Examensarbete

Design and implementation of a decimation filter using a multi-precision multiply and accumulate unit for an audio range delta sigma analog to digital converter

Erik Lindahl

LiTH - ISY - EX - - 08 / 4075 - - SE
Design and implementation of a decimation filter using a multi-precision multiply and accumulate unit for an audio range delta sigma analog to digital converter

Department of electrical engineering, Linköpings Universitet

Erik Lindahl

LiTH - ISY - EX - - 08 / 4075 - - SE

Examensarbete: 30 hp

Level: D

Supervisor: Oscar Gustafsson,
Department of electrical engineering, Linköpings Universitet

Examiner: Oscar Gustafsson,
Department of electrical engineering, Linköpings Universitet

Linköping: februari 2008
Design and implementation of a decimation filter using a mult-precision multiply and accumulate unit for an audio range delta sigma analog to digital converter

Erik Lindahl

Abstract

This work presents the design and implementation of a decimation filter for a three bits sigma delta analog to digital converter. The input is audio with a oversampling ratio of 32. Filter optimization and tradeoffs concerning the design is described. The filter is a multistage filter consisting of two cascaded FIR filters. The arithmetic unit is a multi-precision unit that can handle three or 24 bits MAC operations. The designed decimation filter is synthesized on standard cells of a 0.13 µm CMOS library.

Nyckelord
decimation, digital filter, FIR, hardware implementation, multi precision, delta sigma
Abstract

This work presents the design and implementation of a decimation filter for a three bits sigma delta analog to digital converter. The input is audio with a oversampling ratio of 32. Filter optimization and tradeoffs concerning the design is described. The filter is a multistage filter consisting of two cascaded FIR filters. The arithmetic unit is a multi-precision unit that can handle three or 24 bits MAC operations. The designed decimation filter is synthesized on standard cells of a 0.13 $\mu$m CMOS library.

Keywords: decimation, digital filter, FIR, hardware implementation, multi-precision, delta sigma
Acknowledgements

I would like to thank my supervisor Oscar Gustafsson, my opponent Johannes Lindblom, Hanna Svensson, Oskar Matteusson and Krister Berglund.
Nomenclature

Most of the reoccurring abbreviations and symbols are described here.

Symbols

$h(n)$  impulse response  
$H(z)$  transfer function  
$M$  decimation rate  
$L$  number of subfilters  
$N$  filter order  
$R(\omega T)$  noise power spectral density  
$b$  word length  
$c$  computational load  
$\omega_c T$  passband edge  
$\omega_s T$  stopband edge  
$\delta$  passband ripple

Abbreviations

A/D analog to digital converter  
SNR signal to noise ratio  
NTF noise transfer function  
PE processing element  
acc accumulator  
RTL register transfer language  
DC Design compiler  
VHDL hardware description language  
MAC Multiply and accumulate  
$\Delta\Sigma$ Delta sigma
Contents

1 Introduction
   1.1 The task .............................................. 1
   1.2 Method of solving ................................... 1
   1.3 Report outline ....................................... 3

2 Theory
   2.1 Delta-Sigma A/D converter ............................. 5
   2.2 FIR and IIR filters .................................. 6
   2.3 Decimation filter ..................................... 6
       2.3.1 Downsampling ................................... 6
       2.3.2 Lowpass filter .................................. 7
   2.4 Polyphase decomposition ............................. 8

3 Matlab model
   3.1 Filter specification ................................. 9
       3.1.1 Phase response ................................ 10
   3.2 Multistage decimation ............................... 10
       3.2.1 Computational load ............................. 11
       3.2.2 Half band filter ................................ 12
       3.2.3 Simulation results and computational complexity . 12
   3.3 The architecture ...................................... 13
       3.3.1 Dataflow ......................................... 13
       3.3.2 Data memory ..................................... 13
       3.3.3 Clock frequency ................................ 15
   3.4 Filter optimization .................................. 16
   3.5 Coefficient word length ............................. 17
       3.5.1 Constant zero bits in coefficients .............. 19
   3.6 Schedule .............................................. 20
   3.7 DC level .............................................. 20
   3.8 A test case ........................................... 21

4 Hardware implementation ................................. 23
   4.1 VHDL model ........................................... 23
   4.2 Processing element ................................. 25
       4.2.1 What is it doing? ............................... 25
       4.2.2 The main idea .................................. 25
       4.2.3 Sign extension .................................. 26
       4.2.4 Internal wordlength ............................ 26

Lindahl, 2008.
<table>
<thead>
<tr>
<th>Contents</th>
</tr>
</thead>
<tbody>
<tr>
<td>4.3 Synthesis ........................................ 27</td>
</tr>
<tr>
<td>4.3.1 Synthesis results .............................. 28</td>
</tr>
<tr>
<td>4.3.2 Validation ....................................... 28</td>
</tr>
<tr>
<td>5 Future work ........................................ 31</td>
</tr>
<tr>
<td>A Filter coefficients ................................. 35</td>
</tr>
</tbody>
</table>
Chapter 1

Introduction

To realize high resolution analog to digital converters without high precision analog components one can use the technique of oversampled delta sigma converters. In these converters digital decimation filters are essential parts. This work considers the design and implementation of such a digital decimation filter.

1.1 The task

The task was to implement a synthesizeable decimation filter for a given delta sigma A/D converter ($\Delta\Sigma$). The system in Fig 1.1 shall fulfill the specifications in Table 1.1. The decimation filter shall attenuate the noise created in the $\Delta\Sigma$, see Fig 1.3, reduce the sample rate by a factor 32 and increase the precision from three bits to 16 bits. In Fig 1.2 a graph of the desired filter and the passband ripple requirements is shown.

<table>
<thead>
<tr>
<th>Passband frequency</th>
<th>$\omega_c$</th>
<th>0.4895 (Normalized)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Stopband frequency</td>
<td>$\omega_s$</td>
<td>0.5688 (Normalized)</td>
</tr>
<tr>
<td>Passband ripple</td>
<td>$\delta_c$</td>
<td>&lt; 0.035 dB</td>
</tr>
<tr>
<td>Signal to noise ratio</td>
<td>$SNR$</td>
<td>91 dB</td>
</tr>
<tr>
<td>Decimation rate</td>
<td>$M$</td>
<td>32</td>
</tr>
<tr>
<td>Data sampling rate</td>
<td>$f_s$</td>
<td>44.1 kHz</td>
</tr>
<tr>
<td>Phase response</td>
<td></td>
<td>linear</td>
</tr>
</tbody>
</table>

Table 1.1: System specifications.

The given $\Delta\Sigma$ introduces noise to the analog signal, this noise is referred to as the noise transfer function ($NTF$), see figure 1.3. The filter will be designed to match this $NTF$ as close as possible.

1.2 Method of solving

The starting point in this work is the filter specification. Then Matlab was used to design a model of the filter. The Matlab model and the hardware architecture was designed simultaneously, because some architectural decisions affects the Matlab model and vice versa. The filter model created in Matlab...
Figure 1.1: Overview of the system. This work considers the design and implementation of the decimation filter $H(z)$.

Figure 1.2: Filter specification. Left: Passband ripple requirement. The dashed lines indicates the maximum passband ripple. Right: The desired ideal low pass filter.

