tr>
<td>10.1</td>
<td>Design Improvements</td>
<td>57</td>
</tr>
<tr>
<td>10.2</td>
<td>Implementation Improvements</td>
<td>58</td>
</tr>
<tr>
<td>10.3</td>
<td>Testing</td>
<td>58</td>
</tr>
</tbody>
</table>

Bibliography | 59 |
# Acronyms

<table>
<thead>
<tr>
<th>Acronym</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DSP</td>
<td>Digital Signal Processing</td>
</tr>
<tr>
<td>FIFO</td>
<td>First In First Out (register)</td>
</tr>
<tr>
<td>FIR</td>
<td>Finite Impulse Response (filter)</td>
</tr>
<tr>
<td>FPGA</td>
<td>Field Programmable Gate Array</td>
</tr>
<tr>
<td>HP</td>
<td>High Pass (filter)</td>
</tr>
<tr>
<td>IIR</td>
<td>Infinite Impulse Response (filter)</td>
</tr>
<tr>
<td>LP</td>
<td>Low Pass (filter)</td>
</tr>
<tr>
<td>LUTs</td>
<td>Look Up Tables</td>
</tr>
<tr>
<td>LWDF</td>
<td>Lattice Wave Digital Filters</td>
</tr>
<tr>
<td>MAC</td>
<td>Multiply Accumulate</td>
</tr>
<tr>
<td>RAM</td>
<td>Random Access Memory</td>
</tr>
<tr>
<td>ROM</td>
<td>Read Only Memory</td>
</tr>
</tbody>
</table>
Chapter 1

Introduction

This chapter will provide an introduction to the thesis and this report.

1.1 Background

The information of an audio signal such as music or speech is not linearly spread throughout the frequency spectrum. The density of information rather follows a logarithmic pattern with more information at lower frequencies. When processing an audio signal, performance could be increased by using the fact that the higher frequencies needs less processing. If one Finite Impulse Response (FIR) filter is used for the equalizing, this opportunity is missed, as an FIR filter treats all frequencies with the same precision.

Figure 1.1: Frequency band splitting in octave filter with four frequency bands.

A solution to this problem is to split the frequency spectrum into several bands, downsample each band as much as possible, filter the bands with a designated filter and then rebuild the signal by upsampling and adding the outputs. Because of the information density of an audio signal it is natural to split the bands from one octave to the next, meaning that one band consists of the upper half of the spectrum, the next band of the upper half of the remaining spectrum, and so on.
until the last band, which contains the remaining frequencies. The bands will then be narrower at the lower frequencies, making it possible to downsample these bands more. An example of this where the spectrum is split between four bands can be seen in Figure 1.1. Downsampling the bands allows for more precise processing, but introduces problems associated with the downsampling and the upsampling that comes after.

1.2 Purpose

The purpose of the thesis is to find a cost-effective way to substitute an arbitrary FIR filter for an octave band filter according to figure 1.2, that splits the frequency spectrum into several differently sized bands, processes each band at the lowest possible sample rates and then adds the bands together. The system should then be useable as a fixed (not user-configureable) equalizer for processing an audio signal. When this has been accomplished the filter is to be implemented on a Xilinx Spartan 3E Field Programmable Gate Array (FPGA).

1.3 Intended Readers

This report assumes that the reader has basic knowledge in Digital Signal Processing (DSP) and FIR digital filters. A course in digital signals and systems should be enough for this. The necessary understanding of multirate systems as well as some special FIR filters will be explained in chapter 2 and 3, respectively.

1.4 Methods

Time and frequency domain models of the system were built in Matlab, and functions were made that would rapidly give a more or less acceptable system that could be tested for different non-idealities such as aliasing and imaging distortion. In order to find better solutions, optimization of the system was made in Matlab. Testing of the system was both done by evaluating the amplitude characteristics as well as listening to the actual result of filtering a song or a sinusoidal sweep.

1.5 Limitations and Scope

As the purpose of the new filter was to reduce the computational complexity and memory usage as compared to the reference FIR filter, this provided an upper limit on the number of frequency bands and the order of the filters used. Another limitation was the delay of the system. Because an audio system should be usable when watching film, there was a need for keeping the delay of the filter (and thus the sound) as short as possible.

Only audio signals have been considered for this thesis, but the results may be interesting for processing of signals that have similar characteristics (i.e. where the lower part of the spectrum is more important than the upper part).
Figure 1.2: Structure of the filter proposed by Actiwave. The downsampling blocks splits the signal spectrum into multiple bands, and the upsampling blocks adds them together.
Chapter 2

Multirate Digital Signal Processing

This chapter is a quick refresher of the basic knowledge needed for understanding multirate DSP and reading this report. For a more complete introduction, please consult for example [3][5] or [6].

2.1 Downsampling

Downsampling by a factor M means that every M:th sample of a signal is kept, while the others are discarded [5]. This makes the signal look as though it was originally sampled at a lower rate. The process of downsampling by a factor of two is depicted in figure 2.1. Because the output of the downsampler comes at a frequency that is lower than the original, the spectrum of the signal is stretched as in figure 2.2. The magnitude of the signal is also lowered by the downsampling factor.

By spreading the spectrum, filters working on the new signal can have looser specifications and still get the same work done. Further, the processing rate is lowered, meaning that less data is needed to be processed per cycle on average.

Figure 2.1: A signal is downsampled by a factor of two, meaning that every other sample is kept.
Figure 2.2: When downsampling the signal by a factor (here two), the spectrum is spread out by the same factor.

Before downsampling the spectrum must be bandlimited. Otherwise the situation in 2.2 occurs, where the signal spectrum overlaps. This is called aliasing distortion, and should be avoided.

