DESIGN OF LOW POWER DECIMATION FILTER FOR SIGMA-DELTA ANALOG DIGITAL CONVERTERS

by

Feyza Kayaduman

B.S. in Electronics and Communication Engineering,

Yıldız Technical University, 2008

Submitted to the Institute for Graduate Studies in
Science and Engineering in partial fulfillment of
the requirements for the degree of
Master of Science

Graduate Program in Electrical and Electronics Engineering

Boğaziçi University

2011

ACKNOWLEDGEMENTS

I sincerely thank to my thesis supervisor Prof. Günhan Dündar for giving me support and encouragement throughout my master education and throughout my thesis. I'm very grateful for his guidance, patience and understanding.

I would like to thank to assistants of Electronics Department for their helps.

I am very thankful to TÜBİTAK-BİDEB for supporting me financially during my master education.

I would like to thank to my family for their love and moral support during all difficult times throughout my life.

Finally, I would like to thank to my fiancé, for his love, patience, encouragement and support in every steps of my life since the day I met.

ABSTRACT

DESIGN OF LOW POWER DECIMATION FILTER FOR SIGMA-DELTA ANALOG DIGITAL CONVERTERS

Sigma-delta analog digital converter ($\sum\Delta$ ADC) is widely used for high resolution applications such as audio applications. Sigma-delta ADC uses the oversampling process; therefore, it can reach high resolutions. Decimation process downsamples the sampling frequency and eliminates the redundant data which are generated by the oversampling process. The digital part of the sigma-delta ADC contains decimation and low-pass filters. Digital low-pass filter attenuates the noise which is pushed out of band by the oversampling process. Digital filters perform arithmetic calculations. Thus, they consume power. Low power design of decimation filter is an important issue for achieving a low power sigma-delta ADC.

In this thesis, a low power decimation filter for sigma-delta ADC for audio frequency is implemented. In order to achieve this goal, multistage decimation filter architecture is used. The decimation filter consists of a third order Cascaded Integrator Comb (CIC) filter, one half-band filter and one Finite Impulse Response (FIR) filter. In this design, input signal bandwidth is 20 KHz and sampling frequency is 2.5 MHz.

Half-band and FIR filter coefficients are generated using Parks McCellan algorithm. Multiplierless filter design is used for achieving low power consumption. Filter coefficients are represented by Canonical Signed Digit (CSD) representation. Horizontal Super Subexpression Elimination (HSSE) method is applied to CSD coefficients.

In addition, half-band and FIR filters are designed using GAM algorithm which is proposed by Mustafa Aktan to compare manual and CAD design results.

GAM algorithm reduces the number of signed power of two (SPT) terms in the coefficients while keeping the quantization word length as small as possible. Common Subexpression Elimination (CSE) method is applied to the filter coefficients by the GAM algorithm.

All filters are designed with VHDL and implemented in 0.35μm CMOS process. The power consumption is measured for different supply voltage values. For 3.3V supply, 2.2 mW; for 2.5V supply, 1.1mW; and for 1.2V, 204 μW is dissipated for manual design of the decimation filter. Furthermore, for 3.3V supply, 1.7 mW; for 2.5V supply, 859 μW; and for 1.2V, 159 μW is dissipated for the decimation filter designed using GAM algorithm.

ÖZET

SİGMA-DELTA ANALOG SAYISAL ÇEVİRİCİLER İÇİN DÜŞÜK GÜÇ TÜKETEN ÖRNEK SEYRELTME SÜZGECİ TASARIMI

Sigma-delta analog sayısal çevirici, ses uygulamaları gibi yüksek çözünürlüklü uygulamalarda sıklıkla kullanılır. Sigma-delta çevirici aşırı örnekleme işlemi kullanır ve bu sayede yüksek çözünürlüklere ulaşır. Örnekleme seyreltme işlemi, örnekleme frekansının düşürülmesiyle aşırı örnekleme işlemi sonucu oluşan gereksiz verilerin yok edilmesini sağlar. Sigma-delta analog sayısal çeviricinin sayısal bölümü, örnekleme seyreltme filtresi ve alçak geçiren filtreden oluşur. Sayısal alçak geçiren filtre, aşırı örnekleme işlemi sonucu bant dışına atılan kuantalama gürültüsünü zayıflatır. Sayısal filtreler aritmetik hesaplamalar gerçekleştirdiklerinden ötürü yüksek güç tüketirler. Düşük güç tüketen bir sigma-delta analog sayısal çevirici elde edebilmek için düşük güç tüketen sayısal filtre tasarlayabilmek önemlidir.

Tezde, ses uygulamalarında kullanılacak bir sigma-delta çevirici için düşük güç tüketen örnekleme seyreltme filtresi gerçekleştirilmiştir. Çok katmanlı mimari kullanılmıştır. Örnekleme seyreltme filtresi, üçüncü dereceden bir CIC filtre, bir yarım bant filtre ve bir sonlu dürtü yanıtlı filtre kullanılmıştır. Giriş sinyalinin bant genişliği 20 KHz, örnekleme frekansı ise 2.5MHz olarak alınmıştır.

Yarım bant ve sonlu dürtü yanıtlı filtrelerin katsayıları MATLAB programı kullanılarak elde edilmiştir. Filtreler çarpıcı kullanılmadan tasarlanmıştır. Filtre katsayıları, düşük güç tüketimi elde edebilmek için kanonik işaretli basamak gösterimiyle ifade edilmiş ve bu katsayılara yatay süper alt ifade eleme yöntemi uygulanmıştır.

Ek olarak Mustafa Aktan'ın önermiş olduğu GAM algoritması kullanılarak half-band ve FIR filtreler tasarlanmıştır ve el ile tasarlanan filtreler ile kıyaslanmıştır.

GAM algoritması filtre katsayılarındaki işaretli ikinin kuvvetleri (SPT) terimlerini kelime uzunluğunu mümkün olduğunca küçük tutarak azaltır. GAM algoritması filtre katsayılarına ortak alt ifade paylaşımı metodu uygular.

Örnekleme seyreltme filtresi VHDL ile tasarlanmış ve 0.35µm CMOS teknolojisi ile gerçeklenmiştir. Farklı besleme gerilimleri için güç tüketimi hesaplanmıştır. El ile tasarlanan örnekleme seyreltme filtresi için 3.3V besleme geriliminde 2.2 mW; 2.5V besleme geriliminde 1.1 mW; ve 1.2V besleme geriliminde 204 µW güç tüketildiği görülmüştür. GAM algoritması kullanılarak tasarlanan örnekleme seyreltme filtresinde ise 3.3V besleme geriliminde 1.7 mW; 2.5V besleme geriliminde 859 µW; ve 1.2V besleme geriliminde 159 µW güç tüketildiği görülmüştür.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF SYMBOLS

$y(n)$	Output of the filter
a_k	Filter coefficients
b_k	Filter coefficients
x_{n-k}	Input shifted by k
f_s	Sampling frequency
f_0	Input signal frequency
f_c	Maximum frequency of interest
BW	Band width
OSR	Oversampling ratio
L	Modulator order
Fpass	Passband frequency
Fstop	Stopband frequency
Apass	Passband ripple
Astop	Stopband ripple
R	Decimation ratio
M	Differential delay
Bin	Number of input bits
$\lceil x \rceil$	The smallest integer not less than *x*
$>> n$	Right shift operation
N	Filter order

LIST OF ACRONYMS/ABBREVIATIONS

ADC	Analog digital converter
VLSI	Very large scale integration
CIC	Cascaded integrator-comb
HB	Half-band
FIR	Finite impulse response
IIR	Infinite impulse response
CSE	Common subexpression elimination
CSD	Canonical signed digit
DAC	Digital analog converter
RCA	Ripple carry adder
SPT	Signed power of two
HCS	Horizontal common subexpression
VCS	Vertical common subexpression
HCSE	Horizontal common subexpression elimination
VCSE	Vertical common subexpression elimination
HSS	Horizontal super subexpression
VSS	Vertical super subexpression
HSSE	Horizontal super subexpression elimination
VSSE	Vertical super subexpression elimination
RTL	Register transfer level
VHDL	Very high speed integrated circuit (VHSIC) hardware description language
FM	Figure of merit

1. INTRODUCTION

Low power is a key parameter in most electronics applications. Especially, power consumption is critical for battery-powered mobile applications such as mobile phone, personal music player, and similar devices. Low power design of electronic circuits maximizes the battery life. Low power also leads to smaller power supplies and less expensive batteries.

Signal processing is commonly performed in the digital domain because of the robustness, high speed and low implementation costs of digital circuits. Since real world signals are analog, data converters are needed as an interface between the analog world and digital domain.

Sigma-delta analog digital converters ($\sum\Delta$ ADC) are generally used in many applications because of their high speed and accuracy. The sigma-delta ADC is used for modern voice band, audio, and high resolution precision industrial measurement applications [1].

It is difficult to implement conventional converters in fine-line very large scale integration (VLSI) technology because they need precise analog components. Sigma-delta ADCs utilize oversampling, simple and high tolerance analog components can be used to achieve high resolution, but sigma-delta ADCs require fast and complex digital signal processing. Therefore, they can be implemented in fine line VLSI technology.

Oversampling reduces the quantization noise in the band of interest. Thus, sigma-delta ADCs can reach high resolutions [1].

The power consumption of the sigma-delta ADC is better compared to other converters because it requires less active elements than other ADC architectures such as flash or pipeline ADCs. However, using high clock rates in oversampling increases the power consumption .

The digital part of the sigma-delta ADC contains decimation and low-pass filters. Decimation is the process of reducing the data rate down from the oversampling rate without losing any information. According to the sampling theorem, the required sample rate is two times the input bandwidth in order to reliably reconstruct the input signal without distortion. However, in sigma-delta modulator, the input is oversampled in order to reduce the quantization noise. Thus, there is redundant data that can be eliminated without introducing distortion to the conversion result. Digital filters attenuate noise and interference, and eliminate the redundant data [1].

There are lots of studies for low power sigma-delta ADCs in the literature. However, most of the studies are about reducing the power consumption of the analog part, namely the modulator part of the ADC. In order to provide good power efficiency for the whole converter, the digital part of the converter must be implemented in low power, too.

There are various approaches in order to implement a low power decimation filter in the literature. One approach is Cascaded Integrator-Comb (CIC) filter. CIC filters do not require multipliers and consist of only adders, subtractors and registers. Therefore, CIC filters provide economical hardware implementations and have low power consumptions.

Another approach is power optimized sinc^4 filter approach. This approach proposes a digital sinc^4 filter based on the direct implementation of the convolution relationship between the input samples and the filter coefficients. The coefficients are represented as sums of powers of two terms. These filters do not require multipliers as well [2, 3]. It is observed that CIC filter architecture is more economical compared to power optimized sinc^4 filter approach in terms of hardware when the decimation ratio is higher. Therefore, CIC filter architecture is preferred in this thesis.

There are two basic types of digital filters in order to implement a low-pass filtering function: Finite Impulse Response (FIR) Filter and Infinite Impulse Response (IIR) Filter.

FIR filter is a non recursive filter and its operation is described by the equation (1.1). Here, a_k is the filter coefficient, x_{n-k} is the input sample.

$$y(n) = \sum_{k=0}^{N-1} a_k x_{n-k} \tag{1.1}$$

IIR Filter is a recursive filter and its operation is described by the equation (1.2)

$$y(n) = \sum_{k=0}^{M} a_k x_{n-k} + \sum_{k=0}^{N} b_k y_{n-k} \tag{1.2}$$

FIR and IIR filters have advantages compared to each other. FIR filters are more easily designed than IIR filters. FIR filters are always stable, but IIR filters may be unstable. FIR filters are easily incorporated decimation. FIR filters have linear phase response, however IIR filters have nonlinear phase response. FIR filters are less efficient than IIR filters. FIR filter is commonly used for the back end of the sigma-delta converter because of its stability, ease of implementation, linear phase response and the fact that decimation can be incorporated into the filter itself [4].

Linear phase equiripple FIR half-band filter is a special class of FIR filters. About 50 percent of the half-band filter coefficients are zero, thus it reduces the implementation cost.

There are many methodologies for designing low power FIR filters. In this thesis, Common Subexpression Elimination (CSE) method is used and, filter coefficients are expressed in Canonical Signed Digit (CSD) representation. This number representation reduces hardware complexity and power dramatically.

In this thesis, the aim is to design and implement a low power decimation filter for a second order sigma-delta ADC modulator for audio frequencies. There are a lot of techniques in decimation filter implementation. In this study, multistage decimation filter is preferred. The decimation filter consists of three cascade stages: a Cascaded Integrator-Comb (CIC) filter, a Half-band (HB) filter and a Finite Impulse Response (FIR) filter.

2. SIGMA-DELTA ADC

2.1. Introduction

A sigma-delta ADC consists of a sigma-delta modulator and a low-pass filter as shown in Figure 2.1.

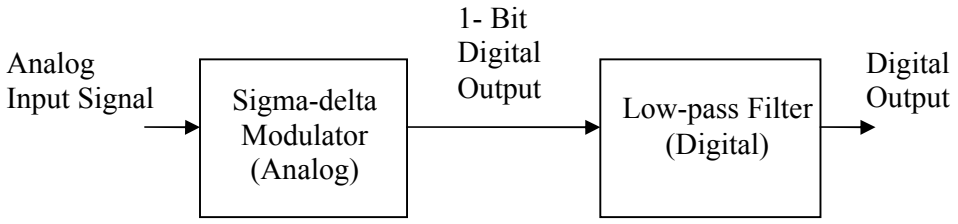

Figure 2.1. Block diagram of a sigma-delta ADC.

The modulator is designed with analogue technique to produce a bit stream and digital low-pass filter is designed to achieve a digital output.

2.1.1. Signal Sampling

Nyquist Sampling Theorem tells that the sampling frequency of a signal must be at least twice the input signal frequency in order to reconstruct the sampled signal without distortion. When a signal is sampled, its input spectrum is copied and mirrored at multiples of the sampling frequency, f_s. Figure 2.2 shows the spectrum of an under sampled signal. The sampling frequency is less than twice the input signal frequency, f_0. The shaded area in the figure shows aliasing which is a distortion that occurs when sampling theorem is violated.

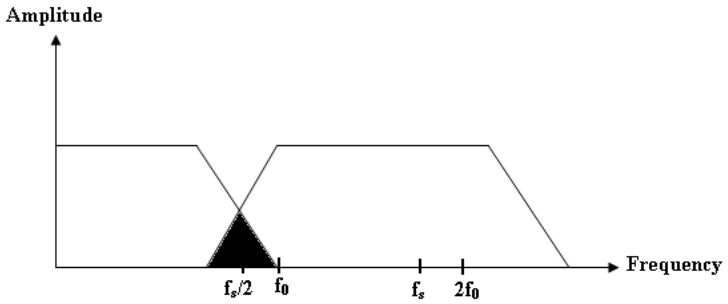

Figure 2.2. Under sampled signal spectrum.

2.1.2. Oversampling

Oversampling is a process of sampling the input signal with a sampling frequency that is significantly higher than twice the bandwidth of the input signal (Nyquist frequency). Oversampling reduces the aliasing and quantization noise in the band of interest. Figure 2.3 shows the spectrum of an oversampled signal. The oversampling process puts the entire input bandwidth at less than $f_s / 2$ and avoids the aliasing.

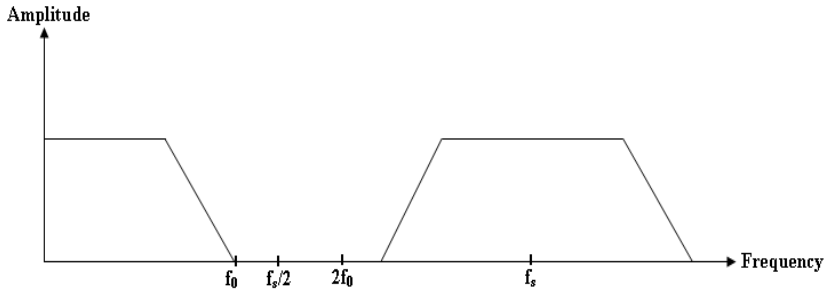

Figure 2.3. Oversampled signal spectrum.

2.1.3. Quantization Noise

Quantization Noise (or Quantization Error) is a round-off error occurs when an analog signal is quantized. An analog signal can take any value however an n bit digital signal can take only 2^n value. Thus, the difference between the analog value and quantized digital value is known as quantization noise.

Figure 2.4 shows the quantization noise spectrum of a typical Nyquist rate converter. Here, f_c is the maximum frequency of interest.