Figure 1.3: Noise transfer function, $NTF$. 

\[ H(z) \]
\[ x(t) \rightarrow \Delta \Sigma \rightarrow \frac{3}{x(n)} \rightarrow H(z) \rightarrow \frac{16}{x(32n)} \]

\[ \text{analog} \]

\[ x(t) \quad 1.41 \text{ MHz} \]

\[ x(n) \quad 44.1 \text{ kHz} \]
was implemented in VHDL using HDL Designer. Finally the VHDL model was synthesized to a gate level design in Design Compiler. A flow graph of the design process can be seen in Fig1.4.

1.3 Report outline

The report follows the design flow, see Fig1.4. A theoretical background is given in chapter 2. Chapter 3 contains the design of a Matlab model. The hardware architecture of the Matlab model is described in chapter 4. Some ideas of improvements in future work can be found in chapter 5.
Chapter 2

Theory

This chapter contains some background theory to this work. First, a brief introduction to delta sigma, then some theory in decimation filtering, and finally polyphase decomposition is explained.

2.1 Delta-Sigma A/D converter

The delta sigma technique makes it possible to realize high-resolution analog to digital conversion without high precision analog components [1]. One feature of the delta sigma is that it instead of sampling at the Nyquist frequency $f_N$ the analog signal is oversampled by an oversampling ratio $M$, this means that the sampling frequency is much larger than the Nyquist frequency. In an A/D converter that samples at the Nyquist frequency the quantization noise is uniformly distributed over the frequency band $0$ to $f_s/2$. An A/D converter that is oversampled by a factor $M$ spreads the noise spectra over a bandwidth that is $M$ times larger. In addition to this the delta-sigma A/D converter moves the quantization noise so most of the noise lands outside the band $0$ to $f_s/(2M)$, this is referred to as noise shaping. See Fig 2.1 [7].

![Figure 2.1:](image)

Figure 2.1: a) Quantization noise spectrum with sampling at the Nyquist rate.
b) The quantization noise spectrum when oversampled by a factor $M$. c) The quantization noise spectrum when oversampled by a factor $M$ and noise shaped in a ΔΣ.

Lindahl, 2008.
2.2 FIR and IIR filters

Filters are usually distinguish between FIR (finite length impulse response) and IIR (infinite length impulse response). IIR filters can only be realized by using recursive algorithms and FIR can be realized with a recursive or non recursive algorithm, though recursive FIR filters are seldom used as they suffer from stability problems [4].

The advantages of FIR over IIR filters are that they can have a linear phase response, they are always stable, and they are easy to implement with polyphase decomposition. On the other hand FIR filters require much higher filter orders and introduce a large group delay [4]. In this case the filter must have a linear phase response and the filter will use different data rates. Because of the linear phase property and that FIR filters are easy to implement in an polyphase decomposition only FIR filters will be considered further on in this work.

The length of an impulse response for an FIR filter of order \( N \) is \( N + 1 \). If the impulse response is symmetric or antisymmetric around \( n = N/2 \) the filter has a linear phase response. The transfer function of an \( N \)th order FIR filter with impulse response \( h(n) \) can be written as:

\[
H(z) = \sum_{n=0}^{N} h(n)z^{-n}
\]  

2.3 Decimation filter

The ∆Σ makes the A/D conversion at a low precision and a high samplerate. The task for the decimation filter is to reduce the sample rate by a factor \( M \) and increase the precision. The decimation is done in two steps consisting of a filter followed by downsampling. The filter is a lowpass filter that shall prevent aliasing. The downsampling reduces the sampling rate by a factor \( M \). In Fig 2.2 an overview of the decimation filter is shown and Fig 2.3 presents an example of how a signal is decimated.

2.3.1 Downsampling

Downsampling of a signal \( x_2(n) \) by a factor \( M \) means that a new signal \( x_3(n) \) is created by extracting every \( M \):th sample in \( x_2(n) \) the other samples are neglected.

\[
x_3(n) = x_2(Mn)
\]
2.3. Decimation filter

Figure 2.3: a) Magnitude response for the signal $X_1(\omega T_1)$. Bandwidth $\omega_m$. b) Ideal lowpass filter with cut off frequency $\pi/M$. c) Magnitude response for $X_2(\omega T_1) = X_1(\omega T_1)H(\omega T_1)$. d) Downsampled version of $X_2(\omega T_1)$. $T_2 = T_1/M$

Figure 2.4: Lowpass filter with cut off frequency $\pi/M$

This operation has effects in the Fourier domain. The Fourier transform to $x_3(n)$ can be derived as.

$$X_3(\omega T_2) = \frac{1}{M} \sum_{k=0}^{M-1} X_2(\frac{\omega T_1}{M} - k2\pi)$$  \hspace{1cm} (2.3)

This means that the spectra $X_2(\omega T_1)$ is repeated $M$ times with the distance $2\pi/M$ between every recurrence, scaled by a factor $1/M$ [2]. In Fig 2.3 an example of the effects of decimation is given.

2.3.2 Lowpass filter

Before downsampling the signal have to be bandlimited to $\pi/M$ to avoid aliasing, therefore the downsampling step have to be preceded by a low pass filter. The filter attenuates the frequency components in the region $\pi/M$ to $\pi$. See Fig 2.4.
Figure 2.5: A direct form FIR polyphase decomposition of a filter with order N.

### 2.4 Polyphase decomposition

The straightforward realization of a decimation filter is to first have a lowpass filter and then neglect every $M$th sample. This means that most of the output samples from the filter is discarded, only every $M$th sample is used. This can be exploited to reduce the computational workload. The main idea of polyphase decomposition is to only calculate those samples that are not discarded. This can be realized by dividing the filter $H(z)$ into $L$ sub filters $H_i(z)$. Where the impulse response for each sub filter is:

$$h_i(n) = h(nM + i), \quad i = 0, 1, \ldots, M - 1$$

For a detailed explanation see [4].
Chapter 3

Matlab model

In this chapter the design of the Matlab model is presented. The starting point is the filter specification.

3.1 Filter specification

The main task for the filter is to attenuate the noise created in the delta-sigma A/D converter and to reduce the sample rate by a factor $M = 32$. To handle this the noise transfer function $NTF(\omega T)$ (see Fig 1.3) from the delta sigma was used to derive a filter specification $H_{spec}(\omega T)$.

The quantization noise power spectral density $R_x(\omega T)$ and the noise power spectral density after filtering $R_y(\omega T)$ is found as [1]

\[
R_x(\omega T) = \frac{Q^2}{12} |NTF(\omega T)|^2 ;\quad Q = 2^{-b_{in}+1} \tag{3.1}
\]

\[
R_y(\omega T) = |H_{spec}(\omega T)|^2 R_x(\omega T) \tag{3.2}
\]

where $Q$ is the quantization step in the A/D conversion, $b_{in}$ is the number of bits that presents the input signal, in this case $b_{in} = 3$. $R_y(\omega T)$ is assumed to be lower than a constant $\varepsilon$. The noise power $P_{noise}$ is then found as

\[
P_{noise} = \frac{1}{\pi} \int_{0}^{\pi} \varepsilon d\omega T\quad \Rightarrow \quad P_{noise} = \varepsilon \geq R_y(\omega T) \tag{3.3}
\]

\[
R_x(\omega T)|H_{spec}(\omega T)|^2 \leq \sigma_{noise}^2 \quad \Rightarrow \quad |H_{spec}(\omega T)| \leq \frac{P_{noise}}{R_x(\omega T)} \tag{3.4}
\]

\[
SNR = 10 \ast \log_{10}\left(\frac{P_{signal}}{P_{noise}}\right) \quad \Rightarrow \quad \sigma_{noise}^2 = \frac{P_{signal}}{10 \ast SNR} \tag{3.5}
\]

The signal is a sinusoid which power is $P_{signal} = 1/2$. By combining equation 3.3, 3.4 and 3.5 the filter specification can be derived as.

\[
|H_{spec}(\omega T)| = \sqrt{\frac{24}{10^{(\frac{SNR}{10})}|NTF(\omega T)|^2}} \tag{3.6}
\]

See Fig 3.1 for a plot of the derived filter specification.

