Decimator, Interpolator and DEM Techniques for Oversampling $\Delta\Sigma$ Data Converters

A Project Report

submitted by

NAGA RAJESH DOPPALAPUDI

in partial fulfilment of the requirements
for the award of the degree of

MASTER OF TECHNOLOGY
Microelectronics & VLSI Design

DEPARTMENT OF ELECTRICAL ENGINEERING
INDIAN INSTITUTE OF TECHNOLOGY MADRAS

MAY 2006

CERTIFICATE

This is to certify that the report titled **Decimator, Interpolator and DEM Techniques for Oversampling $\Delta\Sigma$ Data Converters**, submitted by **Naga Rajesh Doppalapudi** , to the Indian Institute of Technology Madras, for the award of the degree of **Master of Technology**, is a bonafide record of the work done by him under our supervision. The contents of this report, in full or in parts, have not been submitted to any other Institute or University for the award of any degree or diploma.

Dr. Y.Shanthi Pavan	**Sudhir Polarouthu**
Internal Advisor,	External Advisor,
Assistant Professor,	Analog Design Engineer,
Dept. of Electrical Engineering,	Nulife Semiconductor India(P) Ltd,
IIT-Madras, 600036	Chennai, 600018

Place: Chennai

Date: May 2006

ACKNOWLEDGEMENTS

I am grateful to my advisors Dr.Y.Shanthi Pavan and Sudhir Polarouthu for their valuable guidance through out the entire course of this project work.
I wish to thank various faculty members from whose courses I have benefitted as a student here.
I would like to thank Dr.Narendra Rao, Dr.Mark Forbes and Jitendra Kulkarni of Nulife Semiconductor India(P) Ltd, Chennai.
I would also like to thank the employees of Nulife Semicondutor India(P) Ltd and extend my special thanks to Prakash Narayanan, Lakshmanan Balasubramanian, Srikanth Manian, Supraja Krishnan, Chetan Shah, Ravi Vijayaraghavan and Kalyan Chakravarthi Chekuri for their help and discussions.
I express my thanks to my friends and classmates for helping me get through the difficult times and for all the emotional support, entertainment and caring they provided.
The people to whom I owe completely for what I am today are my parents and family. Without their love, affection and support its impossible to conceive of myself.

ABSTRACT

Data converters are key components in modern Digital Signal Processing(DSP) systems. Conventional Nyquist converters require analog components that are precise and highly immune to noise and interference. In contrast, oversampling converters can be implemented using simple and high-tolerance analog components along with cheap and robust digital filtering circuits. Moreover, sampling at high frequency eliminates the need for abrupt cutoffs in the analog filters but it require sharp cutoff digital filtering and sample rate conversion. A technique of noise shaping is used in $\Delta\Sigma$ converters in addition to oversampling to achieve a high-resolution conversion.

A decimator is used to convert the signal from oversampling rate to Nyquist rate. Similarly, an interpolator is used to convert the signal from Nyquist rate to oversampling rate. As these sampling rate converter blocks consume significant amount of power and die area, there is a need for improving their performance for applications in portable battery operated systems.

This work estimates and compares the performance of different low power implementations of $sinc^4$ decimation filters. Also design of a high performance and area efficient interpolator with an interpolation factor of 64, for use in low power oversampling $\Delta\Sigma$ digital-to-analog converters is presented. Finally, performance and hardware complexity of novel class Dynamic Element Matching(DEM) algorithms are compared. The dynamic element matching algorithms are used to improve the linearity of the oversampling dataconverter.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

1.1 Motivation

Oversampling $\Delta\Sigma$ data converters [18] are widely used in high precision, low bandwidth applications. Oversampling means use of a sampling rate that is much higher than the bandwidth of the signal of interest. Digital Signal Processing(DSP) techniques are used for sample rate conversion from oversampling rate to Nyquist rate.

Oversampling of a signal greatly relaxes the anti-aliasing analog filter requirements in analog-to-digital conversion and the reconstruction filter requirements in digital-to-analog conversion [11] [7]. The $\Delta\Sigma$ modulator with its noise shaping topology achieves high resolution. The oversampling $\Delta\Sigma$ data conversion is relatively less sensitive to analog component variation and so, well suited for CMOS implementation which has relatively large component variations and matching tolerances.

So, the oversampling techniques overcome the problem of analog accuracy with traditional converters by trading the digital complexity and speed for the desired insensitivity to analog non-idealities [11]. As the state-of-art deep submicron technologies offer fast and dense circuit realizations, it is a good trade-off and allows to integrate with advanced digital CMOS process for low cost System-On-Chip(SOC) solutions.

The $\Delta\Sigma$ converter achieves high performance by shifting the signal processing(filtering) complexity from analog domian to digital domain. Since most of signal processing operations are done in digital domain, the digital hardware occupies significant amount of area and dissipate power. *Typically, the oversampling* *converter performance is determined by the analog components and its power consumption and die area are governed by the digital filtering circuitry.*

As a trend towards using oversampling converters in portable battery-operated equipment and low-cost consumer market products, the power and die area should be low. This work presents an estimate of power and area for a special class of filters known as *sinc* filters for use in decimator. Three popular implementations are compared for different oversampling ratios and input bit-widths. The work also includes the design of high performance and area efficient interpolation filter for use in a low power oversampling $\Delta\Sigma$ digital-to-analog converter. Finally, a novel class Dynamic Element Matching algorithms for improving the linearity of the oversampling $\Delta\Sigma$ data converter [19] [20] are discussed and their hardware requirements are estimated.

1.2 Organization of the report

Chapter 2 explains the theory of decimation and the decimtion filter design concepts. Performance of *sinc* filters is discussed.

Chapter 3 deals with the *sinc* filter implementations. Three most popular implementations are presented.

Chapter 4 explains interpolation concepts and design aspects.

Chapter 5 presents the design of high performance and area efficient interpolator.

Chapter 6 discusses performance of dynamic element matching algorithms and their hardware requirements.

CHAPTER 2

DECIMATION FILTER

Decimation Filter is a key multirate signal processing component in oversampling $\Delta\Sigma$ ADC [11] [18]. There have been continuous efforts to improve the performance of the digital decimation filters in terms of power dissipation, hardware and speed [15] [22] [12] [6]. This chapter deals with theory of the decimation filter and its design. Implementation aspects of the decimator are discussed in next chapter.

2.1 Role of decimation filter in $\Delta\Sigma$ ADC

The system architecture of an oversampling $\Delta\Sigma$ ADC [11] [18] is shown in Figure 2.1.

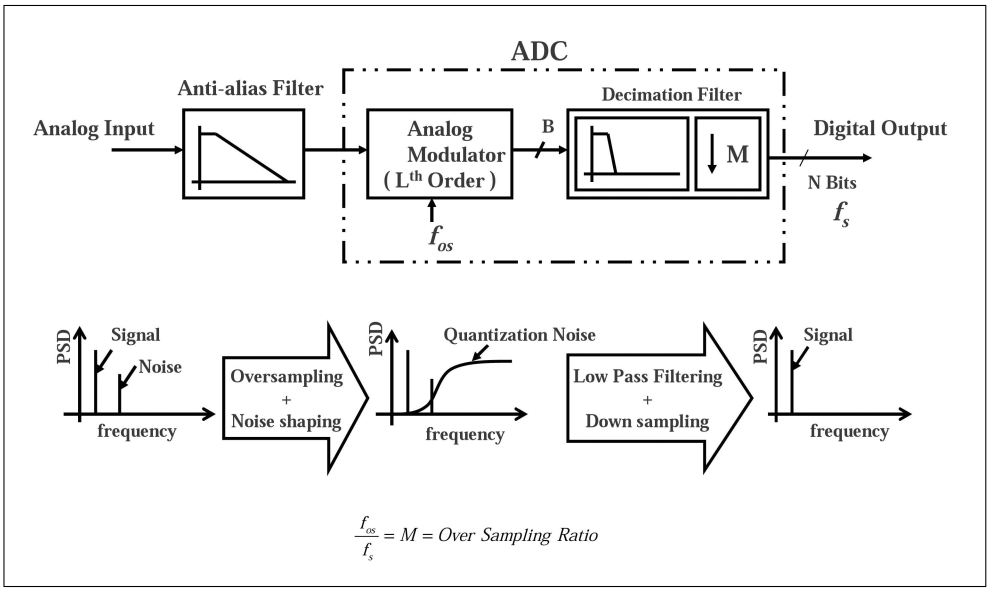

ADC
Anti-alias Filter
Analog Input
Analog Modulator (Lth Order)
B
Decimation Filter
M
Digital Output
N Bits
f_s
f_{os}
PSD
Signal
Noise
frequency
Oversampling + Noise shaping
Quantization Noise
Low Pass Filtering + Down sampling
$\frac{f_{os}}{f_s}$ = M = Over Sampling Ratio

Figure 2.1: Oversampling $\Delta\Sigma$ A/D Conversion

The first stage is a continuous-time anti-aliasing filter and is required to band-limit the input signal to frequencies less than one-half the oversampling frequency, f_{os}. When the oversampling ratio is large, simple anti-aliasing filter can be used.

The output of anti-aliasing filter is processed by analog $\Delta\Sigma$ modulator. With a noise shaping feedback loop, the modulator pushes the quantization noise to out-of-band high frequencies and generates a low-resolution digital signal. *The noise shaped low-resolution digital signal is equivalent to coarse quantization of analog input at oversampling rate.*

The final stage of the converter is a decimator, which converts the oversampled low-resolution digital signal into a high-resolution digital signal at a lower sampling rate ideally equal to twice the frequency of the desired bandwidth of the input signal. High-resolution digital signal is obtained by *averaging* the modulator output samples to interpolate *between* the coarse *quantization levels* of the modulator. From frequency domain point of view, decimation filter has to suppress the out-of-band quantization noise before it downsamples the signal to Nyquist rate. These aspects are discussed in detail in next section.
From the system level perspective, the decimation filter should

- *Suppress the out-of-band quantization noise*
- *Re-sample the signal at lower sampling rate*

2.2 Decimation basics

2.2.1 Decimation theory

Decimation is basically a process of *sampling of discrete time signal* [7] [1]. Mathematically it can be expressed as

$$y[n] = x[nM]$$

A time domain decimation example is shown in Figure 2.2.

Similar to an anti-aliasing filter in sampling of continuous time signal, a Low Pass Filter (LPF) is required to bandlimit discrete time input signal for decimation. The decimation filter can be conceptually thought of as a digital LPF followed by a downsampler, as shown in Figure 2.3. Ideal cutoff frequency of the

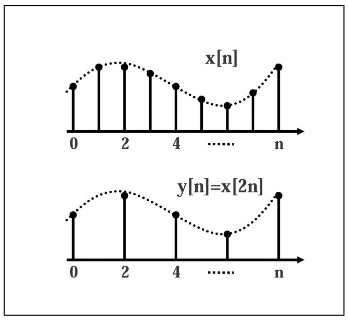


Figure 2.2: Decimation (M=2)

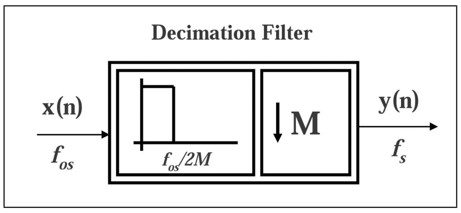


Figure 2.3: Decimation filter

LPF should be equal to one-half of the output sampling rate.

$$\text{Cutoff frequency of the LPF } (f_c) = \frac{f_s}{2}$$

In terms of oversampling frequency (f_{os})

$$f_s = \frac{f_{os}}{M} \Longrightarrow f_c = \frac{f_{os}}{2M}$$

2.2.2 Decimation filter requirements

Sampling in time domain results in frequency translation of input signal spectrum to multiples of sampling frequency. So sampling of a discrete time wideband signal like output of a $\Delta\Sigma$ modulator results in frequency translation as shown in Figure 2.4. Because of this frequency translation, the spectral components at $kf_s \pm f_c$ are aliased to baseband in sampling rate reduction process as shown in Figure 2.5. (where k=0,1,...$\frac{M}{2}$)

"*Hence, the digital LPF used in decimator must attenuate these aliasing components around kf_s to the level much lower than the noise floor in baseband.*"

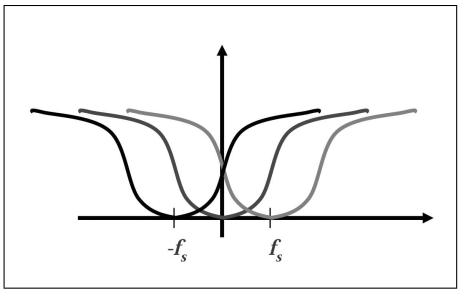


Figure 2.4: Frequency Translation of wideband signal

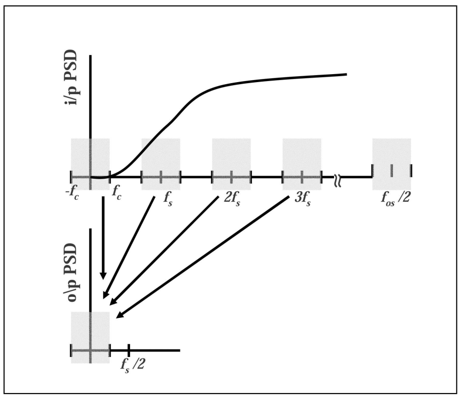


Figure 2.5: Aliasing to baseband

2.3 Decimation filter design

2.3.1 Single stage design

With the knowledge of decimation filter requirements, decimation filter specifications can be derived for a given application. A simple, straight forward approach is a single stage design. In this a digital LPF is designed to meet the specifications and it is followed by a downsampler. Well known classic digital filter design techniques can be used to design the LPF [1] [7].

As an example, design specifications of a digital LPF for CD quality audio application based on single stage approach are shown in Figure 2.6.

From classical filter design theory, the order of a digital LPF is directly related to a function of the required ripples δ_p and δ_s in the passband and stopband respectively, and inversely related to the normalized width of the transition band $(\Delta f/f_{os})$ [18] [1].

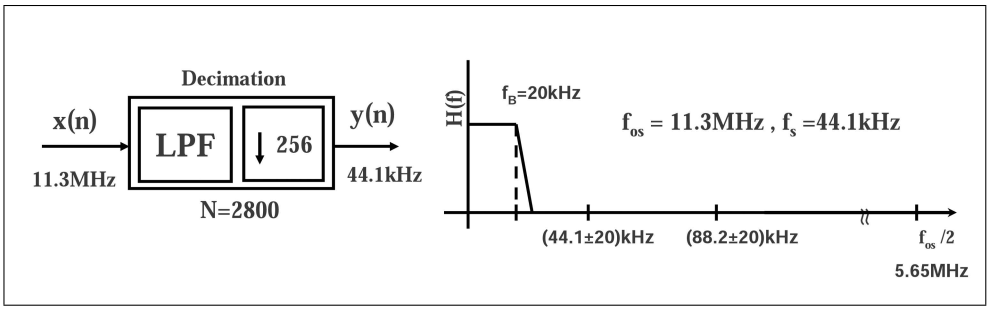


Figure 2.6: Single stage design example

$$\text{Filter Order(N)} = \frac{D(\delta_p, \delta_s)}{\Delta f / f_{os}}$$

A digital FIR filter that meets specifications of the above example would require more than $\frac{f_{os}}{\Delta f} = \frac{11.3MHz}{4.1kHz} \simeq 2800$ coefficients.
Output of LPF is downsampled which means one out of every 256 samples is taken as output. Entire digital LPF operates at oversampling rate. The disadvantages of single stage design approach are

- *Excessively large filter orders* because of narrow transition bandwidth.
- *High power dissipation* as the entire filter operates at f_{os}
- *Inefficient use of hardware* is because one out of every M computed samples is passed to output.

2.3.2 Multi-rate multi-stage design

The disadvantages of single stage design can be overcome by multistage design in which the decimator is designed as a cascade of two or more stages such that the overall decimation ratio is product of the decimation ratio of each of the stages. Advantages of multistage design can be better explained by considering the same example. Let us assume that a decimator of M=256 is designed as cascade of $M_1 = 64$ and $M_2 = 4$ as shown in Figure 2.7. As we can see that considerably lower order filters are required in each stage because of relatively wider transition bands. Increase in relative transition bandwidth ($\frac{\Delta F}{F}$) of the first and last stage filters is due to the increase in ΔF and decrease in F respectively. *In general,*

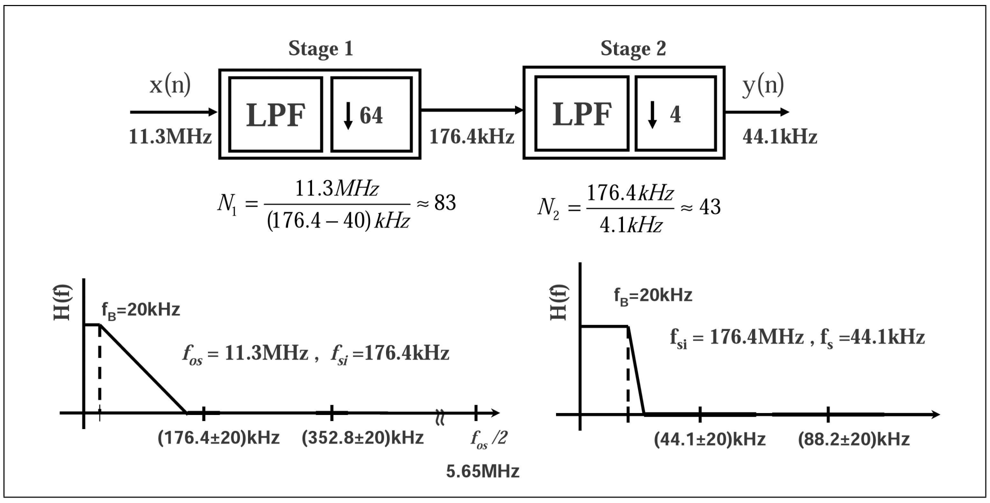

Stage 1
Stage 2
x(n)
y(n)
LPF
↓64
LPF
↓4
11.3MHz
176.4kHz
44.1kHz
$N_1 = \frac{11.3MHz}{(176.4-40)kHz} \approx 83$
$N_2 = \frac{176.4kHz}{4.1kHz} \approx 43$
H(f)
f_B=20kHz
f_{os} = 11.3MHz , f_{si} =176.4kHz
(176.4±20)kHz
(352.8±20)kHz
$f_{os}/2$
5.65MHz
H(f)
f_B=20kHz
f_{si} = 176.4MHz , f_s =44.1kHz
(44.1±20)kHz
(88.2±20)kHz

Figure 2.7: Two Stage Design Example

several stages can be used to reduce computational complexity and power consumption. $M = M_1.M_2....M_P$

With emphasis on optimal designs in terms of minimizing the number of multiplications per second or the required amount of storage, FIR multistage approach can be used [17]. The no. of multiplications can be further reduced by careful choice of filter coefficients for half-band decimators [9]. An economical hardware implementation can be obtained by Cascaded Integrator-Comb (CIC) which require no multipliers and use limited storage [8]. It will be shown in the following sections that CIC is one of the implementation styles of *sinc* filter [3] [7] and it is widely used in first stage of decimation in $\Sigma\Delta$ ADC.

With the trend towards integrating decimation filter with analog $\Delta\Sigma$ Modulator for constructing high resolution A/D converter [18], *sinc* filter in first stage of decimation is the favourite choice for VLSI implementation. Almost all the published decimation designs for $\Delta\Sigma$ ADC use *sinc* filter in their first stage[22] [15].

2.3.3 Multi-stage decimator design example

A published four stage decimation filter [15] with M=256 following a 2^{nd} order $\Delta\Sigma$ modulator and designed for CD quality audio application is shown in Figure 2.8 and the frequency response of each stage filter is shown in Figure 2.9. The third order *sinc* filter reduces the sampling rate by a factor of 64. The remaining sampling rate reduction to the Nyquist output rate of 44.1kHz is accomplished efficiently with two cascaded halfband filters. A low-order droop-correction filter, operating at output rate is used as fourth stage to compensate for the droop in the baseband response of the *sinc* filter.

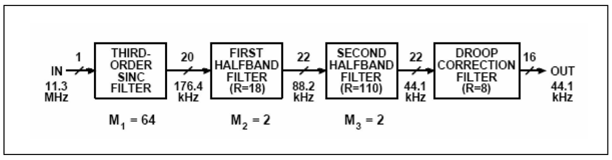


Figure 2.8: Multi-stage decimator Example

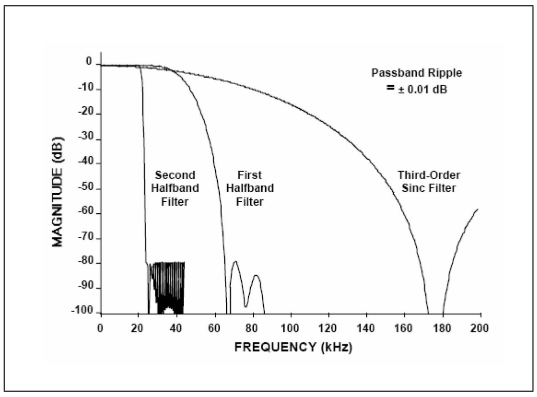


Figure 2.9: Each stage Frequency Response

As we can see in Figure 2.9, transition bandwidth of third stage is less than that of second stage. This is the reason for increase in filter order in last stages. "*By using mutiple stages, simple filters can be used at high sampling rates to deal with aliasing, while higher performance filters are used at the low sampling rates in order to achieve the required overall filter performance. This multistage approach results in a tremendous savings in computation and power.*"

2.4 Sinc Filter

2.4.1 Economical first stage filter

The simplest low-pass FIR filter with no multiplier is a filter with all its coefficients equal to unity, which is a Moving Average (MA) filter [7] [1]. Output of MA filter is equal to the average of M samples as shown in Figure 2.10 and its impulse response is shown in Figure 2.11.

$$y[n] = \frac{1}{M}\left(x[n] + x[n-1] + ... + x[n-M+1]\right)$$

Transfer function is

$$H(z) = \frac{1}{M}\left(1 + z^{-1} + z^{-2} + ... + z^{-(M-1)}\right) = \frac{1}{M}\sum_{i=0}^{M-1} z^{-i} = \frac{1}{M}\left(\frac{1-z^{-M}}{1-z^{-1}}\right)$$

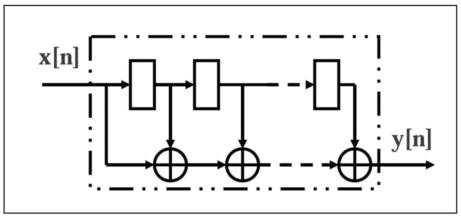


Figure 2.10: MA filter(Direct form structure)

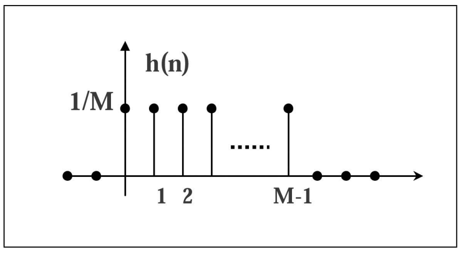


Figure 2.11: Impulse response of MA filter

Frequency Response of MA filter is given by

$$H(e^{j\omega}) = \frac{1}{M}\left(\frac{1-e^{-jM\omega}}{1-e^{-j\omega}}\right) = \frac{1}{M}\left(\frac{sin(\frac{M\omega}{2})}{sin(\frac{\omega}{2})}\right)e^{-j\omega(\frac{M-1}{2})} \text{ ,where } \omega = \frac{2\pi f}{f_{os}}$$

Magnitude Response is shown in Figure 2.12

$$|H(e^{j\omega})| = \frac{1}{M}\left(\frac{sin(\frac{M\omega}{2})}{sin(\frac{\omega}{2})}\right)$$

$$|H(e^{j\omega})| = 0, \text{ when } \frac{M\omega}{2} = \pm k\pi$$

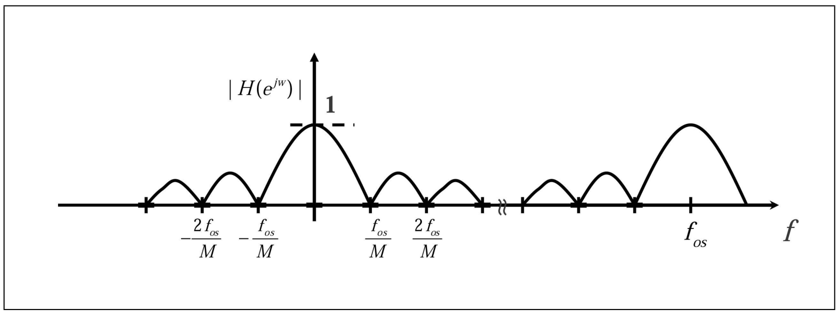


Figure 2.12: Magnitude response of MA filter

As it's magnitude response is *sinc* function, the MA filter is called a *sinc* filter. The magnitude response has nulls at $\frac{kf_{os}}{M} = kf_s$. As explained in section 2.2.2 for decimating the signal from f_{os} to f_s, the regions around kf_s should be attenuated to avoid aliasing to baseband. With the nulls at kf_s positions, SINC filter can attenuate the spectral components around kf_s.