The mathematical representation of downsampling a signal $X$ by a factor $M$ is given by:

$$X(z) \downarrow M = \frac{1}{M} \sum_{k=0}^{M-1} X(z^{\frac{1}{M}} W_k^M), \text{ where } W_k^M = e^{j\frac{2\pi k}{M}} \quad (2.1)$$

As this thesis exclusively will deal with downsampling by a factor two, that is given as an example:

$$X(z) \downarrow 2 = \frac{X(z^{\frac{1}{2}}) + X(-z^{\frac{1}{2}})}{2} \quad (2.2)$$

$X(z^{\frac{1}{2}})$ in the nominator contains the non-aliased signal, while the factor $X(-z^{\frac{1}{2}})$ is the aliased signal. Worth to mention is that the minus sign before $z$ means a frequency shift of $\pi$ radians. This means that two different frequency components are added together (alias distortion), which is bad if the magnitudes are non-negligible. The factor 2 in the denominator means that the magnitude is halved. This is not a problem in an implementation, as a multiplication by two is really easy to do on-chip.

2.1.1 Decimation

As mentioned, downsampling can cause aliasing distortion. In order to avoid this, the signal is generally bandlimited before the downsampling occurs. This
is referred to as decimation, and the block schematic is shown in figure 2.3. In figure 2.4 an ideal Low Pass (LP) filter is used that removes all signal components over $\pi/2$. The downsampling is then made without aliasing distortion.
Figure 2.4: By first bandlimiting the signal, alias distortion is reduced. The drawback is that some of the signal is lost.
2.2 Upsampling

The band-limiting filter does not necessarily have to be of LP type. When downsampling by a factor of two, an High Pass (HP) filter could be used to keep the higher frequencies. This means that the alias component \( X(-\frac{1}{2}) \) becomes the new signal, while what would ”normally” be the signal is now filtered away.

2.1.2 Noble Identity of Downsampling

A useful mathematical tool for calculating the spectrum of a signal when multiple downsampling steps are involved is the noble identity of downsampling [5], displayed in figure 2.5. Later on in this report, this will let us lump a lot of cascaded downsampling blocks together.

\[
\begin{array}{c}
\downarrow M \\
H(z) \\
\leftrightarrow \\
H(z^M) \\
\downarrow M
\end{array}
\]

Figure 2.5: Noble identity of downsampling.

2.2 Upsampling

The opposite of downsampling is upsampling. Upsampling by a factor \( L \) means putting \( L - 1 \) zeros between every sample, as described in figure 2.6. The point of doing this becomes clear when looking at the resulting spectrum. An example is given in figure 2.7. As can be seen in the figure, the spectrum is compressed. As a side effect, images of the compressed spectrum appears. The mathematical representation of upsampling a signal \( X \) by a factor \( M \) is given by:

\[
X(z) \uparrow M = X(z^M)
\]

2.2.1 Interpolation

The images of the spectrum that arises when a signal is upsampled generally needs to be filtered away. Upsampling followed by filtering is called interpolation. If an
Figure 2.7: When upsampling by a factor (here two) the signal spectrum is compressed and repeated.

Figure 2.8: Block diagram of an interpolator of factor $L$.

LP filter is used for the filtering, the case in figure 2.9 is the result. In the time-domain, this means that the signal is smoothened out and that the zeros from the upsampling will instead be put between the amplitudes of the original samples - thus interpolating them.
Figure 2.9: Removing the image with an LP filter.
As with the decimator case, an HP filter can be used instead. This means that the same information is kept, but is instead represented by the higher frequencies.

2.2.2 Noble Identity of Upsampling

There exists a noble identity for upsampling as well [5], depicted in figure 2.10. This will also be useful later on in the report.

Figure 2.10: Noble identity of upsampling.
Chapter 3

Finite Impulse Response Filters

FIR filters have a number of characteristics that are desirable in DSP [6]. For one, they are guaranteed to be stable, as there is no feedback involved. Furthermore, they can be made to be of linear phase type, meaning that all frequencies are delayed by the same amount of time. The drawback of FIR filters is that the lack of recursion make them more inefficient than Infinite Impulse Response (IIR) filters. The basic structure of an FIR filter is quite simple, as it directly implements the convolution between its finite impulse response and the input signal. The transfer function of an FIR filter whose impulse response begins at $n = 0$ is:

$$ H[z] = \sum_{0}^{L-1} h[n]z^{-n} $$

(3.1)

where $L$ is the (finite) length of the impulse response and $h[n]$ is the impulse response at sample $n$. $z^{-n}$ translates to a time delay of $n$ samples in the time domain. An example of an FIR filter can be seen in figure 3.1. The coefficients $b_0$ to $b_7$ are the same as the impulse response values $h_0$ to $h_7$. The order of the filter (the number of delay registers) is six, and the number of taps (equal to the impulse response length) is seven.

Figure 3.1: One possible structure of an FIR filter.
3.1 Linear Phase FIR Filters

Linear phase FIR filters are often preferred over other types, as they delay all frequencies of a signal by the same amount of time (i.e. they have constant phase and group delay). This is useful when for example filtering noise from an electrocardiogram. If all frequencies are not delayed equally, it might look as though the patients heartbeat is very odd! The delay of a linear phase filters is \((N - 1)/2\), where \(N - 1\) is the order (i.e. the number of delay registers) of the filter.

![Magnitude and phase response of a linear phase filter](image)

Figure 3.2: Magnitude and phase characteristics of a linear phase filter.

As this thesis involves splitting and recombining the signal spectrum in several steps, it has been very important to make sure that the phase characteristics are the same for all spectrum components when recombining the signal. This can relatively easily be achieved with the use of linear phase filters. The drawback of this is that the delay of a linear phase FIR filter is large compared to other types of filters.

A linear phase FIR filter can easily be recognized by its impulse response, which is either symmetrical or antisymmetrical around the delay of the filter. An example of this is presented in figure 3.3.

The coefficients of an FIR filter are taken directly from the impulse response. For a linear phase FIR this means that pairs of coefficients will be equal to or negative of each other. That in turn allows us to reduce the number of multiplications by two, demonstrated in figure 3.4.
Figure 3.3: Impulse response of a lowpass linear phase FIR filter of order 10.