Lindahl, 2008.
Figure 3.1: A plot of the filter specification. The dashed line indicates the transition band.

3.1.1 Phase response
The filter specification derived above only limits the magnitude function. The phase response have to be linear, this is achieved by using a linear phase FIR filter.

3.2 Multistage decimation
If the overall sampling rate conversion ratio can be factored into the product

$$
\prod_{i=1}^{L} M_i = M
$$

(3.7)

where each $M_i$ is an integer, the decimation filter can be implemented using $L$ cascaded sub filters. In this chapter the optimum number of sub filters and their corresponding decimation rates will be determined.

All sub filters works in different data rates, the last sub filters have a lower data rate than the first ones. A low data rate result in a low computational work load, the number of computations per output sample is linear dependent of the data rate.

One also have to take into account the wordlength at each sub filter. The wordlength to the first sub filter is only three bits, it will be much higher to the other sub filters, due to multiplications and additions. To make use of this fact, downsampling by a large factor at the first sub filter is preferred. For example the filter structure with downsampling factors 16 and 2 will be examined, but not the opposite case with downsampling factors 2 and then 16.

Seven different filter structures have been investigated. These structures are presented in Fig 3.3. To find the optimum of these structures the Matlab function firpm has been used. Given the filter order, stopband and passband edges the firpm function returns an impulse response that is optimized with the McClellan-Parks-Rabiner algorithm [9].

To find the required filter order for each sub filter, the filter order was iteratively increased until the filter met the filter specification that was derived in
3.2. Multistage decimation

\[
x(n) \xrightarrow{H_z} 1_M \xrightarrow{y(m)} x(n) \xrightarrow{H_1(z)} 1_M \xrightarrow{y(m)} \cdots \xrightarrow{H_k(z)} 1_M \xrightarrow{y(m)} \]

Figure 3.2: Multi stage decimation.

\[
\begin{align*}
1 & \quad x(n) \xrightarrow{H_{11}(z)} 1_2 \xrightarrow{H_{12}(z)} 1_2 \xrightarrow{H_{13}(z)} 1_2 \xrightarrow{H_{14}(z)} 1_2 \xrightarrow{H_{15}(z)} 1_2 \quad y(m) \\
2 & \quad x(n) \xrightarrow{H_{21}(z)} 1_4 \xrightarrow{H_{22}(z)} 1_4 \xrightarrow{H_{23}(z)} 1_4 \xrightarrow{y(m)} \\
3 & \quad x(n) \xrightarrow{H_{31}(z)} 1_8 \xrightarrow{H_{32}(z)} 1_8 \xrightarrow{H_{33}(z)} 1_8 \xrightarrow{y(m)} \\
4 & \quad x(n) \xrightarrow{H_{41}(z)} 1_8 \xrightarrow{H_{42}(z)} 1_8 \xrightarrow{H_{43}(z)} 1_8 \xrightarrow{y(m)} \\
5 & \quad x(n) \xrightarrow{H_{51}(z)} 1_8 \xrightarrow{H_{52}(z)} 1_8 \xrightarrow{y(m)} \\
6 & \quad x(n) \xrightarrow{H_{61}(z)} 1_16 \xrightarrow{H_{62}(z)} 1_16 \xrightarrow{y(m)} \\
7 & \quad x(n) \xrightarrow{H_{7}(z)} 1_32 \xrightarrow{y(m)} 
\end{align*}
\]

Figure 3.3: Seven filter structures that was taken under consideration.

section 3.1. A schematic of the iteration is presented in Fig 3.4. The results are presented in table 3.1.

### 3.2.1 Computational load

Since each sub filter works at different sample rates a sub filter running at a high rate will need more calculations per output sample compared to a filter stage running at a low rate. The computational load \(c\) for each filter structure is estimated through calculation of the number of multiplications per output sample.

In addition to this one have to take into account that each filter stage uses different word lengths. The word length at the first stage is 3, further on I have assumed that the word length at the inputs to all other stages is 24, i.e. the computational load for the first filter stage is a factor \(3/24 = 1/8\) lower.

\[
c = \sum_{i=1}^{L} \left( \frac{N_i + 1}{fsf} \right) \prod_{j=i+1}^{L} M_j \tag{3.8}
\]

\[
fsf = \begin{cases} 
8 & \text{if } i = 1 \\
1 & \text{when others}
\end{cases} \tag{3.9}
\]

**Example:** Filter structure 4 have three sub filters, \((L = 3)\). The decimation factors and filter orders are: \(M_1 = 8, M_2 = 2, M_3 = 2, N_1 = 43, N_2 = 16, N_3 = 59\). \(c\) is calculated with equation 3.8.

\[
c = \frac{43 + 1}{8} \cdot 2 \cdot 2 + (16 + 1) \cdot 2 + 59 + 1 = 116 \tag{3.10}
\]
3.2.2 Half band filter

To implement the antialiasing filters one can use half band filters, which can reduce the computational load. A half band filter with the impulse response $h(n)$ have the property that:

$$h(2p) = 0 \quad \text{for} \quad p \neq 0$$  \hfill (3.11)

Or in other words, every second filter tap in the impulse response will be zero except for the tap at $n = 0$. This means that the number of multiplications required for a half band filter of order $N$ will be [5]:

$$\text{multiplications} = \begin{cases} \frac{N}{2} & \text{if } N \text{ even} \\ \frac{N-1}{2} & \text{if } N \text{ odd} \end{cases}$$  \hfill (3.12)

The drawback of half band filters is that the magnitude function must be symmetric with respect to $\pi/2$. This also means that the stopband and passband ripples must be equal. This limitation will result in higher filter orders and perhaps also more multiplications.

3.2.3 Simulation results and computational complexity

The results in table 3.1 have been evaluated by using the algorithm in Fig 3.4 and the equations 3.8 and 3.9 have been used to estimate the computational complexity of each filter structure. With aid of these results a filter structure was chosen that will be implemented.

Of main interest in table 3.1 is the estimated computational complexity $c$ and $c_{hb}$ for each filter structure. $c_{hb}$ is the computational complexity when half band filters is used. As one can see in the table the lowest value of $c$ or $c_{hb}$ is for filter structure 4 if half band filters are used. Structure 4 results in the lowest complexity even if halfband filters are not used. Forcing a filter into a half band filter puts constraints on the filter that will result in higher filter orders. In Fig 3.5 filters 4 and 6 are compared to the filter specification. In both cases...
3.3. The architecture

Before continuing with the design of the Matlab model the hardware architecture have to be considered.

The main idea of the architecture is to use one memory for storing data, one coefficient memory and one processing element that performs the convolutions. A buffer at the input stores \( l \) input samples. The processing element works in different modes depending on the data wordlength.