The worst case aliasing attenuation is just before the first null and precisely at $f = f_s - f_c$. If $f_s = 2f_c$ (transition bandwidth is zero) which means *sinc* filter is used to directly decimate the signal to Nyquist rate, aliasing attenuation will be poor resulting in more aliasing noise and also excessive passband droop.[1] If $f_c << f_s$ (signal bandwidth is a small fraction of output rate), the worst case attenuation will be higher and passband droop will reduces. This is the reason for using *sinc* filter only in the first stage of decimator but not in the last stages.

2.4.2 Higher order *sinc* filter

An important consideration in the design *sinc* filter for decimation is that the width of the notches in the filter be wide enough to attenuate the aliasing regions$kf_s \pm f_c$. The attenuation and/or width of null regions can be improved by increasing

[1]Undesirable attenuation of higher passband frequencies is called droop.

the order of zeros at null positions. Increasing the order of zeros (k times) is equivalent to

$$H(z) = \left(\frac{1}{M}\left(\frac{1-z^{-M}}{1-z^{-1}}\right)\right)^{k} \Rightarrow |H(e^{j\omega})| = \left(\frac{1}{M}\left(\frac{sin(\frac{M\omega}{2})}{sin(\frac{\omega}{2})}\right)\right)^{k}$$

Magnitude responses of higher order *sinc* filters (k=1,2,3) with M=4 are shown in Figure 2.13. Worst case aliasing rejection increases with increase in $SINC$.

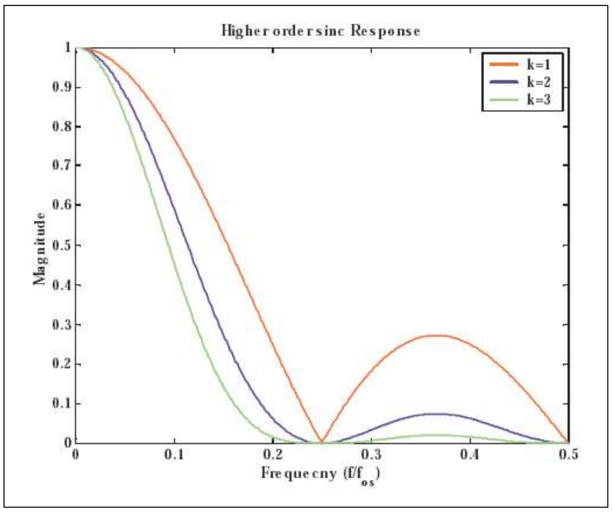


Figure 2.13: Magnitude response of $sinc^k$ with M=4

The main advantage of this higher order *sinc* filter is higher aliasing band rejection, but on otherhand it attenuates the high frequency components within the passband. This droop in passband can be large and intolerable, in which case a compensating filter with inverse *sinc* magnitude response in its passband is used at the final stage of decimator to obtain overall flat passband response as discussed in example of section 2.3.3. Advantage of using droop correction stage operating at the output rate is reduced power consumption. Passband droop and worst case aliasing rejection for $sinc^k$ filters are tabulated in table 2.1 and their magnitude response(dB) plots are shown in Figure 2.14. Output sampling rate is assumed to be twice the Nyquist rate for this example.

Table 2.1: $sinc^k$(M=4) passband droop & rejection

Order(k)	Droop at $f_c = 0.0625f_{os}$ (dB)	Rejection at$f_{st} = 0.1875f_{os}$ (dB)
1	-0.86	-9.95
2	-1.71	-19.89
3	-2.57	-29.84
4	-3.42	-38.78

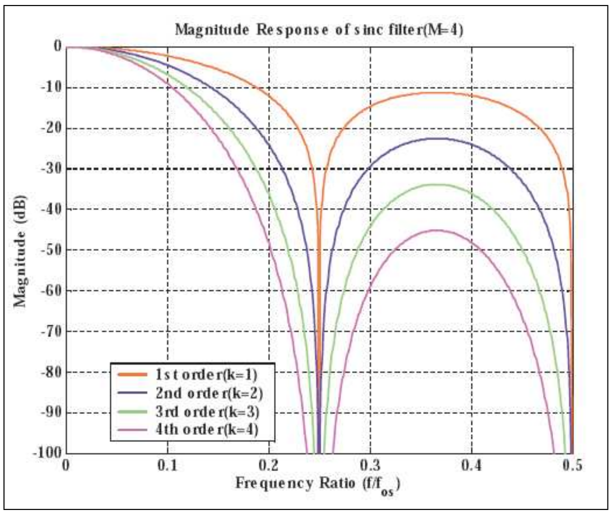


Figure 2.14: Magnitude response(dB) of $sinc^k$ with M=4

2.5 *Sinc* Decimator Design

From hardware implementation point of view, *sinc* filter is the simplest filter. But as explained in section 2.4.1, the performance of the *sinc* filter is better when the passband is a small fraction of the output sampling rate. The *sinc* filters cannot be used to decimate to Nyquist rate because of excessive droop and poor aliasing attenuation [3].

For a given overall decimation ratio(M), the first step in multistage approach is to find the number of stages and decimation factor for each stage. Suppose number of stages are chosen to be two and *sinc* filter is used in first stage, the next step would to find the decimation factor for each stage.

As stated earlier, performance of a *sinc* filter is better when the passband is small a fraction of output sampling rate. In otherwords, the performance of *sinc* filter depends on ratio of output sampling rate to passband width which is nothing but the second stage decimation ratio(M_2). The performance of *sinc* filter for different second stage decimation ratios is discussed in detail in the following two subsections. Based on the overall performance required second stage decimation factorM_2 is chosen. The first stage decimation factor (M_1) can be obtained from M and M_2 from the relation $M = M_1M_2$

Second stage filter can be either single stage or cascade of several stages again. These last stage filters can be either FIR or IIR, choice is entirely application specific.

In general, because of their hardware efficiency, sinc filters are used to decimate oversampled signal as much possible.

2.5.1 Passband performance

A conceptual two stage decimator for design is shown in Figure 2.15 and *sinc*^{k} filter passband droop is plotted in figure 2.16 for different values of intermediate Over Sampling Ratio(M_2), SINC stage decimation factor(M_1) and the order of *sinc* filter.

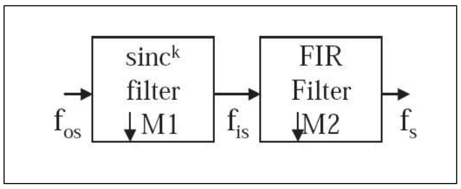


Figure 2.15: Two stage decimator with *sinc* filter

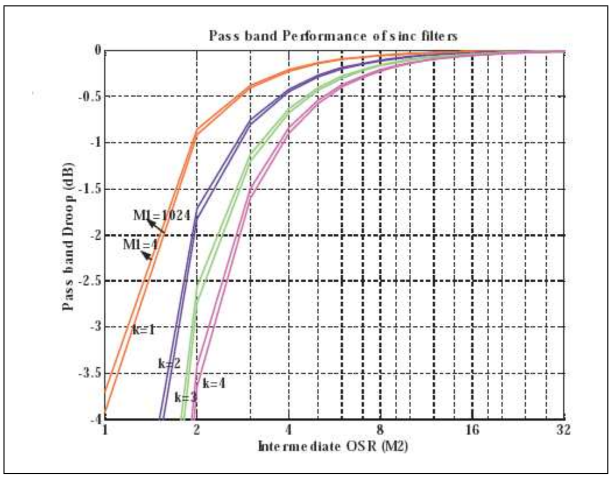


Figure 2.16: Passband droop of $sinc^k$ filter

With fourth-order *sinc* filter and an intermediate oversampling ratio of 4, the droop is about 1.6dB, but it increases rapidly if the intermediate oversampling ratio is lowered. It is usually inconvenient to compensate for a droop more than 3dB. Passband droop is more sensitive to the values of Intermediate OSR (M_2) and order of the filter (k) and is less sensitive to *sinc* stage decimation factor(M_1).

2.5.2 Stopband performance

In addition to suppressing quantization noise, the decimation filter must attenuate out-of-band signals [f_c, $\frac{f_{os}}{2}$) present at the modulator input. Worst case attenuation is at frequency $\frac{f_{os}}{M_1} - f_c$. This worst case rejection is plotted in Figure 2.17 for different order(k) *sinc* filter with different intermediate OSR (M_2) and decimation factor of *sinc* stage (M_1).

For third-order $sinc(k = 3)$ and an intermediate OSR(M_2) of 4, rejection is

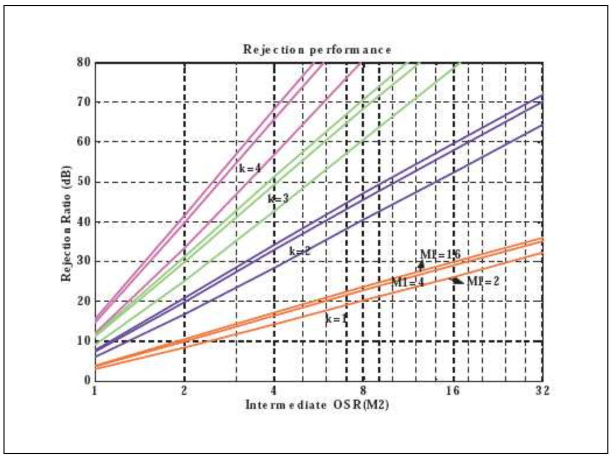


Figure 2.17: Worst case rejection of $sinc^k$ filter

50dB. The *sinc* filter worst case rejection is more sensitive to order (k), M_1 and to an extent on M_2.

2.5.3 *Sinc* stage design

Based on minimum required attenuation in stopband and allowed passband droop, *sinc* filter order and decimation ratio are chosen. In general, the order of the *sinc* filter should be atleast one order higher than that of the $\Delta\Sigma$ modulator in order to attenuate the rising shaped quantization noise as shown in Figure 2.18 [3]. For a L^{th} modulator, order of *sinc* should be atleast L+1.

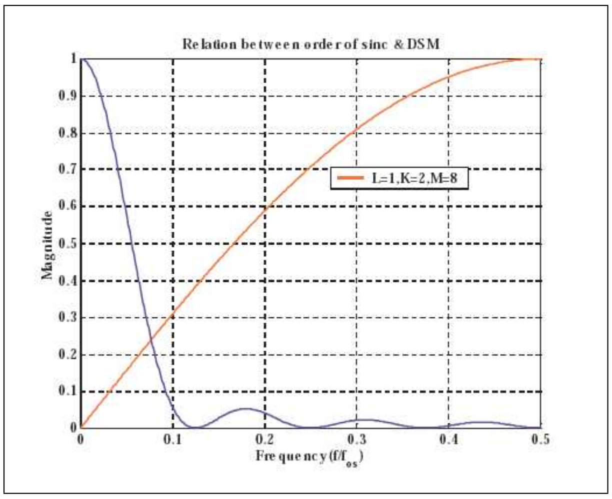


Figure 2.18: $sinc^k$ order selection

2.5.4 *Sinc* decimator example

In some low power and area applications only *sinc* filter is used for decimation and the output sampling rate is chosen based on required performance. A droop correction stage is used to compensate for the *sinc* droop. For example, a 4th order *sinc* decimator with droop correction stage designed for 3rd $\Delta\Sigma$ Modulator and OSR of 64 is shown in Figure 2.19. The figure also shows passband response of the $sinc^4$, droop correction and resultant approximately flat response.

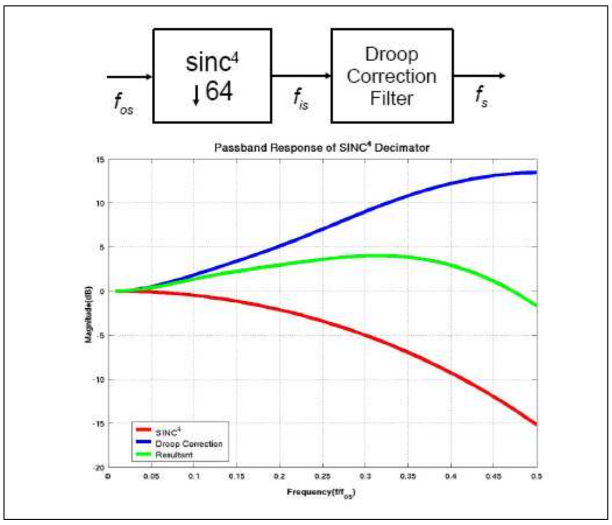


Figure 2.19: $sinc^4$ Decimator with droop correction

The droop correction filter is of 3rd order FIR with inverse *sinc* passband response. The coefficients of the droop correction filter are [2.2857, -1.7143, 0.5714, -0.1429]. Implementation aspects of this filter are discussed in next chapter.

CHAPTER 3

$SINC^K$ DECIMATION FILTER IMPLEMENTATION

Power consumption and area of decimation filters is receiving greater attention in constructing low power $\Delta\Sigma$ A/D converters for battery operated applications [12] [6]. The *sinc* filter discussed in Chapter 2 is widely used in decimation filter of $\Delta\Sigma$ ADC [18]. This chapter presents a comparison of different $sinc^4$ filter implementations with different decimation ratios and bit-resolution inputs. The main objective was to find an implementation efficient in terms of power and area for a given decimation ratio and input bit width.

3.1 Direct form FIR Implementation

The transfer function $H(z)$ of a *sinc* filter of order k and a decimation ratio M given by

$$H(z) = \left(\sum_{M-1}^{i=0} z^{-i}\right)^k$$

A cascaded direct form FIR implementation [1] is shown in Figure 3.1. As explained later, it is very inefficient interms of hardware and power. This requires $k(M-1)$ adders and registers operating at oversampling frequency.

3.2 CIC implementation

A simple and most popular way of implementing the *sinc* filter [7] shown in Figure 3.2 is based on the equation below.

$$H(z) = \sum_{M-1}^{i=0} z^{-i} = \frac{1-z^{-M}}{1-z^{-1}} = \left(\frac{1}{1-z^{-1}}\right)\left(1-z^{-M}\right)$$

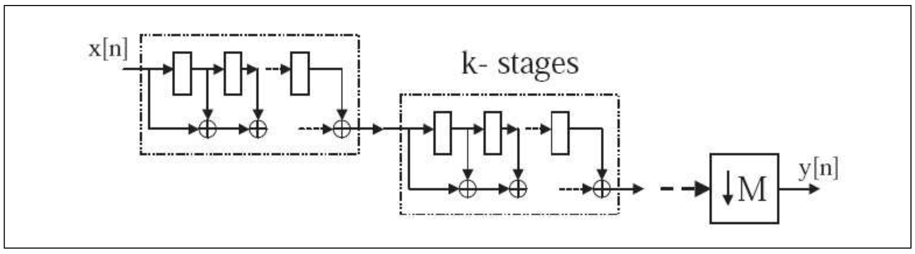


Figure 3.1: Direct Form FIR implementation of $sinc^k$

This implementation is known as Cascade Integrator-Comb (CIC) as it is implemented as a cascade of an integrator and a comb filter(differentiator). This requires one adder and one register for integrator and one adder and one register for differentiator. This structure is very efficient interms of hardware. The same

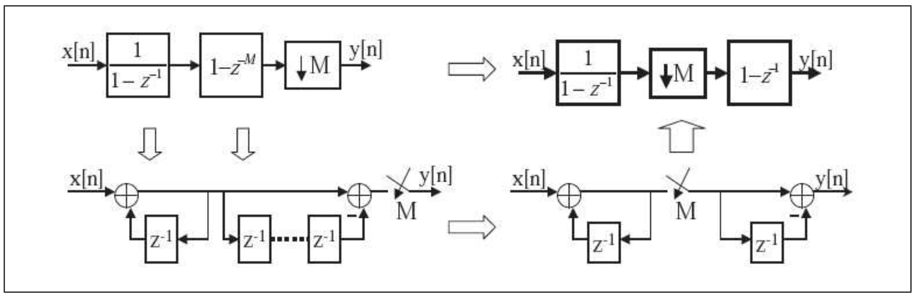


Figure 3.2: CIC implementation of $sinc^1$ filter

theory can be extended to higher order *sinc* filter and is shown in Figure 3.3.

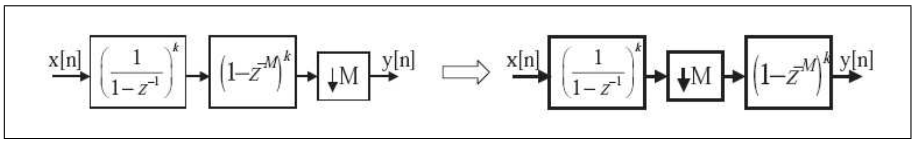

x[n]
$\left(\frac{1}{1-z^{-1}}\right)^k$
$\left(1-z^{-M}\right)^k$
↓M
y[n]

Figure 3.3: CIC implementation of $sinc^k$ filter

In order to avoid instability problems of the integrators, a modulo two arithmetic [5] is used to determine the internal wordlength for most economic VLSI realization. The maximum register width is given by the ratio of maximum output magnitude resulting from the worst possible input signal and the maximum input magnitude, which is equal to M^k in this case [8] [7]. So, the required adder widths in integrators is

$$Adderwidth = B_A = \lceil B_{in} + klog_2 M \rceil$$

For example, with k=4,M=64,B_{in}=4, adders width required is B_A=28 as shown in Figure 3.4.

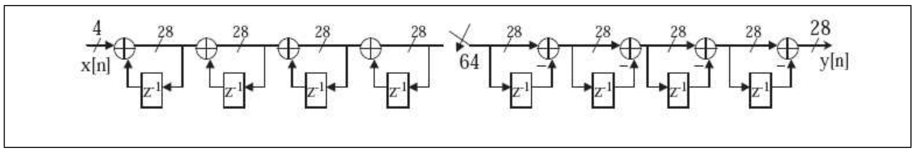


Figure 3.4: CIC implementation of $sinc^4$ with M=64

3.2.1 Re-timed CIC

With large oversampling ratios(M), integrators require longer word lengths and have to operate at the very high oversampling frequency(f_{os}). Therefore the integrators' section limits the applicability of this CIC structure to very high frequency oversampling A/D converters. On the otherhand, power consumption of these adders is high as they operate at oversampling frequency. Critical path can be reduced to certain extent by re-timing [14] each integrator register as shown in Figure 3.5. This re-timing technique has a side benefit of reducing the glitch power dissipation in adders but it increases the latency.

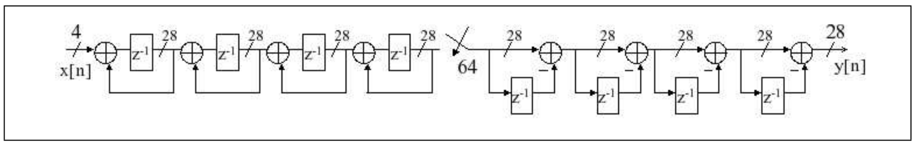


Figure 3.5: Retimed CIC implementation

3.2.2 Pipelined CIC

Pipelining CIC reduces the power consumption. As the output of the integrators is directly connected to the comb stage adder input and the integrator output changes at oversampling rate, there is a large amount of switching in the comb adders, increasing the power dissipation. This undesirable switching power can be reduced by isolating the integrators' output and the comb input by inserting a register operating at comb clock as shown in Figure 3.6. Pipelining is advantageous in both the basic CIC and re-timed CIC.

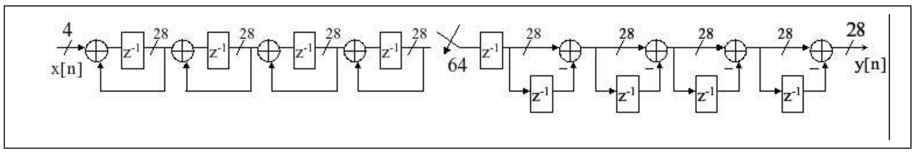


Figure 3.6: Pipelined Retimed CIC

3.2.3 Comparison of CIC architectures

The techniques discussed in previous subsection are applied to implement the SINC decimation filter discussed in the section 2.5.4. RTL was developed for these architectures, sythesized and power consumption was measured through circuit simulation. The results are tabulated in table 3.1.

	CIC	CIC-Pipelined	CIC Retimed	CIC Retimed-Pipelined
Avg. Current(μA)	31.2	17.3	27.3	16.8
Area(μm^2)	39901	41096	40440	41560
Critical path (ps)	76576	48006	66835	41604
Adders(FA + HA)	387 + 10	387 + 10	387 + 10	387 + 10
DFF(RS+BASIC)[1]	103 + 288	103 + 312	103 + 288	103 + 312
Buffer+Inverter	87 + 141	97 + 141	126 + 141	130 + 141

Table 3.1: $sinc^4$(M=64), CIC implementations

Pipelining greatly reduces the power consumption with slight increase in hardware. Critical path is reduced on retiming the registers. Reduction in power on retiming is due to reduced glitch power dissipation in integrtors chain. The increase in buffer count on retiming is because, now the integrator register has to drive two adders instead of one before retiming. Flip-Flops with reset were used in integrator registers and D-FlipFlop without reset were used in comb FIR stages as it reduces the routing and area. This is advantageous especially in deep submicron technologies as the interconnect loading dominates the gate loading. This reduces the power consumption and area to some extent.