Figure 3.4: A linear phase FIR filter can be economically implemented by utilizing the fact that most of the coefficients comes in pairs.
3.2 Complementary Linear Phase Filters

A neat consequence of the linear phase property is that given a linear phase filter you can get its complement, meaning a filter with the opposite magnitude spectrum, without adding much to the overall complexity [6]. In figure 3.5 the complement of a lowpass filter has been plotted. The complement is taken by first delaying the input sample by the same time as the delay of the original filter. This makes sure that all frequencies are in-phase for the next step, which is subtracting the filtered signal from the delayed one. This means that if a specific frequency was highly attenuated by the filter, the complement will be attenuated only a little. Block schematics describing the process are shown in figure 3.6.
3.2 Complementary Linear Phase Filters

(a) Forming of the complementary output.

(b) Block schematic of a linear phase FIR filter of order six and its complementary output.

Figure 3.6: Taking the complementary output. The filter order must be even so that the delay of the original filter is an integer number of samples.
A drawback of using complementary filters is that passband and stopband rippel must be treated equally. The reason for this is that rippel in the passband for the original filter translates to rippel in the stopband for the complementary filter, and vice versa. Also, only even order filters may be used. This is because the delay of the filter must be an integer.

3.3 Minimum Phase FIR Filters

While linear phase FIR filters have the nice property of delaying the whole signal by the same amount of time, that amount of time is quite large in proportion to the order of the filter. When low delay is important to a system, an alternative is to instead use minimum phase FIR filters.

Minimum phase filters tries to keep the phase of the transfer function as close to zero as possible. In a pole-zero diagram this is achieved by putting all the zeros inside the unit circle [6]. The phase contributions from each zero will then work in opposite directions of those of the poles, meaning that the overall phase shift is kept to a minimum. Keeping the phase to a minimum also means keeping the delay low. For a given filter order, a minimum phase filter can typically be made to have better characteristics than a linear phase one. However, a minimum-phase filter cannot be as efficiently implemented as a linear phase, as all coefficients are (generally) unique. A comparison of a minimum phase and linear phase low pass filter can be viewed in figure 3.7 on page 19.

3.4 A Note on Interpolation Filters

The upsampling that is made before the interpolation filtering introduces a number of zeros in between every sample. If the upsampling is made by a factor of two, every other input to the interpolation filter is then zero. This fact can be used to reduce the computational complexity of the filter [6], as a multiplication with zero will always result in zero. How this is done can be seen in figure 3.8.
3.4 A Note on Interpolation Filters

(a) linear phase filter pole placement  (b) minimum phase filter pole placement

(c) spectrum differences between minimum phase and linear phase filters

Figure 3.7: Comparison between linear phase and minimum phase filters with approximately the same transfer function.
Figure 3.8: The filter in (a) uses seven multiplications to filter one output, while (b) uses seven multiplications to filter two outputs. For simplicity, the input here is a stream of ones.
Chapter 4

Octave Band Filter

This chapter presents the filter layout proposed by Actiwave and studied in this thesis. First, an overview of the system is presented followed by some general considerations when designing the filters.

4.1 Overview

The block layout of the system can be seen in figure 4.1 on page 22. The general idea is that the upper half of the input frequency spectrum is sent to a separate processing filter, while the remaining spectrum is downsampled by a factor of two. The process is repeated for the downsampled spectrum until the last downsampler block, where the lowest frequencies are finally processed. After all bands have been processed the outputs are upsampled to the original sample rate, and then added together. It has been desired that the same filters are used for all the upsamplers and downsamplers. This reduces the number of coefficients that needs to be stored in an implementation.

The purpose of doing the downsampling is that the more information-dense lower frequencies (which are downsampled the most) can then be filtered with much greater precision than the higher frequencies. The computational benefit is large: if one more band is added, the order of the last processing filter can be reduced by a factor of two while achieving the same performance. Furthermore, the last band now only needs to be calculated half as many times. While there is a computational benefit to separately processing parts of the spectrum of an audio signal, the main question has been if this gain can outweigh the costs associated with the down- and upsampling.

4.2 Considerations

A number of problems associated with the design had to be solved. Some were effects of the sample rate alterations, while others were requirements of the equalizer.
(a) Structure of the filter. The downsampling blocks are used to split the frequency spectrum into several bands, and the upsampling blocks are used to add the processed bands together to form the output.

(b) Structure of the downsampler and upsampler blocks. The same filters are used in all downsampler and upsampler blocks, making an implementation cheaper.

Figure 4.1: Structure of the filter proposed by Actiwave.
4.2 Considerations

(a) Structures of the downsampler and upsampler. The downsampler splits the input spectrum in two, and the upsampler puts it together again.

(b) The upsampler takes two inputs, interpolates one of them, and adds both to construct the output. A downsampler followed by an upsampler should ideally give the original input signal as output.

Figure 4.2: Principles of the down- and upsampler blocks. The downsampler block splits the spectrum in two parts that are then processed independantly, while the upsampler block recombines the parts to form the output.

4.2.1 Aliasing

Before downsampling, the spectrum has to be bandlimited, as described in section 2.1.1 on page 6. However, because no real filter is ideal, if the cutoff frequency of the filter is placed exactly at the new maximum frequency after downsampling ($\pi/2$ when downsampling by two), there will be aliasing. For this reason the low pass filter of the downsampler had to have a cutoff frequency set below $\pi/2$ [8].

For a comparison of aliasing from filters of different bandwidths, please refer to figure 4.3.

4.2.2 Imaging

As with aliasing, imaging can cause some distorsion to the signal, and must be dealt with by filtering the upsampled signal as described in section 2.2.1 on page 9. Imaging had a larger effect on the system than aliasing. This is because the imaging occurs in the upsamplers, which are placed after the processing filters. As the equalizers considered often has a large boost to a specific part of the system, the highest frequencies of the image had a large amplification. 4.4 is a plot showing this problem. Rather than using higher order filters to reduce this big image, the interpolation filter should be designed so that the attenuation is larger where the
Figure 4.3: Reducing the cutoff frequency of the bandlimiting filter will reduce the aliasing.