3.3.1 Dataflow

In this section a description of how data goes from input to output is given. The input is three bits wide and \( l \) inputs are buffered and stored in the data memory. When the first sub filter shall be evaluated data is read from data memory to the processing element, eight data samples are processed in parallel. Results are written back to the data memory. At last the second sub filter is evaluated through reading data from memory to the processing element and then update the output.

3.3.2 Data memory

The data memory shall store input data and results from the first sub filter. \( l \) inputs are stored in one word in the memory. To make the read and writes as simple as possible \( l \) is a power of two (\( l = 2^n \)), the wordlength in memory is then

![Figure 3.5: Two filters compared to the filter specification. To the left is structure 6 and to the right structure 4.](image-url)
<table>
<thead>
<tr>
<th>Sub Filter</th>
<th>$H_{11}$</th>
<th>$H_{12}$</th>
<th>$H_{13}$</th>
<th>$H_{14}$</th>
<th>$H_{15}$</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>Decimation</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>$N+1$</td>
<td>6</td>
<td>6</td>
<td>10</td>
<td>16</td>
<td>55</td>
<td></td>
</tr>
<tr>
<td>$c$</td>
<td>12</td>
<td>48</td>
<td>40</td>
<td>32</td>
<td>55</td>
<td>187</td>
</tr>
<tr>
<td>$c_{hb}$</td>
<td>6</td>
<td>24</td>
<td>20</td>
<td>16</td>
<td>55</td>
<td>121</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Sub Filter</th>
<th>$H_{21}$</th>
<th>$H_{22}$</th>
<th>$H_{23}$</th>
<th>$H_{24}$</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>Decimation</td>
<td>4</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>$N+1$</td>
<td>17</td>
<td>10</td>
<td>14</td>
<td>63</td>
<td></td>
</tr>
<tr>
<td>$c$</td>
<td>17</td>
<td>40</td>
<td>28</td>
<td>63</td>
<td>148</td>
</tr>
<tr>
<td>$c_{hb}$</td>
<td>17</td>
<td>20</td>
<td>14</td>
<td>63</td>
<td>114</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Sub Filter</th>
<th>$H_{31}$</th>
<th>$H_{32}$</th>
<th>$H_{33}$</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>Decimation</td>
<td>4</td>
<td>4</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>$N+1$</td>
<td>17</td>
<td>34</td>
<td>94</td>
<td></td>
</tr>
<tr>
<td>$c$</td>
<td>17</td>
<td>68</td>
<td>94</td>
<td>179</td>
</tr>
<tr>
<td>$c_{hb}$</td>
<td>17</td>
<td>68</td>
<td>94</td>
<td>173</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Sub Filter</th>
<th>$H_{41}$</th>
<th>$H_{42}$</th>
<th>$H_{43}$</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>Decimation</td>
<td>8</td>
<td>2</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>$N+1$</td>
<td>44</td>
<td>17</td>
<td>60</td>
<td></td>
</tr>
<tr>
<td>$c$</td>
<td>22</td>
<td>34</td>
<td>60</td>
<td>116</td>
</tr>
<tr>
<td>$c_{hb}$</td>
<td>22</td>
<td>17</td>
<td>60</td>
<td>99</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Sub Filter</th>
<th>$H_{51}$</th>
<th>$H_{52}$</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>Decimation</td>
<td>8</td>
<td>4</td>
<td></td>
</tr>
<tr>
<td>$N+1$</td>
<td>44</td>
<td>140</td>
<td></td>
</tr>
<tr>
<td>$c$</td>
<td>22</td>
<td>139</td>
<td>161</td>
</tr>
<tr>
<td>$c_{hb}$</td>
<td>22</td>
<td>139</td>
<td>161</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Sub Filter</th>
<th>$H_{61}$</th>
<th>$H_{62}$</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>Decimation</td>
<td>16</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>$N+1$</td>
<td>177</td>
<td>89</td>
<td></td>
</tr>
<tr>
<td>$c$</td>
<td>44</td>
<td>89</td>
<td>133</td>
</tr>
<tr>
<td>$c_{hb}$</td>
<td>44</td>
<td>89</td>
<td>133</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Sub Filter</th>
<th>$H_{7}$</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>Decimation</td>
<td>32</td>
<td></td>
</tr>
<tr>
<td>$N+1$</td>
<td>1521</td>
<td></td>
</tr>
<tr>
<td>$c$</td>
<td>190</td>
<td>190</td>
</tr>
<tr>
<td>$c_{hb}$</td>
<td>190</td>
<td>190</td>
</tr>
</tbody>
</table>

Table 3.1: Required filter order and computational complexity for each sub filter. $N$ is the filter order, $c$ is the computational complexity when halfband filters are not used, $c_{hb}$ is the computational complexity when half band filters are used. The indexes at each sub filter refers to Fig 3.3.
3.3. The architecture

<table>
<thead>
<tr>
<th>Filter structure</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
</tr>
</thead>
<tbody>
<tr>
<td>c</td>
<td>232</td>
<td>157</td>
<td>177</td>
<td>136</td>
<td>200</td>
<td>121</td>
<td>190</td>
</tr>
</tbody>
</table>

Table 3.2: Computational complexity. Note that these numbers are wrong.

![Diagram of the architecture](image)

Figure 3.6: Main idea of the architecture. \(b_{in}\) is the input word length, \(b_1\) the data word length between the sub filters and \(b_c\) is the coefficient word length. PE is the processing element.

\[ b_1 = 3 \times 2^v \quad (3.13) \]

\(b_1\) is also the number of bits that represents the result from the first filter stage. \(b_1 = 6\) and \(b_1 = 12\) are to few bits and \(b_1 = 48\) too many. Then the best choice is \(b_1 = 24\). Consequently \(l\) is set to \(24/3 = 8\). The data is saved in memory as in Fig 3.7.

The number of words needed in the data memory is determined by the length of the impulse responses of the filters. Because \(l\) inputs are stored in one word in memory, the number of words needed for the first sub filter will be reduced by a factor \(l\). The total number of words in the memory will be:

\[ \left\lceil \frac{N_1 + 1}{l} \right\rceil + N_2 + 1 \quad (3.14) \]

3.3.3 Clock frequency

To determine the clock frequency \(f_{clk}\) one have to know the number of clock cycles used to produce one output. If the output have the frequency \(f_{sample}\) then \(f_{clk}\) is:

a) \[
\begin{array}{cccccccc}
3 & 3 & 3 & 3 & 3 & 3 & 3 \\
\end{array}
\]

b) \[
\begin{array}{c}
24 \\
\end{array}
\]

Figure 3.7: Data formats. a) Eight inputs is stored in one word in the data memory. b) The result from the first sub filter is 24 bits wide one word is stored on each line in data memory.
\[ f_{\text{clk}} = K \times f_{\text{sample}} \]  

(3.15)

Where \( K \) is the number of cycles needed to produce one output. To make a simple design \( K \) should be chosen to a power of two.

\[ K = 2^x \]  

(3.16)

A bottleneck in the design is the data memory which only can perform one read or one write each clock cycle. To produce one output 32 input samples have to be read. Due to decimation 8 samples can be written at one single line in memory, hence \( 32/8 = 4 \) cycles must be used to read input values in memory. The decimation rate at the first filter stage is 16, then the results from this filter stage have to be stored \( 32/16 = 2 \) times per output sample. This ends up in the following equation:

\[ 2 \frac{N_1 + 1}{8} + (N_2 + 1) + 4 + 2 = 2^x \]  

(3.17)

In section 3.4 \( x \) will be determined.