3.3 Multistage FIR implementation

An other alternative to the above implementation can be derived based on non-recursive algorithm,

$$H(z) = \sum_{M-1}^{i=0} z^{-i} = 1 + z^{-1} + z^{-2} + ... + z^{-(M-1)} = \Pi_{i=0}^{log_2^M - 1}\left(1 + z^{-2^i}\right)$$

This can be explained better with an example. For example, the *sinc*^{1} transfer function for M=4 becomes

$$H(z) = 1 + z^{-1} + z^{-2} + z^{-3} = (1 + z^{-1})(1 + z^{-2})$$

The implementation of this equation is shown in Figure 3.7. The same non-

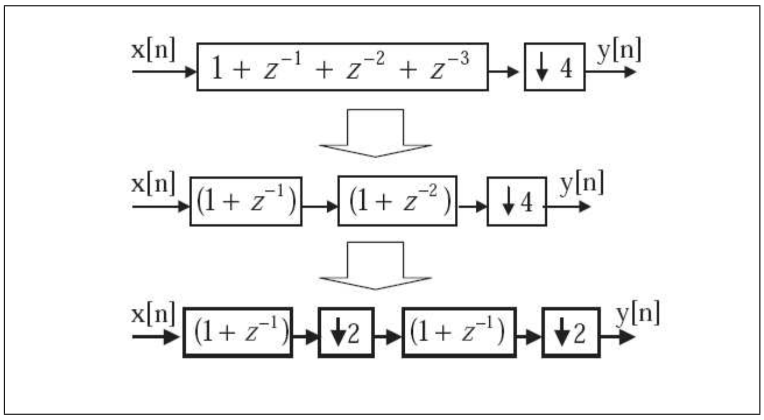


Figure 3.7: Multi-stage FIR implementation of *sinc* with M=4

recursive algorithm [6] can be generalized to higher order *sinc* filters as shown in figure 3.8.

$$H(z) = \left(\sum_{M-1}^{i=0} z^{-i}\right)^k = \Pi_{i=0}^{log_2^M - 1}\left(1 + z^{-2^i}\right)^k$$

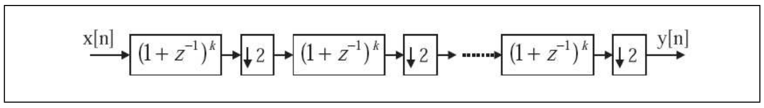


Figure 3.8: Multi-stage FIR implementation of *sinc*^{k}

Every stage has the same low-order (k^{th} order) FIR filter but operates at different sampling rate. Each stage has a maximum gain of k, bit width increases by k-bits from input to output of each stage. So, the word length increases through every stage by k-bits and the word rate (sampling rate) decreases through every stage by a factor of 2 starting from the oversampling frequency (f_{os}) as shown in table 3.2. An example $sinc^4$ with M=64 FIR implementation is shown in Figure 3.9.

	Input	Output
Sampling rate	$f_{os}/2^{i-1}$	$f_{os}/2^i$
Word length	$B_{in} + k(i-1)$	$B_{in} + ki$

Table 3.2: Sampling rate & word length of stage i in Figure 3.8, (i=1,2,..log_2M)

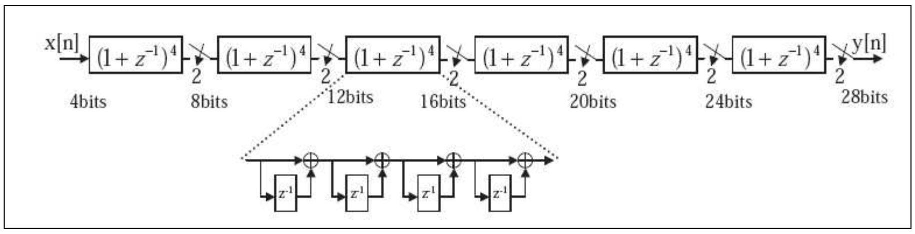


Figure 3.9: Multi-stage FIR implementation of $sinc^4$ with M=64

In this non-recursive algorithm, the word length at oversampling rate is very much less than that of CIC. The word length is short when the sampling rate is high and when the word length increases the sampling rate decreases. Reducing the wordlength as early as possible helps to reduce the power consumption. The main advantage of this non-recursive architecture is it's support to pipelining, theoritically it can be used to achieve infinite throughput with infinite hardware.

3.3.1 Pipelining in multi-stage FIR

Pipelining can be used to isolate the stages which reduces power consumption and critical path. As in CIC implementation without the pipelining high frequency stage output is connected to low frequency stage adder inputs, this can dissipate considerable amout of power. Pipelining register can be added at the input of each stage to break the direct path from previous stage output.This technique is shown in Figure 3.10.

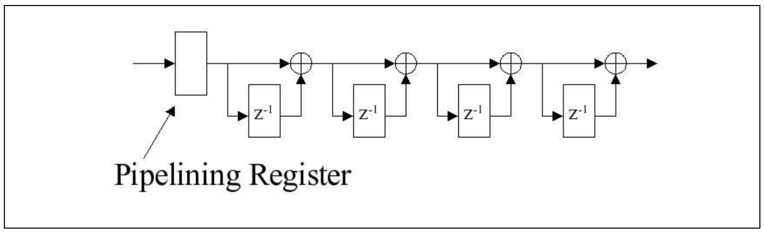


Figure 3.10: Pipelined FIR stage of $sinc^4$

3.3.2 Pipelined vs Non-pipelined

The results of multi-stage FIR implementation of the example discussed in section 2.5.4 are tabulated in table 3.3.

	FIR	FIR-Pipelined
Avg. Current(μA)	41.8	23.4
Area(μm^2)	46191	49935
Critical path (ps)	119826	26478
Adders(FA + HA)	461 + 26	461 + 26
DFF(RS+BASIC)	7 + 462	7 + 542
Buffer+Inverter	34 + 55	39 + 55

Table 3.3: $sinc^4$(M=64), Multi-stage FIR implementations

As expected pipelining reduced the critical path delay and power dissipation, but the area overhead is slightly more that of CIC as it requires a register in between every two stages.

3.4 Polyphase decomposed FIR multistage approach

Frequency of operation of each stage hardware can be further reduced by polyphase decomposition [16] [1] [6] of FIR in each stage as shown in figure 3.11 for fourth-order *sinc* filter. Implementation of each stage is shown in Figure 3.12. Overall $sinc^4$ decimation filter with polyphase decomposition in each stage is shown in Figure 3.13. The same technique of polyphase decomposition can be applied to any decimator stage with abitrary decimation ratio. But in some cases(with higher

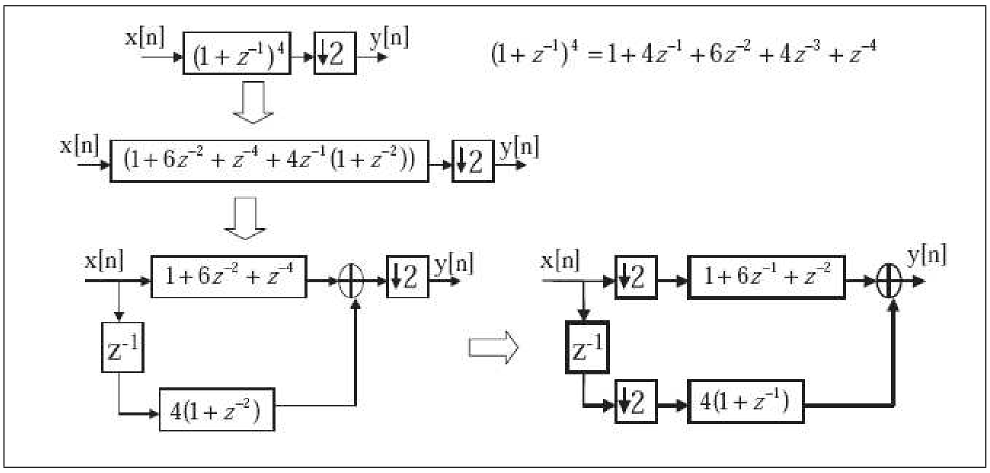


Figure 3.11: Polyphase decomposed FIR stage of $sinc^4$

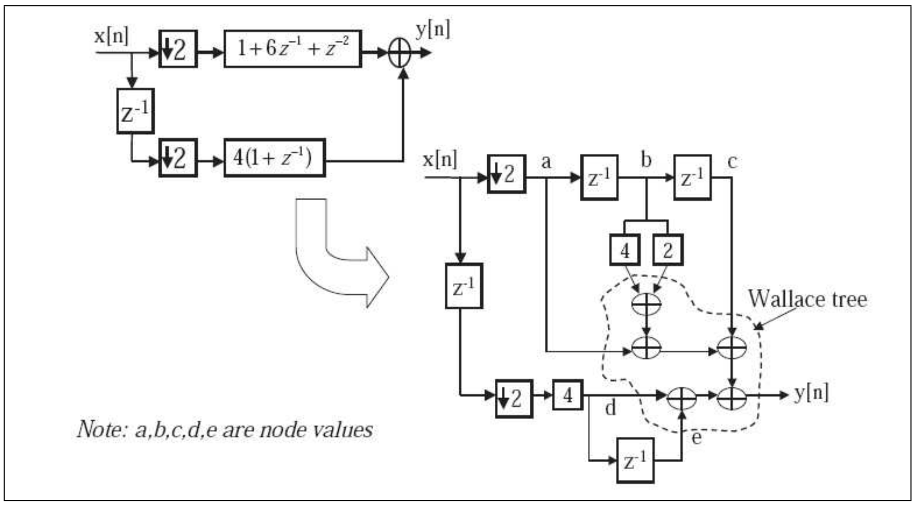


Figure 3.12: Implementation of each stage

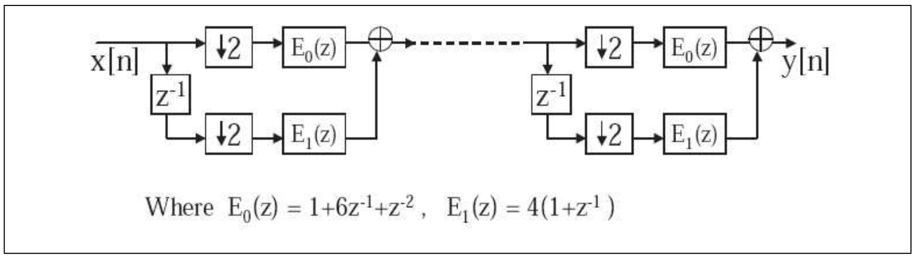


Figure 3.13: Polyphase decomposed FIR stage of $sinc^4$

decomposition factors) reduced clock frequency does not compensate for added complexity of polyphase decomposition [6].Technique of pipelining can be used to reduce power consumption. The results of the presented in the table 3.4.

	POLY	POLY-Pipelined
Avg. Current(μA)	36.2	22.7
Area(μm^2)	48610	52449
Critical path (ps)	116069	20418
Adders(FA + HA)	541 + 6	541 + 6
DFF(RS+BASIC)	7 + 432	7 + 512
Buffer+Inverter	48 + 60	48 + 60

Table 3.4: $sinc^4$(M=64), Polyphase implementations

As we can observe that polyphase decomposition of FIR in each stage reduced the power consumption. Pipelining reduced the power consumption with an area overhead. On comparing with FIR results, Without the pipelining reduction in power by polyphase decomposition is more compared power reduction with pipelining. This is because of stage isolation on pipelining.

3.5 Architecture for $sinc^4$ with M=64

These different implementations discussed so far can be better compared with pareto graph [14] drawn between area and power as shown in Figure 3.14 and tabulated in table 3.5. Efficient architecture is the one closer to origin in the graph

	CIC	CIC-P	CICRT	CICRT-P	FIR	FIR-P	POLY	POLY-P
Avg. Current(μA)	31.2	17.3	27.3	16.8	41.8	23.4	36.2	22.7
Area(μm^2)	39901	41096	40440	41560	46191	49935	48610	52449

Table 3.5: $sinc^4$(M=64), Architectures comparison

3.14. Pipelined CIC architectures are efficient as they dissipate less power compared to other architectures. Even though stages in FIR implementation operate at low frequency, because of its more gate count, it requires more interconnects. Especially in this (0.13μm) deep submicron technologies power consumption due to interconnect loading dominates over gate loading. Area efficient CIC is power efficient as it requires less interconnects at M=64.

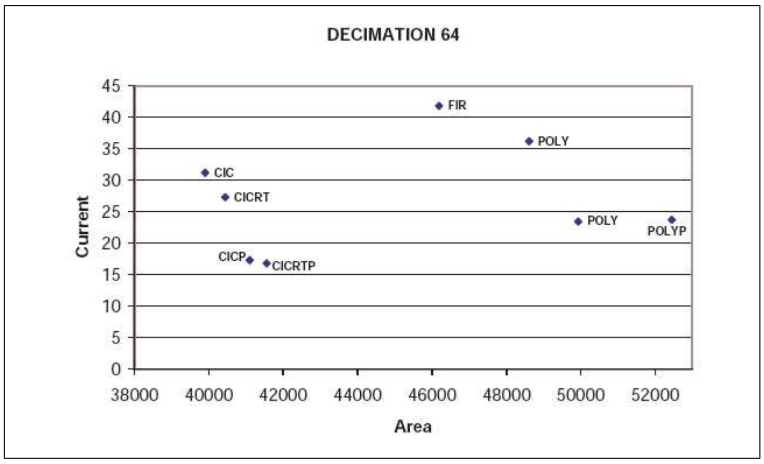


Figure 3.14: Architectures comparison

3.6 Architectures comparison

As the objective is to find an efficient implementation in terms of power & area of $sinc^4$ filter for a given decimation ratio and input bit width, a parameterized RTL in verilog HDL is developed, synthesized with $0.13\mu m$ standard cell library and power estimations are done through spice simulation. As the target is to find the low power architecture which rules out the non-pipelined architectures. The results are presented for different input bit widths and oversampling ratios of three architectures (pipelined CIC,FIR,POLY) discussed in the last sections .

3.6.1 Oversampling ratio of 2,(M=2)

The power dissipation & synthesis results for $sinc^4$ filter with M=2 and different values of input bit width (B_{in})of CIC,FIR and POLY are shown in tables 3.6,3.7 and 3.8 respectively.

As plotted in Figure 3.15,with M=2, CIC consumes more power and occupies more area, this is because of long adder widths. Polyphase and FIR have

	$B_{in} = 1$	$B_{in} = 2$	$B_{in} = 3$	$B_{in} = 4$	$B_{in} = 5$	$B_{in} = 6$
Avg. Current(μA)	3.45	4.4	5.49	5.99	6.11	6.25
Area(μm^2)	4356	5304	6263	7167	8218	9102
Critical path(ps)	16639	18690	22208	21766	23285	24803
Adder(FA+HA)	24+4	32+4	40+4	48+4	56+4	64+4
DFF(RS+BASIC)	21+30	25+36	29+42	33+48	37+54	41+60

Table 3.6: $sinc^4$(M=2), CIC-Pipelined

	$B_{in} = 1$	$B_{in} = 2$	$B_{in} = 3$	$B_{in} = 4$	$B_{in} = 5$	$B_{in} = 6$
Avg. Current(μA)	0.6091	1.02	1.31	1.51	1.72	1.97
Area(μm^2)	1279	1721	2174	2585	3036	3457
Critical path(ps)	18446	21290	22837	24344	26020	27665
Adder(FA+HA)	6+4	10+4	14+4	18+4	22+4	26+4
DFF(RS+BASIC)	1+15	1+20	1+25	1+30	1+35	1+40

Table 3.7: $sinc^4$(M=2), FIR-Pipelined

	$B_{in} = 1$	$B_{in} = 2$	$B_{in} = 3$	$B_{in} = 4$	$B_{in} = 5$	$B_{in} = 6$
Avg. Current(μA)	0.838	1.19	1.44	1.65	1.76	1.8
Area(μm^2)	1103	1581	2095	2586	3066	3568
Critical path(ps)	14474	16654	18420	20441	22754	26457
Adder(FA+HA)	11+0	16+0	21+0	26+0	31+0	36+0
DFF(RS+BASIC)	1+9	1+14	1+19	1+24	1+29	1+34

Table 3.8: $sinc^4$(M=2), POLY-Pipelined

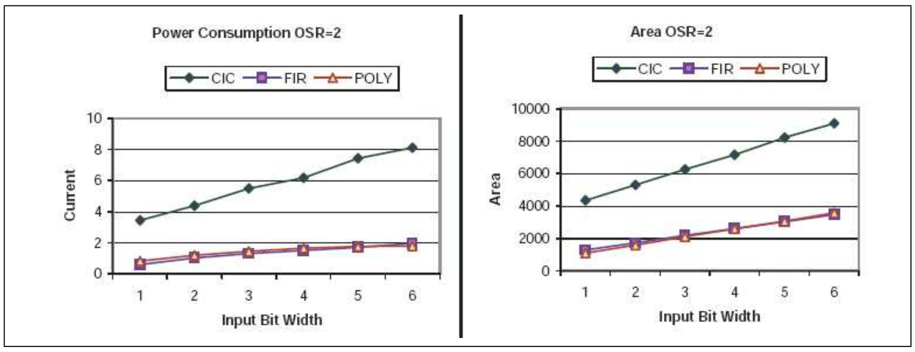


Figure 3.15: Comparison of architectures for M=2

approximately same performance but better than CIC.

3.6.2 Oversampling ratio of 4,(M=4)

The results for the three architectures(CIC,FIR,POLY) with M=4 are shown in tables 3.9,3.10 and 3.11 respectively.

	$B_{in}=1$	$B_{in}=2$	$B_{in}=3$	$B_{in}=4$	$B_{in}=5$	$B_{in}=6$
Avg. Current(μA)	5.19	5.36	5.56	6.06	6.59	6.85
Area(μm^2)	8124	9131	10068	11030	11965	12959
Critical path(ps)	23332	25038	28004	27869	29714	33225
Adder(FA+HA)	56+4	64+4	72+4	80+4	88+4	96+4
DFF(RS+BASIC)	38+54	42+60	46+66	50+72	54+78	58+84

Table 3.9: $sinc^4$(M=4), CIC-Pipelined

	$B_{in}=1$	$B_{in}=2$	$B_{in}=3$	$B_{in}=4$	$B_{in}=5$	$B_{in}=6$
Avg. Current(μA)	3.28	4.59	5.4	5.79	6.73	6.96
Area(μm^2)	4212	5122	5953	6824	7695	8542
Critical path(ps)	18446	21290	22837	24344	26020	27665
Adder(FA+HA)	28+8	36+8	44+8	52+8	60+8	68+8
DFF(RS+BASIC)	2+50	2+60	2+70	2+80	2+90	2+100

Table 3.10: $sinc^4$(M=4), FIR-Pipelined

	$B_{in}=1$	$B_{in}=2$	$B_{in}=3$	$B_{in}=4$	$B_{in}=5$	$B_{in}=6$
Avg. Current(μA)	2.13	2.82	3.28	3.72	4.03	4.57
Area(μm^2)	4163	5095	6121	7102	8071	9076
Critical path(ps)	14474	16654	18420	20441	22754	26457
Adder(FA+HA)	42+0	52+0	62+0	72+0	82+0	92+0
DFF(RS+BASIC)	2+38	2+48	2+58	2+68	2+78	2+88

Table 3.11: $sinc^4$(M=4), POLY-Pipelined

With M=4, CIC consumes more power and occupies more area because of long width adders. Polyphase implementation is advantageous interms of power compared to other two.

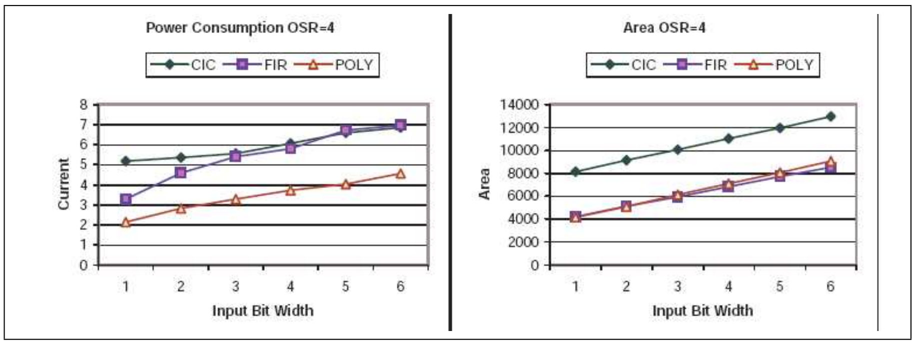


Figure 3.16: Comparison of architectures for M=4

3.6.3 Oversampling ratio of 8,(M=8)

The results for M=8 are shown in tables 3.12, 3.13 and 3.14 of CIC,FIR and POLY respectively.

	$B_{in} = 1$	$B_{in} = 2$	$B_{in} = 3$	$B_{in} = 4$	$B_{in} = 5$	$B_{in} = 6$
Avg. Current(μA)	5.22	5.38	6.08	7.58	7.16	7.67
Area(μm^2)	11995	12960	13810	14766	15775	16736
Critical path(ps)	29684	32970	35252	36830	35888	39755
Adder(FA+HA)	88+4	96+4	104+4	112+4	120+4	128+4
DFF(RS+BASIC)	55+78	59+84	63+90	67+96	71+102	75+108

Table 3.12: $sinc^4$(M=8), CIC-Pipelined

	$B_{in} = 1$	$B_{in} = 2$	$B_{in} = 3$	$B_{in} = 4$	$B_{in} = 5$	$B_{in} = 6$
Avg. Current(μA)	4.92	5.14	7.06	7.76	8.19	8.57
Area(μm^2)	8983	10244	11534	12816	14112	15409
Critical path(ps)	18446	21290	22837	24344	26020	27665
Adder(FA+HA)	66+12	78+12	90+12	102+12	114+12	128+12
DFF(RS+BASIC)	3+105	3+120	3+135	3+150	3+165	3+180

Table 3.13: $sinc^4$(M=8), FIR-Pipelined

With M=8, CIC consumes more power and occupies more area because of long width adders. Compared M=4, FIR,POLY curves are approaching CIC both in power & area. FIR,POLY architectures are advantageous over CIC at M=8.