4.2.3 Delay

The delay of a system is often not as important as its other properties. However, it was desired that the system was usable with a video system, which meant that the delay must not be too long. For television broadcasting the audio should not lag more than 45 milliseconds relative to the video [2], and for film the limit is considered to be 22 milliseconds [1]. The aim was therefore to keep the delay to around or below 10 milliseconds (or 441 samples for a signal at 44100Hz) in order to provide some slack for delays from other parts of the full audio system.

Looking at figure 4.1, if the delay through the first downsampling block is $D$, then the delay through the next is $2D$ because of the lower rate of processing. The total delay through the downsampling stages is therefore a geometrical sum, with
4.2 Considerations

Figure 4.4: Plot of imaging. Notice that the boost of the processing filters gives rise to a higher image.

Each new stage approximately doubling the total delay. It was soon discovered that if all parts of the system (downsampler, upsampler and processing) used linear phase filters, the delay would become too large. This means that there is a trade-off: increasing the number of bands reduces the computational complexity, but increases the delay.

4.2.4 Phase Distorsion

If the signal is delayed by different amounts of time for each path through the filter, the phase distorsion of a signal component will depends on which path it took. This became a problem in the filter architecture that was to be implemented, as parts of the signal spectrum would pass through two different processing bands with approximately the same strength, but with one of the paths going through a lower number of filters, and thus being less delayed. If linear phase filters are used, this can be compensated for by adding delay elements in the shorter paths, so that the total delay through all paths was equal. The effect of not doing this compensation can be seen in figure 4.5.

4.2.5 Complexity and Memory

The purpose of the new filter topology suggested is to reduce the computational complexity of an audio equalizer. The reference filter was of FIR type, and compared to that gains could be made in both computational complexity and memory usage.
Figure 4.5: Plot demonstrating the harmful effects of unequal delay through the system paths. If delay elements are used to keep everything in phase, the result is quite close to the ideal one.
Chapter 5

Choise of Filters

In this chapter, the choises made for the different filters in the system are presented and motivated. First off are the filters of the down- and upsampling stages, also called the analysis bank and synthesis bank, respectively. After that, the filters used for the actual processing are described.

The mathematical description of the system is left for a later chapter. For now, the discussion will be kept simple.

5.1 Downsampler

![Figure 5.1: Structures of the downsampler and upsampler. The downsampler splits the input spectrum in two, and the upsampler puts it together again.](image)

The downsampler is splitting its input frequency spectrum into two parts as well as downsampling the lower frequency part. In order to do the downsampling without significant aliasing, the band is split below half the Nyquist frequency. It has been very important to make sure that the signal can be split and reconstructed without serious distortion. To achieve this, a couple of different approaches has been studied. Linear phase filters provides the most obvious and simple solution (because the distortion to the phase is that of a pure delay), but comes at the price of a long delay. To avoid this delay a couple of different alternatives were tried. These were minimum phase FIR filters and cascaded second-order IIR filters
(also called digital biquad filters). The problem with both of these were that they introduced different phase distortions to the different bands of the system. Because of this, linear phase filters were used for the downsampler. The aim was to reduce alias components to a level that was inaudible. The less critical high pass filter was implemented as the complement of the low pass. For this reason, both filters had to be of even order.

5.2 Upsampler

For the same reasons as for the downsampler, linear phase FIR were used for the upsampler as well. The cutoff frequency of the lowpass filter was placed below half the Nyquist frequency. This improved the attenuation of images that appeared close to that frequency. A side effect is that the lower band signal was slightly attenuated at a part of the spectrum in the upsampling process. As long as this attenuation is small enough, the effect can be compensated for in the processing filters. For simplicity, the order of the filter in the upsampler was set to be the same as in the downsampler. It might pay off to make this filter slightly larger than the downsampler filter, as imaging was a bigger problem than aliasing. The order should be even, as this gives an integer delay, which can then be compensated for.

5.3 Processing Filters

The processing filters are the largest filters of the system. For this reason, letting them be of linear phase type would cause too much time delay of the audio signal. Luckily, having them as minimum phase filters worked out. The phase distortion is not as large as for a bandlimiting filter, and each processing filter only affects the phase of its own band, which gives only a small difference between the different bands that can be compensated for in the system optimization. The different processing filters may very well be of different sizes. As a rule of thumb, the last should be at least the size of the reference filter divided by two to power of the number of downsamplers.
Figure 5.3: The upsampler takes two inputs, interpolates one of them, and adds both to construct the output. A downsampler followed by an upsampler should ideally give the original input signal as output.
Chapter 6

System Analysis

This chapter will deal with the mathematical description of the system. The formulas derived here have been used in testing different solutions and types of filters, as well as in the optimization of the final filter.

6.1 Transfer Function

One approach of determining the characteristics of a system is to apply an impulse to it, do the time-domain calculations, and finally take the Fourier transform of the impulse response to get the magnitude and phase characteristics. This works less well for the system in this thesis, for the reason that the impulse response is dependant of when the impulse is applied. In other words, the system is not time-invariant. The reason for this is the downsampler. As it only keeps some of the samples and rejects the others, an impulse will give different responses depending on when it enters the system.

For this reason, the easiest way to get the characteristics of the function is to go through the math, and do all calculations in the frequency domain.

6.1.1 Analysis Bank

The analysis will start with the outputs from the downsampler stages (commonly referred to as the analysis bank). In figures 6.1 and 6.2 these outputs are called $X_1$ to $X_6$, but any number of bands should be usable with the model. Using the noble identity of downsampling to lump the downsampler stages together (as in figure 6.2) one gets the following expression:

$$X_n(z) = X(z) \prod_{i=0}^{M-n} LPd(z^{2^{M-n-i}}) HPd(z^{2^{M+1-n}}) \downarrow 2^{M+1-n}, \text{ when } n \neq 1$$

$$X_1(z) = X(z) \prod_{i=0}^{M-1} LPd(z^{2^{M-n-i}}) \downarrow 2^{M+1-n}$$

(6.1)
Figure 6.1: System layout using six frequency bands. The different stages are called the analysis, processing and synthesis banks.
Figure 6.2: Equivalent system layout using six frequency bands.
where $M$ is the number of downsampling stages ($M+1$ is the number of bands) and $n$ is the band being studied. For example, the input to processing filter 4 in figure 6.2 would be:

$$X_{n}(z) = X(z) \prod_{i=0}^{M-n} LPd(z^{2^{M-n-i}})HPd(z^{2^{M+1-n}}) \downarrow 2^{M+1-n}$$

$$= X(z) \prod_{i=0}^{5-4} LPd(z^{2^{5-4-i}})HPd(z^{2^{6-4}}) \downarrow 2^{6-4}$$