### 3.4 Filter optimization

The filters derived in chapter 3.2 fulfills the specification with a large margin and the only requirement in the specification is a question of \( SNR \). This might imply that the filter orders can be lower.

In order to get an optimal filter, the filter coefficients given by Parks-McClellan algorithm was optimized using the \texttt{fminimax} function in Matlab. The optimization maximizes the \( SNR \) subject to the requirements on the ripple in the passband in table 1.1. The optimization problem is formulated as:

\[
\text{maximize } SNR \quad \text{(3.18)}
\]

subject to \( 1 - \delta_c \leq |H(\omega_{pb})| \leq 1 + \delta_c \quad \omega_{pb} \in [0 : \omega_c] \)  

(3.19)

In order to determine \( 2^x \) in equation 3.17 four different values of \( 2^x \) have been examined. For each \( 2^x \) \( N_1 \) and \( N_2 \) have been chosen so they fulfill equation 3.17 and \( 2N_1 \approx N_2 \). An \( SNR \) value have been derived with the optimization technique for each \( N_1, N_2 \). See table 3.3.

\[
\begin{array}{ccc}
2^x & N_1 & N_2 & SNR \\
32 & 47 & 19 & 58.4 \\
64 & 79 & 37 & 102.8 \\
128 & 207 & 95 & 104.4 \\
256 & 399 & 199 & 105.0 \\
\end{array}
\]

Table 3.3: \( SNR \) for different number of cycles per output. \( 2^x = 64 \) is chosen.

According to the results in table 3.3 \( 2^x = 64 \) is chosen, because \( 2^x = 32 \) results in a too low \( SNR \) and \( 2^x = 128 \) \( 2^x = 256 \) yield a small improvement in \( SNR \) compared to \( 2^x = 64 \) and an \( SNR \) at 102.8 dB is sufficient. Compare
3.5 Coefficient word length

Figure 3.8: SNR for different values of $N_1$ and $N_2$. The best trade off is reached for $N_1 = 63$ and $N_2 = 41$ with a $SNR = 103.3$ dB.

these filter orders to the results given in chapter 3.2, the difference is more than an factor two. Equation 3.17 can now be rewritten as:

$$2\frac{N_1 + 1}{8} + (N_2 + 1) + 4 + 2 = 64 \Rightarrow N_2 = 57 - 2\frac{N_1 + 1}{8} \quad (3.20)$$

When the number of cycles is set to 64, one have to decide how to divide these cycles between the two filter stages. 24 filter structures with different values of $N_1$ and $N_2$ was optimized to find the best trade off between $N_1$ and $N_2$. A $SNR$ value is calculated and plotted in Fig 3.8. The best $SNR$ is reached for $N_1 = 63$ and consequently $N_2 = 41$. Hence a $SNR = 103.3$ db is obtained.

The resulting filter after optimization have a much higher attenuation in the stopband compared to the filter derived with firpm. The filter coefficients are presented in appendix A.

### 3.5 Coefficient word length

A Matlab model can have (almost) infinite precision in the filter coefficients. In hardware the filter coefficients are represented with a finite number of bits. This affects both the passband ripple and the $SNR$. The Table 3.4 and 3.5 shows how the $SNR$ and passband ripple are affected by the word length.

According to the results in table 3.4 the $SNR$ is only dependent on the wordlength of the first sub filter $b_{H_1}$. Hence this table was used to determine $b_{H_1}$. If the wordlength is chosen to $b_{H_1} = 16$ or $b_{H_1} = 17$ the $SNR$ results almost in its ideal value. Hence $b_{H_1}$ is chosen to 16.

To decide the wordlength for the second sub filter ($b_{H_2}$) table 3.5 was used. According to the filter specification (see table 1.1) the passband ripple have to be less then 0.035 dB. If $b_{H_2}$ is chosen to 13 the passband ripple will be 0.0232 dB. Then a design margin of 0.0118 dB is achieved at a low cost.

It is possible to obtain a better filter if coefficient wordlength was written as a constraint in the optimization. It is hard to write such a constraint and optimization would take a lot of time for a small improvement in the filter design.
Figure 3.9: Magnitude response for the optimized filter compared with the filter derived with the Matlab function firpm. Note that the attenuation is much higher for the optimized filter. The two lower plots are the magnitude responses for the two sub filters $H_1$ and $H_2$.

Figure 3.10: Enlargement of the passband.
3.5. Coefficient word length

<table>
<thead>
<tr>
<th></th>
<th>11</th>
<th>12</th>
<th>13</th>
<th>14</th>
</tr>
</thead>
<tbody>
<tr>
<td>$b_{H_1}$</td>
<td>14</td>
<td>92</td>
<td>92</td>
<td>92</td>
</tr>
<tr>
<td></td>
<td>15</td>
<td>97</td>
<td>97</td>
<td>97</td>
</tr>
<tr>
<td></td>
<td>16</td>
<td>102</td>
<td>102</td>
<td>102</td>
</tr>
<tr>
<td></td>
<td>17</td>
<td>103</td>
<td>103</td>
<td>103</td>
</tr>
</tbody>
</table>

Table 3.4: The table shows how the SNR are affected by different wordlengths at the filter coefficients. $b_{H_1}$ and $b_{H_2}$ are coefficient wordlength for filter stage one end two respectively. The SNR values are printed in $[dB]$.

<table>
<thead>
<tr>
<th></th>
<th>11</th>
<th>12</th>
<th>13</th>
<th>14</th>
</tr>
</thead>
<tbody>
<tr>
<td>$b_{H_2}$</td>
<td>14</td>
<td>0.0311</td>
<td>0.0281</td>
<td>0.0241</td>
</tr>
<tr>
<td></td>
<td>15</td>
<td>0.0324</td>
<td>0.0293</td>
<td>0.0234</td>
</tr>
<tr>
<td></td>
<td>16</td>
<td>0.0332</td>
<td>0.0301</td>
<td>0.0232</td>
</tr>
<tr>
<td></td>
<td>17</td>
<td>0.0329</td>
<td>0.0298</td>
<td>0.0232</td>
</tr>
</tbody>
</table>

Table 3.5: The table shows how the passband ripple are affected by different wordlengths at the filter coefficients. $b_{H_1}$ and $b_{H_2}$ are coefficient wordlength for filter stage one end two respectively. The passband ripple values are printed in $[dB]$.