	$B_{in} = 1$	$B_{in} = 2$	$B_{in} = 3$	$B_{in} = 4$	$B_{in} = 5$	$B_{in} = 6$
Avg. Current(μA)	3.68	4.6	5.53	6.2	6.58	7.14
Area(μm^2)	9206	10682	12116	13542	15110	16515
Critical path(ps)	14474	16654	18420	20441	22754	26457
Adder(FA+HA)	93+0	108+0	123+0	138+0	153+0	168+0
DFF(RS+BASIC)	3+87	3+102	3+117	3+132	3+147	3+162

Table 3.14: $sinc^4$(M=8), POLY-Pipelined

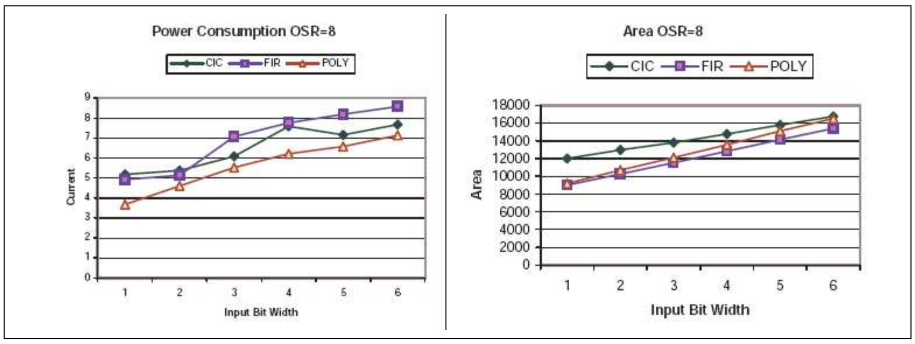


Figure 3.17: Comparison of architectures for M=8

3.6.4 Oversampling ratio of 16,(M=16)

The synthesis & power results for CIC,FIR & POLY implementations with M=16 are shown in tables 3.15, 3.16 and 3.17.

	$B_{in} = 1$	$B_{in} = 2$	$B_{in} = 3$	$B_{in} = 4$	$B_{in} = 5$	$B_{in} = 6$
Avg. Current(μA)	5.81	6.52	7.12	8.04	8.71	8.76
Area(μm^2)	15742	16695	17732	18775	19639	20606
Critical path(ps)	38209	39734	41092	40905	44180	46075
Adder(FA+HA)	120+4	128+4	136+4	144+4	152+4	160+4
DFF(RS+BASIC)	72+102	76+108	80+114	84+120	88+126	92+132

Table 3.15: $sinc^4$(M=16), CIC-Pipelined

Compared M=8, power of FIR is almost double, this is because of increase of area. For M=16, power & area of FIR and poly increases rapidly with input bit width.CIC is the low power architecture for M=16.

	$B_{in} = 1$	$B_{in} = 2$	$B_{in} = 3$	$B_{in} = 4$	$B_{in} = 5$	$B_{in} = 6$
Avg. Current(μA)	8.04	11.5	12	12.3	12.8	14.8
Area(μm^2)	15359	17073	18816	20585	22215	23931
Critical path(ps)	18446	21290	22837	24344	26020	27665
Adder(FA+HA)	120+16	136+16	152+16	168+16	184+16	200+16
DFF(RS+BASIC)	4+180	4+200	4+220	4+240	4+260	4+280

Table 3.16: $sinc^4$(M=16), FIR-Pipelined

	$B_{in} = 1$	$B_{in} = 2$	$B_{in} = 3$	$B_{in} = 4$	$B_{in} = 5$	$B_{in} = 6$
Avg. Current(μA)	7.62	8.11	10.9	12.1	13.1	13.9
Area(μm^2)	16212	18128	20075	22069	23992	25966
Critical path(ps)	14474	16654	18420	20441	22754	26457
Adder(FA+HA)	164+0	184+0	204+0	224+0	244+0	264+0
DFF(RS+BASIC)	4+156	4+176	4+196	4+216	4+236	4+256

Table 3.17: $sinc^4$(M=16), POLY-Pipelined

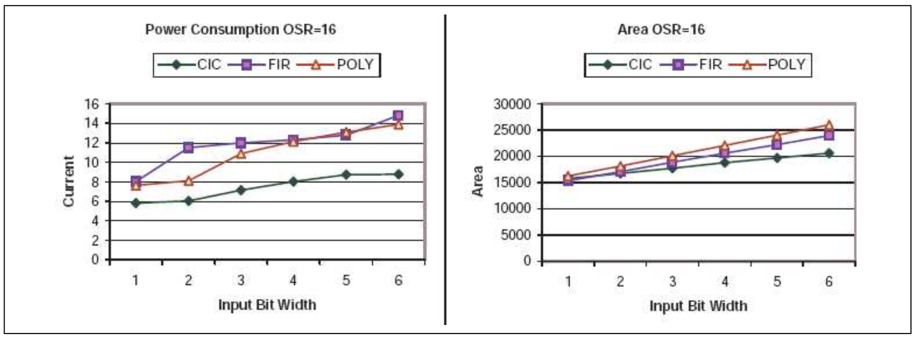


Figure 3.18: Comparison of architectures for M=16

3.6.5 Oversampling ratio of 32,(M=32)

The power dissipation & synthesis results for CIC,FIR & POLY implementations with M=32 are shown in tables 3.18,3.19 and 3.20.

	$B_{in} = 1$	$B_{in} = 2$	$B_{in} = 3$	$B_{in} = 4$	$B_{in} = 5$	$B_{in} = 6$
Avg. Current(μA)	7.02	7.78	8.02	8.56	9.06	9.63
Area(μm^2)	19705	20578	21502	22621	23551	24494
Critical path(ps)	42717	46660	46028	47549	50923	51152
Adder(FA+HA)	152+4	160+4	168+4	176+4	184+4	192+4
DFF(RS+BASIC)	89+126	93+132	97+138	101+144	105+150	109+156

Table 3.18: $sinc^4$(M=32), CIC-Pipelined

	$B_{in} = 1$	$B_{in} = 2$	$B_{in} = 3$	$B_{in} = 4$	$B_{in} = 5$	$B_{in} = 6$
Avg. Current(μA)	12.6	15.8	16.8	17.3	18	18.5
Area(μm^2)	23378	25420	27595	29765	31944	34078
Critical path(ps)	18446	21290	22837	24344	26020	27665
Adder(FA+HA)	190+20	210+20	230+20	250+20	270+20	290+20
DFF(RS+BASIC)	5+275	5+300	5+325	5+350	5+375	5+400

Table 3.19: $sinc^4$(M=32), FIR-Pipelined

	$B_{in} = 1$	$B_{in} = 2$	$B_{in} = 3$	$B_{in} = 4$	$B_{in} = 5$	$B_{in} = 6$
Avg. Current(μA)	11.5	12.6	15.4	16.1	17.1	17.9
Area(μm^2)	24993	27377	29788	32224	34756	37289
Critical path(ps)	14474	16654	18420	20441	22754	26457
Adder(FA+HA)	255+0	280+0	305+0	330+0	355+0	380+0
DFF(RS+BASIC)	5+245	5+270	5+295	5+320	5+345	5+370

Table 3.20: $sinc^4$(M=32), POLY-Pipelined

As we can see in Figure 3.19, CIC architecture is the lowest power of the three architectures. Polyphase decomposed architecture consumes less power than FIR with M=32.

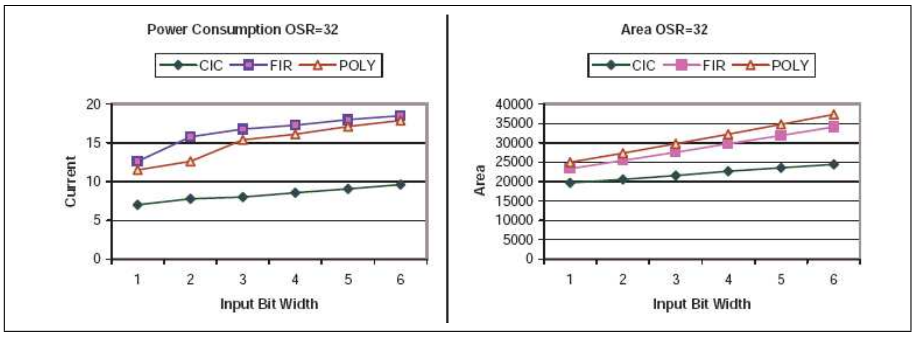


Figure 3.19: Comparison of architectures for M=32

3.6.6 Oversampling ratio of 64,(M=64)

The power dissipation & synthesis results for CIC,FIR & POLY implementations with M=64 are shown in tables 3.21,3.22 and 3.23.

	$B_{in} = 1$	$B_{in} = 2$	$B_{in} = 3$	$B_{in} = 4$	$B_{in} = 5$	$B_{in} = 6$
Avg. Current(μA)	7.5	7.78	8.25	8.75	9.72	10.8
Area(μm^2)	23599	24454	25456	26323	27312	28107
Critical path(ps)	49291	50952	52572	53878	56093	58821
Adder(FA+HA)	184+4	192+4	200+4	208+4	216+4	224+4
DFF(RS+BASIC)	106+150	110+156	114+162	118+168	122+174	126+180

Table 3.21: $sinc^4$(M=64), CIC-Pipelined

	$B_{in} = 1$	$B_{in} = 2$	$B_{in} = 3$	$B_{in} = 4$	$B_{in} = 5$	$B_{in} = 6$
Avg. Current(μA)	15.1	17.3	18.2	19	20.1	22.6
Area(μm^2)	33160	35719	38203	40863	43455	46089
Critical path(ps)	18446	21290	22837	24344	26020	27665
Adder(FA+HA)	276+24	300+24	324+24	348+24	372+24	396+24
DFF(RS+BASIC)	6+390	6+420	6+450	6+480	6+510	6+540

Table 3.22: $sinc^4$(M=64), FIR-Pipelined

As we can see in Figure 3.20, CIC architecture has the least power & area.

3.6.7 Oversampling ratio of 128,(M=128)

The power dissipation & synthesis results for CIC,FIR & POLY implementations with M=128 are shown in tables 3.24, 3.25 and 3.26.

	$B_{in} = 1$	$B_{in} = 2$	$B_{in} = 3$	$B_{in} = 4$	$B_{in} = 5$	$B_{in} = 6$
Avg. Current(μA)	17.4	18.4	21	23.2	23.7	25.8
Area(μm^2)	35760	38811	41777	44698	47483	50514
Critical path(ps)	14474	16654	18420	20441	22754	26457
Adder(FA+HA)	366+0	396+0	426+0	456+0	486+0	516+0
DFF(RS+BASIC)	6+354	6+384	6+414	6+444	6+474	6+504

Table 3.23: $sinc^4$(M=64), POLY-Pipelined

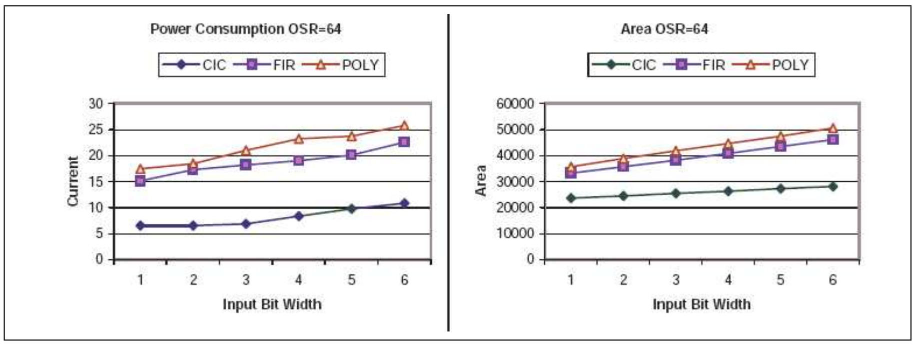


Figure 3.20: Comparison of architectures for M=64

	$B_{in} = 1$	$B_{in} = 2$	$B_{in} = 3$	$B_{in} = 4$	$B_{in} = 5$	$B_{in} = 6$
Avg. Current(μA)	9.25	10.7	11.8	12.5	13.4	13.8
Area(μm^2)	27367	28248	29216	30089	31061	32102
Critical path(ps)	55802	59171	60711	60399	62035	66009
Adder(FA+HA)	216+4	224+4	232+4	240+4	248+4	256+4
DFF(RS+BASIC)	123+174	127+180	131+186	135+92	139+198	143+204

Table 3.24: $sinc^4$(M=128), CIC-Pipelined

	$B_{in} = 1$	$B_{in} = 2$	$B_{in} = 3$	$B_{in} = 4$	$B_{in} = 5$	$B_{in} = 6$
Avg. Current(μA)	19	22.8	23.8	24.5	25.3	26.4
Area(μm^2)	44709	47631	50759	53783	56768	59856
Critical path(ps)	18446	21290	22837	24344	26020	27665
Adder(FA+HA)	378+28	406+28	434+28	462+28	490+28	518+28
DFF(RS+BASIC)	7+525	7+560	7+595	7+630	7+665	7+700

Table 3.25: $sinc^4$(M=128), FIR-Pipelined

	$B_{in} = 1$	$B_{in} = 2$	$B_{in} = 3$	$B_{in} = 4$	$B_{in} = 5$	$B_{in} = 6$
Avg. Current(μA)	21.6	25.1	27.4	28.1	29	30.5
Area(μm^2)	48750	52086	55954	59273	62535	65832
Critical path(ps)	14474	16654	18420	20441	22754	26457
Adder(FA+HA)	497+0	532+0	567+0	602+0	637+0	672+0
DFF(RS+BASIC)	7+483	7+518	7+553	7+588	7+623	7+658

Table 3.26: $sinc^4$(M=128), POLY-Pipelined

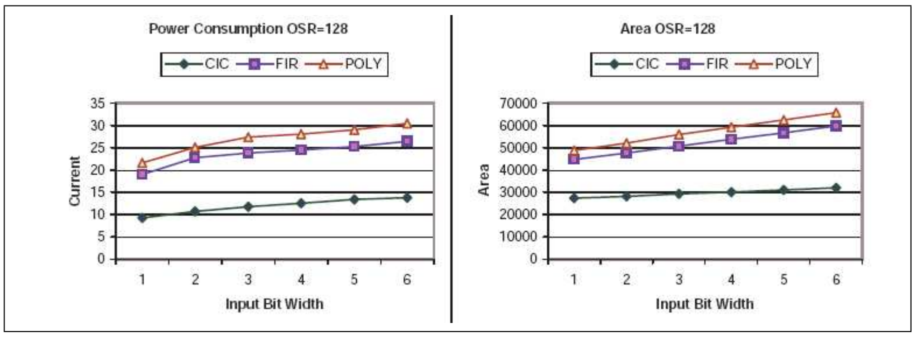


Figure 3.21: Comparison of architectures for M=64

It can observed in Figure 3.21 that, CIC architecture is the least power & area implementation.

From the above results, FIR & POLY gives high performance over CIC at low oversampling ratios (2,4,8) and CIC is advantageous at higher oversamping ratios (16,32,64,128) with its compactness and low power consumption. The main reason for increase of power in FIR & POLY implementation at higher oversamping ratios is that increased hardware and interconnects.

CHAPTER 4

INTERPOLATION FILTER

In this work an interpolation filter is designed for use in low power & high resolution oversampling $\Delta\Sigma$ DAC. The concepts of interpolation and its design basics are discussed in this chapter.

4.1 Signal reconstruction basics

4.1.1 Ideal signal reconstruction from samples

According to Nyquist sampling theory any signal can be reconstructed from its samples if the signal is sampled greater than or equal to twice the signal bandwidth [1]. This reconstruction process involves an ideal Digital-to-Analog Conversion (D/A) followed by a Low Pass Filtering (LPF) to remove the images and extract the band around dc. This ideal impulse reconstruction process is shown in figure 4.1. The analog LPF is called reconstruction or anti-imaging filter. The analog LPF should have a sharp cutoff at $\frac{f_s}{2}$ (ideally).

4.1.2 Signal reconstruction with ZOH

As it is easy to implement an Zero-Order-Hold D/A than a impulse D/A, in-practice ZOH D/A interface is generally used. Reconstruction with ZOH D/A interface is shown in figure 4.2.

The disadvantage of this ZOH is passband droop which needs to be compensated by analog LPF with inverse SINC response in passband. On otherside, ZOH increases the image attenuation which can be used to relax the stopband specifications.

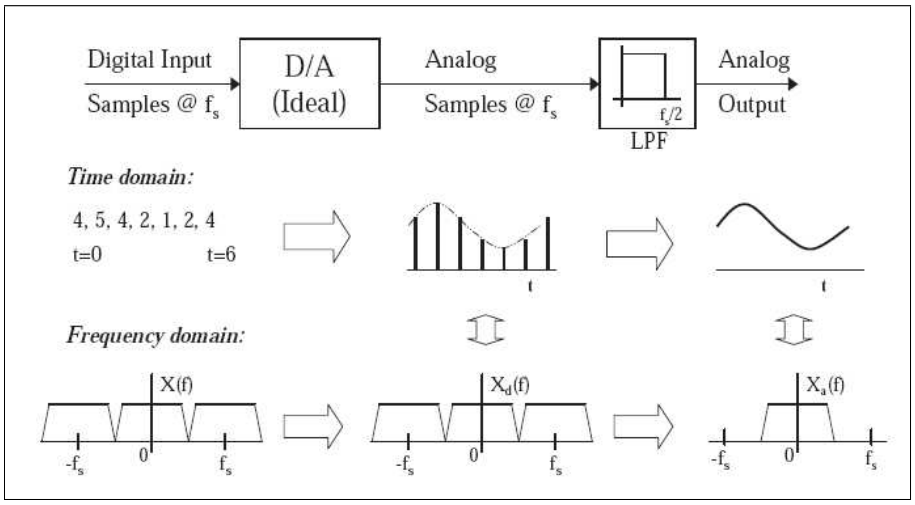


Figure 4.1: Ideal signal reconstruction

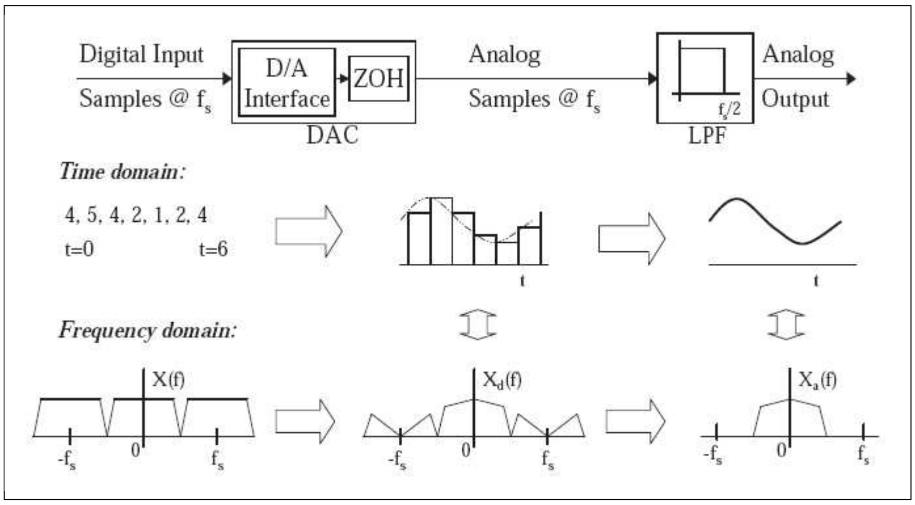


Figure 4.2: Signal reconstruction with ZOH

4.1.3 Practical reconstruction

Ideally sharp cutoff is required for the filter at the output but this sharp cutoff demands higher order filters. To relax the requirements of the filter the signal is sampled slightly greater than nyquist rate(f_N). This allows a transition band of $f_s - f_N$ for the filter as shown in figure 4.3.

$$f_s > f_N \text{ and } f_N = 2f_B$$

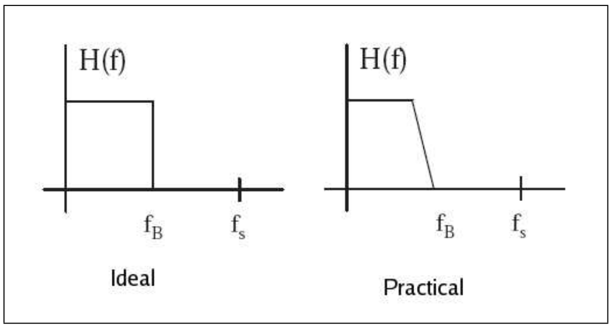


Figure 4.3: Ideal & Practical LPF characteristics

For example, in a typical CD quality audio application, $f_B = 20kHz$ and $f_s = 44.1kHz$ allowing a transition band of $4.1kHz$.

4.2 Oversampling

Reconstruction filter requirements can be further relaxed and passband droop due to ZOH can be reduced by oversampling the signal ($f_{os} >> f_s$)as shown in figure 4.4. This oversampling ($f_{os} = Mf_s$) allows a transition bandwidth of $f_{os} - 2f_B$.

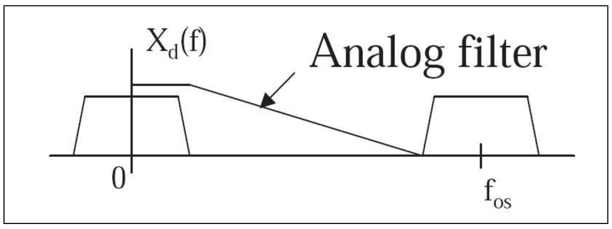


Figure 4.4: LPF with oversampling

This wide transition band requires a simple filter at the output but the price paid here is DAC need to operated at this oversampling frequency and a special digital hardware is convert the signal from nyquist rate to oversampling rate. Nyquist rate signal reconstruction and oversampling signal reconstruction are shown in figure 4.5.

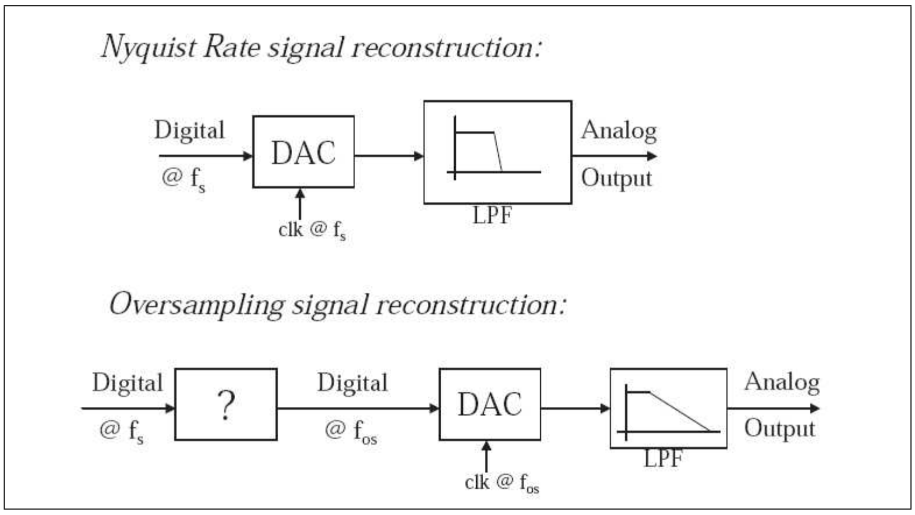


Figure 4.5: Nyquist & oversampling reconstruction

The digital block that converts the nyquist rate signal to oversampling rate is called as interpolator as interpolate between the nyquist rate samples as shown in figure 4.6.