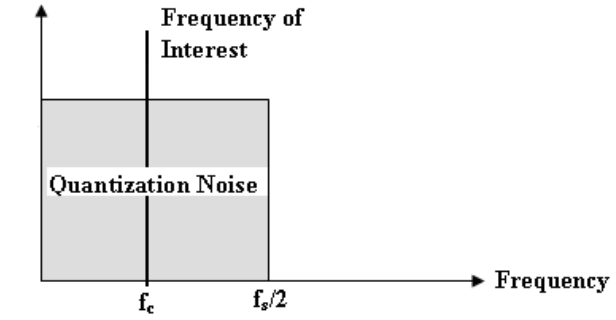

Figure 2.4. Quantization noise spectrum of Nyquist converter.

Figure 2.5 shows the quantization noise spectrum of a sigma-delta ADC. The effect of oversampling over quantization noise can be seen obviously. In an oversampled ADC, the quantization noise is spread over a wider spectrum. Although the total quantization noise is the same, the quantization noise in the bandwidth of interest is greatly by this technique [4].

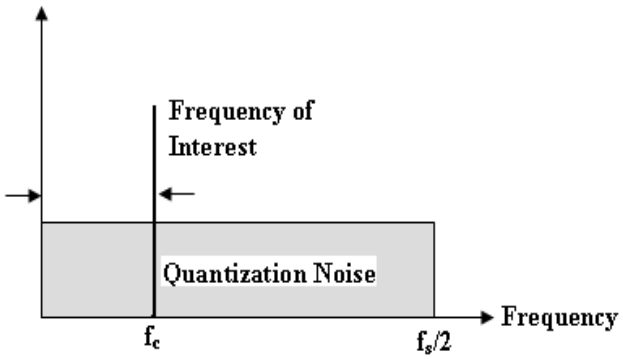

Figure 2.5. Quantization noise spectrum of sigma-delta converter.

2.2. Modulator

Figure 2.6 shows the block diagram of a first order sigma-delta modulator. The modulator consists of an integrator, comparator and a one bit digital analog converter (DAC) in the feedback loop. The analog input signal comes into the modulator via summing amplifier. Then this signal passes the integrator which feeds a comparator that acts as a one bit quantizer. The comparator output is fed back to the input via one bit DAC. The feedback loop forces the average of the DAC output signal to be equal to the input signal [4].

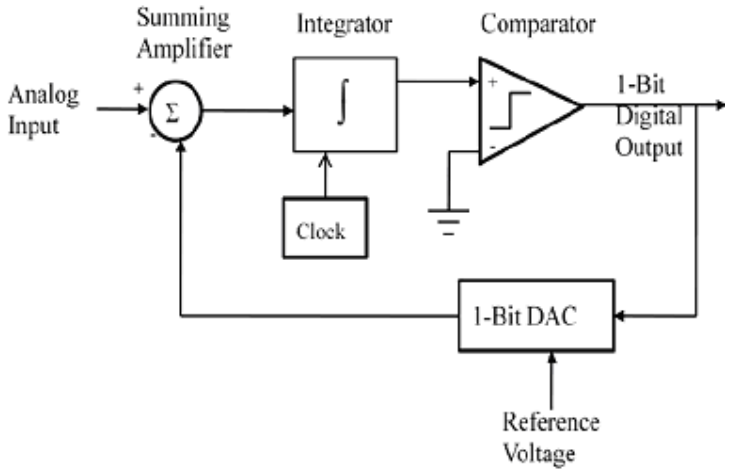

Figure 2.6. First order sigma-delta modulator.

The name first order comes from that there is only one integrator in the circuit. Delta modulation is based on quantizing the change in the signal from sample to sample rather than the absolute value of the signal at each sample. Sigma stands for summing or integrating which is performed at the input stage on the digital output from one bit DAC and the input signal before delta modulation [5]. Thus, this technique of analog to digital conversion is called sigma-delta modulation.

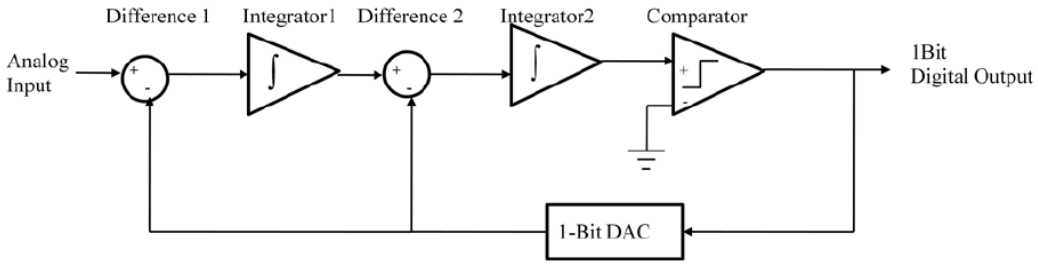

Figure 2.7. Second order sigma-delta modulator.

Figure 2.7 shows the block diagram of a second order sigma-delta modulator. Second order sigma-delta modulator works similar to the first order modulator. The only difference is the integration is performed twice on the data.

2.3. Decimator

Decimation is a technique that downsamples the sampling frequency from oversampling rate to the minimum required, i.e. Nyquist frequency. Decimation process eliminates the redundant data at the output. Figure 2.8 shows decimation process in time domain and Figure 2.9 shows decimation by 4 in frequency domain.

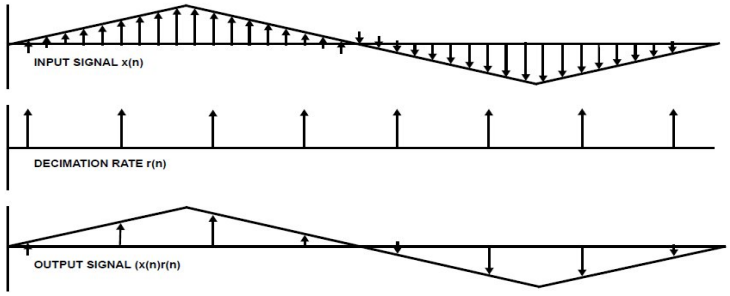

Figure 2.8. Decimation in time domain [4].

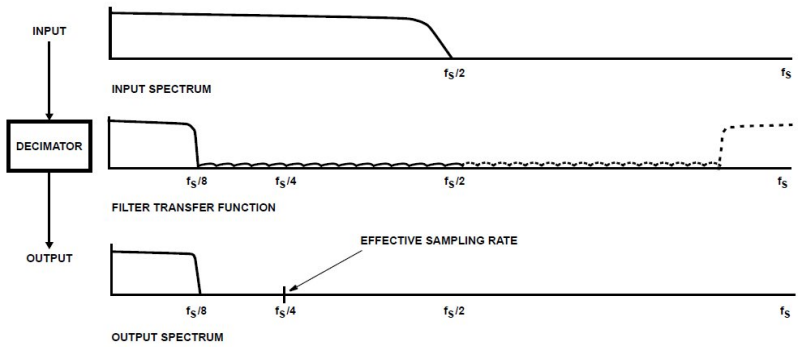

Figure 2.9. Decimation by 4 in frequency domain [4].

The low-pass digital filter attenuates the signals and noise that are outside the band of interest. Figure 2.10 and Figure 2.11 shows the digital filter eliminates the quantization noise that the modulator pushed out to the higher frequencies [4].

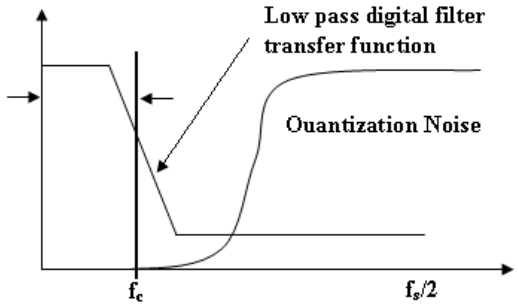

Figure 2.10. Quantization noise before filtering.

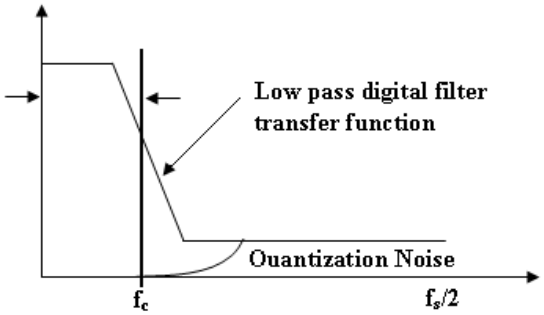

Figure 2.11. Quantization noise after filtering.

Table 2.1. shows the sigma-delta converter specifications used in this thesis. Table 2.2. shows the required decimation filter characteristics for this modulator.

Table 2.1. Sigma-delta converter specifications.

Parameter	Value
Signal Bandwidth	BW = 20 KHz
Sampling Frequency	f_s = 2.5 MHz
Oversampling Ratio	OSR = 62.5
Modulator Order	L = 2
Number of bits at output of the modulator	B = 1
Number of bits at output of the filter	B = 10

Table 2.2. Decimation filter characteristics.

Filter Parameter	Value
Sampling Frequency	f_s = 2.5 MHz
Decimation Ratio	R= 60
Passband Frequency	Fpass = 20 KHz
Stopband Frequency	Fstop= 40 KHz
Passband Ripple	Apass = 0.01
Stopband Attenuation	Astop = 70 dB

In order to meet the characteristics of decimation filter which is given in Table 2.2,a standard 446 tap FIR filter is required. The filter can be easily obtained with suitable software such as MATLAB. It's hard to implement such kind of filter in hardware. Thus, it is necessary to choose multistage decimation filter architecture. The advantage of a multistage decimation filter is that a sharp low-pass filter can be realized only at the last stage. The proposed decimation filter architecture is a three stage decimation filter. One CIC filter, one half-band and one FIR filter. Figure 2.12 shows the proposed decimation filter.

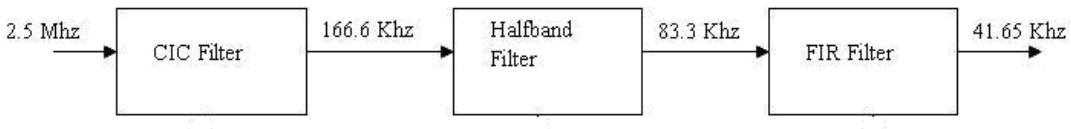

Figure 2.12. Decimation filter architecture.

3. CASCADED INTEGRATOR-COMB (CIC) FILTER

3.1. Introduction

The Cascaded Integrator-Comb (CIC) filter was proposed by Hogenauer for decimation and interpolation [6]. Integrator and comb blocks are the basic blocks of a CIC filter. Cascaded ideal integrator stages operate at a high sampling rate and equal numbers of comb stages operate at a low sampling rate. Figure 3.1 shows the basic structure of the CIC decimation filter [6]. It can be seen that CIC filters can be implemented using only adders, registers and subtractors, they do not require multipliers. Thus, CIC filters are preferred as low power decimation filters.

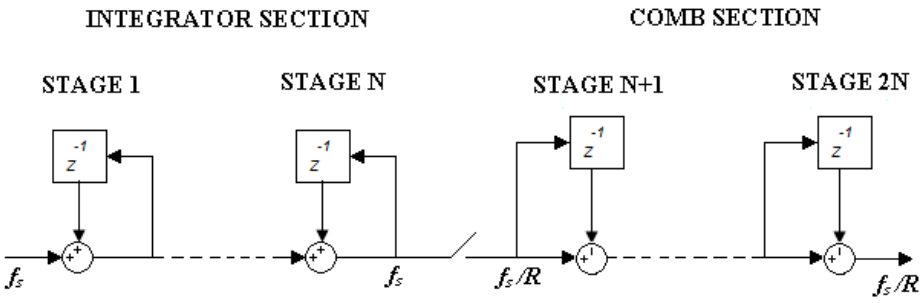

Figure 3.1. Basic structure of the CIC filter.

An integrator is a single pole IIR filter with a unity feedback coefficient. The time domain representation of an integrator is shown in equation (3.1).

$$y[n] = y[n-1] + x[n] \tag{3.1}$$

An integrator is also known as an accumulator. The transfer function of the integrator in z plane is shown in equation (3.2)

$$H_I(z) = \frac{1}{1 - z^{-1}} \tag{3.2}$$

Figure 3.2 shows the basic integrator block. It consists of a sum and a delay element. Delay element can easily be designed with a simple register that delays the input signal by one clock period.

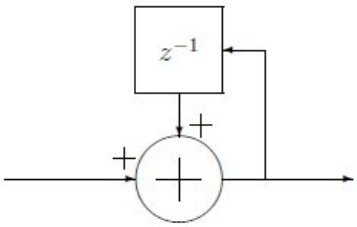

Figure 3.2. Basic integrator.

The magnitude and the phase response of the integrator in frequency domain is given by the equations (3.3) and (3.4).

$$\left|H_I(e^{jw})\right|^2 = \frac{1}{2(1 - \cos w)} \tag{3.3}$$

$$ARG[H_I(e^{jw})] = -\tan^{-1}\left[\frac{\sin w}{1 - \cos w}\right] \tag{3.4}$$

The magnitude response plot of an integrator is shown in Figure 3.3 It is a low-pass filter with a -20dB/dec roll off and, it has infinite gain at DC. Due to the single pole at *z=1* the integrator itself is unstable.

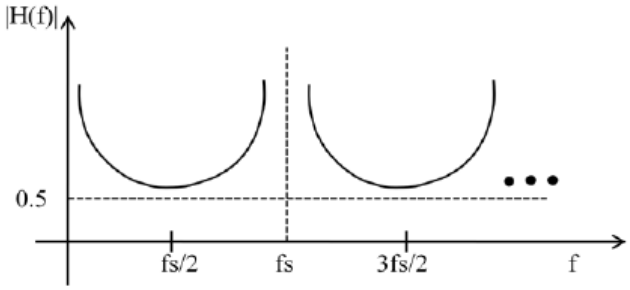

Figure 3.3. Magnitude response of an integrator.

A comb filter running at a low sampling rate f_s/R is an odd symmetric FIR filter. It is also known as differentiator. The time domain representation of a comb is shown in equation (3.5)

$$y[n] = x[n] - x[n - RM] \tag{3.5}$$

Here R is the rate change factor namely decimation ratio, M is the differential delay. M can be any positive integer but it is usually restricted to 1 or 2.

The transfer function of the comb filter in z plane is shown in equation (3.6).

$$H_C(z) = 1 - z^{-RM} \tag{3.6}$$

The comb sections are combined with a rate changer. Using a technique for multirate analysis of linear time-invariant systems, the comb sections can be pushed through rate changer and defined by equation (3.7). The advantages of (3.7) are: half of the CIC filter is slowed down and efficiency is increased; the number of delay elements required in the comb section is reduced and the integrator and comb stages are independent of rate change factor. [7]

$$y[n] = x[n] - x[n - M] \tag{3.7}$$

Figure 3.4 shows the basic comb block.

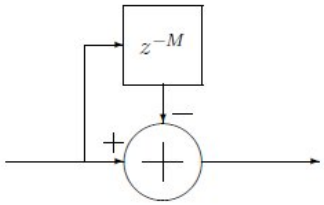

Figure 3.4. Basic comb.

The magnitude and phase response of the comb in frequency domain is given by the equations (3.7) and (3.8).

$$\left|H_C(e^{jw})\right|^2 = 2(1-\cos RMw) \tag{3.8}$$

$$ARG\left[H_C(e^{jw})\right] = -\frac{RMw}{2} \tag{3.9}$$

The magnitude response plot of the comb is shown in Figure 3.5. In the case of *R=1* and M=1 the magnitude response is a high pass function with a 20dB/dec gain [8].

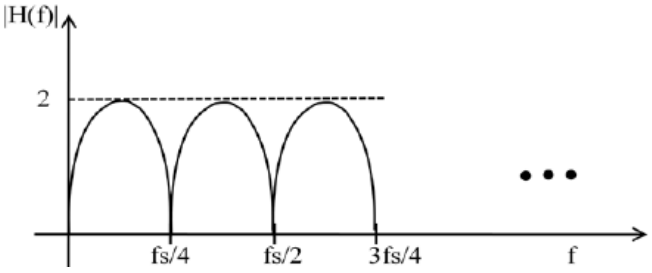

Figure 3.5. Magnitude response of a comb.