3.5.1 Constant zero bits in coefficients

If the most significant bits are zero for all filter coefficients in an impulse response it is unnecessary to store these zeros and even more unnecessary to multiply with bits that are constant zero. To find out how many bits that are constant zero, the following expression was used where $h$ is the impulse response and $b_{zeros}$ is the number of bits that are constant zero.

$$\max(|h(n)|) \cdot 2^{b_{zeros}} \leq 1 \Rightarrow b_{zeros} \leq \log_2 \left( \frac{1}{\max(|h(n)|)} \right)$$

(3.21)

And consequently the number of active bits $b_{active}$ is

$$b_{active} = b_H - b_{zeros}$$

(3.22)

In table 3.6 $b_{zeros}$ and $b_{active}$ that are derived from equation 3.21 and 3.22 for both sub filters. The number of active bits is equal for the two sub filters, this will make it easy to design the hardware in an efficient way. The impulse responses are multiplied with two constants. To compensate for this the output is divided by the same constants, see the post processing block in section 4.1.

$$h_{1\text{new}}(n) = 2^4 h_1(n)$$

(3.23)

$$h_{2\text{new}}(n) = 2^1 h_2(n)$$

(3.24)
<table>
<thead>
<tr>
<th>sub filter</th>
<th>( b_{zeros} )</th>
<th>( b_{active} )</th>
</tr>
</thead>
<tbody>
<tr>
<td>( H_1 )</td>
<td>4</td>
<td>12</td>
</tr>
<tr>
<td>( H_2 )</td>
<td>1</td>
<td>12</td>
</tr>
</tbody>
</table>

Table 3.6: The number of bits equal to zero (\( b_{zeros} \)) and the number of active bits (\( b_{active} \)). Note that the number of active bits is equal for both filters.

Figure 3.11: Schedule

3.6 Schedule

There are 64 cycles available to produce one output to decide how these cycles should be divided between the two sub filters a schedule was made. A bottleneck in the design is the data memory, one word can be written or read each cycle. Input data have to be written four times to the memory and results from the first sub filter have to be written twice. When the convolutions for the two sub filters is computed data is read from the data memory. This is scheduled as in Fig 3.11.

3.7 DC level

The output DC level shall be zero. To handle this a constant (\( c_{DC} \)) is added at the output. Because the input is in the range \([0 : in_{max}]\) the DC level will not be zero.

\[
in_{max} = \sum_{i=1}^{3} 2^{-i} = 0.875 \tag{3.25}
\]

This will cause a constant DC offset. To compensate for this offset a constant \( c_{DC} \) is added after the filter. In the equation below ‘\(*\)’ is the convolution operator.

\[
x(n) = \frac{in_{max}}{2} \tag{3.26}
\]

\[
c_{DC} = -x(n) * h(n) \tag{3.27}
\]
3.8 A test case

To test the Matlab model of the filter a test signal was derived from the $\Delta\Sigma$ modulator that precedes the filter, see section 1.1. The test signal is a sine wave with a frequency at $\pi/2$, oversampled by a factor 32. To this signal noise from the $\Delta\Sigma$ is added. The frequency spectra of the test signal is plotted in Fig 3.12. In Fig 3.13 the filtered test signal is plotted. Note that the peak in the resulting output is located at $\pi/2$. Also note that inband signal is intact and that the noise in the stopband is attenuated.

c_{DC} is found to $-0.4366$.

---

Figure 3.12: Magnitude spectra of a test signal. The dashed line indicates the transition band.

Figure 3.13: The test signal filtered by the Matlab model. The plot to the left is the signal only filtered, not downsampled, the dashed line indicates the transition band. The rightmost plot is filtered and downsampling signal.
Chapter 4

Hardware implementation

This chapter describes how the Matlab model was implemented in hardware. First a VHDL model was created, here the processing element is described in detail. Secondly I explain how the VHDL model was synthesized to a gate level design.

4.1 VHDL model

To implement the VHDL model HDL Designer was used. To start with a detailed architecture was created, see Fig 4.2. A brief description to each block is given below. The processing element is discussed in more detail in section 4.2.

control The control block keeps track of the schedule (see section 3.11). It sends control signals to the other blocks.

in buffer The in buffer is a serial to parallel block. It buffers eight inputs. The output is \(8 \times 3 = 24\) bits wide.

Coefficient memory The coefficient memory stores the filter coefficients. Eight coefficients can be read from the memory in parallel.

Data memory The Data memory stores input data and results from the first sub filter. There are 50 words in memory, each word has a wordlength of 24 bits. See Fig 4.1. The given standard cell library provides a register file that was used.

Memory pointer The Memory pointer is a pointer to the data memory. The wordlength needed is:

\[
b_{\text{pointer}} = \lceil \log_2(50) \rceil
\]  

(4.1)

Post process The post process block cares about the DC level (see section 3.7). It also compensates for a constant gain discussed in section 3.5.1. Here the output is saturated and truncated to match the output format of 16 bits [6].
Figure 4.1: The data memory. Eight words is needed to store inputs to the first sub filter. 42 words is needed to store inputs to the second sub filter.

Figure 4.2: The architecture.
4.2 Processing element

This section describes the Processing element.

4.2.1 What is it doing?

The task for the processing element is to calculate the convolution for the two sub filters. The convolution operation consists of multiplying and accumulation see Fig 4.3. There are two inputs, input data and coefficient data.

![Diagram of multiply and accumulate (MAC) operation for calculating convolutions. d and c are input data and coefficient data respectively. acc is accumulator output.](image)

4.2.2 The main idea

The main idea is to use the same multiplier for calculating the convolution for both the first and second sub filter, even though the wordlength is 3 and 24 bits respectively.

In the case when the input data is 24 bits wide, the data will be split up in eight parts where each part is three bits wide. Each part of the input data is multiplied by the coefficient to produce eight partial products. These are shifted left and added so the product \( p_2 = c * d \) is evaluated. See Fig 4.4.

\[
p_2 = \sum_{i=0}^{7} cd_i 2^{3i} \quad (4.2)
\]

When the convolution to the first sub filter is to be evaluated the input is three bits wide, grouped eight words together see Fig 3.7. Now no shifts are performed. When these partial products are added the result will be eight MAC operations each clock cycle. See Fig 4.4.

\[
p_1 = \sum_{i=0}^{7} cd_i \quad (4.3)
\]

The architecture for implementing this can be seen in Fig 4.4. This architecture enables the processing element to either calculate the multiply and accumulate (MAC) for eight inputs per cycle if the input wordlength is three or one MAC if the input wordlength is 24.
4.2.3 Sign extension

If we want to change the wordlength we have to copy the sign bit this is referred to as sign extension. see example below:

\[ x_0 \ x_1 \ x_2 = x_0 \ x_0 \ x_0 \ x_1 \ x_2 \]

In the multiplexers in Fig 4.4 sign extension will cause a significant load on the sign bit. This can be avoided by inverting the sign bit and adding a compensation vector see Fig 4.5. This is similar to the Baugh-Wooley’s multiplier [6]. The sign extension technique reduce the load on the sign bits at the cost of one extra addition in the adder tree.

4.2.4 Internal wordlength

The internal wordlength \( (b_{internal}) \) in the processing element have to be long enough to prevent the occurrence of overflow. The worst case input would need the following number of bits to represent the output for the first sub filter:

\[
b_{internal1} = \left\lceil \log_2 \left( 7 \sum_{n=0}^{N_1} |h_1(n)| \right) \right\rceil = 20 \tag{4.4}\]

The number 7 derives from the input range which maximum is 7. The result from the first sub filter is only 20 bits, the memory is 24 bits wide, so 4 bits in each word in memory is not used when storing results from the first sub filter. This also means that one of the multiplier (the leftmost in Fig 4.4) will not be used for the second sub filter.