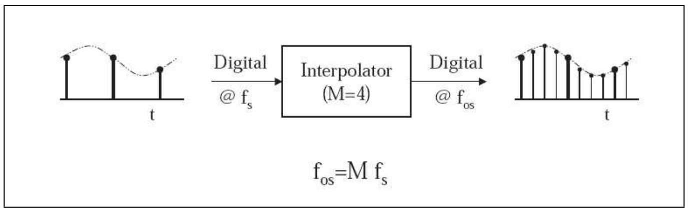


Figure 4.6: Interpolator (black box)

4.3 Interpolator

The interpolator can be conceptually thought as a zero stuffing followed by digital LPF as shown in figure 4.7 [1] [11] [16] [7]. In time domain, with zero stuffing between the input samples maintains the same time period in which frequency domain appears as images. So, the digital LPF is used to remove the images and this calculates the values for samples between input samples in time domain. On upsampling or zero stuffing the power of signal goes down by the upsampling factor and to restore the same power level at the output the LPF should have a gain equal to upsampling factor.

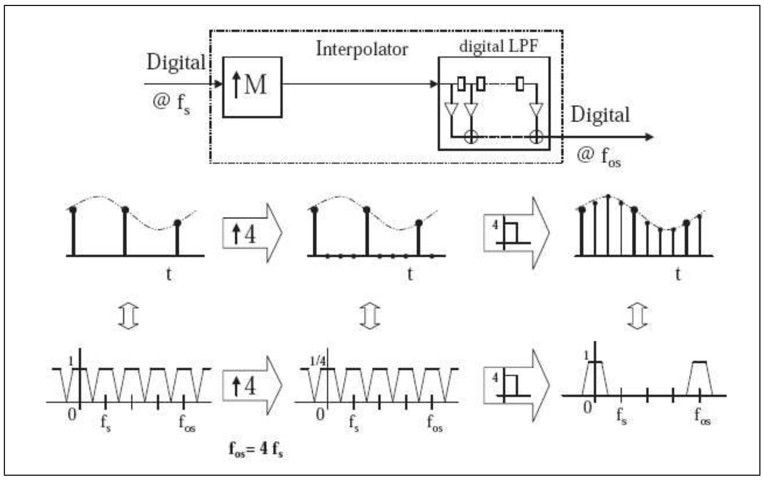


Figure 4.7: Interpolation of a signal - analysis

Ideally cutoff frequency of digital LPF should be f_B and a passband gain of M for an interpolation factor of M. Similar to analog reconstruction filter in nyquist signal reconstruction, sharp cutoff of digital LPF requires higher order digital filter. So, with oversampling the problem of sharp cutoff of filter has been transfered from analog domain to digital domain. The digital filter order can be reduced by selecting lower sampling rate slightly above nyquist rate. Design and implementation of higher order digital filter is comparatively easier than its analog counter part so this technique is widely used.

4.4 Oversampling with Noise shaping

An important and widely used dimension of oversampling is in building a high resolution D/A converter [11] [18]. A noise shaping modulator is used at oversampling rate to push the quantization noise to out-of-band high frequencies. With this technique of noise shaping a low-bit resolution DAC operating at oversampling frequency is used as shown in figure 4.8. The analog LPF suppress the high frequency quantization noise and in general it's order is one order higher than the $\Delta\Sigma$ modulator as similar to the case of SINC decimtor order following the analog $\Delta\Sigma$ modulator (section 2.5.3).

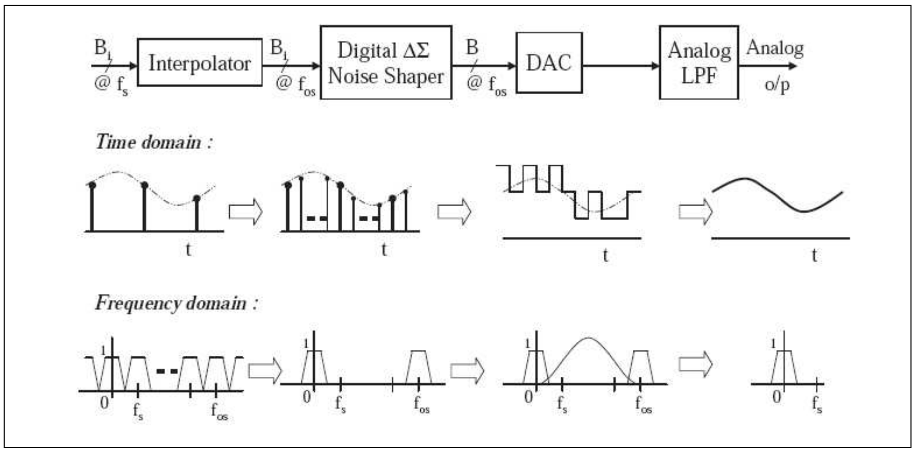


Figure 4.8: Oversampling $\Delta\Sigma$ DAC

4.5 Interpolator design basics

Conceptually, The interpolator consists of an upsampler followed by a digital LPF as shown in figure 4.9. Both the interpolator and decimator are dual to each other, both requires a digital LPF. The design concepts developed for decimator can be applied to the design of interpolator. Similar to decimator single stage approach results in higher order LPF because of sharp transition and entire filter operates at output oversampling frequency [18] [1]. In general, multistage approach is used as it allows to relax requirements of the filters operating at high frequency. For more detailed information, reader can refer to [18] [1] [7]. At higher sampling rates only image rejection needs to be considered , so simple and economic filters are used to reduce hardware and power consumption.

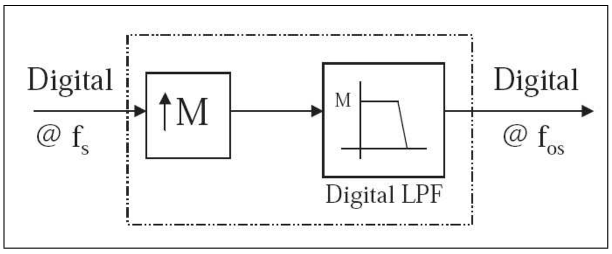


Figure 4.9: Interpolator block diagram

4.6 Interpolator design specifications

Interpolation by factor of 64 as shown in figure 4.10and dynamic range of 16-bit is required for the target application. Input and output are 18-bit fixed point numbers with 17bits of fractional length. Interpolator should map +/-0.5 signal at input to +/-0.75 at the output which means a gain of 1.5.

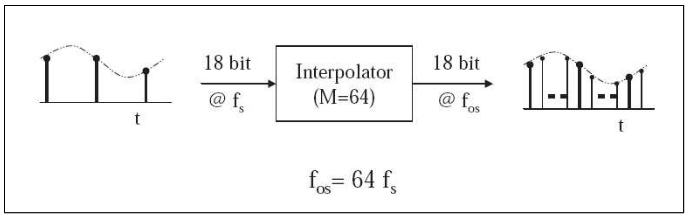


Figure 4.10: Interpolator specifications

With $M = 64 = 2^6$, multi-stage design approach requires 6 stages with each stage interpolating by a factor of 2. This multistage approach results in low power and less hardware. In this chapter a basic design is discussed and a comparative design to meet the specs is given in next chapter.

4.7 Interpolator design 1

A 6-stage design is shown in figure 4.11. A 4^{th} IIR filter is used in the first stage of filter for sharp cutoff with minimum hardware. Last five stages use 4^{th} order FIR filters.

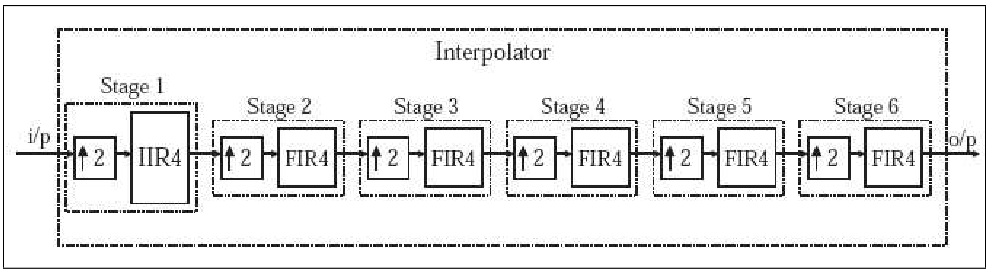


Figure 4.11: 6-stage interpolator

4.7.1 Interpolator stage 1(Elliptic IIR4)

A fourth order elliptic IIR is implemented as cascade of two Second-Order-Section(SOS) as shown in figure 4.12.

$$H_1(z) = 0.4921875 \times \frac{0.125 + 0.25z^{-1} + 0.125z^{-2}}{1 - 1.00390625z^{-1} + 0.375z^{-2}} \times \frac{0.5 + 1.125z^{-1} + 0.5z^{-2}}{1 - 0.5z^{-1} + 0.75z^{-2}} \times 2$$

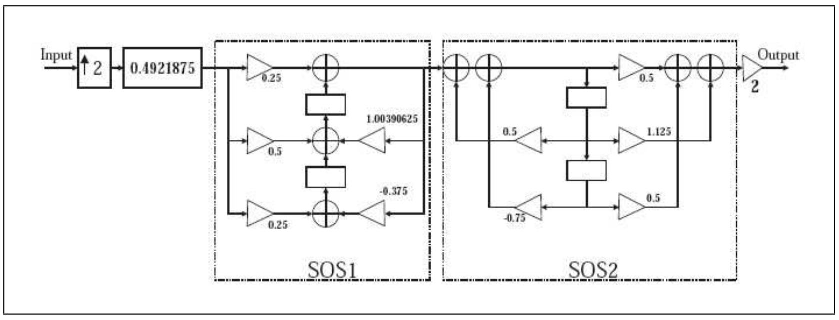


Figure 4.12: Interpolator stage 1 (IIR4)

To prevent overflow in adders, interpolative gain of 2 is provided at the output and the signal is scaled down at the input with 0.4921875. Frequency response and pole zero plot are shown in figure 4.13.

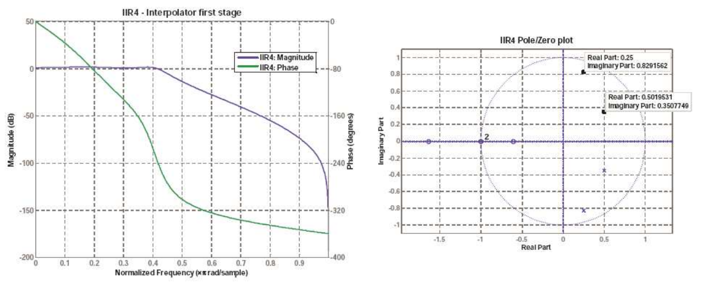


Figure 4.13: Frequency response & PZ plot of IIR4

4.7.2 Interpolator stage 2 (FIR4)

The second stage 4^{th} order FIR is designed to achieve maximum image rejection without effecting passband and its magnitude response is shown in figure 4.14

$$H_2(z) = (0.03125 + 0.25z^{-1} + 0.5z^{-2} + 0.25z^{-3} + 0.03125z^{-4}) \times 2$$

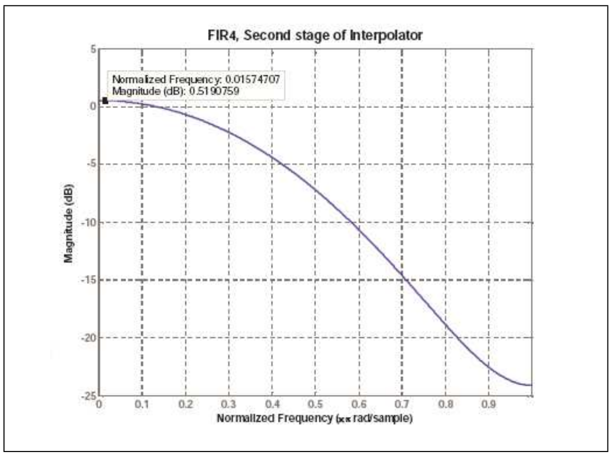


Figure 4.14: Frequency response of stage 2(FIR4)

Polyphase Implementation

Polyphase implementation technique is used as shown in figure 4.15 to reduce hardware and power consumption.

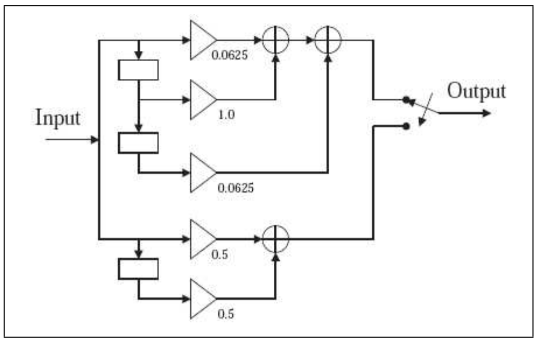


Figure 4.15: Implementation of stage 2(FIR4)

4.8 Last four interpolator stages

The magnitude responses of 4^{th} order FIR used in last four stages are shown in figure 4.16 and their transfer functions are given below.

$$H_3(z) = (0.25 + 0.25z^{-1} + 0.0625z^{-2} + 0.25z^{-3} + 0.25z^{-4}) \times 2$$
$$H_4(z) = (0.25 + 0.25z^{-1} + 0.015625z^{-2} + 0.25z^{-3} + 0.25z^{-4}) \times 2$$
$$H_5(z) = (0.25 + 0.25z^{-1} + 0.0078125z^{-2} + 0.25z^{-3} + 0.25z^{-4}) \times 2$$
$$H_6(z) = (0.25 + 0.25z^{-1} + 0.00390625z^{-2} + 0.25z^{-3} + 0.25z^{-4}) \times 2$$

Polyphase decomposition is used in each stage similar to second stage.

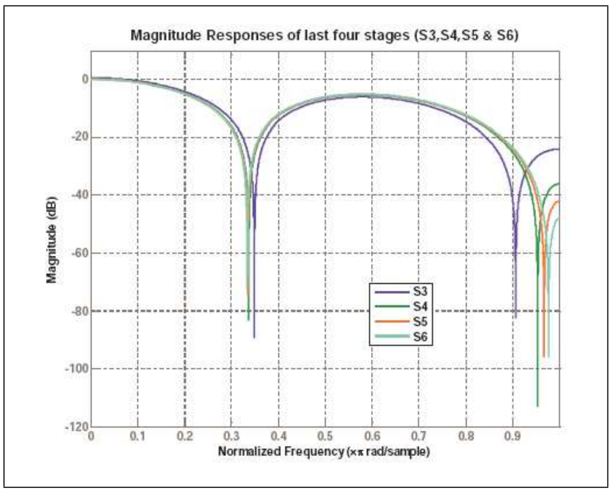


Figure 4.16: Magnitude response of last four stages

4.9 Interpolator design 1 - performance

Overall frequency response

Overall magnitude response of interpolator is shown in figure 4.17. It has a worst case image rejection of nearly 40dB.

The passband magnitude response is shown in figure 4.18.

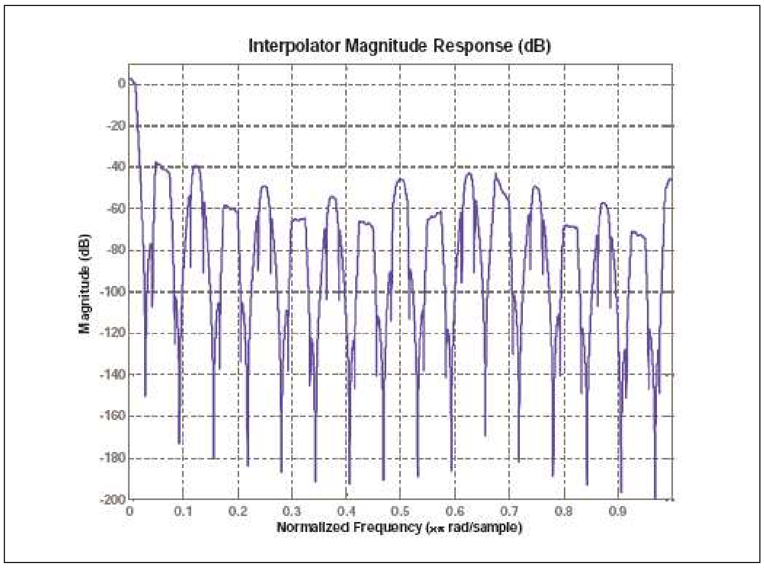


Figure 4.17: Magnitude response of interpolator

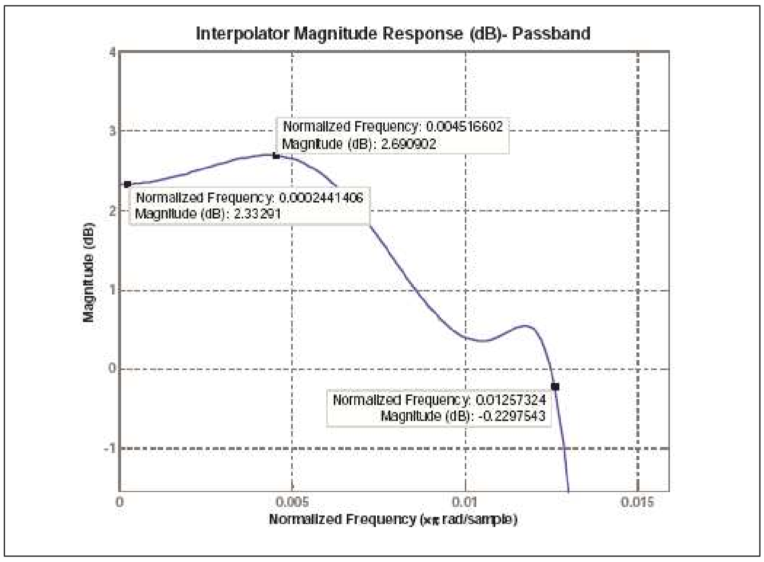


Figure 4.18: Passband performance

DC gain

The dc gain of the interpolator can be obtained by multiplying each stage dc gain. Each stage dc gain can be obtained by sum of coefficients.

$DCgain = 1.1274 \times 1.0625 \times 1.0625 \times 1.015625 \times 1.0078125 \times 1.00390625 = 1.3078$

So with a gain of 1.3, the interpolator maps +/-0.5 to +/-0.654 at the output.

Dynamic range of the interpolator

An uniform data path width of 18bits is maintained throughout the filter. The spectrum of the response to -2.5dBFS sinusoidal input with $f_{in} = 600Hz$ is shown in figure 4.19 and SIgnal-to-Noise-And-Distortion (SINAD)is 78.7dB.

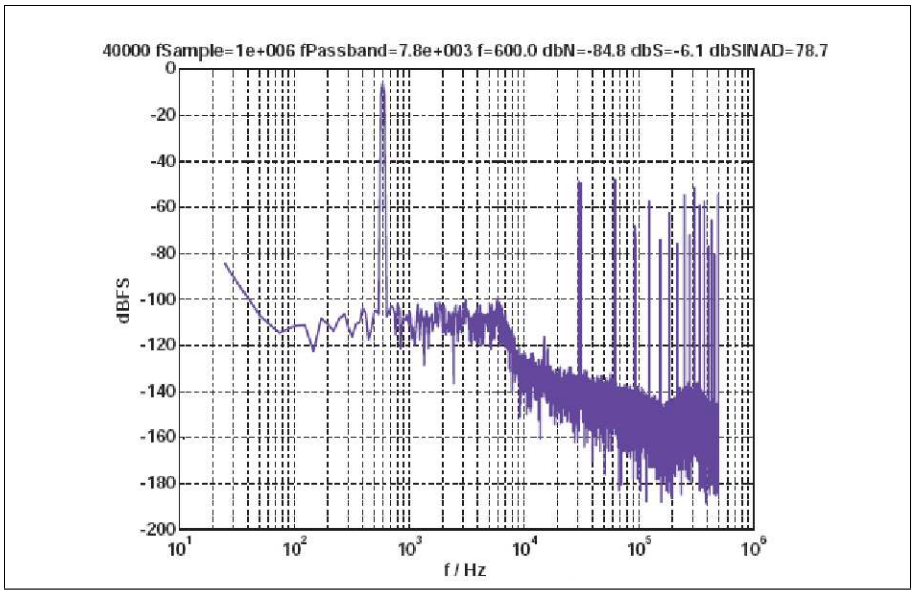


Figure 4.19: Spectrum of response to sinusoidal tone

4.10 Interpolator design 1 - summary

The dynamic range of interpolator is only 78.7dB (nearly 13bit ENOB). Output contains strong high frequency images as shown in figure 4.19. Passband gain the interpolator maps input +/-0.5 to +/-0.654 only. Finally, relaxed requirements of high frequency filters were not used efficiently.

On analysis through the chain it is found that poor SINAD performance is because of excessive scaling of the signal in IIR in first stage of interpolator. The improvements to design 1 to achieve the performance that meets the specification is discussed in next chapter.

CHAPTER 5

INTERPOLATOR DESIGN

In this chapter an area efficient and high performance interpolation filter design is discussed. The filter leverages several architectural and implementation features at various levels to achieve low computational complexity and performance. The features of this design are

- *Dynamic range(ENOB) of 16-bits*
- *Improved high frequency image rejection*
- *Area efficient implementation*

The area efficient implementation is achieved by *register sharing* and *substrcuture sharing* techniques.

5.1 Interpolator block diagram

The interpolator block diagram is shown in figure 5.1. The IIR4 of first design is used but with different implementation to improve the performance. High frequency rejection is improved by use of efficient filters in 3^{rd} and 4^{th} stages. The relaxed requirements of high frequency last stage filters is efficiently used to save hardware. Design of each stage is seperately discussed in following sections.

5.2 Concepts used in analysis

For fixed point implementation of the filters, dynamic range of signal at each node in the system is required [14] [1]. The dynamic range can be estimated from L1 norm analysis. These concepts are discussed in this section.