3.1.1. Frequency Characteristics of CIC Filter

From equations (3.2) and (3.5) the transfer function of the CIC filter is given by equation (3.10).

$$H(z) = H_I^N(z)H_C^N(z) = \frac{(1-z^{-RM})^N}{(1-z^{-1})^N} = \left(\sum_{k=0}^{RM-1} z^{-k}\right)^N \tag{3.10}$$

Here N is the number of stages, R is the rate change factor and, M is the differential delay. CIC filter is the same with N FIR filters that, each FIR filter has a rectangular impulse response. The magnitude response of a CIC filter is given by equation (3.11).

$$\left|H(f)\right| = \left|\frac{\sin \pi Mf}{\sin \frac{\pi f}{R}}\right|^N \tag{3.11}$$

For large rate change factor, R, the magnitude response can be approximated to equation (3.12).

$$\left|H(f)\right| \approx \left|RM \frac{\sin \pi Mf}{\pi Mf}\right|^N \quad for \quad 0 \le f < \frac{1}{M} \tag{3.12}$$

Figure 3.6 shows the magnitude response plot of the CIC filter. It can be seen that nulls exist at multiples of *f=1/M* and aliasing occurs at the region around nulls. If the passband cutoff frequency is *fc* then the aliasing region is described in equation (3.13) [6].

$$\left(i - f_C\right) \le f \le (i + f_C) \tag{3.13}$$

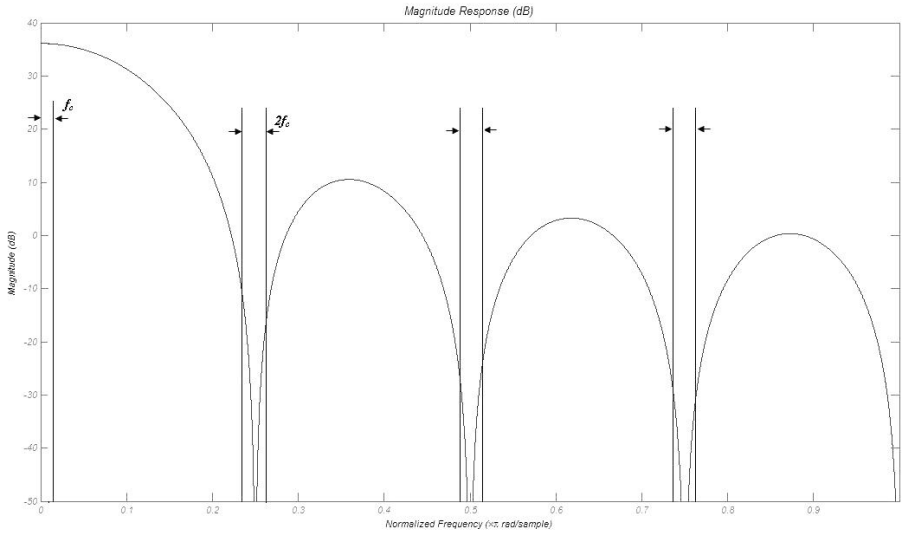

Figure 3.6. Magnitude response of a CIC filter.

3.1.2. Register Growth

Data may be lost because of the register growth in CIC filters. Data cannot be lost due to overflow as long as registers are long enough to store the largest word when using 2's complement representation.

The maximum register growth is depicted as the maximum output magnitude resulting from the worst possible input signal relative to the maximum input magnitude. This growth is used to ensure that no data are lost due to overflow. The gain at the output of the comb section is defined by (3.14).

$$G_{\max} = (RM)^N \tag{3.14}$$

In order to avoid the runtime overflow, required internal word length can be calculated from (3.15) Here *Bin* is the number of input bits and, $\lceil x \rceil$ is the smallest integer not less than *x* [9].

$$W = \lceil (1 \text{ sign bit}) + Bin + N \log_2(RM) \rceil \tag{3.15}$$

3.2. Third Order CIC Filter Design

In this study, a third order CIC filter is designed. This CIC filter reduces the sampling rate with a rate change factor of R=15. When the sigma-delta modulator order is L, the number of cascade stages is chosen L+1, in order to obtain the best attenuation [10]. In this study, the modulator order is 2, therefore, 3 cascades of stages are designed. The differential delay M is chosen as 1. Figure 3.7 shows the basic building blocks of the CIC filter. Figure 3.8 shows magnitude response of the CIC filter generated in MATLAB.

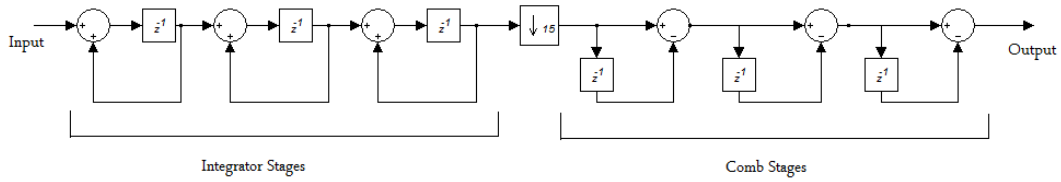

Figure 3.7. Blocks of proposed third order CIC filter.

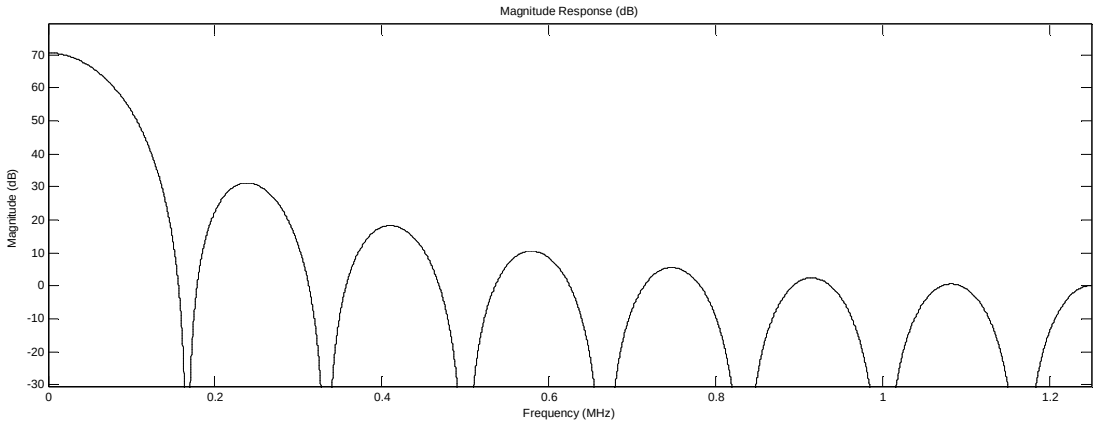

Figure 3.8. Magnitude response of proposed CIC filter.

The required internal word length can be calculated from equation (3.15).

$$W = \lceil 1 + 1 + 3\log_2(15) \rceil = 14$$

According to equation (3.15), 14 bit adders, subtractors and registers are used for all stages in CIC filter.

As explained before, 2's complement number representation is preferred in order to avoid register overflow. One bit modulator output is converted to 2's complement format. For this purpose a coder circuit is generated.

Comb stages operate at a clock rate of $f_s/15$. A clock divider circuit which divides the clock by 15 is generated for comb stages.

Figure 3.9 shows the designed CIC filter.

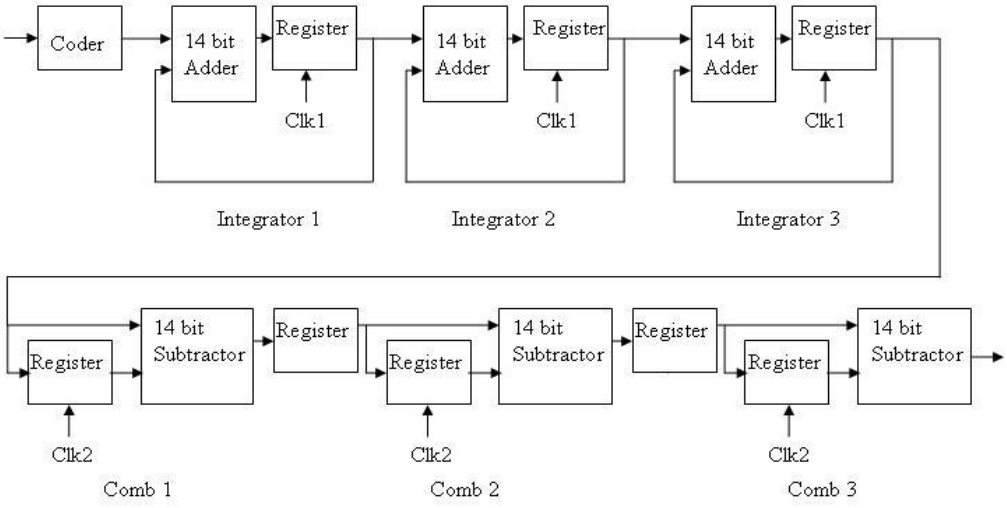

Figure 3.9. Designed CIC filter.

3.2.1. Coder Circuit

As explained before coder circuit converts the input bit to 2's complement format. Bit 1 is represented by +1 and bit 0 is represented by -1. (3.16 a) and (3.16 b) show the 14 bit representation of input bit.

$$\text{Bit } 1 \Rightarrow 00000000000001 = +1 \tag{3.16 a}$$

$$\text{Bit } 0 \Rightarrow 11111111111111 = -1 \tag{3.16 b}$$

Figure 3.10 shows the gate level schematic of the designed coder circuit.

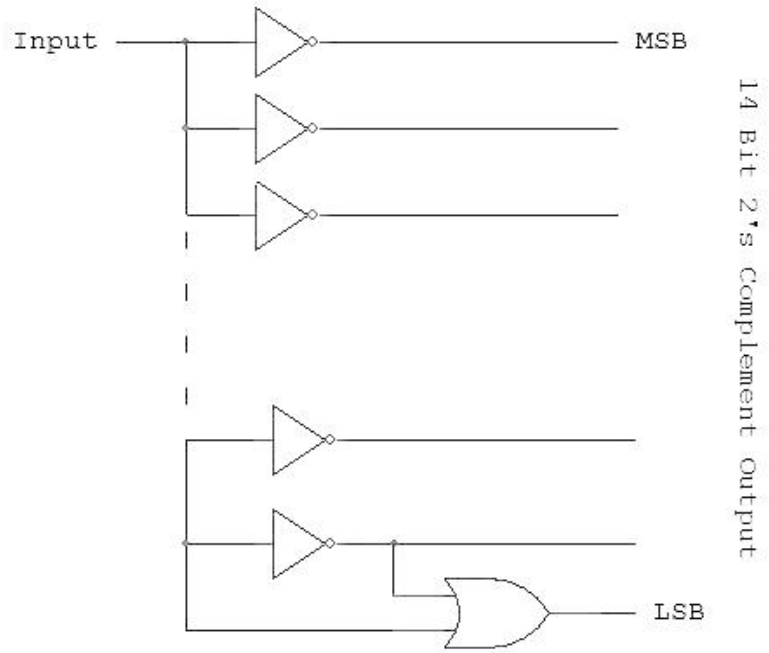

Figure 3.10. Gate level schematic of coder circuit.

3.2.2. Clock Divider Circuit

Clock divider divides the sampling frequency by rate change factor R=15 because comb stages clocked at that lower frequency. Clock divider is designed using D flip-flop loop. There is an input circuit and output circuit of D flip-flop loop. Input circuit is connected to first D flip-flop input terminal. Other input terminals of flip flops are connected directly with the output terminals of following flip flops. Finite state machine (FSM) technique is used to design the clock divider.

The procedure of clock divider design is as follows. First the number of states required is decided. 15 states are needed in order to obtain divide by 15 divider. One code is assigned to each state and each code can be used for only once. In the code system the previous code is obtained by left shift of the next code. Either 1 or 0 can be added the left shifted code's LSB bit. The last code is obtained by left shift the first code [11]. Figure 3.11 shows the code system of the divider.

$$0000 \rightarrow 1000 \rightarrow 0100 \rightarrow 1010 \rightarrow 0101 \rightarrow 0010 \rightarrow 1001 \rightarrow 1100$$
$$\rightarrow 0110 \rightarrow 1011 \rightarrow 1101 \rightarrow 1110 \rightarrow 0111 \rightarrow 0011 \rightarrow 0001 \rightarrow 0000$$

Figure 3.11. Code system of the divider.

The transition table is given in Table 3.1. Karnough map which is shown in Figure 3.12 is drawn to obtain the input equation.

Table 3.1. Transition table.

Current State $Q_A\ Q_B\ Q_C\ Q_D$	**Next State** D_A	**Output**
0000	1	0
1000	0	0
0100	1	0
1010	0	0
0101	0	0
0010	1	0
1001	1	0
1100	0	0
0110	1	0
1011	1	0
1101	1	0
1110	0	0
0111	0	0
0011	0	0
0001	0	1

Q_AQ_B \ Q_CQ_D	00	01	11	10
00	1	0	0	1
01	1	0	0	1
11	0	1	X	0
10	0	1	1	0

Figure 3.12. Karnough map.

According to Karnough map, obtained input equation (3.17) is shown below.

$$\text{Next } D_A = (A + D)' + AD(CB)' \tag{3.17}$$

The output circuit is performed according to how much duty cycle is needed. Output equation is given in (3.18) for 6%duty cycle.

$$Output = (A + B + C)'D \tag{3.18}$$

Figure 3.13 shows the gate level schematic of the clock divider circuit.

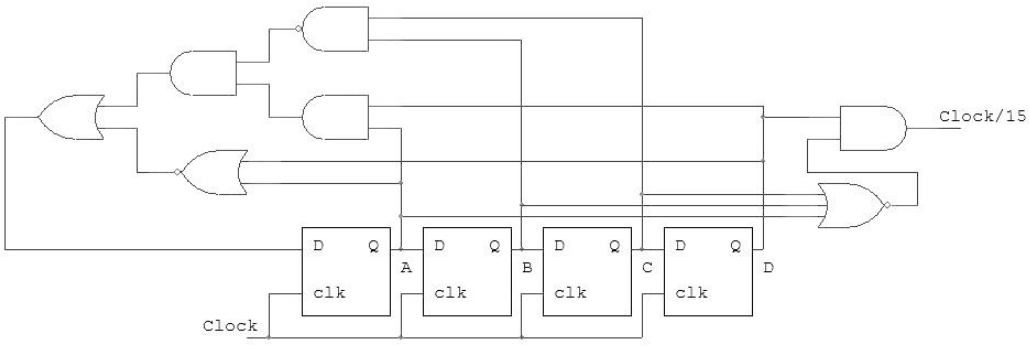

Figure 3.13. Gate level schematic of clock divider circuit.

Figure 3.14 shows simulation result of the clock divider which is generated in Modelsim.

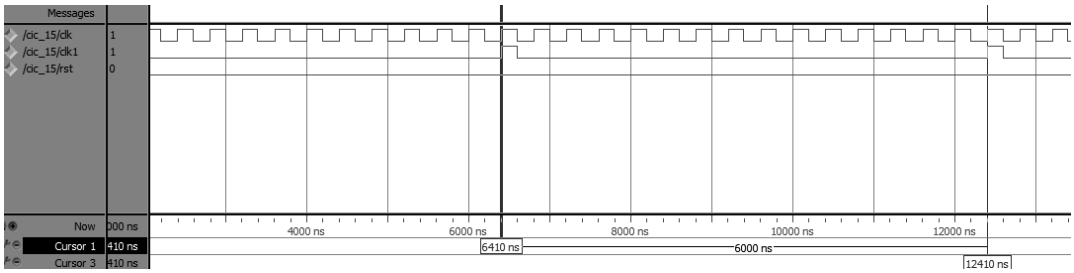

Figure 3.14. Simulation results of the clock divider.

3.2.3. Adder

It can be seen from Figure 3.9 that CIC filter needs 14 bit adders. Ripple carry adders (RCA) are preferred for the whole filter design because they are efficient in terms of hardware namely power when compared to other types of adder structures [0].

Figure 3.15 shows the structure of an n bit RCA.

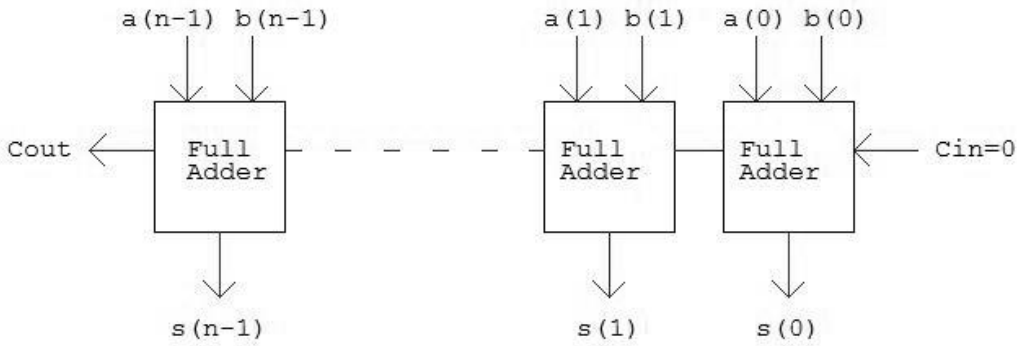

Figure 3.15. n bit RCA.

3.2.4. Subtractor

The designed CIC filter needs 14 bit 2's complement subtractors. Subtractors are obtained from adders. Figure 3.16 shows the structure of an n bit 2's complement subtractor and, this subtractor structure is used for the whole filter.