The number of bits needed to represent the result from the second sub filter is:

\[
b_{internal} = \left\lceil \log_2 \left( 7 \sum_{n=0}^{N_1} |h_1(n)| \ast \sum_{n=1}^{N_2+1} |h_2(n)| \right) \right\rceil = 34 \tag{4.5}\]
4.3. Synthesis

\[
\begin{array}{cccccccccccc}
0 & 0 & 0 & x_0 & x_1 & x_2 & x_3 & x_4 & x_5 & x_6 & \ldots \\
+ & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
\hline
x_0 & x_0 & x_0 & x_0 & x_1 & x_2 & x_3 & x_4 & x_5 & x_6 & \ldots \\
\end{array}
\]

\[
\begin{array}{cccccccccccc}
0 & 0 & 0 & 0 & 0 & 0 & y_0 & y_1 & y_2 & y_3 & \ldots \\
+ & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 \\
\hline
y_0 & y_0 & y_0 & y_0 & y_0 & y_0 & y_0 & y_1 & y_2 & y_3 & \ldots \\
\end{array}
\]

\[
x + y =
\begin{array}{cccccccccccc}
0 & 0 & 0 & x_0 & x_1 & x_2 & x_3 & x_4 & x_5 & x_6 & \ldots \\
0 & 0 & 0 & 0 & 0 & 0 & y_0 & y_1 & y_2 & y_3 & \ldots \\
+ & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
\hline
0 & 0 & 0 & x_0 & x_1 & x_2 & x_3 & x_4 & x_5 & x_6 & \ldots \\
0 & 0 & 0 & 0 & 0 & 0 & y_0 & y_1 & y_2 & y_3 & \ldots \\
+ & 1 & 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 \\
\end{array}
\]

Figure 4.5: Sign extension and Addition of the binary numbers \(x\) and \(y\). Sign extension by inverting the sign bit and adding a compensation vector. When adding several numbers the compensation vectors can be summed and precomputed. Therefore the load on the most significant bit is reduced at the cost of one extra add.

\[
x_{33} x_{32} \, \boxed{x_{31}} \, x_{30} \ldots x_{18} \, x_{17} \, \boxed{x_{16}} \, x_{15} \, x_{14} \ldots x_1 \, x_0
\]

Figure 4.6: 16 bits to the output selected from the accumulator in the processing element.

**\(L_\infty\) norm**

To determine which 16 of the 34 bits that shall represent the output a measure of the size of the signal was needed. For this purpose the \(L_\infty\) norm was used which is defined as \([4]\):

\[
\|X(\omega T)\|_\infty = \max\{|X(\omega T)|\}
\]

The number of bits needed will be:

\[
b = \lceil \log_2(\max\{7 \ast |H(\omega T)|\}) \rceil = 32
\]

The 16 output bits will be chosen as in Fig 4.6

4.3 Synthesis

To translate the VHDL model to a gate level design the synthesis tool Design Compiler (DC) was used. To synthesize DC need a RTL hardware description and a standard cell library. DC can then produce a gate level netlist which is
Table 4.1: Area and power consumption for the circuit. The area have no unit

<table>
<thead>
<tr>
<th>block</th>
<th>area</th>
<th>%</th>
<th>power [µW]</th>
<th>%</th>
</tr>
</thead>
<tbody>
<tr>
<td>Processing element</td>
<td>23253</td>
<td>50.6</td>
<td>35.4</td>
<td>73.3</td>
</tr>
<tr>
<td>Post process</td>
<td>1777</td>
<td>3.9</td>
<td>2.27</td>
<td>4.7</td>
</tr>
<tr>
<td>In buffer</td>
<td>1185</td>
<td>2.6</td>
<td>1.65</td>
<td>3.4</td>
</tr>
<tr>
<td>Data memory</td>
<td>14059</td>
<td>30.6</td>
<td>3.31</td>
<td>6.9</td>
</tr>
<tr>
<td>Memory pointer</td>
<td>1708</td>
<td>3.7</td>
<td>1.83</td>
<td>3.8</td>
</tr>
<tr>
<td>Control</td>
<td>390</td>
<td>0.8</td>
<td>0.8</td>
<td>1.7</td>
</tr>
<tr>
<td>Coefficient ROM</td>
<td>3353</td>
<td>7.3</td>
<td>2.76</td>
<td>5.7</td>
</tr>
<tr>
<td>total</td>
<td>45975</td>
<td></td>
<td>47.95</td>
<td></td>
</tr>
</tbody>
</table>

a complete description of the RTL hardware description where all components are standard cells (for example AND-gates and OR-gates) [8].

4.3.1 Synthesis results

Timing

The longest path in the design is called the critical path and the time to execute the critical path is denoted $T_{CP}$. If $T_{CP}$ is lower than the clock period time ($T_{clk}$) the timing constraints on the circuit will be fulfilled. $T_{clk}$ is found as:

$$T_{clk} = \frac{1}{f_s \times 32 \times 2} = 355 \text{ [ns]}$$

(4.8)

$T_{CP}$ is found with the Design Compiler function timing_report. The critical path is a path starting in the control block, going through coefficient memory and processing element block and ends in the post processing block.

$$T_{CP} = 28.84 \text{ [ns]}$$

(4.9)

$T_{CP}$ is lower then $T_{clk}$, then the timing constraint is fulfilled.

Area and power consumption

The area and power consumption for the circuit was estimated by Design Compiler. The functions report_area and report_power have been used. In table 4.1 the area and power consumption is presented. The area has no unit, the numbers in the table are only to make a comparison between the blocks. The processing element and the memory are the blocks that uses most area and power. Especially the processing element which uses half of the area and 73 % of the total power.

4.3.2 Validation

To validate the behavior of the gate level design a testbench was created (Fig 4.7). A test vector was created, see section 3.8, this test vector was transformed to a file which could be read from Model Sim. Model Sim is a simulation tool which can simulate VHDL designs. The gate level design was translated to a VHDL netlist which also can be simulated in Model Sim. The results from the
4.3. Synthesis

Simulation in Model Sim was translated back to Matlab. At last a comparison between the Model Sim simulation and the Matlab model was made in Matlab. This comparison resulted in a pass.

This test was made for several test vectors, all of them resulted in a pass.
Chapter 5

Future work

The filter that have been presented in this work is of course not perfect, below are some examples of how the filter can be improved.

- To start with I think that the estimations of the filter orders in chapter 3.2 are not reliable, because these results differ a lot to the filter orders derived with the optimization technique. Compare 177 and 89 to 63 and 41, the difference is more than a factor two. To improve the filter design one should examine other filter structures in detail with the optimization technique. Of main interest is structure 4 (decimation with 8, 2 and 2).

- The filter coefficients are represented with a finite number of bits. If this is written as a constraint in the optimization formulation a better filter could be achieved.