(6.2)

$$= X(z) \prod_{i=0}^{1} LPd(z^{2^{1-i}})HPd(z^{4}) \downarrow 4$$

$$= X(z)LPd(z)LPd(z^{2})HPd(z^{4}) \downarrow 4$$

which match with the figure. The downarrow symbolizing the downsampling is a mathematical operator affecting everything to the left of it. Referring to the equation of downsampling, 2.1 on page 6, and using this in 6.1, the final expression of the analysis bank outputs is:

$$X_{n}(z) = \frac{1}{2^{M+1-n}} \sum_{k=0}^{M-n} \left[ X(z^{2^{n-M-1-k}}) \omega^{k}_{2^{M+1-n}} \right] \cdot \prod_{i=0}^{M-n} \left[ LPd(z^{2^{i-M-1+n}}) \omega^{2^{i-k}}_{2^{M+1-n}} \right] HPd(z), \text{ when } n \neq 1$$

(6.3)

$$X_{1}(z) = \frac{1}{2^{M}} \sum_{k=0}^{2^{M-1}} \left[ X(z^{2^{-M+k}}) \omega^{2^{i+k}}_{2^{M}} \prod_{i=0}^{M-1} \left[ LPd(z^{2^{i-M}}) \omega^{2^{i+k}}_{2^{M+1-n}} \right] \right]$$

where $\omega^{k}_{a} = e^{\frac{b \cdot 2\pi}{a}}$.

### 6.1.2 Synthesis Bank

Next, the outputs of the upsampling stages (commonly referred to as the synthesis bank) will be studied. This expression is much tidier:

$$Y_{out} = \sum_{n=1}^{M+1} \left[ Y_{n}(z) \uparrow 2^{M+1-n} \prod_{i=0}^{M-n} LPu(z^{2^{i}}) \right]$$

(6.4)

where the signal names are taken from figure 6.2. The expression inside the sum is the contribution from each band in the system. A note here is that if $n$ is greater than $M$ (which is true for the highest band), the product is said to be one.

### 6.1.3 Processing Stages

The processing stages are simple FIR-filters, which creates the outputs $Y_{n}$ from the inputs $X_{n}$ as $Y_{n} = F_{n}X_{n}$. The delay elements needed in each system path are incorporated together with the processing filters, as the sample rate is lowest there for each path through the system.
6.1.4 Total Transfer Function

Now everything may be summed up in one expression. Substituting $Y_n$ in equation 6.4 for $F_nX_n$, where $X_n$ is taken from equation 6.1 gives:

$$Y_{out} = \sum_{n=2}^{M+1} \left[ X(z) \prod_{i=0}^{M-n} LPd(z^{2M-n}) HPd(z^{2M+1-n}) \downarrow 2^{M+1-n} F_n(z) \uparrow 2^{M+1-n} \right] \cdot \prod_{i=0}^{M-n} LPu(z^{2^i}) + X(z) \prod_{i=0}^{M-1} LPd(z^{2^{M-1}}) \downarrow 2^{M+1-n} F_1(z) \uparrow 2^{M+1-1} \cdot \prod_{i=0}^{M-1} LPu(z^{2^i})$$

(6.5)

Finally, in order to get an expression that can be implemented in matlab, the arrows are substituted by equations 2.1 and 2.3:

$$Y_{out} = \sum_{n=2}^{M+1} \left[ \frac{1}{2^{M+1-n}} \sum_{k=0}^{2^{M+1-n}-1} X(z \omega_2^{k} \prod_{i=0}^{M-n} LPd(z^{2^i \omega_2^{2iM+1-n}})) \prod_{i=0}^{M-n} LPu(z^{2^i}) \right] \cdot HPd(z^{2^{M+1-n}}) F_n(z^{2^{M+1-n}}) \prod_{i=0}^{M-n} HPu(z^{2^i}) + \frac{1}{2^M} \sum_{k=0}^{2^M-1} X(z \omega_2^{k} \prod_{i=0}^{M-n} LPd(z^{2^i \omega_2^{2iM+n-1}})) \prod_{i=0}^{M-n} HPu(z^{2^i}) \right] \cdot F_1(z^{2^M}) \prod_{i=0}^{M-1} HPu(z^{2^i})$$

(6.6)

This is the formula that has been used when simulating and optimizing the system.

6.2 Time Domain Representation

While the system optimizations has been done using a frequency domain model, a time domain model had to be built in order to actually being able to use the filter. Using this model, an audio file could be filtered and then played. This was very useful in testing the system, as problems with audible aliasing, imaging and delays could be discovered. That information could then be used to change the optimization parameters.

6.3 Delay

Because the delay of the system should not be too high, a formula for the delay was needed. Looking at figure 6.1, the longest signal path is the one that goes through the lowest frequency band. On its way through the system, the signal passes through the first low pass filter at full sample rate, the second low pass filter at
half the sample rate, the third at a quarter, and so on. For every downsampling stage, the delay is doubled. If the number of taps of the low pass filters in the downsampler is \( N \) and the number of bands is \( M + 1 \), then the delay from \( X_{\text{in}} \) to the input to \( F_1 \) is:

\[
\sum_{i=1}^{M+1} \frac{N - 1}{2} 2^{M-i} = \frac{N - 1}{2} (2^M - 1)
\]  

(6.7)

If the same filter order is used for the filter in the upsampler stage, then the delay from \( Y_1 \) to \( Y_{\text{out}} \) is identical to the delay from \( X_{\text{in}} \) to \( X_1 \). The delay from the processing filters are ignored, as these are minimum phase and therefore have a shorter delay. The total delay through the system is then:

\[
D_{\text{tot}} = (N - 1)(2^M - 1)
\]  

(6.8)

In order to make sure that all paths through the system had this delay, delay elements were added to each path in conjunction with their processing blocks. The expression says that the delay of the system grows exponentially with the number of bands.