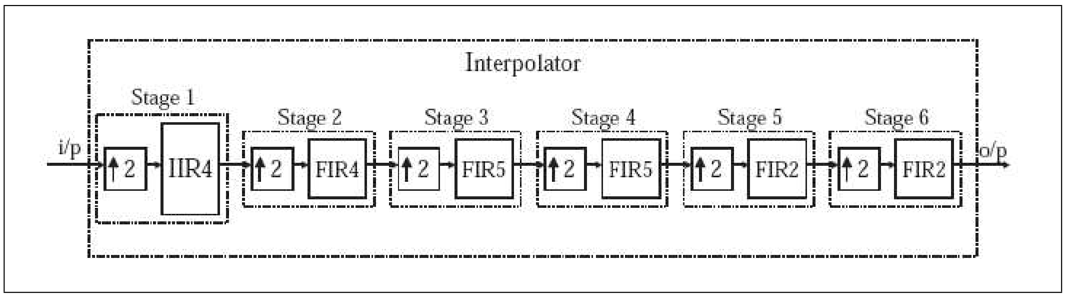


Figure 5.1: Interpolator design 2, block diagram

5.2.1 L1 Norm

For a discrete time Linear Time Invariant(LTI) system shown in figure **??**,

$$y[n] = h[n] \odot x[n] = \sum_{k=-\infty}^{\infty} h(k) \times x(n-k)$$

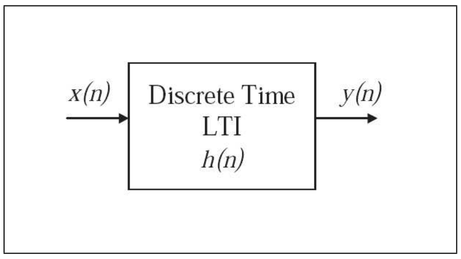


Figure 5.2: DT-LTI system

$$DCgain = \sum_{k=-\infty}^{\infty} h(k)$$

$$Max.\ value\ of\ output = L1\ norm = \sum_{k=-\infty}^{\infty} |h(k)|$$

The worst case input that produce this maximum value [7] [14] [10] is

$$x_{max}[n] = sign(h[D-n])$$

where D is chosen to be larger than impulse response period

5.2.2 Noise Power Gain(NPG)

For a stationary random noise input with variance σ_i, output noise power is given by [1] [14] [2]

$$\sigma_o^2 = \sigma_i^2 \sum_{k=-\infty}^{\infty} [h[k]]^2$$
$$\Rightarrow Noise\ Power\ Gain(NPG) = \sum_{k=-\infty}^{\infty} [h[k]]^2$$

5.2.3 L1 norm in interpolator

Based on theory presented in reference [13], interpolator with upsampler followed by a digital filter can be polyphase decomposed and L1 norm at output is maximum of L1 norm of the different paths. This is explained with M=2 example in figure 5.3.

$$DC\ gain = Avg.\left(\sum_{k=-\infty}^{\infty} h_e(k), \sum_{k=-\infty}^{\infty} h_o(k)\right)$$
$$L1\ norm = Max.\left(\sum_{k=-\infty}^{\infty} |h_e(k)|, \sum_{k=-\infty}^{\infty} |h_o(k)|\right)$$

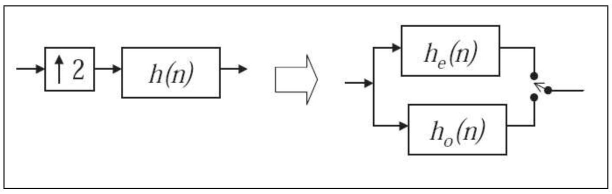


Figure 5.3: L1 norm in interpolator, M=2

These techniques are extensively for analysis in the design.

5.3 Interpolator stage 1(IIR4)

5.3.1 Noise analysis on IIR4

The IIR4 filter implementation of the two designs is shown in figure 5.4. In the first design, the signal is scaled down at the input and scaled up at the output to avoid overflow in adders but this downscaling at the input increased the round-off noise at IIR4 output. This increase in round-off noise degraded the overall interpolator performance.

The round-off noise is reduced in the second design by scaling up the signal as close as possible to the input and moving the downscaling block towards output.

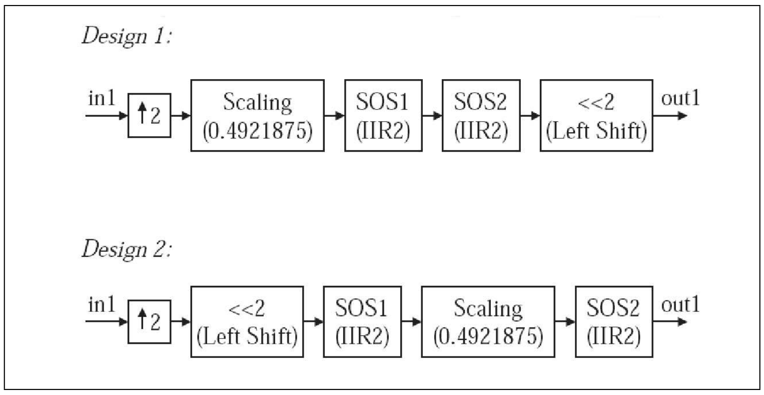


Figure 5.4: IIR4 implementations comparison

The round-off noise is quantified by calculating Noise Power Gain (NPG) from each adder output node to the IIR4 output in both the designs and are shown in table 5.1.

	Design 1	Design 2
Adder 1	56.873	3.4443
Adder 2	56.873	3.4443
Adder 3	56.873	3.4443
Adder 4	56.873	3.4443
Adder 5	56.873	3.4443
Adder 6	56.873	3.4443
Adder 7	0.2995	5.2389
Adder 8	20.956	5.2389
Adder 9	20.956	5.2389
Adder 10	20.956	5.2389
Adder 11	4	1
Adder 12	4	1
Adder 13	4	1
Total NPG	**416.4055**	**44.6214**

Table 5.1: IIR4 noise analysis

$$\frac{\sigma^2_{design1}}{\sigma^2_{design2}} = \frac{416.4055}{44.6214} = 9.33$$

The round-off noise at the output of second design is 9.33 times less than that of first design. The spectrum response to sinusoidal tone (-2.5dBFS at 600Hz) of IIR4 in second design implementation is shown in figure 5.5.

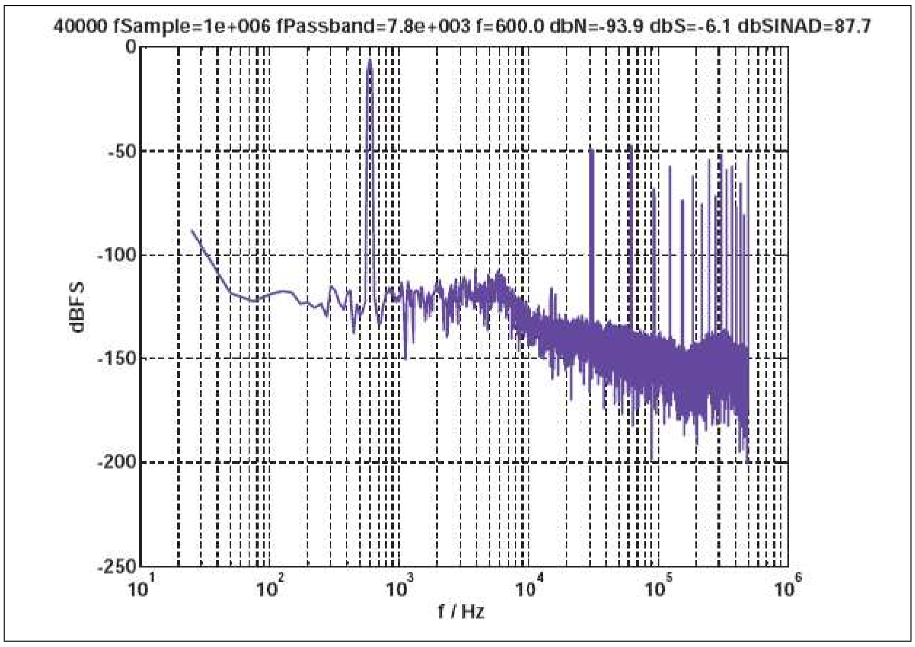


Figure 5.5: Spectrum of the response to sinusoid tone input

5.3.2 Overflow check - L1 norm analysis

It can be observed that SINAD is improved 9dB,(87.7dB compared to 78.7dB in first design). This scaling up of signal to reduce round-off noise increases the chances of overflow in the filter, it should be thoroughly checked to prevent overflow as it introduces lot of distortion. The possible overflow can be detected by calculating L1 norm at each node [1]. The L1 norm analysis results are shown in table 5.2.

The above L1 norm analysis shows that adder 13 has a chance of overflow. The worst case maximizing input pattern for the adder 13 output node and response at adder 13 output is shown in figure 5.6. So from the adder 13 output extra head room must be provided to prevent overflow but it is a costly solution from hardware point of view. As L1 norm at adder 13 output is slightly above one and overflow occurs with worst case input pattern amplitude greater than 1/1.1646 which is about 1.32dB below FS. In target application these high amplitude levels occur rarely, so a saturating adder is used. Distortion in signal is less with saturation compared to overflow. The saturating threshold of adder 13 should be chosen to avoid overflow in subsequent stages.

Adder	L1 norm
Adder 1	0.32684
Adder 2	0.54087
Adder 3	0.77566
Adder 4	0.79131
Adder 5	0.29559
Adder 6	0.78824
Adder 7	0.38796
Adder 8	0.70107
Adder 9	0.68050
Adder 10	0.51038
Adder 11	0.76557
Adder 12	0.97320
Adder 13	**1.16460**

Table 5.2: L1 norm analysis on IIR4

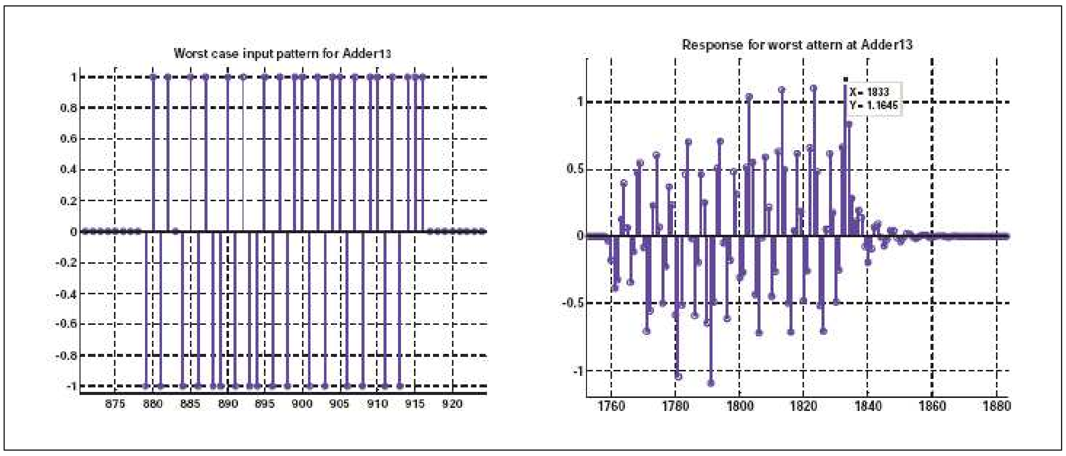


Figure 5.6: Response to worst case input pattern at adder 13 output

5.3.3 Improving IIR4 performance

The IIR4 filter output noise can be further reduced by increasing the adder widths. Based on NPG, the adder widths can be calculated. The increase in adder width reduces the noise source power at the node and so the output noise.It is found that adder widths required in IIR4 are 5bits more than effective resolution required at output.So, for 16bit ENOB 21bit adders are required. The performance of IIR4 with 21bit adders for sinusoidal test tone is shown in figure 5.7and output SINAD, Signal level and noise floor level is quoted in table 5.3 for different ENOB signals at input (-2.5dBFS at 600Hz). The above table shows that maximum output

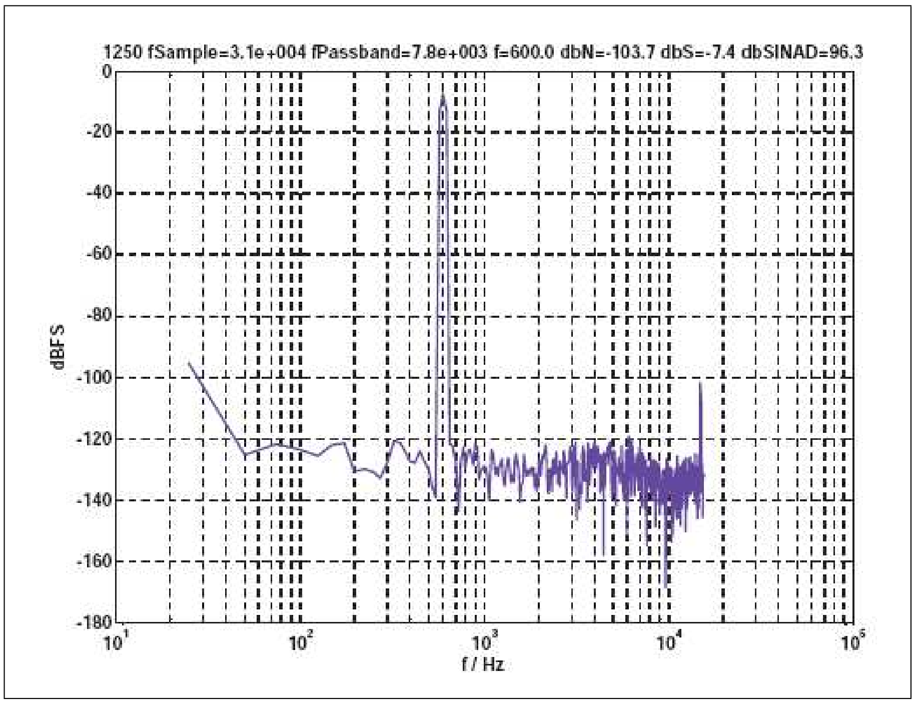


Figure 5.7: IIR4 with improved performance

Input ENOB	SINAD(dB)	Signal (dB)	Noise (dB)
16	94.3	-7.4	-101.7
17	96.3	-7.4	-103.7
18	96.8	-7.4	-104.2
19	96.8	-7.4	-104.2

Table 5.3: IIR4 performance test

SINAD is 96.8dB and it is limited by roundoff noise of IIR4.

5.4 Interpolator stage 2 (FIR4)

The stage 2 filter is same as that of first design but with slight change in implementation. The technique of register sharing between the polyphase paths is used as shown in figure 5.8. This saves one 18-bit register. A gain of 1.25 is added

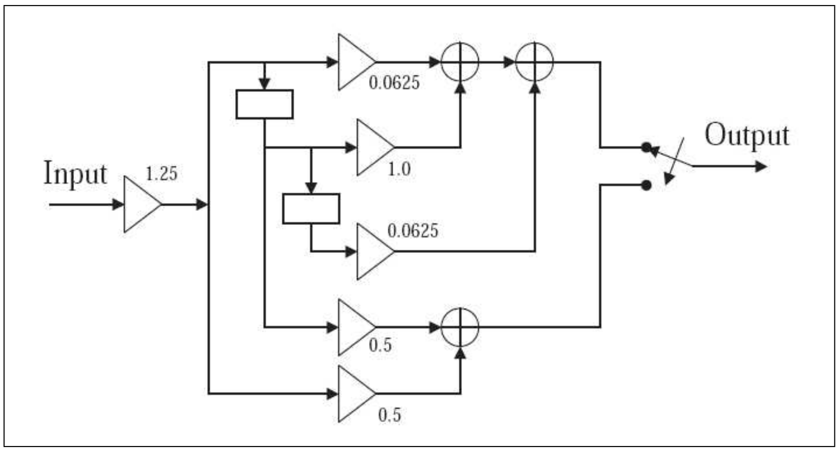


Figure 5.8: Stage 2 implementation with register sharing

at the input to increase the overall interpolator gain to 1.5. The advantage of introducing gain early in second stage is to reduce effect of round-off noise. The L1 norm analysis results of stage 2 are shown in table 5.4.

Adder	L1 norm
Adder 1	1.328125
Adder 2	1.40625
Adder 3	1.25

Table 5.4: Stage 2, L1 norm analysis

In order to avoid overflow in adder 1 & adder 2, the input signal should be limited to 1/1.40625 = 0.7111. Adder 13 of IIR4 should be saturated for +/- 0.7109375 (quantized threshold).

5.5 Interpolator stage 3 (FIR5)

5.5.1 Stage 3 filter design

The poor high frequency rejection in design 1 is because of lack of zero for the filter tranfer function at $\omega = \pi$. The design challenge is improving the image rejection without degrading the passband performance. The worst case image rejection target for the application is 55dB.

The image rejection is improved by introducing zeros at $\omega = \pi$ and $\omega = 0.5\pi$. As the sinc filter have a zeros at $\omega = \pi$, zero can be introduced by cascading sinc with FIR4 of design 1. The different filter alternatives for stage 3 filter are listed below.

$FIR4$: single zero at $\omega = 0.35\pi$ and $\omega = 0.9\pi$

$H_{3a}(z) = (0.25 + 0.25z^{-1} + 0.0625z^{-2} + 0.25z^{-3} + 0.25z^{-4}) \times 2$

$FIR4 + SINC^1$: single zero at $\omega = 0.35\pi$, $\omega = 0.9\pi$ and $\omega = \pi$

$H_{3b}(z) = (0.25 + 0.25z^{-1} + 0.0625z^{-2} + 0.25z^{-3} + 0.25z^{-4}) \times 2 \times (0.5 + 0.5z^{-1})$

$FIR4 + SINC^2$: single zero at $\omega = 0.35\pi$, $\omega = 0.9\pi$ and double zero at $\omega = \pi$

$H_{3c}(z) = (0.25 + 0.25z^{-1} + 0.0625z^{-2} + 0.25z^{-3} + 0.25z^{-4}) \times 2 \times (0.5 + 0.5z^{-1})^2$

$FIR3$: single zero at $\omega = 0.5\pi$ and $\omega = \pi$

$H_{3d}(z) = (0.25 + 0.25z^{-1} + 0.25z^{-2} + 0.25z^{-3}) \times 2$

$FIR4A$: single zero at $\omega = 0.5\pi$ and double zero $\omega = \pi$

$H_{3e}(z) = (0.125 + 0.25z^{-1} + 0.25z^{-2} + 0.25z^{-3} + 0.125z^{-4}) \times 2$

$FIR5$: single zero at $\omega = 0.5\pi$ and three zero $\omega = \pi$

$H_{3f}(z) = (0.0625 + 0.1875z^{-1} + 0.25z^{-2} + 0.25z^{-3} + 0.1875z^{-4} + 0.0625z^{-5}) \times 2$

$FIR6$: single zero at $\omega = 0.5\pi$ and four zero $\omega = \pi$

$H_{3g}(z) =$

$(0.03125 + 0.125z^{-1} + 0.21875z^{-2} + 0.25z^{-3} + 0.21875z^{-4} + 0.125z^{-5} + 0.03125z^{-6}) \times 2$

$FIR6A$: double zero at $\omega = 0.5\pi$ and double zero $\omega = \pi$

$H_{3h}(z) =$

$(0.0625 + 0.125z^{-1} + 0.1875z^{-2} + 0.25z^{-3} + 0.1875z^{-4} + 0.125z^{-5} + 0.0625z^{-6}) \times 2$

The comparison of filters with respect to their passband droop and image rejection is shown in table 5.5. Of all the seven alternatives, FIR5 gives sufficient

Filter	Passband droop(dB)	Image rejection(dB)
$FIR4$	0.253	24.4
$FIR4 + SINC^1$	0.28	49.25
$FIR4 + SINC^2$	0.31	69.8
$FIR3$	0.134	23.14
$FIR4A$	0.1607	43.96
$FIR5$	0.1894	66.22
$FIR6$	0.215	88.16
$FIR6A$	0.2706	43.96

Table 5.5: Stage 3 alternatives comparison

image rejection with better passband flatness compared to others.

5.5.2 Stage 3 filter implementation

An efficient realization is obtained through the techniques of polyphase decomposition, register sharing and substructure sharing. The substructure sharing is achieved based on equations shown below

$$H_3(z) = 0.125 + 0.375z^{-1} + 0.5z^{-2} + 0.5z^{-3} + 0.375z^{-4} + 0.125z^{-5}$$
$$= (0.125 + 0.5z^{-2} + 0.375z^{-4}) + z^{-1}(0.375 + 0.5z^{-2} + 0.125z^{-4})$$
$$=$$
$$(0.125 + 0.5z^{-2} + 0.125z^{-4}) + z^{-1}(0.125 + 0.5z^{-2} + 0.125z^{-4}) + 0.25z^{-4} + 0.25z^{-1}$$

The structure of the filter is shown in figure 5.9 and the detailed hardware implementation is shown in figure 5.10. This implementation requires one extra adder

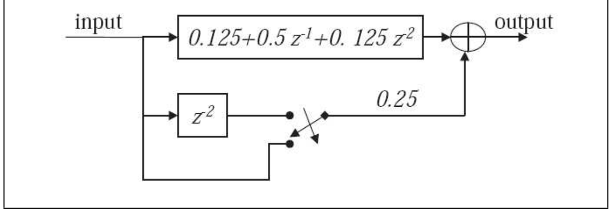


Figure 5.9: Stage 3 filter structure

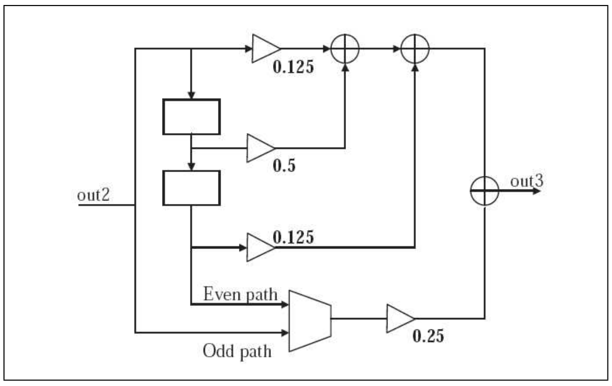


Figure 5.10: Stage 3 filter hardware implementation

operating at output frequency compared to FIR4 of design 1.

5.6 Interpolator Stage 4 (FIR5)

The FIR4 used in first design lacks zero at $\omega = \pi$ and its rejection is poor. The same FIR5 filter designed for stage 3 is used. This further increases the stage 3 image rejection because of a zero at mid-band. The overall magnitude response of first four stages of interpolator is shown in figure 5.11

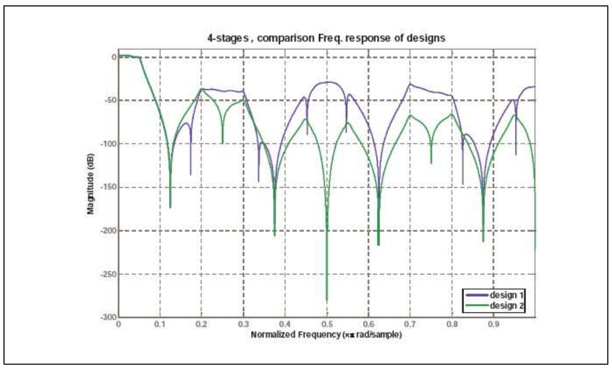


Figure 5.11: Magnitude response of 4-stages

5.7 Interpolator Stage 5 & Stage 6(FIR2)

5.7.1 Design

Requirements of high frequency stages are relaxed as band of interest is a small fraction of sampling frequency. The sinc filters can be particularly used at last stages. This not only improves image rejection but the passband flatness is also better than FIR4. So, $sinc^2$ filters are used in last two stages of the interpolator.

$$H_5(z) = (0.5 + 0.5z^{-1})^2$$
$$H_6(z) = (0.5 + 0.5z^{-1})^2$$

The magnitude response of $sinc^2$ is shown in figure 5.12 along with FIR4 of design 1. It can be seen that $sinc^2$(FIR2) filter better than FIR4 in all aspects.