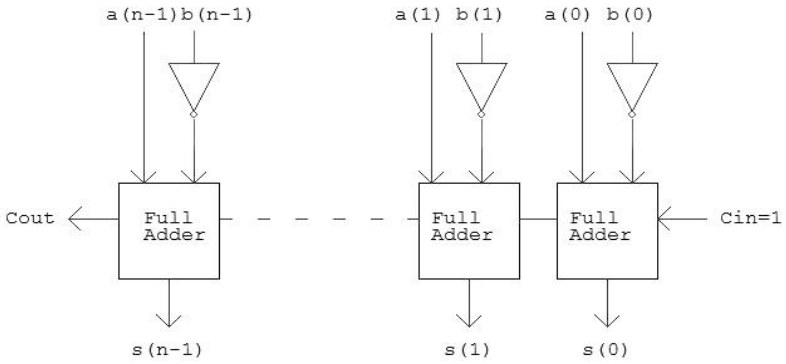

Figure 3.16. n bit 2's complement subtractor.

3.2.5. Register

CIC filter requires registers as delay elements. Registers are designed using D type flip flops. Figure 3.17 shows the structure of an n bit register and, these types of registers are used for the whole filter.

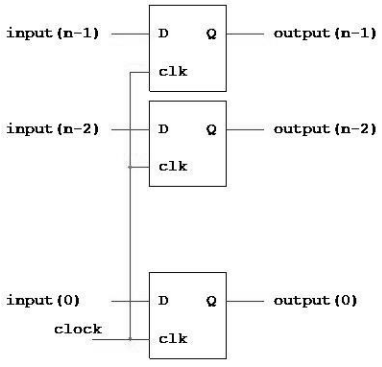

Figure 3.17. n bit register.

4. LOW POWER FINITE IMPULSE RESPONSE (FIR) FILTER

4.1. Introduction

Finite impulse response (FIR) filters are commonly used since they are simple and easy to understand. An FIR filter is a digital filter whose impulse response is finite. FIR filters are non-recursive filters, thus they have no feedback loops.

An FIR filter is implemented using an array of multipliers, adders and delay elements. Figure 4.1 shows the basic block diagram of an FIR filter. Here, N is the filter order, a_k values are filter coefficients and x_n values are input samples. The filter multiplies the most recent input samples by a coefficient array and adds the elements of the resulting array in order to obtain the output, $y(n)$. A coefficient/delay pair is called a tap.

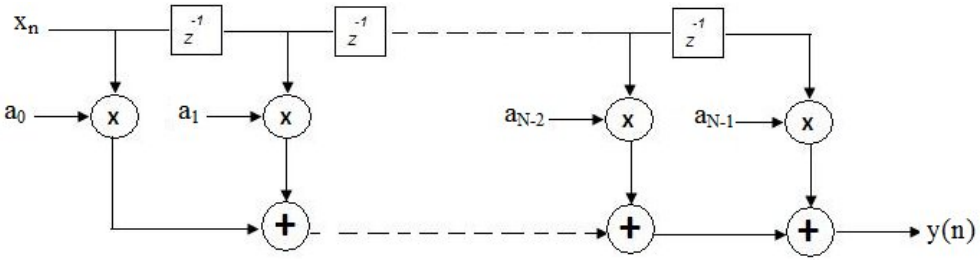

Figure 4.1. Block diagram of an FIR filter.

The time domain representation of FIR filter is shown in equation (4.1)

$$y(n) = \sum_{k=0}^{N-1} a_k x_{n-k} \tag{4.1}$$

The transfer function of FIR filter in z plane is shown in equation (4.2).

$$H(z) = \sum_{n=0}^{N-1} a_n z^{-n} \tag{4.2}$$

The frequency response of FIR filter is given in equation (4.3).

$$H(w) = \sum_{k=0}^{N-1} a_k e^{-jwk} \tag{4.3}$$

Using Euler's formula the frequency response can be expressed as equation (4.4). The frequency response of FIR filter is periodic and conjugate symmetric.

$$H(w) = \sum_{k=0}^{N-1} a_k \cos wk - j\sum_{k=0}^{N-1} a_k \sin wk \tag{4.4}$$

FIR filters have linear phase property which their phase response is a linear function of frequency. The delay through the filter is the same at all frequencies. Therefore, the filter does not cause phase distortion or delay distortion. This is an important advantage of FIR filters over IIR filters. In addition to that, linear phase FIR filters have symmetric coefficients. For an N order FIR filter, coefficients are symmetric as shown in equation (4.5).

$$a_k = a_{N-1-k} \tag{4.5}$$

Since, FIR filter requires sequential arithmetic calculations, it consumes large power. Therefore, power aware design of FIR filter is essential. A lot of design methods of low power FIR filters are proposed in the literature, for example, reducing multiples or add calculations [13-15], reducing the power consumption of adders and multipliers [16], reducing the amount of calculation using CSD (Canonical Signed Digit) expression [17], reduction of the Signed Power of Two (SPT) terms [18], minimizing the number of non-zero digits in the coefficients [19], minimization of the switching activity [20], and so on.

In this study, multiplierless FIR filter design is preferred because multipliers consume high power. Filter coefficients are chosen constant, so multiplication by a constant value can be performed by adders, subtractors and shift operations. Filter coefficients are represented by Canonical Signed Digit (CSD) format.

This representation provides less hardware complexity than 2's complement representation implementation of an FIR filter, because CSD representation reduces the number of nonzero bits in coefficients. Hence, the number of adding operation decreases.

4.2. Canonical Signed Digit (CSD) Representation

The Canonical Signed Digit (CSD) number system is a signed digit number system which minimizes the number of nonzero digits. In this number system each digit can be either -1,0, or 1 and, consecutive CSD digits are never both zero. Hence, for an n bit number, numbers of digits are at most n/2. However, in 2's complement number representation all n digits can be non zero.

For example a 16 bit number -4369 can be represented in 2's complement format as 1110111011101111, but this number can be represented in CSD format as $000\,\bar{1}\,000\,\bar{1}\,000\,\bar{1}\,000\,\bar{1}$. In CSD format -1 is represented by $\bar{1}$. It can be seen obviously from the above example CSD representation greatly reduces the number of non-zero digits.

On average, CSD numbers contain about 33% fewer non-zero digits than 2's complement numbers. Table 4.1. shows conversion from 2's complement numbers to CSD numbers [21].

Table 4.1. Conversion from 2's complement to CSD.

x_{i+1}	x_i	carry-in	carry-out	c_i
0	0	0	0	0
0	1	0	0	1
1	0	0	0	0
1	1	0	1	$\bar{1}$
0	0	1	0	1
0	1	1	1	0
1	0	1	1	$\bar{1}$
1	1	1	1	0

4.3. Common Subexpression Elimination (CSE) Method

As mentioned above multiplication by a constant value is no more than a sequence of shift, add and subtract operations. For instance, multiplication of an 8 bit number by the fixed point CSD value $[0.100\bar{1}0100\bar{1}]$ is shown in Figure 4.2. Here, circles denote addition or subtraction operation, the numbers in the circles denote word length of the results, >> *n* notation denotes right shift operation. Shift operation can be done by hardwiring. No extra circuit is required for shift operation. Thus, only adders and subtractors are considered in terms of reducing power consumption.

Common Subexpression Elimination (CSE) method is used for reducing the number of addition/subtraction operation with eliminating redundant computations in order to obtain low power FIR filter [22].

The CSE methods expressed in the literature [23-26] use horizontal common subexpressions (HCS) which occur within CSD coefficients and vertical common subexpressions (VCS) which occur among the adjacent coefficients.

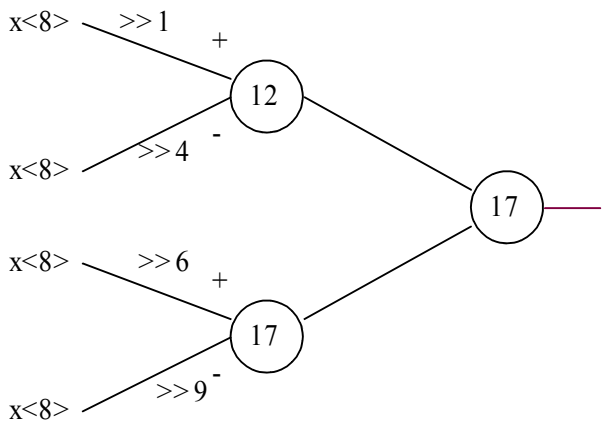

Figure 4.2. Multiplication with a CSD number.

4.3.1. Horizontal Common Subexpression Elimination (HCSE) Method

The coefficient $a_k = [0.10110\bar{1}010\bar{1}00101]$ is used as an example to describe the HCSE method. Multiplying a number x_1 by a_k without CSE method can be represented directly by equation (4.6). In order to implement this operation, there are 7 adders/subtractors are needed.

$$y_k = a_k x_1 = x_1 >> 1 + x_1 >> 3 + x_1 >> 4 - x_1 >> 6 + x_1 >> 8 - x_1 >> 10 + x_1 >> 13 + x_1 >> 15 \tag{4.6}$$

When the coefficient a_k is analyzed it can be seen that bit patterns [101] and [10$\bar{1}$] are repeated twice. These patterns can be expressed as horizontal common subexpressions (HCS) as shown in equation (4.7) and equation (4.8).

$$x_2 = [101] = x_1 + x_1 >> 2 \tag{4.7}$$

$$x_3 = [10\bar{1}] = x_1 - x_1 >> 2 \tag{4.8}$$

The output, y_k can be expressed using common subexpressions as described in equation (4.9).

$$y_k = x_2 >> 1 + x_3 >> 4 + x_3 >> 8 + x_2 >> 13 \tag{4.9}$$

In this case, only 5 adders/subtractors are used to perform the multiplication. 2 adders/subtractors are used to perform common subexpressions and 3 adders are used to perform output. There is a saving of 2 adders with HCSE method when compared to direct implementation.

4.3.2. Vertical Common Subexpression Elimination (VCSE) Method

Table 4.2 is used as an example to describe the VCSE method. The numbers in the first row express the number of right shifts. The output of the filter, y_k is expressed by the equation (4.10). Direct implementation of the filter is expressed in equation (4.11). Here, x_1 is the input signal and x_1 [-1] denotes the delayed version of it.

$$y_k = a_0 x_1 + a_1 x_1[-1] \tag{4.10}$$

$$y_k = x_1 >> 1 + x_1 >> 4 - x_1 >> 8 - x_1[-1] >> 1 + x_1[-1] >> 4 - x_1[-1] >> 8 \tag{4.11}$$

Table 4.2. Filter coefficients.

	1	**2**	**3**	**4**	**5**	**6**	**7**	**8**
a_0	1	0	0	1	0	0	0	$\bar{1}$
a_1	$\bar{1}$	0	0	1	0	0	0	$\bar{1}$

Vertical common subexpressions (VCS) exist in FIR filter since the adjacent coefficients have similar bit patterns. From Table 4.2 it can be seen that $\begin{bmatrix} 1 \\ \bar{1} \end{bmatrix}$ and $\begin{bmatrix} 1 \\ 1 \end{bmatrix}$ are vertical common subexpressions and they are expressed in equation (4.12) and equation (4.13).

$$x_2 = \begin{bmatrix} 1 \\ \bar{1} \end{bmatrix} = x_1 - x_1[-1] \tag{4.12}$$

$$x_3 = \begin{bmatrix} 1 \\ 1 \end{bmatrix} = x_1 + x_1[-1] \tag{4.13}$$

The output, y_k can be expressed using common subexpressions as described in equation (4.14).

$$y_k = x_2 >> 1 + x_3 >> 4 - x_3 >> 8 \tag{4.14}$$

Direct implementation needs 5 adders/subtractors, however, 4 adders/subtractors are needed when VCSE method is applied.

Linear phase FIR filters have symmetric coefficients (a_k=a_{n-k}). VCSE method reduces the number of adders/subtractors but in the case of VCS like $\begin{bmatrix} 1 \\ \overline{1} \end{bmatrix}$ or $\begin{bmatrix} \overline{1} \\ 1 \end{bmatrix}$ the symmetry cannot be exploited. Therefore, the filter's output cannot be directly obtained from equation (4.10) by applying simple delay, so, extra adders/subtractors are needed.

4.3.3. Horizontal Super Subexpression Elimination (HSSE) Method

HSSE method is an optimization of HCSE method. Horizontal super subexpressions can be obtained when there is identical shift between an HCS and a non zero bit or between two HCS. Therefore, redundant computations of HCSs are eliminated [27].

To illustrate HSSE method, consider the example shown in Table 4.3. A 6 tap linear phase FIR filter coefficients are given in this table.

Table 4.3. Coefficients of a 6 tap linear phase FIR filter.

	1	**2**	**3**	**4**	**5**	**6**	**7**	**8**
a_0	1	0	1	0	$\bar{1}$	0	0	0
a_1	0	1	0	$\bar{1}$	0	1	0	1
a_2	0	1	0	1	0	$\bar{1}$	0	0
a_3	0	1	0	1	0	$\bar{1}$	0	0
a_4	0	1	0	$\bar{1}$	0	1	0	1
a_5	1	0	1	0	$\bar{1}$	0	0	0

According to Table 4.3. performed HCSs are given by equation (4.15), equation (4.16).

$$x_2 = [101] = x_1 + x_1 >> 2 \tag{4.15}$$

$$x_3 = [10\bar{1}] = x_1 - x_1 >> 2 \tag{4.16}$$

There is one shift between HCS $[101]$ and $\bar{1}$ and, this pattern is repeated in the coefficients. Thus, $[1010\bar{1}]$ is performed as a HSS shown in equation (4.17). Similarly, there is one shift between HCS $[10\bar{1}]$ and $[101]$. This pattern is repeated in the coefficients as well. Hence, $[10\bar{1}0101]$ is performed as a HSS shown in equation (4.18).

$$x_4 = [1010\bar{1}] = x_2 - x_1 >> 4 \tag{4.17}$$

$$x_5 = [10\bar{1}0101] = x_3 + x_2 >> 4 \tag{4.18}$$

The output of the filter can be expressed by the following equation (4.19).

$$y_k = x_4 >> 1 + x_5[-1] >> 2 + x_4[-2] >> 2 + x_4[-3] >> 2 + x_5[-4] >> 2 + x_4[-5] >> 1 \quad (4.19)$$

With HCSE method the number of addition/subtraction is reduced from 19 to 13. With HSSE method the number of addition/subtraction is reduced from 13 to 9.

4.3.4. Vertical Super Subexpression Elimination (VSSE) Method

VSSE method is an optimization of VCSE method. Super subexpression technique, used in HCSE method, can also be applied to VCSE method [27]. To illustrate VSSE method, consider the example shown in Table 4.4. A 6 tap linear phase FIR filter coefficients are given in this table.

Table 4.4. Coefficients of a 6 tap linear phase FIR filter.

	1	**2**	**3**	**4**	**5**	**6**	**7**	**8**
a_0	1	0	$\bar{1}$	0	0	1	0	1
a_1	$\bar{1}$	0	1	0	0	1	0	1
a_2	0	$\bar{1}$	0	0	0	0	0	0
a_3	0	$\bar{1}$	0	0	0	0	0	0
a_4	$\bar{1}$	0	1	0	0	1	0	1
a_5	1	0	$\bar{1}$	0	0	1	0	1

According to Table 4.4. performed VCSs are given by equation (4.20) and equation (4.21).

$$x_2 = \begin{bmatrix} 1 \\ \bar{1} \end{bmatrix} = x_1 + x_1[-1] \quad (4.20)$$

$$x_3 = \begin{bmatrix} 1 \\ 1 \end{bmatrix} = x_1 + x_1[-1] \tag{4.21}$$

There is one shift between VCS $\begin{bmatrix} 1 \\ \bar{1} \end{bmatrix}$ and $\begin{bmatrix} \bar{1} \\ 1 \end{bmatrix}$ across coefficients a_0 and a_1. These two VCS perform a VSS. Similarly, there is one shift between two VCS $\begin{bmatrix} 1 \\ 1 \end{bmatrix}$ across coefficients a_0, a_1 and they also perform a VSS. Equation (4.22) and (4.23) show the performed VSSs.

$$x_4 = \begin{bmatrix} 10\bar{1} \\ \bar{1}01 \end{bmatrix} = x_2 - x_2 >> 2 \tag{4.22}$$

$$x_5 = \begin{bmatrix} 101 \\ 101 \end{bmatrix} = x_3 + x_3 >> 2 \tag{4.23}$$

The output of the filter can be expressed by the following equation (4.24).

$$y_k = x_4 >> 1 + x_5 >> 6 - x_1[-2] >> 2 - x_1[-3] >> 2 - x_4[-4] >> 1 + x_5[-4] >> 6 \tag{4.24}$$

With VCSE method the number of addition/subtraction is reduced from 17 to 11. With VSSE method the number of addition/subtraction is reduced from 11 to 9.