- Moreover the processing element consumes 73 % of the total power consumption. If the implementation can be improved area can be saved. Perhaps one can save power by implementing the multipliers using carry save adders instead of regular adders.
Bibliography

[1] Henrik Ohlsson, Behzad Mesgarzadeh, Kenny Johansson, Oscar Gustafsson, Per Löwenborg, Håkan Johansson, Atila Alvandpour, A 16 GSPS 0.18 μm CMOS Decimator for Single-Bit ΣΔ - Modulation


# Appendix A

## Filter coefficients

<table>
<thead>
<tr>
<th>$i$</th>
<th>$h_1(i)$</th>
<th>$i$</th>
<th>$h_1(i)$</th>
<th>$i$</th>
<th>$h_1(i)$</th>
<th>$i$</th>
<th>$h_1(i)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>-1</td>
<td>16</td>
<td>994</td>
<td>32</td>
<td>3635</td>
<td>48</td>
<td>819</td>
</tr>
<tr>
<td>1</td>
<td>-1</td>
<td>17</td>
<td>1186</td>
<td>33</td>
<td>3598</td>
<td>49</td>
<td>664</td>
</tr>
<tr>
<td>2</td>
<td>-1</td>
<td>18</td>
<td>1393</td>
<td>34</td>
<td>3526</td>
<td>50</td>
<td>527</td>
</tr>
<tr>
<td>3</td>
<td>2</td>
<td>19</td>
<td>1612</td>
<td>35</td>
<td>3422</td>
<td>51</td>
<td>409</td>
</tr>
<tr>
<td>4</td>
<td>9</td>
<td>20</td>
<td>1838</td>
<td>36</td>
<td>3288</td>
<td>52</td>
<td>309</td>
</tr>
<tr>
<td>5</td>
<td>21</td>
<td>21</td>
<td>2069</td>
<td>37</td>
<td>3127</td>
<td>53</td>
<td>227</td>
</tr>
<tr>
<td>6</td>
<td>40</td>
<td>22</td>
<td>2299</td>
<td>38</td>
<td>2943</td>
<td>54</td>
<td>160</td>
</tr>
<tr>
<td>7</td>
<td>68</td>
<td>23</td>
<td>2525</td>
<td>39</td>
<td>2741</td>
<td>55</td>
<td>108</td>
</tr>
<tr>
<td>8</td>
<td>108</td>
<td>24</td>
<td>2741</td>
<td>40</td>
<td>2525</td>
<td>56</td>
<td>68</td>
</tr>
<tr>
<td>9</td>
<td>160</td>
<td>25</td>
<td>2943</td>
<td>41</td>
<td>2299</td>
<td>57</td>
<td>40</td>
</tr>
<tr>
<td>10</td>
<td>227</td>
<td>26</td>
<td>3127</td>
<td>42</td>
<td>2069</td>
<td>58</td>
<td>21</td>
</tr>
<tr>
<td>11</td>
<td>309</td>
<td>27</td>
<td>3288</td>
<td>43</td>
<td>1838</td>
<td>59</td>
<td>9</td>
</tr>
<tr>
<td>12</td>
<td>409</td>
<td>28</td>
<td>3422</td>
<td>44</td>
<td>1612</td>
<td>60</td>
<td>2</td>
</tr>
<tr>
<td>13</td>
<td>527</td>
<td>29</td>
<td>3526</td>
<td>45</td>
<td>1393</td>
<td>61</td>
<td>-1</td>
</tr>
<tr>
<td>14</td>
<td>664</td>
<td>30</td>
<td>3598</td>
<td>46</td>
<td>1186</td>
<td>62</td>
<td>-1</td>
</tr>
<tr>
<td>15</td>
<td>819</td>
<td>31</td>
<td>3635</td>
<td>47</td>
<td>994</td>
<td>63</td>
<td>-1</td>
</tr>
</tbody>
</table>

Table A.1: Filter coefficients sub filter one.

<table>
<thead>
<tr>
<th>$i$</th>
<th>$h_2(i)$</th>
<th>$i$</th>
<th>$h_2(i)$</th>
<th>$i$</th>
<th>$h_2(i)$</th>
<th>$i$</th>
<th>$h_2(i)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>7</td>
<td>11</td>
<td>-210</td>
<td>22</td>
<td>482</td>
<td>33</td>
<td>48</td>
</tr>
<tr>
<td>1</td>
<td>-58</td>
<td>12</td>
<td>296</td>
<td>23</td>
<td>-1155</td>
<td>34</td>
<td>-181</td>
</tr>
<tr>
<td>2</td>
<td>112</td>
<td>13</td>
<td>161</td>
<td>24</td>
<td>-111</td>
<td>35</td>
<td>25</td>
</tr>
<tr>
<td>3</td>
<td>-61</td>
<td>14</td>
<td>-452</td>
<td>25</td>
<td>680</td>
<td>36</td>
<td>135</td>
</tr>
<tr>
<td>4</td>
<td>-84</td>
<td>15</td>
<td>-66</td>
<td>26</td>
<td>-66</td>
<td>37</td>
<td>-84</td>
</tr>
<tr>
<td>5</td>
<td>135</td>
<td>16</td>
<td>680</td>
<td>27</td>
<td>-452</td>
<td>38</td>
<td>-61</td>
</tr>
<tr>
<td>6</td>
<td>25</td>
<td>17</td>
<td>-111</td>
<td>28</td>
<td>161</td>
<td>39</td>
<td>112</td>
</tr>
<tr>
<td>7</td>
<td>-181</td>
<td>18</td>
<td>-1155</td>
<td>29</td>
<td>296</td>
<td>40</td>
<td>-58</td>
</tr>
<tr>
<td>8</td>
<td>48</td>
<td>19</td>
<td>482</td>
<td>30</td>
<td>-210</td>
<td>41</td>
<td>7</td>
</tr>
<tr>
<td>9</td>
<td>217</td>
<td>20</td>
<td>3453</td>
<td>31</td>
<td>-163</td>
<td>42</td>
<td>7</td>
</tr>
<tr>
<td>10</td>
<td>-163</td>
<td>21</td>
<td>3453</td>
<td>32</td>
<td>217</td>
<td>43</td>
<td>7</td>
</tr>
</tbody>
</table>

Table A.2: Filter coefficients sub filter two.
Copyright

The publishers will keep this document online on the Internet - or its possible replacement - for a period of 25 years from the date of publication barring exceptional circumstances. The online availability of the document implies a permanent permission for anyone to read, to download, to print out single copies for your own use and to use it unchanged for any non-commercial research and educational purpose. Subsequent transfers of copyright cannot revoke this permission. All other uses of the document are conditional on the consent of the copyright owner. The publisher has taken technical and administrative measures to assure authenticity, security and accessibility. According to intellectual property law the author has the right to be mentioned when his/her work is accessed as described above and to be protected against infringement. For additional information about the Linköping University Electronic Press and its procedures for publication and for assurance of document integrity, please refer to its WWW home page: http://www.ep.liu.se/

Upphovsrätt

Detta dokument hålls tillgängligt på Internet - eller dess framtida ersättare - under 25 år från publiceringsdatum under förutsättning att inga extraordinära omständigheter uppstår. Tillgång till dokumentet innebär tillstånd för var och en att läsa, ladda ner, skriva ut enstaka kopior för enskilt bruk och att använda det oförändrat för ickekommersiell forskning och för undervisning. Överföring av upphovsrätten vid en senare tidpunkt kan inte upphäva detta tillstånd. All annan användning av dokumentet kräver upphovsmannens medgivande. För att garantera äktheten, säkerheten och tillgängligheten finns det lösningar av teknisk och administrativ art. Upphovsmannens ideella rätt innefattar rätt att bli nämnd som upphovsman i den omfattning som god sed kräver vid användning av dokumentet på ovan beskrivna sätt samt skydd mot att dokumentet ändras eller presenteras i sådan form eller i sådant sammanhang som är kränkande för upphovsmannens litterära eller konstnärliga anseende eller egenart. För ytterligare information om Linköping University Electronic Press se förlagets hemsida http://www.ep.liu.se/

© 2008, Erik Lindahl