### 6.4 Computational Complexity

The main motivation of the new system was reducing the computational complexity of the equalizing. For that reason, a formula is provided for calculating the complexity of the system when using an arbitrary number of bands and arbitrarily large filters.

An important observation is that because the filters work at different rates, the average number of multiplications that needs to be made for each new sample is much less than the total number of filter taps in the system. Looking at the analysis bank, the first downsampler needs to be calculated for every new sample (as it runs at full speed), the second needs only to be run every other sample, the third every fourth sample, and so on. All in all, an average of two downsampler stages needs to be calculated for each sample. The same relation holds for the processing and synthesis banks. The average number of multiplications for each sample depends of course on the number of coefficients for the upsampler, downsampler, and the different processing filters.

### 6.5 Memory Requirement

While it is possible to save a lot of computations due to the downsampling, the memory usage will not benefit in the same way. Each filter needs to store its internal variables as well as coefficients. Furthermore, the delay elements that must be introduced to avoid harmful phase distortion translates to memory usage in the implementation. A comparison between systems of different number of bands is provided in figure 6.3.
Figure 6.3: Resource requirements and delay of systems with comparable amplitude responses but different number of bands.
Chapter 7

Filter Design

This section describes the work of designing the different filters in the system.

7.1 Method

The filters were designed using optimization in two stages with the Matlab function \textit{fminimax}. In order to make the optimization times shorter, initial guesses were first made for all filters. After the guessing, the analysis and synthesis bank filters (i.e. $LPd$, $HPd$ and $LPu$ in figure 6.1) were optimized separately from the processing filters. Then the processing filters were optimized while keeping the analysis and synthesis bank filters constant. An optimization of the full system was planned but never implemented due to lack of time.

7.2 Analysis and Synthesis Banks

The matlab function \textit{firpm} designs equiripple linear-phase filters from a given specification using the Parks-McClellan algorithm [4]. By giving this function the desired filter order and characteristics, the initial guesses for the filters in the analysis and synthesis banks were provided.

The optimization of these filters were then made such that the magnitude function of the system was as close to a constant one as possible when no processing was used. Furthermore, the optimization made sure that the attenuation of the interpolation filter was extra high for the highest frequencies, as these had larger problems with imaging. The formulation of the optimization problem is:
minimize $f$
subject to:

$$||H(e^{j\Omega}) - 1|| \leq f$$
$$|H_{LPd}(e^{j\Omega_{s0}})| \leq \frac{f}{W}$$
$$|H_{HPd}(e^{j\Omega_{s1}})| \leq \frac{f}{W}$$
$$|H_{LPu}(e^{j\Omega_{s2}})| \leq \frac{f}{W}$$
$$|H_{LPu}(e^{j\Omega_{s2end}})| \leq \frac{f}{100 \cdot W}$$

Where $f$ is the variable that is to be minimized, $W$ is a weight that can be used to shift the balance between passband flatness and stopband ripples, $H$ is the transfer function of the system, $H_{LPd}$ is the transfer function of the downsampler low pass filter $LPd$, $H_{HPd}$ is the transfer function of $HPd$, $H_{LPu}$ is the transfer function of $LPu$. $\Omega_{s0}$ is the stop band of $LPd$, $\Omega_{s1}$ is the stop band of $HPd$, $\Omega_{s2}$ is the stop band of $LPu$, and $\Omega_{s2end}$ is the highest part of the stop band of $LPu$, where extra attenuation is desired. By experimenting with different weights for the inequalities, the result could possibly be improved. For example, the weight of the $LPd$ stopband should probably be higher than that of $HPd$.

### 7.3 Processing Filters

The initial guesses of the processing filters were made using the Parks-McClellan algorithm, and then transforming the linear-phase results to minimum-phase filters with roughly the same magnitude functions. Different parts of the reference filter was used as reference for `firpm` for determining the different processing filters. For instance, the lowest frequency filter had the lowest parts of the reference filter as reference, while the highest used the whole spectrum as reference, with the exception that it should be attenuated below a certain frequency.

These guesses were then used in the optimization. The optimization had two goals. Firstly, it tried to minimize the amplitude error (in dB) compared to the reference filter. Secondly, a constraint was made such that the minimum-phase property was guaranteed. Imaging and aliasing were not considered in this optimization, as those effects were taken care of by the design of the up- and downsampler filters. The different results of guessing and optimizing an equalizer system is presented in figure 7.1.
The formulation of the optimization problem for the processing bank were as follows:

\[
\begin{align*}
\text{minimize} & \quad f \\
\text{subject to:} & \quad \left| \frac{d B}{d \Omega} \right| H(e^{j\Omega}) - \left| \frac{d B}{d \Omega} \right| G(e^{j\Omega}) \cdot W(e^{j\Omega}) \leq f \\
& \quad |\text{phase}(F_n(e^{j\pi}))| \leq \frac{\pi}{2}
\end{align*}
\]

Where \( G \) is the transfer function of the reference filter and \( W \) are weights for each point in the optimization. The weights can be used to put more emphasis on reducing the low frequency ripple, but will add some extra optimization time. The optimization points were logarithmically spread over the frequencies. When optimizing the system, care must be taken that the number of optimization points is large enough. The second line of the expression is to make sure that the processing filters \( F_n \) are of minimum phase type. This works because the phase shift of an FIR filter is an integer number of \( \pi \) at the Nyquist frequency (normalized to \( \pi \) in the expression) and the phase shift for the same frequency for a minimum phase filter is 0. This line may not be needed in the optimization, as successful optimizations have been done without it.
Figure 7.1

(a) Result before optimization

(b) Result after optimization of ten iterations and no weighting
(c) Result after optimization of ten iterations and more weighting on the lower frequencies

Figure 7.1: Amplitude response of the system before and after optimization.
Chapter 8

Implementation

An implementation of the system was made on an FPGA. Because the FPGA has limited resources, the implementation was made as cheaply as possible in terms of dedicated resources such as multipliers and memory. The implementations allow for a modular design, with different sizes for the different filters and a variable number of bands. For this reason, for a particular filter choice the implementation will not be completely optimized and may require some extra work. The implementation is based on the block schematic in figure 8.1. The downsampler high pass filter is calculated as the complements of the low pass filter, and the upsampler low pass filter is implemented in its polyphase form. For the reason of making the hardware implementation simple, the output from the downsampler high pass filter is negated. This means that instead of negating the output of the lowpass filter (which is temporarily stored in an accumulator) and adding the delayed input, the delayed input is subtracted directly from the low pass input. In this way a few clock cycles and some hardware is saved. In order to compensate for the error that arises in the sign, the upsampler uses subtraction of its inputs instead of addition.