4.4. Low Power FIR Half-band Filter Design

Half-band filters have important applications in multi rate signal processing. They split the operating frequency into two equal parts. Typical magnitude response of a half-band filter is shown in Figure 4.3. [28]. The cutoff frequency for a half-band filter is always 0.5π. Moreover, the passband and stopband ripples are identical.

About half of the coefficients of a half-band filter are zero. Thus, they have higher computational efficiency than ordinary FIR filters. Half-band filters can also be used in decimation for ADC. FIR half-band filters are easily designed to prove exactly linear phase response.

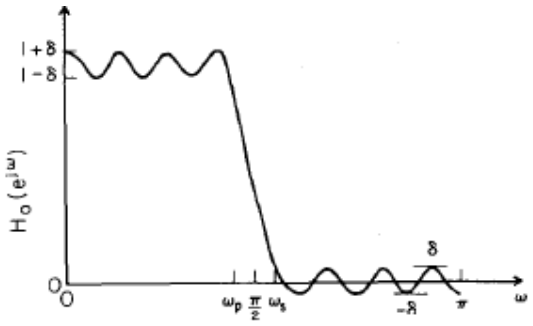

Figure 4.3. Typical magnitude response of a half-band filter.

In this thesis, a low-pass linear phase FIR half-band filter is designed as a second stage of the decimation filter. Table 4.5. shows the required filter characteristics.

Table 4.5. Half-band filter characteristics.

Filter Parameter	**Value**
Sampling frequency	f_s =166.6 KHz
Passband frequency	Fpass= 20 KHz
Passband ripple	Apass=0.01

Half-band filter is designed using MATLAB. Filter coefficients are generated with MATLAB FDATOOL which uses Parks-McCellan algorithm. In order to meet the characteristics, 14 tap half-band filter is designed. Figure 4.4. shows the corresponding magnitude response of the filter.

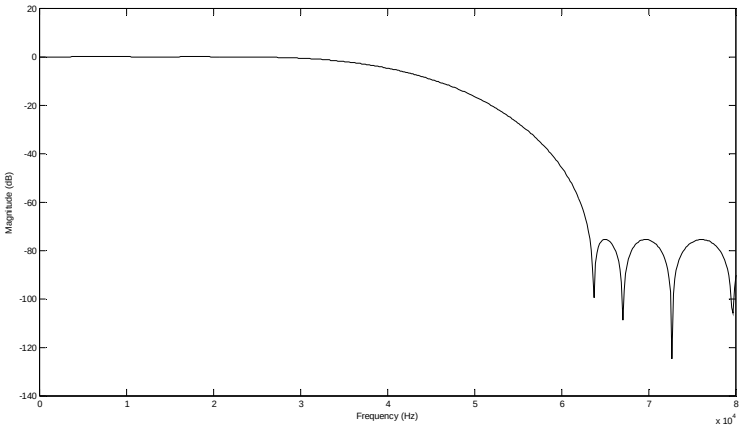

Figure 4.4. Magnitude response of the half-band filter.

Table 4.6. Actual vs. quantized coefficients of the half-band filter.

	Actual coefficients	**Quantized coefficients**
a_0	-0.034141571978849104	-0.034
a_1	0	0
a_2	0.01973469376492494	0.0198
a_3	0	0
a_4	-0.071304070035135961	-0.0713
a_5	0	0
a_6	0.30490019723637624	0.3049
a_7	0.5	0
a_8	0.30490019723637624	0.3049
a_9	0	0
a_{10}	-0.071304070035135961	-0.0713
a_{11}	0	0
a_{12}	0.01973469376492494	0.0198
a_{13}	0	0
a_{14}	-0.034141571978849104	-0.034

Half-band filter coefficients are quantized to 13 bit fixed point format using coefficient scaling theorem explained in [29]. Table 4.6. shows actual coefficients versus quantized coefficients of half-band filter.

Figure 4.5. represents the actual coefficients magnitude response versus quantized coefficients magnitude response.

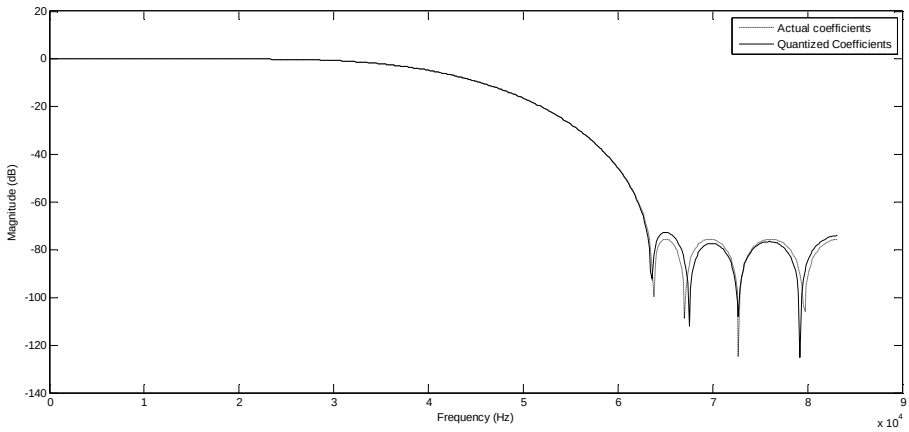

Figure 4.5. Magnitude responses of actual coefficients vs. quantized coefficients.

Table 4.7. 2's complement vs. CSD representation of half-band filter coefficients.

	2's complement format	**CSD format**
a_0	1.111111110010	$0.0000000\,\bar{1}\,0010$
a_1	0.000000000000	0.000000000000
a_2	0.000001010001	0.000001010001
a_3	0.000000000000	0.000000000000
a_4	1.111011011100	$0.000\,\bar{1}\,00\,\bar{1}\,00\,\bar{1}\,00$
a_5	0.000000000000	0.000000000000
a_6	0.010011100001	$0.010100\,\bar{1}\,00001$
a_7	0.100000000000	0.100000000000
a_8	0.010011100001	$0.010100\,\bar{1}\,00001$
a_9	0.000000000000	0.000000000000
a_{10}	1.111011011100	$0.000\,\bar{1}\,00\,\bar{1}\,00\,\bar{1}\,00$
a_{11}	0.000000000000	0.000000000000
a_{12}	0.000001010001	0.000001010001
a_{13}	0.000000000000	0.000000000000
a_{14}	1.111111110010	$0.0000000\,\bar{1}\,0010$

Quantized coefficients of half-band filter are represented in CSD format in order to reduce the number of non-zero digits within the coefficients. Table 4.7. shows the coefficients in 2's complement format and CSD format. There are 55 non-zero digits in 2's complement format, but there are only 25 non-zero digits in CSD format. About 63% reduction of non-zero digits are provided by CSD representation.

Table 4.8. CSD coefficients and applied HSSE method of half-band filter.

	1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**
a_0	0	0	0	0	0	0	0	$\bar{1}$	0	0	1	0
a_1	0	0	0	0	0	0	0	0	0	0	0	0
a_2	0	0	0	0	0	1	0	1	0	0	0	1
a_3	0	0	0	0	0	0	0	0	0	0	0	0
a_4	0	0	0	$\bar{1}$	0	0	$\bar{1}$	0	0	$\bar{1}$	0	0
a_5	0	0	0	0	0	0	0	0	0	0	0	0
a_6	0	1	0	1	0	0	$\bar{1}$	0	0	0	0	1
a_7	1	0	0	0	0	0	0	0	0	0	0	0
a_8	0	1	0	1	0	0	$\bar{1}$	0	0	0	0	1
a_9	0	0	0	0	0	0	0	0	0	0	0	0
a_{10}	0	0	0	$\bar{1}$	0	0	$\bar{1}$	0	0	$\bar{1}$	0	0
a_{11}	0	0	0	0	0	0	0	0	0	0	0	0
a_{12}	0	0	0	0	0	1	0	1	0	0	0	1
a_{13}	0	0	0	0	0	0	0	0	0	0	0	0
a_{14}	0	0	0	0	0	0	0	$\bar{1}$	0	0	1	0

After coefficients are obtained in CSD format, horizontal super subexpression elimination (HSSE) method which is explained in section 4.2.3 is applied to the coefficients.

HSSE method is preferred instead of VSSE method, because it's easy to implement this method and coefficient symmetry can be exploited. CSD coefficients of FIR half-band filter and applied HSSE method is shown in Table 4.8.

According to Table 4.8, performed HCSs are given by the equations (4.25), (4.26), (4.27) and (4.28).

$$x_2 = [\bar{1}001] = -x_1 + x_1 >> 3 \tag{4.25}$$

$$x_3 = [101] = x_1 + x_1 >> 2 \tag{4.26}$$

$$x_4 = [\bar{1}00\bar{1}] = -x_1 - x_1 >> 3 \tag{4.27}$$

$$x_5 = [\bar{1}00001] = -x_1 + x_1 >> 5 \tag{4.28}$$

It can be seen from Figure 4.6, $[1010001]$, $[\bar{1}00\bar{1}00\bar{1}]$ and $[10100\bar{1}00001]$ are obtained as HSS from HCS. These HSSs can be expressed in equations (4.29), (4.30) and (4.31).

$$x_6 = [1010001] = x_3 + x_1 >> 6 \tag{4.29}$$

$$x_7 = [\bar{1}00\bar{1}00\bar{1}] = x_4 - x_1 >> 6 \tag{4.30}$$

$$x_8 = [10100\bar{1}00001] = x_3 + x_5 >> 5 \tag{4.31}$$

The output of the filter can be expressed by the following equation (4.32).

$$y_k = x_2 >> 8 + x_6[-2] >> 6 + x_7[-4] >> 4 + x_8[-6] >> 2 + x_1[-7] >> 1 + x_8[-8] >> 2 + x_7[-10] >> 4 + x_6[-12] >> 6 + x_2[-14] >> 8 \quad (4.32)$$

It is observed that with HSSE method the number of adders/subtractors reduced from 24 to 15. About 37% reduction of adders/subtractors is obtained.

Figure 4.6 shows the proposed FIR half-band filter structure. The filter is implemented with a tree structure which performs parallel addition. 15 adders/subtractors and 14 registers are required in order to implement the filter. Here, the numbers denote the amount of right shift. Overflow due to adders/subtractors is omitted in observed CSE methods. In order to prevent wrong calculations because of overflow problem, one bit sign extension is applied to all terms in the filter structure.

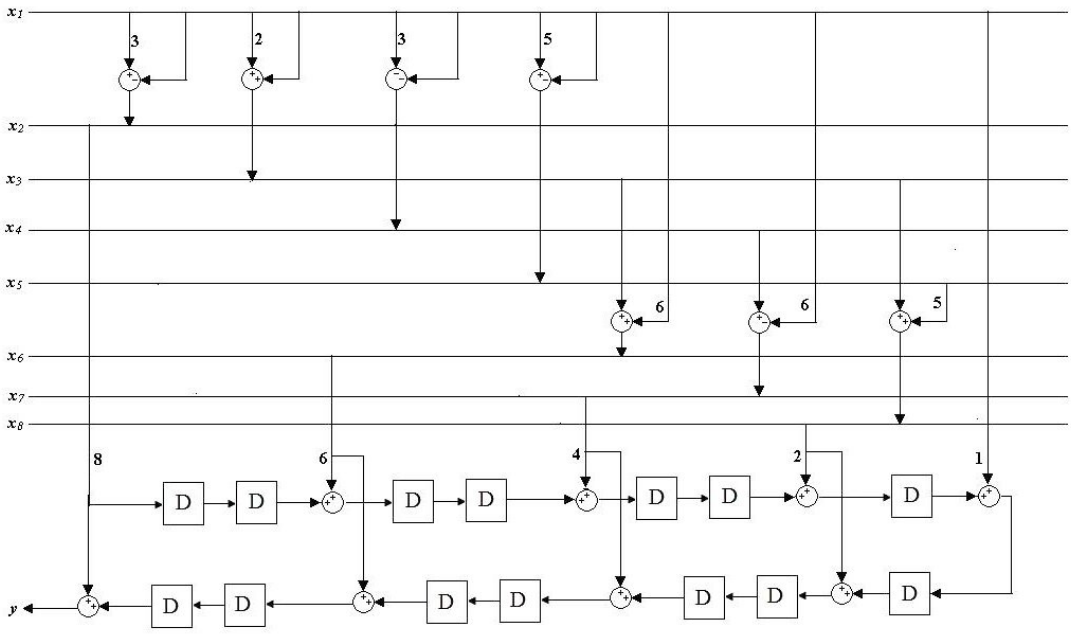

Figure 4.6. Proposed FIR half-band filter structure.

As explained before, Ripple Carry Adders (RCA) are used to implement n bit adders and subtractors. In addition, D flip-flops are used to implement n bit registers.

4.5. Low Power FIR Filter Design

A low-pass linear phase FIR filter is designed as a third and last stage of the multistage decimation filter. FIR filter is designed as the same procedure as half-band filter design.

Table 4.9 shows the required filter characteristics. Like half-band filter, FIR filter is designed using MATLAB FDATOOL. 13 tap FIR filter is needed to satisfy the required characteristics given in Table 4.9.

Table 4.9. FIR filter characteristics.

Filter Parameter	**Value**
Sampling frequency	f_s =83.3 KHz
Passband frequency	Fpass= 20 KHz
Stopband frequency	Fstop=41 KHz
Passband ripple	Apass=0.01
Stopband attenuation	Astop=70 dB

Figure 4.7 shows magnitude response of the required 13 tap FIR filter.

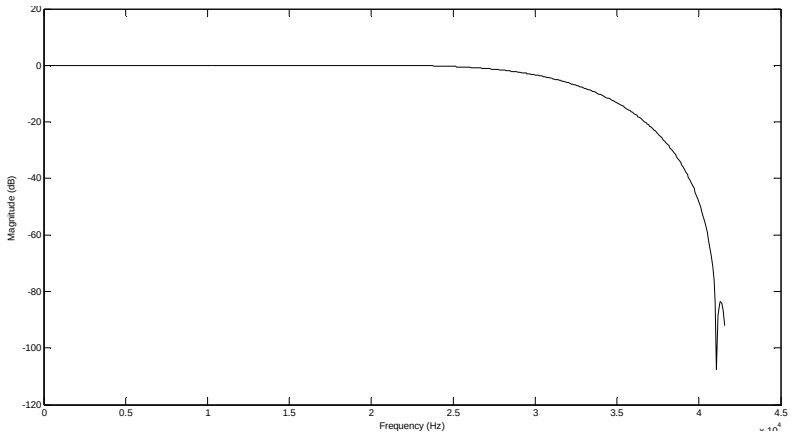

Figure 4.7. Magnitude response of the FIR filter.

According to the coefficient scaling theorem [28], FIR filter coefficients are quantized to 13 bit fixed point format. Table 4.10 shows actual coefficients versus quantized coefficients of FIR filter.

Table 4.10. Actual vs. quantized coefficients of the FIR filter.

	Actual coefficients	**Quantized coefficients**
a_0	-0.0018805467722019604	-0.0020
a_1	0.010616895088591529	0.0105
a_2	0.028639752701422945	-0.0286
a_3	0.044923861988961887	0.0449
a_4	0.26654252678170245	-0.0266
a_5	-0.085228278077465483	-0.0852
a_6	0.58692331564458	0.5869
a_7	0.58692331564458	0.5869
a_8	-0.085228278077465483	-0.0852
a_9	0.26654252678170245	-0.0266
a_{10}	0.044923861988961887	0.0449
a_{11}	0.028639752701422945	-0.0286
a_{12}	0.010616895088591529	0.0105
a_{13}	-0.0018805467722019604	-0.0020

The actual coefficients magnitude response versus quantized coefficients magnitude response is shown in Figure 4.8.

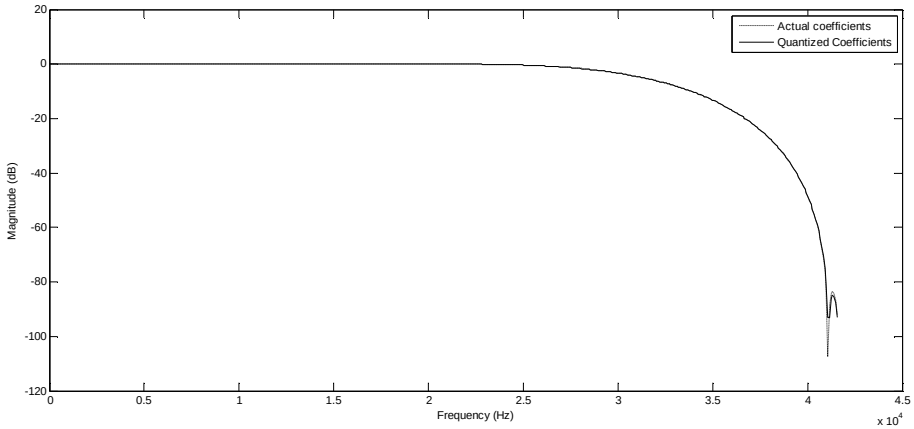

Figure 4.8. Magnitude responses of actual coefficients vs. quantized coefficients.