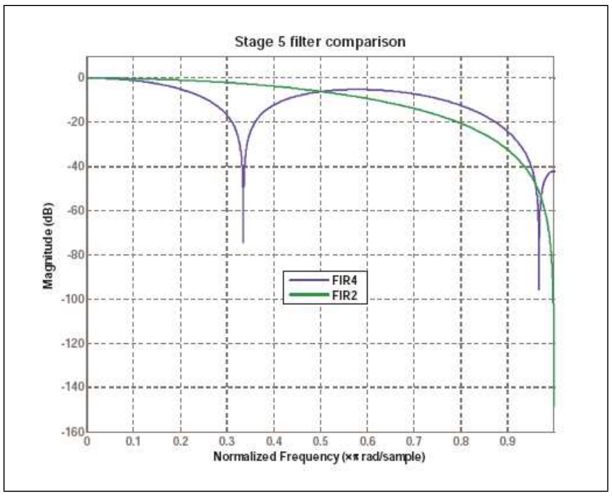


Figure 5.12: Magnitude response of stage 5

5.7.2 Implementation

Polyphase decomposed implementation is used and it is shown in figure 5.13 This polyphase implementation just need one adder, one register and a multiplexer for each stage.

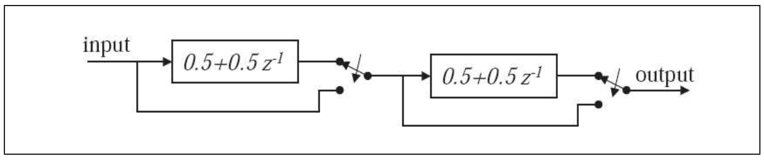


Figure 5.13: Implementation of stage 5

5.8 Interpolator design 2 - performance

5.8.1 Magnitude response

The magnitude response plot of interpolator design 2 is shown in figure 5.14 along with design 1 plot. It can be seen that image rejection at high frequencies is improved compared to design 1.

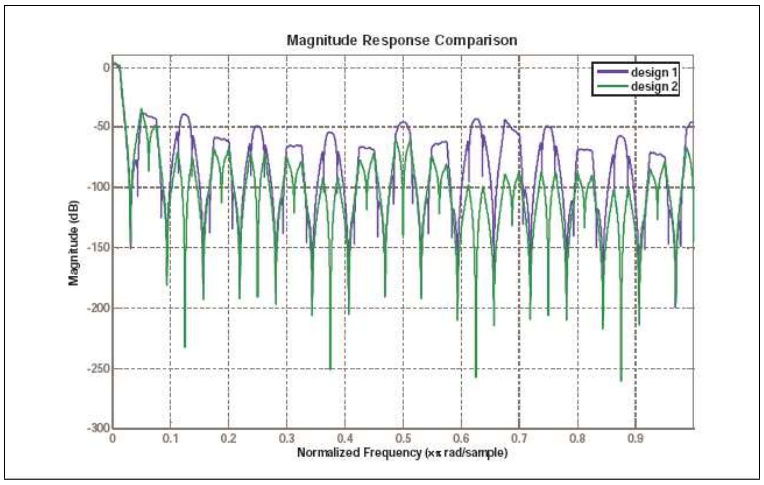


Figure 5.14: Magnitude response of interpolator

5.8.2 Passband performance

The passband response of two designs are plotted in figure 5.15. The DC gain of the interpolator has been improved to 3.511dB (1.5) from 2.33dB (1.3).

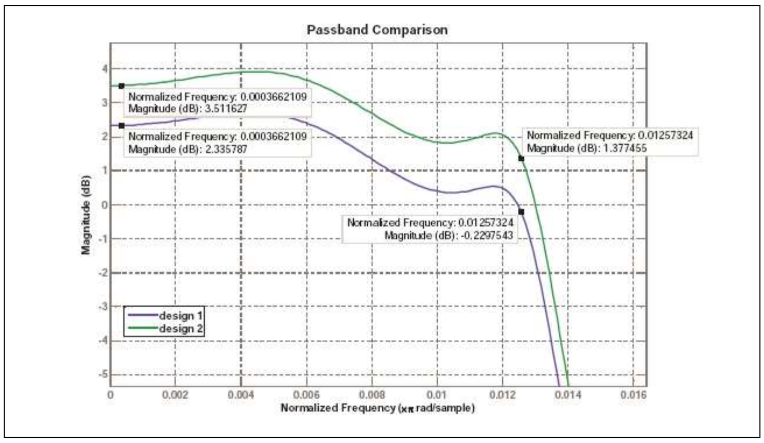


Figure 5.15: Passband performance

5.8.3 SINAD performance

The spectrum response to sinusoidal input of fullscale amplitude at frequency of 600Hz is shown in figure 5.16. The overall SINAD is 97.2dB at FS output. The

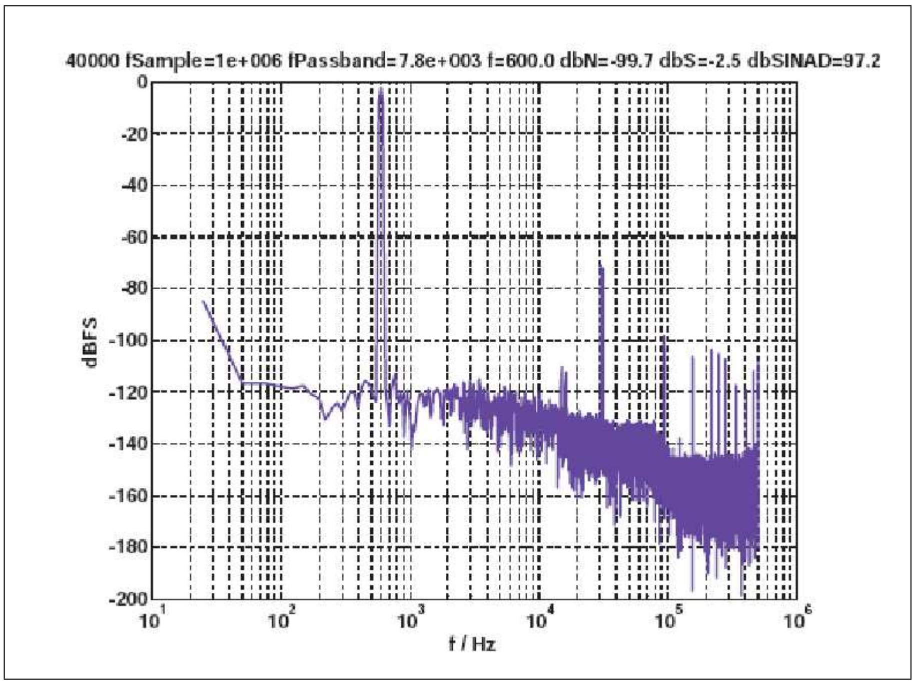
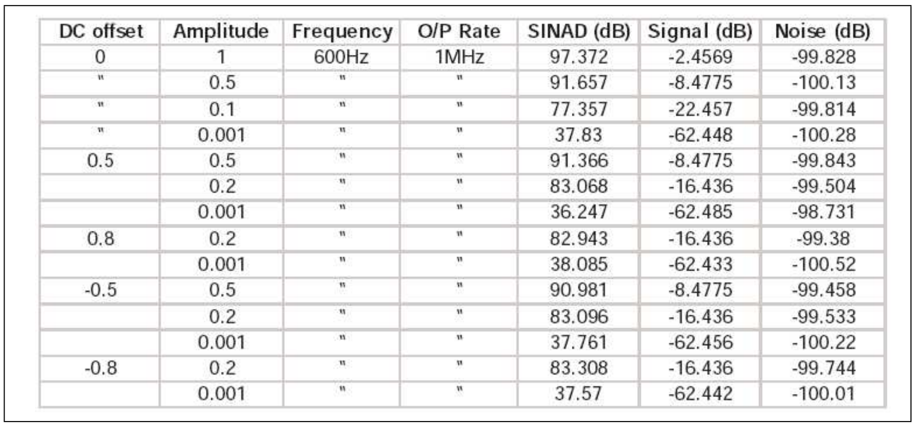


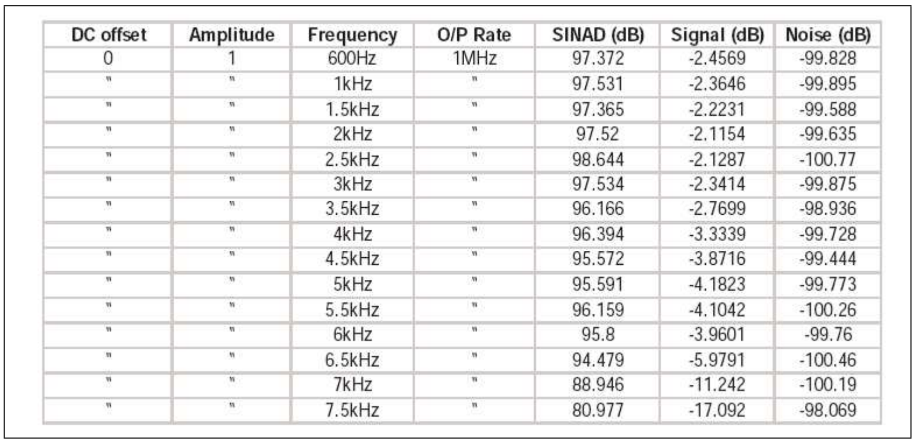
Figure 5.16: Spectrum of response to sinusoidal input

interpolator is extensively tested with several input cases and results are tabulated in tables 5.17, 5.18 and 5.19.



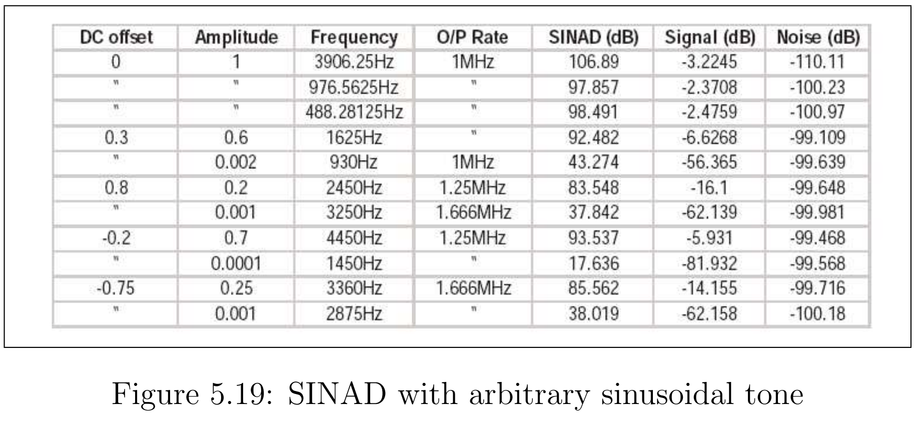
DC offset	Amplitude	Frequency	O/P Rate	SINAD (dB)	Signal (dB)	Noise (dB)
0	1	600Hz	1MHz	97.372	-2.4569	-99.828
"	0.5	"	"	91.657	-8.4775	-100.13
"	0.1	"	"	77.357	-22.457	-99.814
"	0.001	"	"	37.83	-62.448	-100.28
0.5	0.5	"	"	91.366	-8.4775	-99.843
	0.2	"	"	83.068	-16.436	-99.504
	0.001	"	"	36.247	-62.485	-98.731
0.8	0.2	"	"	82.943	-16.436	-99.38
	0.001	"	"	38.085	-62.433	-100.52
-0.5	0.5	"	"	90.981	-8.4775	-99.458
	0.2	"	"	83.096	-16.436	-99.533
	0.001	"	"	37.761	-62.456	-100.22
-0.8	0.2	"	"	83.308	-16.436	-99.744
	0.001	"	"	37.57	-62.442	-100.01

Figure 5.17: SINAD vs. Amplitude



DC offset	Amplitude	Frequency	O/P Rate	SINAD (dB)	Signal (dB)	Noise (dB)
0	1	600Hz	1MHz	97.372	-2.4569	-99.828
"	"	1kHz	"	97.531	-2.3646	-99.895
"	"	1.5kHz	"	97.365	-2.2231	-99.588
"	"	2kHz	"	97.52	-2.1154	-99.635
"	"	2.5kHz	"	98.644	-2.1287	-100.77
"	"	3kHz	"	97.534	-2.3414	-99.875
"	"	3.5kHz	"	96.166	-2.7699	-98.936
"	"	4kHz	"	96.394	-3.3339	-99.728
"	"	4.5kHz	"	95.572	-3.8716	-99.444
"	"	5kHz	"	95.591	-4.1823	-99.773
"	"	5.5kHz	"	96.159	-4.1042	-100.26
"	"	6kHz	"	95.8	-3.9601	-99.76
"	"	6.5kHz	"	94.479	-5.9791	-100.46
"	"	7kHz	"	88.946	-11.242	-100.19
"	"	7.5kHz	"	80.977	-17.092	-98.069

Figure 5.18: SINAD vs. Frequency



DC offset	Amplitude	Frequency	O/P Rate	SINAD (dB)	Signal (dB)	Noise (dB)
0	1	3906.25Hz	1MHz	106.89	-3.2245	-110.11
"	"	976.5625Hz	"	97.857	-2.3708	-100.23
"	"	488.28125Hz	"	98.491	-2.4759	-100.97
0.3	0.6	1625Hz	"	92.482	-6.6268	-99.109
"	0.002	930Hz	1MHz	43.274	-56.365	-99.639
0.8	0.2	2450Hz	1.25MHz	83.548	-16.1	-99.648
"	0.001	3250Hz	1.666MHz	37.842	-62.139	-99.981
-0.2	0.7	4450Hz	1.25MHz	93.537	-5.931	-99.468
"	0.0001	1450Hz	"	17.636	-81.932	-99.568
-0.75	0.25	3360Hz	1.666MHz	85.562	-14.155	-99.716
"	0.001	2875Hz	"	38.019	-62.158	-100.18

Figure 5.19: SINAD with arbitrary sinusoidal tone

5.8.4 Step response

The positive maximum and negative maximum step response plots are shown in figure 5.20 and 5.21.

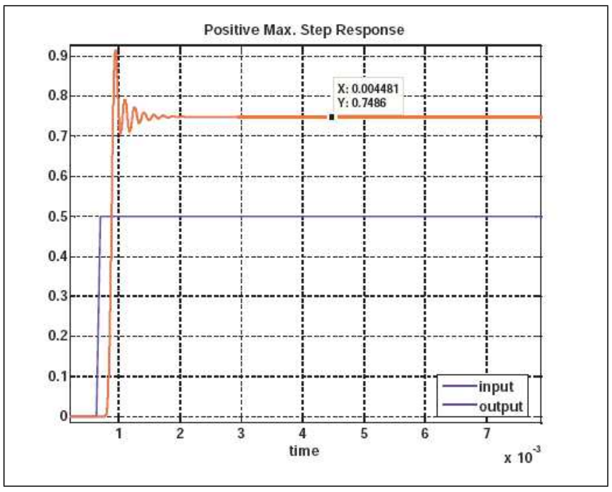


Figure 5.20: Positive max. step response

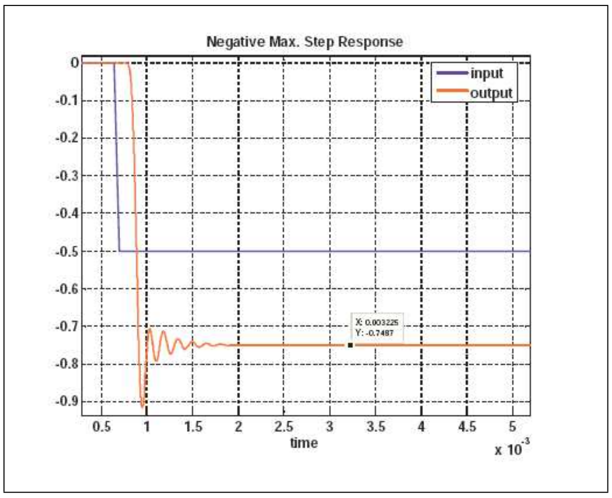
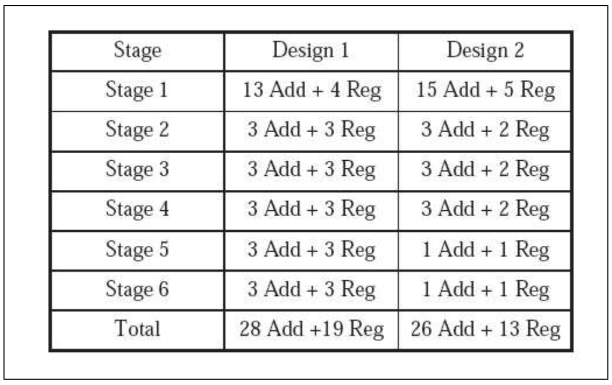


Figure 5.21: Negative max. step response

5.9 Hardware comparison

The hardware requirements of the two discussed designs are compared in table 5.22.



Stage	Design 1	Design 2
Stage 1	13 Add + 4 Reg	15 Add + 5 Reg
Stage 2	3 Add + 3 Reg	3 Add + 2 Reg
Stage 3	3 Add + 3 Reg	3 Add + 2 Reg
Stage 4	3 Add + 3 Reg	3 Add + 2 Reg
Stage 5	3 Add + 3 Reg	1 Add + 1 Reg
Stage 6	3 Add + 3 Reg	1 Add + 1 Reg
Total	28 Add +19 Reg	26 Add + 13 Reg

Figure 5.22: Hardware comparison

So, it can be seen that second design occupies less area but gives better performance compared to first design. Second design saves *six* 18-bit registers and *two* 18-bit adders.

CHAPTER 6

DYNAMIC ELEMENT MATCHING(DEM) ALGORITHMS

In this chapter a novel class of dynamic element matching algorithms for improving the linearity of DACs in multi-bit oversampling $\Delta\Sigma$ data converters are discussed. The hardware requirements and performance of Randomization, Clocked Averaging and Data Weighted Averaging algorithms are estimated and compared.

6.1 Introduction

Linearity of a DAC depends on the matching of its elements. High linearity requires precise component matching. But the fabrication processes can only guarantee up to a certain extent of matching. Alternative approaches like calibration, DEM are used to improve the linearity with modest element matching [18]. DEM is a simple and cost effective solution. It shuffles the DAC unit elements to obtain fairly precise *average* output levels. A 2 element DAC is shown in figure [20].

DEM converts static DAC errors to high frequency noise

6.2 Role of DEM in $\Delta\Sigma$ converters

In $\Delta\Sigma$ DAC, a low-bit-resolution D/A is used at the output as discussed in section 4.4.

A DAC is used in the feedback path of analog $\Delta\Sigma$ modulator, so the nonlinearity in DAC effects the output performance [19] in both the cases.

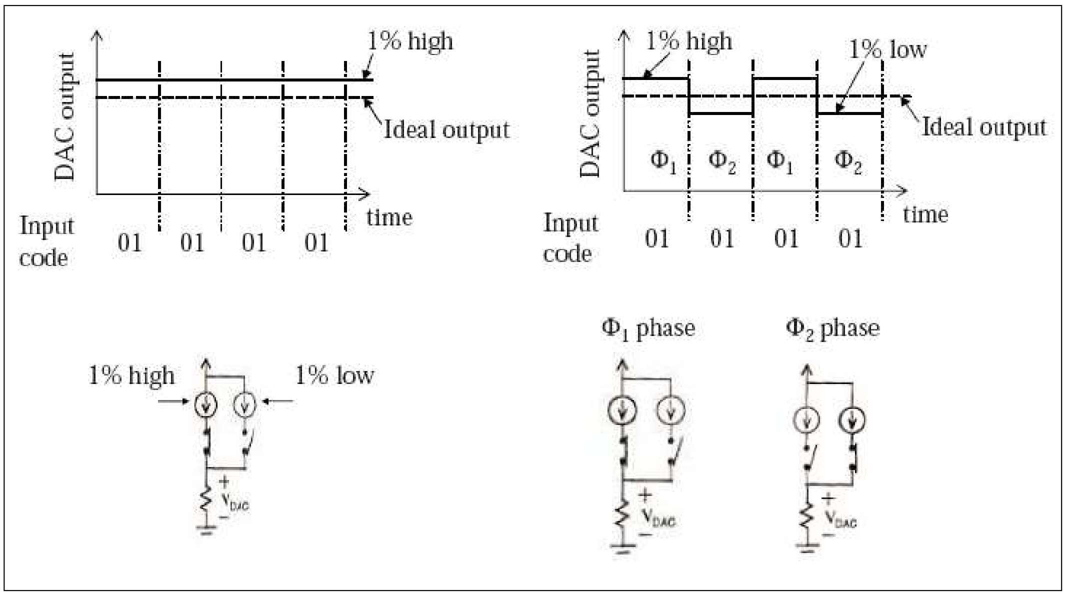


Figure 6.1: DEM for DAC with 2 elements

6.3 Topology of DAC with DEM

A topology of DAC empolying DEM to improve linearity is shown in figure 6.2. The DEM block in between encoder and thermometer coded DAC dynmically the change the element usage. This selection process significantly reduces the in-band distortion.

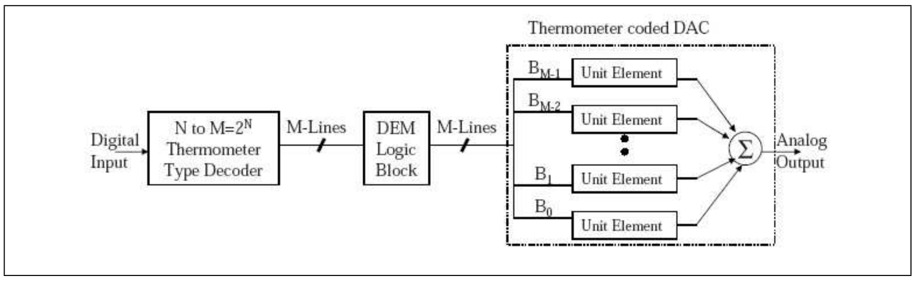


Figure 6.2: Topology of DAC with DEM

The DEM logic block determines the type of DEM algorithm used.

6.4 DEM Techniques

The DEM techniques can be broadly classified as data independent selection algorithms(randomization, clocked averging and data dependent selection algorithms(individual level averaging, data weighted averaging).

6.4.1 Carley Randomization

In this algorithm [4], the DAC elements are randomly chosen with number of elements selected equal to input level. The DAC mismatch error is converted into white noise. As only a small portion of noise falls into baseband, system linearity is improved. The implementation of this technique proposed in reference [4] consists of an aribtrary switch connection block and a random sequence generator (like LFSR) as shown in figure 6.3. The aribtrary connection block is implemented with butterfly switches as shown in figure 6.4. This butterfly connection ensures that there is path from every input to every other output.