8.1 Hardware Choice

The aim was to use a clock frequency that matched the reference implementation, but this had to be reduced. The reason for the trouble of meeting the original goal was the control unit, and the way it generates the addresses to the memories. This could probably be fixed by improving the design of the control unit, that was unproportionally large.

The computational complexity of the system was much lower than that of the reference filter. This allowed for the implementation to use only one Multiply Accumulate (MAC) block. The extra overhead associated with reading data, switching filters etc. turned out to be small, meaning that two equalizers could be scheduled on one MAC. Many times the sound will come in two channels that requires the same processing. For this reason the final implementation was made to
Figure 8.1: Block diagram of the filter. LPu1 and LPu2 means the first and second phase of the polyphase implementation of the interpolation filter.

filter two data streams simultaneously. The main focus in this chapter will however be on the implementation of filtering one data stream.

### 8.2 Block Layout

A simplified schematic of the filter implemented on one channel is provided in figure 8.2. The implementation uses one MAC unit. The target FPGA (Xilinx Spartan 3E) is a low-cost model and has no dedicated hardware MACs [7]. Instead, the MAC in the figure is implemented as a dedicated multiplier followed by an accumulation (that synthesizes into distributed logic and registers), along with a mux for bypassing the multiplier. The path from the Random Acces Memory (RAM), through the multiplier and to the addition in the MAC had to be pipelined so that it was not a bottleneck in the system. Pipelining was made before and after the multiplicator, but could be made more times if wanted, as long as some control signals and address buses are delayed accordingly.

The new input samples are first buffered into the input First In First Out (FIFO). All coefficients are stored in a Read Only Memory (ROM), and all internal states of the filters and delay elements are stored in a dual-port RAM, from which one word can be read and one written per cycle. The outputs of the filters are calculated using a MAC, and the finished result is loaded into an output FIFO. Because of the multirate structure of the system, not all sub-filters are calculated...
every sample. As long as the average number of clock cycles needed for calculating one sample is below the number of clock cycles on average available per sample, it does not matter if the maximum number of cycles to calculate is higher. The only effect will be that one or more extra samples will be buffered in the input FIFO.

### 8.3 Downsampler and Upsampler Implementation

The linear phase property of the filters of the down- and upsamplers can be utilized to save on the number of multiplications that needs to be made. In this implementation this fact has not been used. This is because only one word can be read out of the RAM per clock cycle. As the MAC can make one multiplication per cycle as well, it does not make sense to first read two outputs from the RAM, add them, and then multiply with a coefficient.

The upsampler has been implemented in its polyphase form, according to section 3.4. This allows us to calculate only half the number of multiplications each cycle, which offers at least some savings. The drawback is that the control unit must keep track of which phase of the upsampling filter that should be used.

### 8.4 Control Unit

A drawback of the system is that it requires much more control logic than a conventional FIR filter. Not only are there more filters to control, the control unit also need to keep track of which bands that needs be to calculated and when, and
Table 8.1: Scheduling of a system with three processing bands. '1' means that the band is calculated and that the interpolation filter of the upsampler in the band is run in the first phase, while a '2' means that the interpolation filter is run at the second phase. The lowest frequency band has no corresponding upsampler, so a run with this filter is indicated by 'x'.

which phase that should be used for the lowpass filter. The scheduling is shown in table 8.1.

If the control unit is not currently busy, it will start whenever the input FIFO is not empty. It then starts the filtering with the downsampler closest to the input, each clock cycle making one MAC operation and storing one state. The multiplications begin with the state closest to the FIR output, and ends with the input. When a state has been used in the multiplication it is moved forward in the chain of delay elements. Lastly, the complement is taken. The complement is actually negated here for the purpose of easing the implementation, but this error is compensated for later. All states as well as the outputs are stored on predetermined addresses in the block RAM. When the first downsampler has been calculated the control unit will move on to the next filter. Depending on which sample the system is currently processing are on, this could either be another downsampler or the top frequency processing filter.

The processing filters are simple FIR filters, and are run in very much the same way as the downsamplers (but with only one output). When these have been calculated, it is time to move on to the upsamplers. In order to calculate everything in the correct order, the control unit now starts with the lowest band that is to be calculated and moves upward. When reading the inputs, the accumulator also compensates for the negation made when taking the complement in the downsamplers.

When the system run is completed, the control unit will store the output in a FIFO, update its internal state (determining which filters that will be calculated next sample) and wait until another sample is ready in the input FIFO and then start again. A more detailed layout of the hardware and the control signals can be viewed in figure 8.3. Not included in this schematic are pipeline registers, scaling of the MAC output and logic for saturating the output if needed. The write enable signal for RAM port A (we-port A) is always low, as port A is only used for reading from the memory. Port B is used for writing to the memory, so the we-port enable signal for port B is usually high, but low during the cycles that nothing is written. The add/sub signal of the accumulator is used to specify whether the input should be added to or subtracted from the accumulated value. Values are only accumulated when the count enable signal (ce) is high, and the clear signal is low.
Figure 8.3: Detailed hardware layout of the system. Fat wires are signal and address buses and thin wires are control signals.
Chapter 9

Results

The octave band filtering scheme proposed can be used to efficiently implement audio equalizers where the requirements are harder on the lower part of the spectrum than the higher. Because of the need of using linear phase filters for splitting the frequency band into several parts, the delay and memory usage of the system grows quickly if many bands are used. Also, most of the savings in complexity are made if only about three to five bands are used.