CSD representation is used for quantized coefficients of FIR filter so that the number of non-zero digits is decreased.

Table 4.11. 2's complement vs. CSD representation of FIR filter coefficients.

	2's complement format	**CSD format**
a_0	1.111111111000	$0.00000000\,\bar{1}\,000$
a_1	0.000000101011	$0.0000010\,\bar{1}\,0\,\bar{1}\,0\,\bar{1}$
a_2	1.111110001011	$0.0000\,\bar{1}\,0010\,\bar{1}\,0\,\bar{1}$
a_3	0.000010111000	$0.00010\,\bar{1}\,00\,\bar{1}\,000$
a_4	1.111110010011	$0.0000\,\bar{1}\,001010\,\bar{1}$
a_5	1.111010100011	$0.00\,\bar{1}\,01010010\,\bar{1}$
a_6	0.100101100100	$0.1010\,\bar{1}\,0\,\bar{1}\,00100$
a_7	0.100101100100	$0.1010\,\bar{1}\,0\,\bar{1}\,00100$
a_8	1.111010100011	$0.00\,\bar{1}\,01010010\,\bar{1}$
a_9	1.111110010011	$0.0000\,\bar{1}\,001010\,\bar{1}$
a_{10}	0.000010111000	$0.00010\,\bar{1}\,00\,\bar{1}\,000$
a_{11}	1.111110001011	$0.0000\,\bar{1}\,0010\,\bar{1}\,0\,\bar{1}$
a_{12}	0.000000101011	$0.0000010\,\bar{1}\,0\,\bar{1}\,0\,\bar{1}$
a_{13}	1.111111111000	$0.00000000\,\bar{1}\,000$

Table 4.11 shows the coefficients in 2's complement format and CSD format. There are 98 non-zero digits in 2's complement format, but there are only 52 non-zero digits in CSD format. About 46% reduction of non-zero digits are provided by CSD representation.

Similar to FIR half-band filter, HSSE method which is explained in section 4.2.3 is applied to FIR filter coefficients in order to implement a low power FIR filter. CSD coefficients of the designed FIR filter and applied HSSE method is shown in Table 4.12.

Table 4.12. CSD coefficients and applied HSSE method of FIR filter.

	1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**
a_0	0	0	0	0	0	0	0	0	$\bar{1}$	0	0	0
a_1	0	0	0	0	0	1	0	$\bar{1}$	0	$\bar{1}$	0	$\bar{1}$
a_2	0	0	0	0	$\bar{1}$	0	0	1	0	$\bar{1}$	0	$\bar{1}$
a_3	0	0	0	1	0	$\bar{1}$	0	0	$\bar{1}$	0	0	0
a_4	0	0	0	0	$\bar{1}$	0	0	1	0	1	0	$\bar{1}$
a_5	0	0	$\bar{1}$	0	1	0	1	0	0	1	0	$\bar{1}$
a_6	1	0	1	0	$\bar{1}$	0	$\bar{1}$	0	0	1	0	0
a_7	1	0	1	0	$\bar{1}$	0	$\bar{1}$	0	0	1	0	0
a_8	0	0	$\bar{1}$	0	1	0	1	0	0	1	0	$\bar{1}$
a_9	0	0	0	0	$\bar{1}$	0	0	1	0	1	0	$\bar{1}$
a_{10}	0	0	0	1	0	$\bar{1}$	0	0	$\bar{1}$	0	0	0
a_{11}	0	0	0	0	$\bar{1}$	0	0	1	0	$\bar{1}$	0	$\bar{1}$
a_{12}	0	0	0	0	0	1	0	$\bar{1}$	0	$\bar{1}$	0	$\bar{1}$
a_{13}	0	0	0	0	0	0	0	0	$\bar{1}$	0	0	0

According to Table 4.12, performed HCSs are given by the equations (4.33), (4.34) and (4.35).

$$x_2 = [10\bar{1}] = x_1 - x_1 >> 2 \tag{4.33}$$

$$x_3 = [101] = x_1 + x_1 >> 2 \tag{4.34}$$

$$x_4 = [\bar{1}001] = -x_1 + x_1 >> 3 \tag{4.35}$$

Performed HSSs are given in equations from (4.36) to (4.43).

$$x_5 = [10\bar{1}0\bar{1}0\bar{1}] = x_1 - x_3 >> 4 \tag{4.36}$$

$$x_6 = [\bar{1}0010\bar{1}0\bar{1}] = x_4 - x_3 >> 5 \tag{4.37}$$

$$x_7 = [10\bar{1}00\bar{1}] = x_2 - x_1 >> 5 \tag{4.38}$$

$$x_8 = [\bar{1}001010\bar{1}] = x_4 + x_2 >> 5 \tag{4.39}$$

$$x_9 = [\bar{1}0101] = -x_2 + x_1 >> 4 \tag{4.40}$$

$$x_{10} = [\bar{1}01010010\bar{1}] = x_9 + x_2 >> 7 \tag{4.41}$$

$$x_{11} = [1010\bar{1}] = x_3 - x_1 >> 4 \tag{4.42}$$

$$x_{12} = [1010\bar{1}0\bar{1}001] = x_{11} + x_4 >> 6 \tag{4.43}$$

The output of the FIR filter is shown in equation (4.44).

$$\begin{aligned} y = &-x_1 >> 9 + x_5[-1] >> 6 + x_6[-2] >> 5 + x_7[-3] >> 4 + x_8[-4] >> 5 + x_{10}[-5] >> 3 \\ &+ x_{12}[-6] >> 1 + x_{12}[-7] >> 1 + x_{10}[-8] >> 3 + x_8[-9] >> 5 + x_7[-10] >> 5 \\ &+ x_6[-11] >> 5 + x_5[-12] >> 6 - x_1[-13] >> 9 \end{aligned} \tag{4.44}$$

When applied HSSE method is analyzed it can be seen that the number of adders/subtractors reduced from 51 to 24. The reduction rate of adders/subtractors is about 52%.

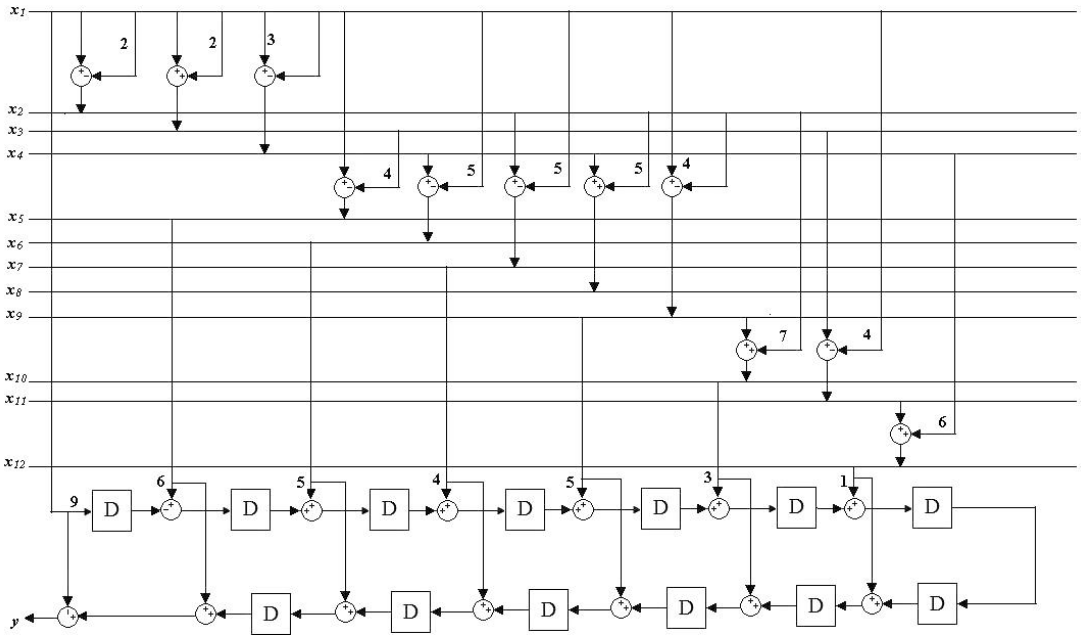

Figure 4.9. Proposed FIR filter structure.

The proposed FIR filter structure can be seen from Figure 4.9 The FIR filter is implemented using parallel addition as half-band filter.

24 adders/subtractors and 13 registers are required in order to implement the filter. Again due to overflow problem, one bit sign extension is applied to all terms in the structure.

The resolution of sigma-delta ADC is 10 bits. Therefore, output word length of the decimation filter must be 10 bits. Table 4.13 shows input and output word length of the designed filters.

Table 4.13. Input and output word length of the filters.

	Input Word length	**Output Word length**
CIC Filter	1	14
Half-band Filter	14	27
FIR Filter	27	40

It can be seen from Table 4.13 output word length of the designed decimation filter is 40 bits. In order to obtain 10 bits at the output, 30 bits must be truncated. For achieving the lowest error, 30 LSB bits must be truncated at the end of the filter. However, the filter consumes high power when this method is used. Instead of this, 30 bits are truncated within the filters shown in Table 4.14.

Table 4.14. Input and output word length of optimized filters.

	Input Word length	**Output Word length**
CIC Filter	1	14
Half-band Filter	14	16
FIR Filter	16	10

Figure 4.10, Figure 4.11 and Figure 4.12 show the simulation outputs of the filter for different input values and different time samples in Modelsim. Here, *output* signal represents the output of the filter when all 30 LSB bits are truncated at the end of the filter. However, *output_disc* signal represents the output of the filter when 30 LSB bits are truncated within the filters.

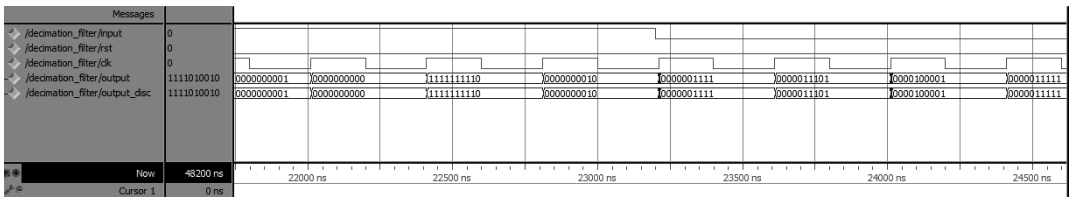

Figure 4.10. Output comparison of the filter.

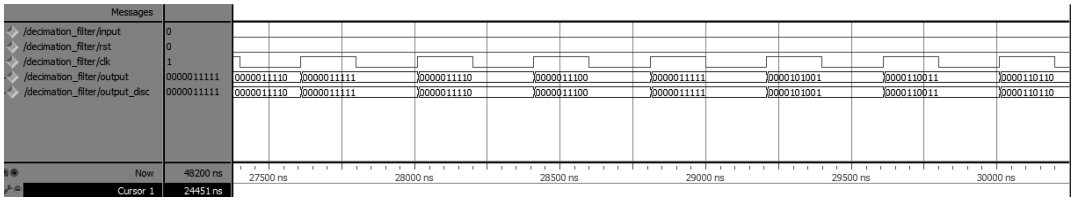

Figure 4.11. Output comparison of the filter.

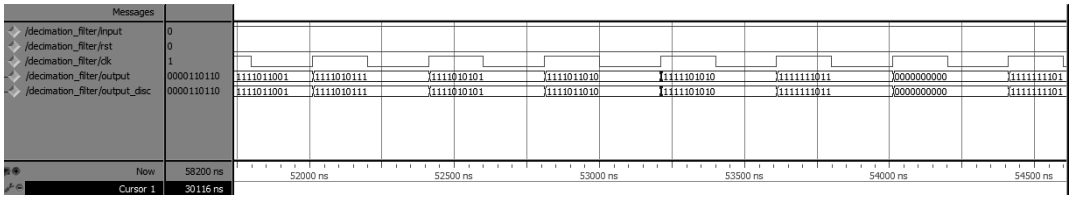

Figure 4.12. Output comparison of the filter.

It can obviously be seen that there is no difference between *output* and *output_disc* because the outputs for each filter are designed for achieving minimum error.

5. GAM ALGORITHM

GAM is an algorithm proposed by Mustafa Aktan for designing low power/hardware efficient linear phase FIR filters by reducing the number of signed power of two (SPT) terms in the coefficients while keeping the quantization word length as small as possible [30]. The algorithm finds the filter coefficients according to the given filter frequency characteristics. GAM is a branch-and-bound (BAB) based algorithm that fixes a coefficient to a certain value. The value is designated by finding the boundary values of the coefficient using linear programming [30].

GAM algorithm iteratively finds the coefficients of a linear phase FIR filter according to its zero-phase magnitude response for given word length B and filter length N. The frequency response of an FIR filter is given in (4.3). Equation (4.3) can be written in terms of amplitude and phase terms.

$$H(w) = A(w)e^{-jw(M-1)} \tag{5.1}$$

M is approximately half the length of the filter, N, and described by (5.2)

$$M = \left\lceil \frac{N+1}{2} \right\rceil \tag{5.2}$$

The amplitude, *A(w)* is a real function and defined by (5.3). Here, $T_m(w)$ is the trigonometric function which is determined by the type of symmetry and length of the filter.

$$A(w) = \sum_{m=0}^{M-1} h[m] T_m(w) \tag{5.3}$$

In filter design, the aim is to find a filter with a frequency response approximates to the desired frequency response. If *D(w)* is desired frequency response and *δ(w)* is the approximation error, linear phase FIR filter should satisfy equation (5.4)

$$\left|A(w)-D(w)\right| \le \delta(w),\ w \in [0,\pi] \tag{5.4}$$

An FIR filter is defined with some disjoint frequency bands, $\Omega_k \subset [0,\pi]$, desired frequency response $D_k(w,)$ and fixed error margin δ_k, k =1,2,…K. Equation (5.3) can be rewritten as (5.5).

$$\left|A(w)-D_k(w)\right| \le \delta_k(w),\ w \in [0,\pi] \tag{5.5}$$

The width of a frequency band, $\Omega = [w_1, w_2]$ is expressed by (5.6). Here, w_1 and w_2 are edge frequencies.

$$\left|\Omega\right| = w_2 - w_1 \tag{5.6}$$

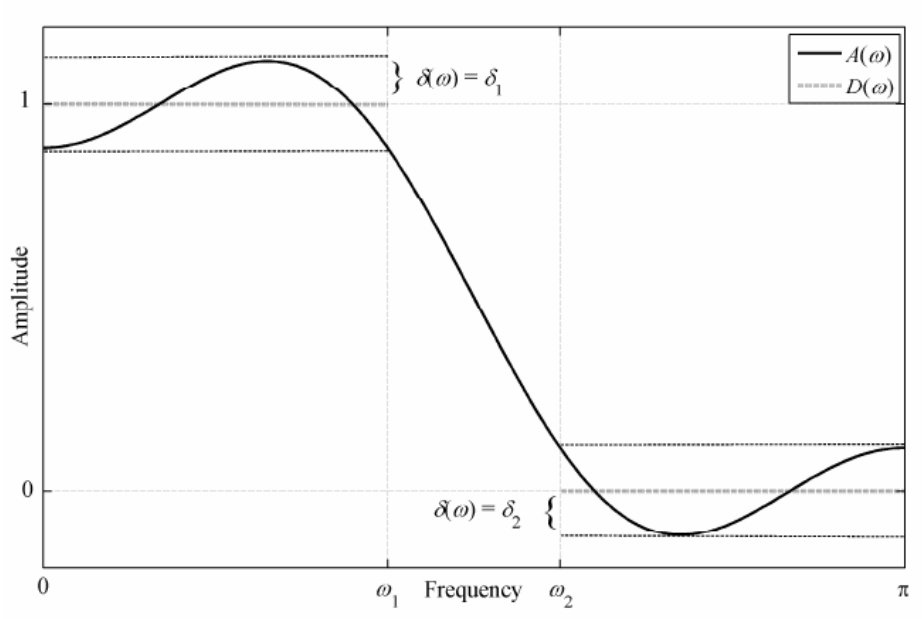

Figure 5.1. Zero-phase magnitude response of a low-pass FIR filter.