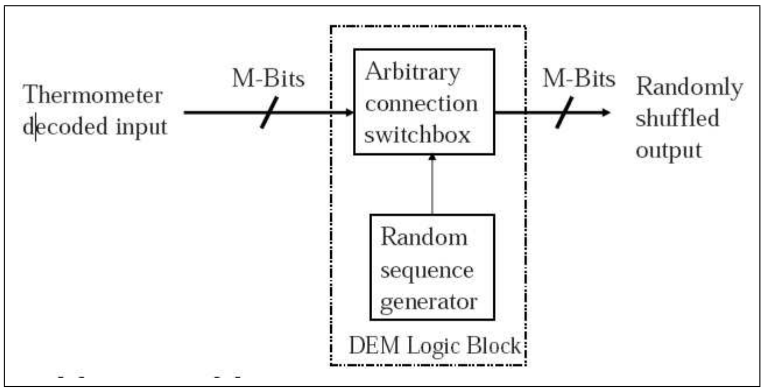

Thermometer decoded input
M-Bits
Arbitrary connection switchbox
M-Bits
Randomly shuffled output
Random sequence generator
DEM Logic Block

Figure 6.3: Randomization implementation

6.4.2 Clocked Averaging(CLA)

The amount of hardware can be reduced by operating each stage switches with a clocks(f_{clk}) as shown in 6.5 [19]. Periodically permuting the DAC elements at given clock frequency(f_{clk}), moves the distortion components into regions around multiples of clock frequency (f_{clk}). For more detailed analysis reader can refer

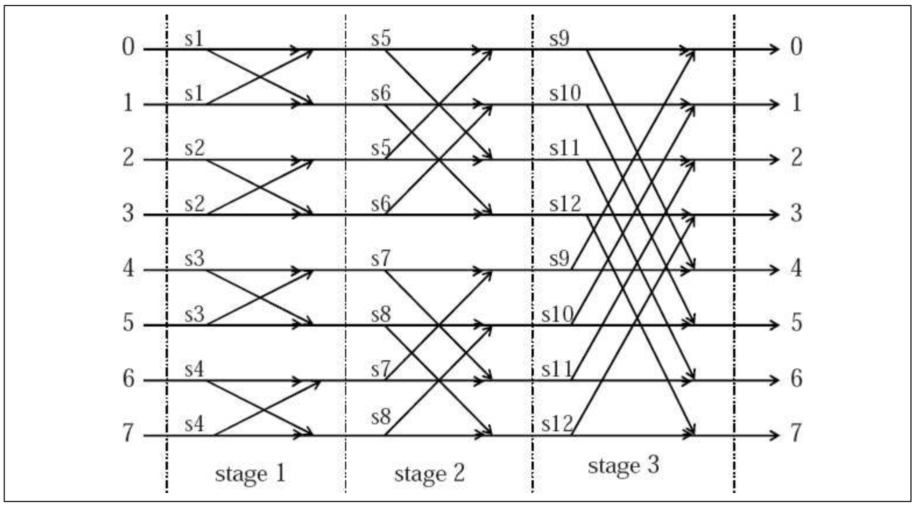


Figure 6.4: Arbitrary switch connection diagram

to the reference [19]. The main disadvantage of CLA is that it can interact with

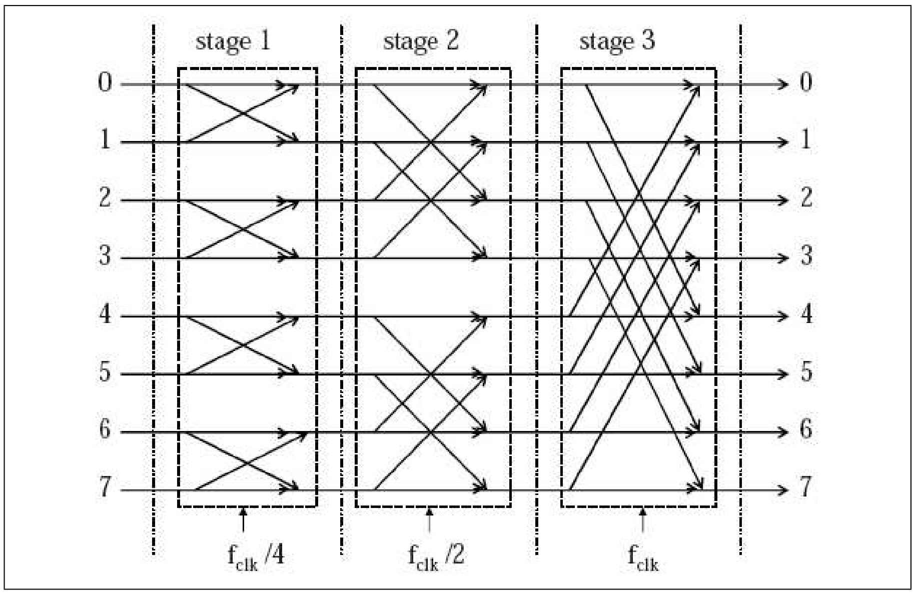


Figure 6.5: CLA implementation

input and may not be effective in moving the DAC error with some inputs.

6.4.3 Individual Leval Averaging (ILA)

The individual level averaging algorithm elements are selected based on input level. It selects the elements by remembering every state of every level but this requires larger and more complex hardware. This is not implemented in this work.

Because of data dependent element selection, the frequency response of DAC error exhibits high pass response.

6.4.4 Data Weighted Averaging (DWA)

The elements are cyclically selected based on the input level as shown in figure 6.6 [20] [21]. The element averaging controlled by the input data sequence, so it is

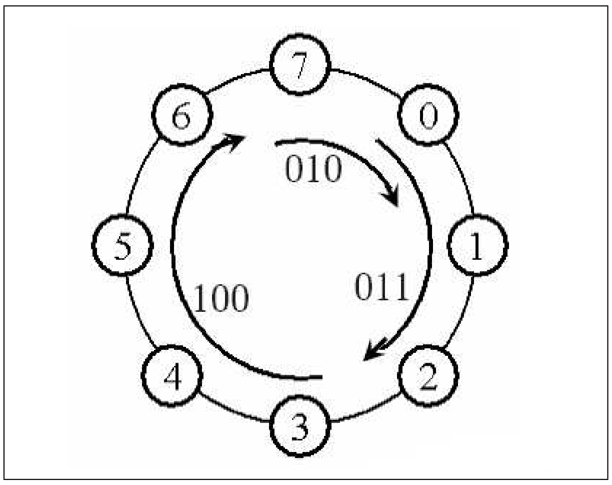


Figure 6.6: DWA operation

referred to as DWA. Using elements at maximum possible rate ensures that DAC errors will quickly sum to zero, moving distortion components to high frequencies. Due to cyclic sequencing no element of the DAC is selected an inordinate number of times, even in a short time interval. It preserves the noise shaping of the modulator. The DWA implementation proposed in reference [21] is shown in figure 6.7.

6.5 Performance comparison

To compare the performance of above discussed algorithms, a third order $\Delta\Sigma$ ADC with 3bit quantizer is taken as an example. The in-band spectrum of output of ADC with ideal DAC elements and non-ideal elements is shown in figure **??**. *Harmonic distortion is observed with errors in DAC*

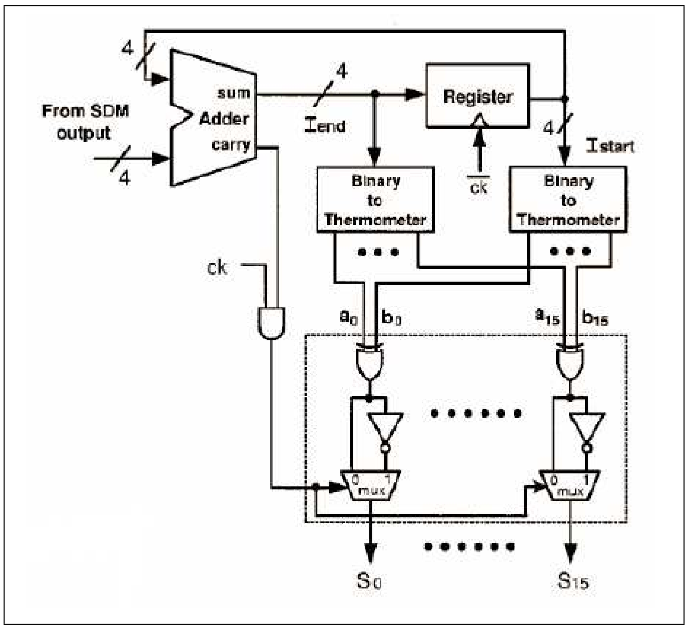


Figure 6.7: DWA implementation

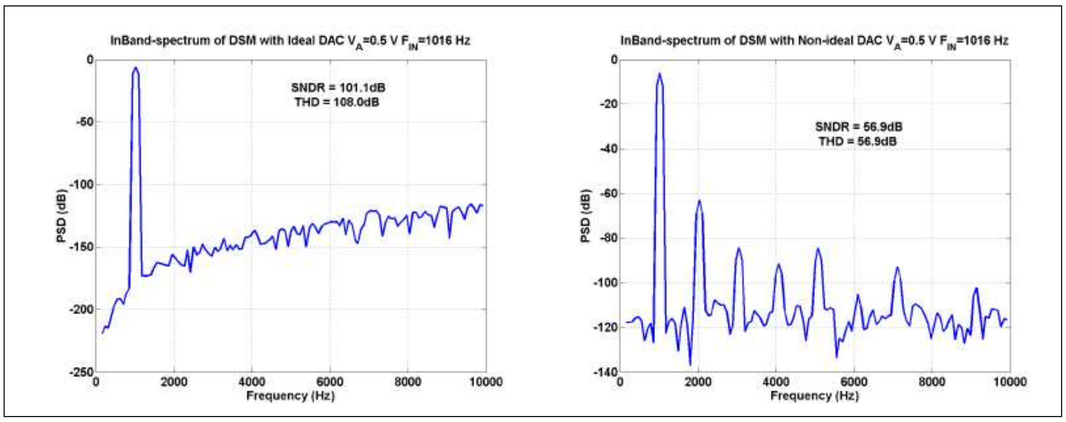


Figure 6.8: In-band spectrum with ideal and non-ideal DAC

Randomization

The in-band output spectrum and error spectrum are shown in figure 6.9.

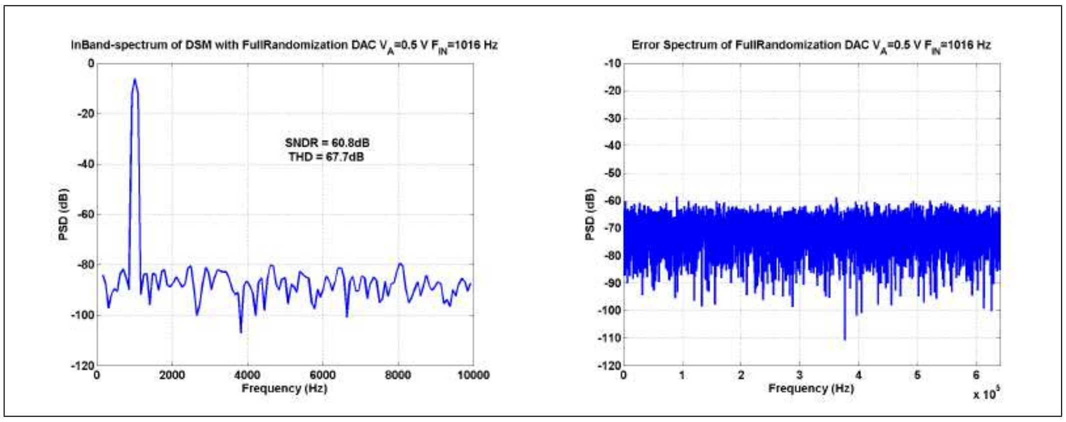


Figure 6.9: In-band output spectrum & error spectrum

Randomization converted DAC errors to white noise

Clocked Averaging (CLA)

The in-band output spectrum and error spectrum are shown in figure 6.10.

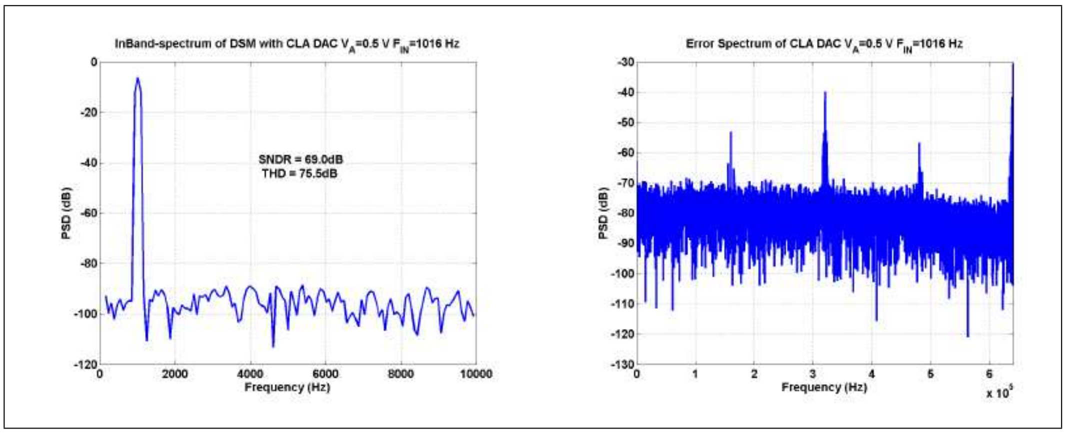


Figure 6.10: In-band output spectrum & error spectrum with CLA

Distortion components are translated to submultiples of clock frequency.

Individual Level Averaging(ILA)

The in-band output spectrum and error spectrum are shown in figure 6.11.

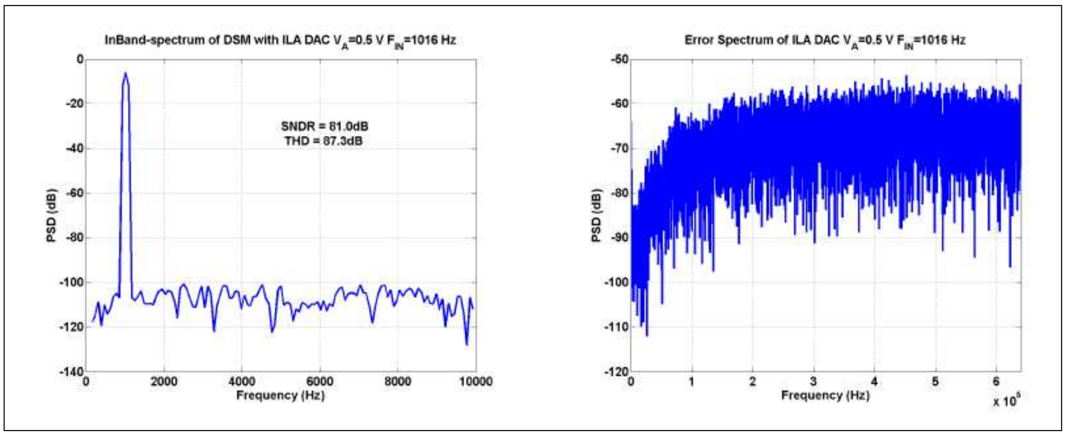


Figure 6.11: In-band output spectrum & error spectrum with ILA

DAC error noise has been pushed to high frequencies.

Data Weighted Averaging(DWA)

The in-band output spectrum and error spectrum are shown in figure 6.12.

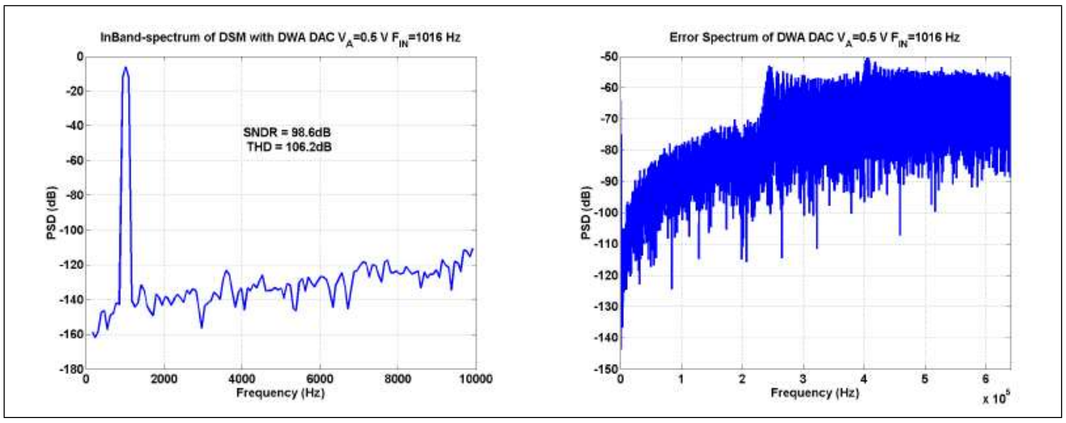


Figure 6.12: In-band output spectrum & error spectrum with DWA

DWA is more effective in pushing DAC errors to high frequencies.

6.6 Hardware complexity comparison

The above discussed implementations for randomization, CLA and DWA are coded in RTL and synthesized for different number of DAC elements(4,8,16,32,64). The hardware requirements are tabulated in tables 6.1, 6.2 and 6.3.

	$B_{in} = 4$	$B_{in} = 8$	$B_{in} = 16$	$B_{in} = 32$	$B_{in} = 64$
Area(μm^2)	288	857	2210	5491	13118
Avg. Current (μA)	0.771	1.54	2.06	6.48	14.9
Delay (ps)	1064	1512	1939	2392	2886

Table 6.1: Randomization hardware with differnet input widths

	$B_{in} = 4$	$B_{in} = 8$	$B_{in} = 16$	$B_{in} = 32$	$B_{in} = 64$
Area(μm^2)	158	432	1068	2736	6371
Avg. Current (μA)	0.48	0.981	1.76	7.12	23.6
Delay (ps)	1967	2441	2905	5727	5623

Table 6.2: CLA hardware with differnet input widths

From the above results it is clear that DWA is hardware efficient and gives better performance compared to others.

	$B_{in} = 4$	$B_{in} = 8$	$B_{in} = 16$	$B_{in} = 32$	$B_{in} = 64$
Area(μm^2)	242	510	919	1783	3580
Avg. Current (μA)	0.88	1.39	2.72	7.45	12.9
Delay (ps)	2625	3481	4710	6848	8541

Table 6.3: DWA hardware with differnet input widths

CHAPTER 7

CONCLUSION

Three popular implementations for fourth-order *sinc* decimation filters were compared in terms of power dissipation and area. It was observed that the CIC implementation consumes least power and area for decimation ratios of 16 and above compared to other two (FIR and Polyphase decomposed FIR). The FIR and its polyphase derivative are efficient at lower oversampling ratios. Also it was observed that pipelining in this multirate filter reduces the power consumption significantly.

A high performance and area efficient interpolator was designed for an oversampling $\Delta\Sigma$ DAC. The multi-rate, multi-stage design reduced the hardware in last stages. Polyphase decomposition, register sharing and sub-structure sharing techniques were used to obtain an efficient implementation.

Finally, the performance of four different Dynamic Element Matching(DEM) algorithms were compared. The hardware requirements of randomization, clocked averaging(CLA) and data weighted averaging(DWA) were estimated.

REFERENCES

[1] **Alan V.Oppenheim, R. W.** and **J. R.Buck**, *Discrete-time Signal Processing*. Pearson Education, 2004. ISBN 81-7808-244-6.

[2] **Burrus, S. C. . C. S.**, Roundoff noise in multirate digital filters. *Circuits Systems Signal Process*, volume 3. 1984.

[3] **Candy, J. C.**, Decimation for sigma delta modulation. *IEEE Transactions on Communications*, volume 34. IEEE, 1986.

[4] **Carley, L. R.**, A noise-shaping coder topology for 15+ bit converters. *IEEE Journal of Solid-State Circuits*, volume 24. 1989.

[5] **E.Dijkstra, C., O.Nys** and **M.Degrauwe**, On the use of modulo arithmetic comb filters in sigma delta modulators. *IEEE Proc. ICASSP'88*. 1988.

[6] **H.Aboushady**, Efficient polyphase decomposition of comb decimation filters in $\delta\sigma$ analog-to-digital converters. *IEEE Transactions on Circuits and Systems - II*, volume 48. IEEE, 2001.

[7] **Harris, F. J.**, *Multirate Signal Processing for Communication Systems*. Prentice Hall, 2004. ISBN 0-13-146511-2.

[8] **Hogenauer, E. B.**, An economical class of digital filters for deciamtion and interpolation. *IEEE Transactions on ASSP*, volume 29. IEEE, 1981.

[9] **J.Goodman, D.** and **M. J. Carey**, Nine digital filters for decimation and interpolation. *IEEE Transactions on ASSP*, volume 29. IEEE, 1977.

[10] **Joan Carletta, F. K. . Z. F., Robert Veillette**, Determining appropriate precisions of signals in fixed-point iir filters.

[11] **Johns, D.** and **K. Martin**, *Analog Integrated Circuit Design*. John Wiley & Sons, 2004. ISBN 9814-12-647-0.

[12] **L.Ascari, A.** and **C. Morandi**, Low power implementation of a sigma delta decimation filter for cardiac applications. *IEEE Instrumentation and Measurement Technology Conference*. IEEE, 2001.

[13] **Olsson, M.**, Scaling of multistage interpolators.

[14] **Parhi, K. K.**, *VLSI Digital Signal Processing Systems*. Wiley, 1986. ISBN 9812-53-023-1.

[15] **P.Brandt, B.** and **B. A. Wooley**, A low-power, area-efficient digital filter for decimation and interpolation. *IEEE Journal of Solid State Circuits*, volume 29. IEEE, 1994.

[16] **P.P.Vaidyanathan**, *Multirate Systems and Filter Banks*. Pearson Education, 2004. ISBN 81-297-0685-7.

[17] **R.E.Crochiere** and **L. R. Rabiner**, Optimum fir digital fitler implementations for decimation, interpolation and narrow-band filtering. *IEEE Transactions on ASSP*, volume 23. IEEE, 1975.

[18] **Steven R. Norsworthy, R.** and **G.C.Temes**, *Delta-Sigma Data Converters - Theory, Design and Simulation*. IEEE Press, 1997. ISBN 0-7803-1045-4.

[19] **Sutarja, B. H. . S.**, Multibit $\delta\sigma$ a/d converter incorporating a novel class of dynamic element matching techniques. *IEEE Transactions on Circuits and Systems-II*, volume 39. 1992.

[20] **T.Baird, R.** and **T. S. Fiez**, Linearity enhancement of multibit $\delta\sigma$ a/d and d/a converters using data weighted averaging. *IEEE Transactions on Circuits and Systems-II*, volume 42. 1995.

[21] **T.H.Kuo, K.** and **H.R.Yeng**, A wideband cmos sigmadelta modulator with incremental data weighted averaging. *IEEE Journal of Solid-State Circuits*, volume 24. 2002.

[22] **T.Saramaki** and **H.Tenhunen**, Efficient vlsi-realizable deciamtors for sigma-delta analog-to-digital converters. *IEEE Proc. ISCAS'88*. 1988.