An environment has been created for designing and implementing a complete system for use on the target FPGA. The results from each of those parts will be treated separately.

9.1 Design

Design of the system is made by first providing data about filter sizes etc. and then optimizing the system using this data. For the equalizer that was tested, the initial guess gave a magnitude error of less than one dB. After optimizing this for a few iterations, this error was reduced below a quarter of one dB. A comparison of different results is shown in figure 9.1. The worst part of the resulting imaging distortion can be viewed in figure 9.2, where the highest frequency filter has been left out in order to visualize the imaging on the upper half of the spectrum.

Something that has hardly been considered in this thesis is the phase characteristics of the system. Because the processing filters are of minimum phase type, there will be slightly different delays for a signal depending on which path it takes through the system. For frequencies that goes through two processing filters with comparable strength, this could result in strange transients. Although the time difference is small (below a tenth of a millisecond for the tested filters), it should be verified that this does not disturb a listener.

Example plots on different costs are shown in figure 9.3. As a rule of thumb, if around three to five bands are used, savings can be made for both memory and computations, while keeping the delay at reasonable levels. The plots represents only one example, but the same pattern applies for other designs. Good to bear in mind is that the delay of the system is only affected by the order of the upsampler
(a) Result before optimization

(b) Result after optimization of ten iterations and no weighting

Figure 9.1
(c) Result after optimization of ten iterations and more weighting on the lower frequencies

Figure 9.1: Amplitude response of the system before and after optimization.

and downsampler filters, and the number of bands used. A large delay will directly translate to a large memory usage, but if more bands are used the last processing filter can be smaller than otherwise, meaning that the ROM can be smaller.

9.2 Implementation

The implementation assumes that the computational complexity is not larger than that it can be implemented using only one MAC. If higher complexity is needed, a good start would be to increase the clock frequency. Currently, the frequency could be increased by around 50%, but the goal is to double it. If the control unit is reduced in size, the frequency requirements will probably follow as well.

A typical implementation would require 1 dedicated multiplier and 3 block RAMs of 1024 18-bit words for one channel, or 5 block RAMs for two channels (the block RAM used as a ROM to store the coefficients can be reused). The control unit would require about 300 to 400 Look Up Tables (LUTs), which is much compared with the reference system. If the implementation is to be used, the control unit should be trimmed. The rest of the system requires around 200 LUTs for the FIFOs, additions, muxes etc.
Figure 9.2: The worst imaging occurs in the upper half of the output spectrum, because the imaging from all the upsamplers affects that part.
9.2 Implementation

Figure 9.3
Figure 9.3: Resource requirements and delay of systems with comparable amplitude responses
Chapter 10

Future Work

There are a number of things that could be improved in this thesis. This chapter will point out a few areas in both the design and implementation that could benefit from some extra work.

10.1 Design Improvements

One way to potentially improve the results of the design phase is to optimize the full system (both the analysis and synthesis banks and the processing filters) at the same time. This was tested, but never made to work. If this is done, one has to make sure that the stopband attenuation of the anti-alias and anti-imaging filters are not compromised. An educated guess is that the savings would not be huge - as these filters should have already been made as small as possible (for reasons of the delay), it may not be possible to change their passband properties much. If an optimization of the whole system is made, it may also be good to look into using different filters for the different up- and downsampler stages. This would require some extra memory for storing new filter coefficients in the implementation, but on the other hand there is the potential of saving some on the processing filters instead.

Because the analysis bank low pass filter ($LP_d$) is directly followed by a down-sampling by two, this filter could be implemented in a poly phase form such that only half the multiplications needs to be made each cycle. However, because the high pass filter ($HP_d$) is implemented as the complement of $LP_d$, this fact has not been used. An alternative to the current design would therefore be to design these two filters separately. $LP_d$ would then only require half the number of multiplications each sample, and $HP_d$ could be designed independently of $LP_d$. Because the function of $HP_d$ is not as critical, this filter could be made smaller.

The fact that the stop band attenuation of $HP_d$ is not as critical as that of $LP_d$ could be utilized better in the optimization, where the weighting of the two stop bands have been the same. Some quick tests has suggested that by reducing the weight on the $HP_d$ stopband, the overall performance may be increased, possibly making it feasible to reduce the order for said filters by a small amount. Overall,
experimenting with different optimization parameters such as cutoff, rolloff and stop band and pass band weighting for the different filters may improve the design considerably.

As the delay of the system has been a bottleneck, the use of Lattice Wave Digital Filters (LWDF) has been considered. These may be designed to have a near-linear phase property with considerably reduced complexity compared to linear phase FIR filters. Designing these filters is quite a task in itself, and requires some optimization. Further, the implementation is not as straight-forward as that of FIR filters.

### 10.2 Implementation Improvements

The main thing that needs to be improved in the hardware part of the thesis is the control unit. The first step should be to make a hardware block for generating the addresses to the memories. In the current implementation purely behavioral code is used for this, probably resulting in more hardware than necessary.

Currently, a full block RAM is used for the ROM storing the coefficients. The RAM can hold 1024 words, but depending on the implementation not all of these may be used. As the block RAM has two ports, the extra memory could probably be used for another application if wanted.

The Matlab functions for generating code for the implementation could also be improved by calculating limits for how large different integer signals and variables can be. Currently, this must be done manually (if not, all integers will be larger than necessary, resulting in wasteful hardware use).

### 10.3 Testing

Testing of the system has been made on all levels down to hardware simulations. Because of problems with a sound card, it has not been possible to do precise testing with audio. Because the system does not have the exact same properties as the reference filter, the system should be subject to blind testing to see if these differences are audible. The main difference is that the phase of the new filter is not like that of either linear phase or minimum phase filters. This may lead to a different perception of the sound, especially in the band-splitting regions. Further, the magnitude properties may not be close enough to the reference system even after optimization.
Bibliography