There are lots of coefficients that satisfy equation (5.4). However, when the filter length, N, is fixed, the range of all possible values for each coefficient, $h[i]$, $i=0,1,...M-1$ can be determined by solving set of linear optimization problems independently.

$$\left|\sum_{m=0}^{M-1} h[m]T_m(w) - D(w)\right| \le \delta(w) \tag{5.7}$$

$$h_{min}[i] = \text{minimize } h[i]$$

$$h_{max}[i] = \text{maximize } h[i]$$

Coefficient $h[i]$ is in the range of $[h_{min}[i],\ h_{max}[i]]$. If coefficient word length is not finite, coefficients can take infinite numbers. However, for finite word length, B, all possible values of $h[i]$ form a finite set and each set must satisfy (5.4).

The algorithm finds the feasible value set. If the number of taps (N) or word length is less than needed, an empty value set is generated, the algorithm returns an empty set.

Figure 5.2 shows the pseudo-code of the GAM algorithm.

```
GAM(M , L, ⋃_{m=0}^{M} V_m )
  H* = ∅;
  Obtain h^s_min[0] and h^s_max[0] by solving equations (2.10) and (2.11);
  V_0^s = VALUE_SELECT(L, h^s_min[0], h^s_max[0], V_0);
  i = 0;
  WHILE (i ≥ 0)
    IF (V_i^s ≠ ∅)
      h[i] = v* such that v* is the first element of ordered set V_i^s;
      H* = H* ∪ {h[i]};
      V_i^s = V_i^s − {v*};
        IF (i < M−1)
          i = i + 1;
          Obtain h^s_min[i] and h^s_max[i] by solving equations (2.10) and (2.11);
          V_i^s = VALUE_SELECT(L, h^s_min[i], h^s_max[i], V_i);
        ELSE-IF (problem is feasible)
          H* is a solution to the problem;
        END-IF
    ELSE
      H* = H* − {h[i]};
      i = i − 1;
    END-IF
  END
```

Figure 5.2. GAM Algorithm.

Assume V_i is a digital value set of $h[i]$. The operation of the algorithm can be explained as follows: the algorithm selects iteratively a value v^* from refined value set V_i^s of tap coefficient $h[i]$ from starting the initial tap ($i=0$). v^* is closest to the average of minimum ($h_{\min}^s[i]$) and maximum ($h_{\max}^s[i]$) values of $h[i]$. This value is assigned to $h[i]$ and v^* is removed from value set and moved to solution set, H^*. Therefore, the first element in H^* is $h[0]$ and the last element is $h[M-1]$. At each iteration only one tap is fixed and following optimization equations are solved for the next tap:

$$h_{\min}^s[i] = \text{minimize } h[i]$$

$$\left|\sum_{m=0}^{M-1} h[m]T_m(w) - D(w)\right| \le \delta(w)$$

$$h_{\max}^s[i] = \text{maximize } h[i]$$

$$\left|\sum_{m=0}^{M-1} h[m]T_m(w) - D(w)\right| \le \delta(w)$$

where $w \in [0, \pi]$ and $h[m] = \begin{cases} fixed & 0 \le m < i \\ free & i \le m < M \end{cases}$ for both problems.

The superscript s denotes refined variables. The algorithm refines the search space for the i'th tap after fixing $i-1$ taps. Fixing $i-1$ taps moves the boundary values of i'th tap closest to each other, therefore the number of possible values for this tap is reduced. If the refined value set V_i^s of $h[i]$ is empty, then the selected value for $h[i-1]$ is removed from H^* and the next value in the value set of $h[i-1]$ is selected for the next iteration and this is repeated until a nonempty value set of $h[i]$ is obtained.

The algorithm finds a solution until all M taps are selected. The solution is found in at most L^M iterations, where L is a parameter used to limit the maximum size of the refined value sets V_i^s. Practically, setting $L = 2$ is enough when the coefficient word length B is set to its lowest possible value.

Half-band filter described in section 4.4 and FIR filter described in section 4.5 are designed manually. In the following sections the same filters are designed using GAM algorithm in order to compare these two processes.

5.1. FIR Half-band Filter Design Using GAM Algorithm

In order to design FIR half-band filter using GAM algorithm, five different programs are used consecutively: FRQ, HMX, GAM, CSE and VHD [30]. The program FRQ generates a file that contains frequency response characteristics of the filter. Figure 5.3 shows the frequency band specifications which are required by FRQ.

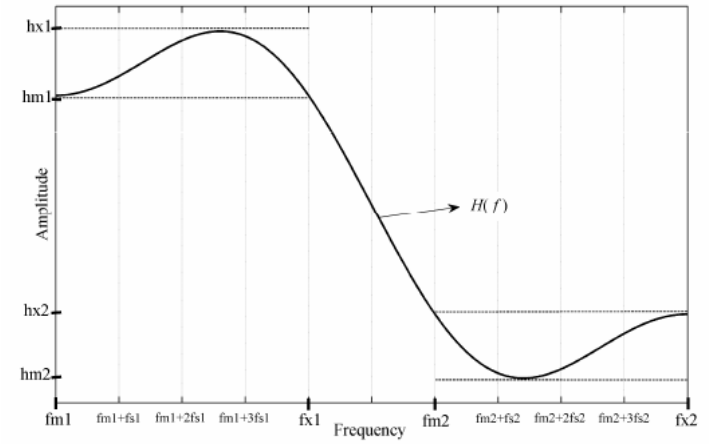

Figure 5.3. Frequency band specifications of the filter.

The parameters in Figure 5.3 denote as follows:

fm1 : cut-in frequency
fx1 : cut-out frequency
fs1 : frequency step
hm1 : minimum amplitude
hx1 : maximum amplitude

First, the required frequency characteristics of the filter are determined. The frequency characteristics of half-band filter are the same with the filter described in section 4.4. GAM algorithm uses zero-phase frequency response characteristics. Figure 5.4 shows zero-phase response of the designed half-band filter.

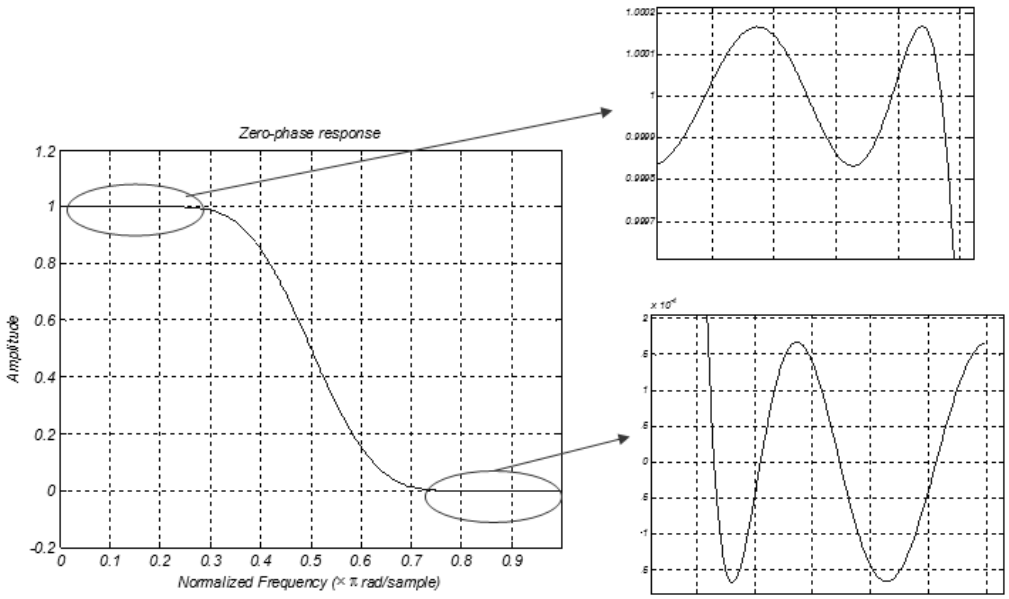

Figure 5.4. Zero-phase response of the half-band filter.

The program HMX is used to determine the boundary values of the coefficients given the filter length N, quantization word length B and the frequency response that is generated by FRQ. Here, N is 15 and B is 13. Coefficients are represented in 2's complement format.

Table 5.1. Minimum and maximum values of the half-band filter coefficients.

h[*0*]	1111111011100	0000000001001
h[*1*]	1111111011001	0000000100111
h[*2*]	0000000011110	0000010000100
h[*3*]	1111110000011	0000001111101
h[*4*]	1111010100000	1111100010111
h[*5*]	1111100011111	0000011100001
h[*6*]	0010011000111	0010011111100
h[*7*]	0011011110010	0100100001110

The GAM program applies the algorithm. Here, the algorithm is used for minimization of SPT terms in CSD notation because it gives the best result in terms of low power. Figure 5.5 shows the frequency response of the designed half-band filter using GAM algorithm.

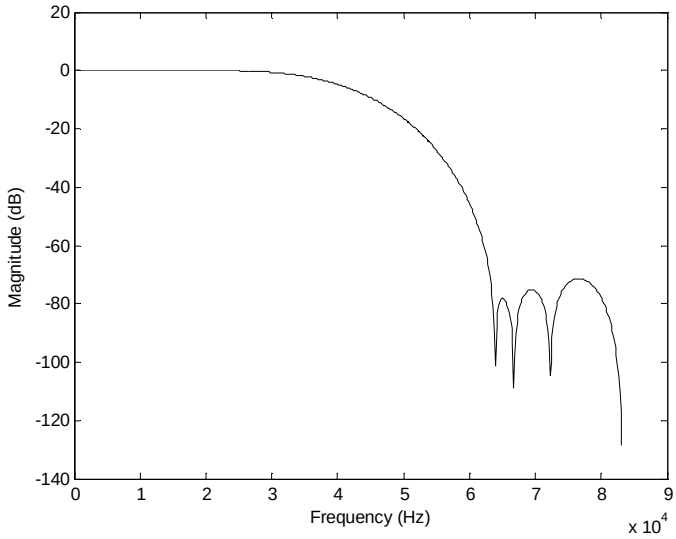

Figure 5.5. Frequency response of the half-band filter.

After coefficients are obtained, the program CSE applies CSE algorithm to the CSD coefficients of the filter. Table 5.6 gives the CSD coefficients of the half-band filter. Here, N represents -1.

Table 5.2. CSD coefficients of the half-band filter.

h[*0*]	00000000N0010	*h*[*8*]	0010100N00001
h[*1*]	0000000000000	*h*[*9*]	0000000000000
h[*2*]	0000001010001	*h*[*10*]	0000N00N00N00
h[*3*]	0000000000000	*h*[*11*]	0000000000000
h[*4*]	0000N00N00N00	*h*[*12*]	0000001010001
h[*5*]	0000000000000	*h*[*13*]	0000000000000
h[*6*]	0010100N00001	*h*[*14*]	00000000N0010
h[*7*]	0100000000000		

The program CSE generates netlist of the designed filter. The algorithm uses multiplierless FIR filter design architecture in terms of hardware. Adders/subtractors are replaced by multipliers because filter coefficients are constant. Ripple carry adders and D type flip flops are used.

```
a2 =   (in << 3) - (in << 0)
a4 =  (in << 0) + (in << 4)
a5 =  (a4 << 0) + (in << 6)
a6 =  (in << 0) + (in << 3)
a7 =  (a6 << 0) + (in << 6)
a8 =  (in << 0) + (in << 10)
a9 =  (a8 << 0) + (a2 << 5)

d0 =     - (a2 << 1)
d1 =  d0
t2 =  d1 + (a5 << 0)
d2 =  t2
d3 =  d2
t4 =  d3 - (a7 << 2)
d4 =  t4
d5 =  d4
t6 =  d5 + (a9 << 0)
d6 =  t6
t7 =  d6 + (in << 11)
d7 =  t7
t8 =  d7 + (a9 << 0)
d8 =  t8
d9 =  d8
t10 =  d9 - (a7 << 2)
d10 =  t10
d11 =  d10
t12 =  d11 + (a5 << 0)
d12 =  t12
d13 =  d12
t14 =  d13 - (a2 << 1)
o14 =  t14
```

Figure 5.6. Netlist of the half-band filter.

Figure 5.6 shows generated netlist of the half-band filter. Here, a denotes adder, t denotes tap adder, d denotes delay element, in denotes input, and o denotes output.

After all, the program VHD generates vhd netlist of the designed filter for given input word length.

5.2. FIR Filter Design Using GAM Algorithm

FIR filter is designed using same procedure as half-band filter described in section 5.1. First, zero-phase response of the FIR filter is obtained in order to apply the algorithm. Designed FIR filter using GAM algorithm has the same frequency response with the filter designed manually in section 4.5. Figure 5.7 shows zero-phase response of the FIR filter.

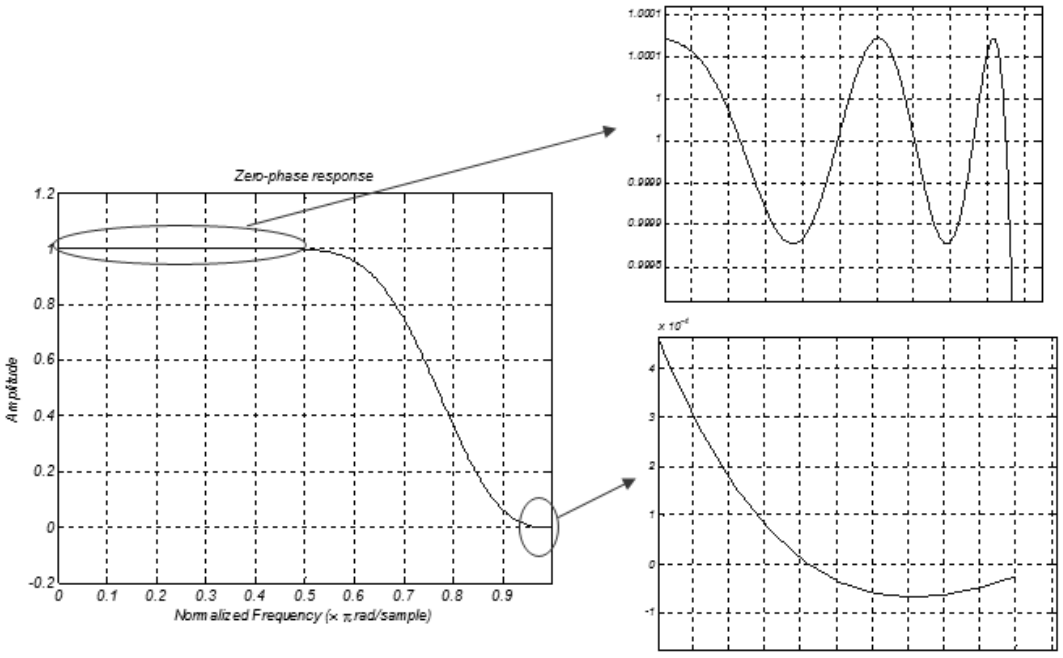

Figure 5.7. Zero-phase response of the FIR filter.

The boundary values of coefficients are given in Table 5.3. Here, N is 14 and B is 12.

Table 5.3. Minimum and maximum values of the FIR filter coefficients.

h[*0*]	111111110011	000000000100
h[*1*]	000000000001	000000101101
h[*2*]	111110101101	111111011010
h[*3*]	000001010100	000001100100
h[*4*]	111110011000	000000000010
h[*5*]	111100000111	111110010011
h[*6*]	010010010100	010011010100

Like half-band filter, minimization of SPT terms in CSD notation is used in the GAM algorithm for FIR filter. Figure 5.8 shows the frequency response of the FIR filter using GAM algorithm and Table 5.4 gives CSD coefficients of the FIR filter.

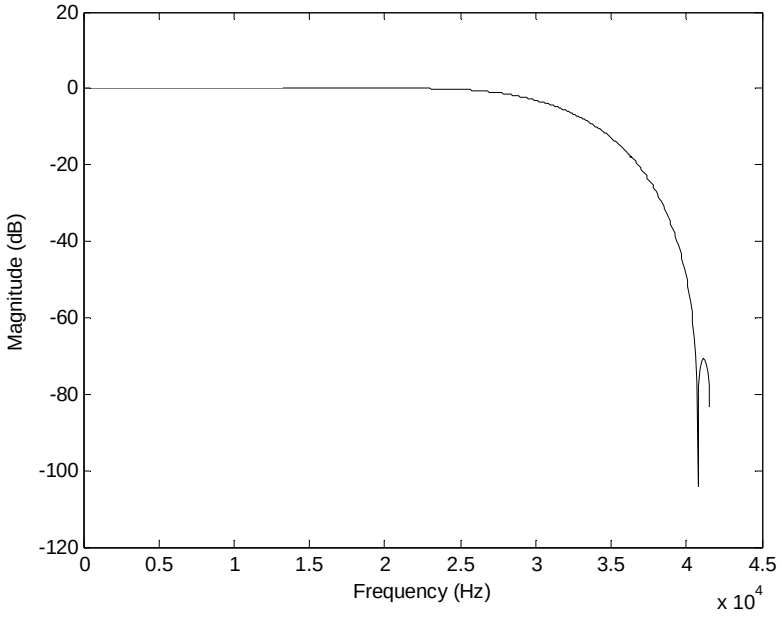

Figure 5.8. Frequency response of the FIR filter.

Table 5.4. CSD coefficients of the FIR filter.

h[*0*]	000000000N00	*h*[*8*]	01010N0N0010
h[*1*]	00000010N0N0	*h*[*9*]	000N01010000
h[*2*]	00000N000101	*h*[*10*]	00000N001010
h[*3*]	000010N00N00	*h*[*11*]	000010N00N00
h[*4*]	00000N001010	*h*[*12*]	00000N000101
h[*5*]	000N01010000	*h*[*13*]	00000010N0N0
h[*6*]	01010N0N0010	*h*[*14*]	000000000N00

Figure 5.9 gives the generated netlist of the designed FIR filter using algorithm.

```
a2 =  (in << 0) - (in << 2)
a3 =  (in << 0) + (a2 << 3)
a5 =  (in << 0) + (in << 2)
a6 =  (in << 0) - (a2 << 3)
a7 =  (in << 0) - (in << 5)
a8 =  (in << 0) + (a3 << 3)
a9 =  (a8 << 0) - (in << 10)

d0 =     - (a5 << 0)
t1 =  d0 + (a6 << 0)
d1 =  t1
t2 =  d1 + (a7 << 1)
d2 =  t2
t3 =  d2 - (a3 << 2)
d3 =  t3
t4 =  d3 + (a2 << 4)
d4 =  t4
t5 =  d4 + (a3 << 3)
d5 =  t5
t6 =  d5 - (a9 << 0)
d6 =  t6
t7 =  d6 - (a9 << 0)
d7 =  t7
t8 =  d7 + (a3 << 3)
d8 =  t8
t9 =  d8 + (a2 << 4)
d9 =  t9
t10 =  d9 - (a3 << 2)
d10 =  t10
t11 =  d10 + (a7 << 1)
d11 =  t11
t12 =  d11 + (a6 << 0)
d12 =  t12
t13 =  d12 - (a5 << 0)
o13 =  t13
```

Figure 5.9. Netlist of the FIR filter.

Finally, the program VHD is used to generate the vhd netlist of the designed filter. Here, there is no word length optimization within half-band and FIR filters. Only the output word lengths of the filters are truncated in order to reach 10 bit resolution.

6. RESULTS

VHDL is a high level description language for system and circuit design. The language supports several levels of abstraction [31]. CIC filter, manually generated half-band and FIR filter described in 4.4 and 4.5, and filters that are generated using GAM algorithm described in 5.1 and 5.2 are written in VHDL. The filter blocks are then synthesized with Leonardo Spectrum. Leonardo Spectrum synthesizes all levels of abstraction, and minimizes the amount of logic required, resulting in a final netlist description in chosen technology [31]. The schematics of the filter blocks in Mentor Graphics are obtained from importing the netlists which are generated with Leonardo Spectrum. The circuits are designed using standard cell library of 0.35μm CMOS process.

The designed filter blocks are simulated in order to measure the power consumption. The simulations are performed for different supply voltages, VDD, as 1.2V, 2.5V and 3.3V.

Table 6.1 gives measured average currents of the designed filters. HB and FIR denote half-band filter and FIR filters designed manually, HB_GAM and FIR_GAM denote the designed half-band filter and FIR filter using GAM algorithm.

Table 6.1. Measured average currents of designed filters.

	3.3V	**2.5V**	**1.2V**
CIC	74.913 μA	51.758 μA	20.545 μA
HB	266.01 μA	175.61 μA	65.29 μA
HB_GAM	197.81 μA	123.88 μA	48.02 μA
FIR	322.3 μA	214.15 μA	84.231 μA
FIR_GAM	252.33 μA	168.7 μA	64.258 μA

According to measured current values Table 6.2 shows power consumption of filters for different supply voltages. Here the last two rows give the total power dissipation of the decimation filter. Again, DEC represents decimation filter designed manually and DEC_GAM represents decimation filter that is designed using GAM algorithm.

Table 6.2. Power consumption of designed filters.

	3.3V	**2.5V**	**1.2V**
CIC	247.2 µW	129.3 µW	24.6 µW
HB	877.8 µW	439 µW	78.3 µW
HB_GAM	650 µW	309 µW	57.6 µW
FIR	1063.59 µW	535.3 µW	101.07 µW
FIR_GAM	832.68 µW	421 µW	77.1 µW
DEC	2188 µW	1103 µW	203.97 µW
DEC_GAM	1729 µW	859 µW	159.2 µW

It can be seen from Table 6.2, the power consumptions for half-band and FIR filters are less than manual designs when the GAM algorithm is used.

Table 6.3. Components of designed filters.

	# FA	**# DFF**
CIC	84	116
HB	374	378
HB_OP	345	327
HB_GAM	291	335
FIR	886	520
FIR_OP	525	299
FIR_GAM	397	305
DEC	1344	1014
DEC_OP	954	742
DEC_GAM	772	756

Also, it is obvious that if the supply voltage is decreased the power consumption will reduce. Therefore, keeping the supply voltage as low as possible while taking the circuit speed into consideration provides efficient designs in terms of power consumption.

Table 6.3 gives the number of full adders (FA) and D flip flops (DFF) used for each filter and whole decimation filter. Here, HB, FIR and DEC represent non optimized half-band, FIR and total decimation filters; HB_OP, FIR_OP, DEC_OPGAM represent optimized optimized half-band, FIR and total decimation filters and HB_GAM, FIR_GAM and DEC_GAM represent filters designed using GAM algorithm. GAM algorithm provides less hardware complexity than manual design.

Table 6.4 gives comparison of designed decimation filters and different decimation filters used for low power sigma-delta ADC structures in the literature in terms of power consumption. Here, DEC represents manual design of the decimation filter and DEC_GAM represents the designed decimation filter using GAM algorithm.

In Table 6.4 technologies, supply voltages, output word lengths (resolution of converters), sampling frequencies and measured power values are represented for different studies and proposed studies. In order to compare the different designs clearly, power figure of merit is generated as shown in equation (6.1).

$$FM = \frac{Power}{Output\ Wordlength \times Sampling\ Frequency} \tag{6.1}$$

Table 6.4. Comparison of different studies.

	Technology	Supply Voltage (V)	Resolution (bits)	Sampling Frequency (KHz)	Power (μW)	FM (pJ/bits)
[2]	0.35 μm CMOS	3.3	8	8	0.310	4.85
[9]	-	1	12	19.2	20	86.8
[32]	0.18 μm CMOS	1.8	14	5644.8	9000	138
[33]	1 μm CMOS	3	16	11300	6500	35.9
[34]	0.18 μm CMOS	1.8	16	3072	100	2.034
[35]	0.7 μm CMOS	3.3	24	6144	20000	135.63
[36]	0.6 μm CMOS	3.3	16	25000	155000	387.5
DEC	0.35 μm CMOS	1.2	10	2500	204	8
DEC	0.35 μm CMOS	2.5	10	2500	1103	44
DEC	0.35 μm CMOS	3.3	10	2500	2188	88
DEC_GAM	0.35 μm CMOS	1.2	10	2500	159	6.3
DEC_GAM	0.35 μm CMOS	2.5	10	2500	859	34
DEC_GAM	0.35 μm CMOS	3.3	10	2500	1729	69

It can be seen from Table 6.4, power performance of designed decimation filters lies in the middle of the reference decimators.

7. CONCLUSION & FUTURE WORK

A low power decimation filter for sigma-delta ADC is designed and implemented in 0.35μm CMOS technology. Multistage decimation filter architecture is preferred. The architecture is composed of one CIC filter, one FIR half-band filter and one FIR filter.

A third order CIC filter is implemented as a first stage. CIC filter consists of three stage integrator and three stage differentiator blocks. It also includes a coder circuit and a clock divider. CIC filter requires no multipliers; it consists of only adders, subtractors and registers. Thus, this filter is a good choice for low power implementations.

Filter specifications according to required decimation filter characteristics are generated with MATLAB. The required filter order and filter coefficients are obtained using MATLAB. Filter coefficients are quantized to proper values. These coefficients are converted to CSD number representation format. Multiplierless FIR filter architecture is used because filter coefficients are constant. According to multiplier less FIR filter design, multiplication is performed using addition/subtraction and shift operations. Shift operation is done by hardwiring. Thus, multiplication is realized using only adders and subtractors. Filters are designed with HSSE method which is an optimization method of CSE method and, parallel addition is performed.

In addition, half-band and FIR filters are designed using GAM algorithm. The algorithm optimizes SPT terms in the coefficients given the filter frequency response characteristics. CSE method is applied to obtained coefficients and filters are generated.

All filters are designed with VHDL and synthesized in 0.35μm CMOS technology. Power consumption is measured for different supply voltages. Designed decimation filters are compared with the filters in the literature.

As future work, the designed decimation filter can be combined with second order sigma-delta modulator and overall ADC structure performance can be measured.

REFERENCES

1. Norsworthy S.R., R. Schreier, G.C. Temes, *Delta-Sigma Data Converters Theory, Design , and Simulation*, IEEE Press, New York, 1998.

2. Gerosa A., A. Neviani, " A Low Power Decimation Filter For A Sigma-Delta Converter Based On A Power-Optimized Sinc Filter", *Proceedings of the 2004 International Symposium on Circuits and Systems*, Vol. 2, 23-26 pp. II – 245-8, 2004.

3. Gürsoy Ö., O. Sağlamdemir, M. Aktan, S. Talay, G. Dündar, "Low Power Decimation Filter Architectures for Sigma-Delta ADC's," *Proceedings of ELECO'05*, pp. 72-75, 7-11, 2005.

4. Jarman D., *A Brief Introduction to Sigma-Delta Conversion*, 1995, http://www.intersil.com/data/an/an9504.pdf

5. Hoeschele D.F., *Analog-to-Digital and Digital-to-Analog Conversion Techniques*, John Wiley & Sons, 1994.

6. Hogenauer E.B., "An Economical Class of Digital Filters for Decimation and Interpolation", *IEEE Transactions on Acoustics, Speech and Signal Processing*, Vol. ASSP-29, No.2, pp. 155-162, 1981.

7. Magendran P.K., S.Sumathi, "Area Efficient High Speed FPGA Implementation of DUC for Ultrasonic NDE Signal Processing Techniques", *International Journal of Computer Communication and Information System,* Vol. 2, No.1, ISSN:0976-1349, 2010.

8. Donadio M.P., *CIC Filter Introduction,* 2000, http://www.mikrocontroller.net/attachment/51932/cic2.pdf

9. Chong K.F., P. K. Gopalakrishnan, T. H. Teo, "Low Power Approach for Decimation Filter Hardware Realization," *Proceedings of World Academy of Science, Engineering and Technology,* Vol. 32, pp. 2070-3740, 2008.

10. Candy J.C., "Decimation for Sigma-Delta Modulation", *IEEE Transactions on Communications,* Vol.COM-34, pp. 72-76, 1986.

11. Arora M., *Clock Dividers Made Easy*, 2002, http://read.pudn.com/downloads95/sourcecode/others/381992/Clock_Dividers_Made_Easy.pdf

12. Zimmermann R., *Computer Arithmetic: Principles, Architectures and VLSI Design*, 1999, http://www.iis.ee.ethz.ch/~zimmi/publications/comp_arith_notes_pdf

13. Chua C.C., B. Gwee, J. S. Chang, "A Low Voltage Micro Power Asynchronous Multiplier for a Multiplierless FIR Filter", *Proc. of ISCAS 2003*, pp.V381-384, 2003.

14. Bhattacharya M., J. Astola, " Multiplierless Implementation of Recursive Digital Filters Based on Coefficient Translation Methods in Low Sensitivity Structures", *The IEE International Symposium on Circuits and Systems (ISCAS) 2001*, pp. II697-II700, 2001.

15. Park J., W. Jeong, H. Choo, "High Performance and Low Power FIR Filter Design Based on Sharing Multiplication", *International Symposium on Low Power Electronics and Design (ISLPED) 2002*, 11.3, 2002.

16. Kang H.J., H.Kim, I.C. Park "FIR Filter Synthesis Algorithms for Minimizing the Delay and the Number of Adders", *International Conference on Computer Aided Design (ICCAD*), pp51-54, 2000.

17. Park I.C., H.J. Kang, " Digital Filter Synthesis Based on Minimal Signed Digit Representation", *The Desing Automation Conference (DAC)*, pp468-473, 2001.

18. Aktan M., A. Yurdakul, G. Dündar, "An Algorithm for The Design of Low-Power Hardware Efficient FIR Filters," *IEEE Transactions on Circuits and Systems – I*, Vol. 55, No. 6, pp. 1536-1545, 2008.

19. Aktan M., G.Dündar, M.Koca, "Low-Power Hardware Efficient MMSE Equalizer Design", *IEEE Transactions on Circuits and Systems*, pp.307-311,2008

20. Aktan M., G. Dündar, "Design of Digital Filters for Low Power Applications Using Integer Quadratic Programming", *Proceedings of PATMOS'05*, pp.137-145, 2005.

21. Hewlitt R.M., E. S. Swartzlander, "Canonical Signed Digit Representation for FIR Digital Filters", *IEEE Workshop on Signal Processing Systems*, pp.416-426, 2000.

22. Richard I. H., "Subexpression Sharing in Filters Using Canonic Signed Digit Multipliers", *IEEE Transactions on Circuits and Systems- II: Analog and Digital Signal Processing*, Vol. 43, No. 10, 1996.

23. Yao C.Y., H.H. Chen, T.F. Lin, C.J. Chien, C.T. Hsu, "A Novel Common Subexpression Elimination Method for Synthesizing Fixed-point FIR Filters," *IEEE Trans. Circuits Syst. I,* Vol 1, No. 11, pp. 2215-2221, 2004.

24. Mahesh R., A. P. Vinod, "A New Common Subexpression Elimination Algorithm for Implementing Low Complexity FIR Filters in Software Defined Radio Receivers," *in Proceedings of IEEE International Symposium on Circuits and Systems,* Vol. 4, pp. 4515-4518, 2006.

25. Vinod A.P., E.M-K Lai, "Comparison of The Horizontal and The Vertical Common Subexpression Elimination Methods for Realizing Digital Filters", *Proceedings of IEEE International Symposium on Circuits and Systems,* Vol. 1, pp.496-499, 2005.

26. Xu F., C.H. Chang, C.C. Jong, "Contention Resolution Algorithm for Common Subexpression Elimination in Digital Filter Design," *IEEE Trans. Circuits Syst. II,* Vol. 52, No. 10, pp. 695-700, 2005.

27. Vinod A.P., M.K. Lai, "On The Implementation of Efficient Channel Filters for Wideband Receivers by Optimizing Common Subexpression Elimination Methods," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Syst.,* Vol.24, No. 2, pp. 295-304, 2005.

28. Vaidyanathan P.P., T. Q. Nguyen, "A "TRICK" for The Design of FIR Half-band Filters," *IEEE Trans. Circuits Syst.*, Vol. CAS-34, pp. 297–300, 1987.

29. Yates R., *Practical Considerations in Fixed-point FIR Filter Implementations*, 2010, http://www.digitalsignallabs.com/fir.pdf

30. Aktan M., *High Level Power Efficient Synthesis of FIR Based Digital Systems,* Ph.D. Thesis, Boğaziçi University, 2008

31. *Leonardo Spectrum for Altera HDL Synthesis Manual*, 2001, http://www.eng.auburn.edu/~agrawvd/COURSE/E6200_Fall08/PROJECT/VHDLSynthesisGuide.pdf

32. Hao L., H. Yan, R.C.C. Cheung, H. Xiaoxia, M. Shaoyu, Y. Peng, Z. Dazhong, " A High Performance Low Power ΣΔ ADC for Digital Audio Applications", *Journal of Semiconductors*, Vol. 31, No. 5, 2010.

33. Brandt B.P., B.A. Wooley, " A Low Power, Area Efficient, Digital Filter for Decimation and Interpolation", *IEEE Journal of Solid State Circuits*, Vol. 29, No.6, 1994.

34. Parameswaran S., N. Krishnapura, "A 100 μW Decimator for a 16 bit 24 KHz Bandwidth Audio ΣΔ Modulator", *International Symposium on Circuits and Systems,* 2010.

35. Fujimori I., K. Koyama, D. Trager, F. Tam, L. Longo, "A 5V Single Chip Delta Sigma Audio A/D Converter with 111 dB Dynamic Range", *IEEE Journal of Solid State Circuits*, Vol. 32, No.3, 1997.

36. Maulik P.C., M.S. Chadha, W.L. Lee, P.J. Crawley, "A 16-Bit 250 KHz Delta-Sigma Modulator and Decimation Filter", *IEEE Journal of Solid-State Circuits*, Vol.35, No.4, 2010.