ISTANBUL TECHNICAL UNIVERSITY ★ GRADUATE SCHOOL OF SCIENCE ENGINEERING AND TECHNOLOGY

A 14-BIT 100-kHZ CONTINUOUS-TIME DELTA-SIGMA ANALOG-TO-DIGITAL CONVERTER FOR HALL EFFECT BASED CURRENT SENSOR APPLICATION

M.Sc. THESIS

Alper GİRGİN

Department of Electronics and Communication Engineering

Electronics Engineering Programme

December 2017

ISTANBUL TECHNICAL UNIVERSITY ★ GRADUATE SCHOOL OF SCIENCE ENGINEERING AND TECHNOLOGY

A 14-BIT 100-kHZ CONTINUOUS-TIME DELTA-SIGMA ANALOG-TO-DIGITAL CONVERTER FOR HALL EFFECT BASED CURRENT SENSOR APPLICATION

M.Sc. THESIS

Alper GIRGIN
(504141202)

Department of Electronics and Communication Engineering

Electronics Engineering Programme

Thesis Advisor: Assist. Prof. Dr. Tufan Coşkun KARALAR

December 2017

ISTANBUL TEKNİK ÜNİVERSİTESİ ★ FEN BİLİMLERİ ENSTİTÜSÜ

HALL ETKİSİ BAZLI AKIM SENSÖRÜ UYGULAMASI İÇİN 14-BİT 100-kHz SÜREKLİ ZAMANLI DELTA-SİGMA ANALOG DİJİTAL ÇEVİRİCİ

YÜKSEK LİSANS TEZİ

Alper GİRGİN
(504141202)

Elektronik ve Haberleşme Anabilim Dalı

Elektronik Mühendisliği Programı

Tez Danışmanı: Yard. Doç. Dr. Tufan Coşkun KARALAR

Aralık 2017

Alper Girgin, a M.Sc student of İTU Graduate School of Science Engineering and Technology student ID 504141202, successfully defended the thesis entitled "A 14-Bit 100-kHZ Continuous-Time Delta-Sigma Analog-to-Digital Converter for Hall Effect Based Current Sensor Application", which he prepared after fulfilling the requirements specified in the associated legislations, before the jury whose signatures are below.

Thesis Advisor :	**Assist.Prof. Dr. Tufan Coşkun KARALAR** ISTANBUL Technical University	.....
Jury Members :	**Assist. Prof. Dr. Mustafa Berke YELTEN** ISTANBUL Technical University	.....
	Assist. Prof. Dr. Hakan DOĞAN Medipol University	.....

Date of Submission : 17 November 2017
Date of Defense : 12 December 2017

FOREWORD

I would like to thank my advisor Assist. Prof. Dr. Tufan Coşkun KARALAR for the oppurtunity of working in this project so that I can have such kind of thesis which include all aspects of integrated circuit design such as schematic design, layout and lab measurements etc.

I also would like to thank to the guys in ITU VLSI LABs for making a nice and friendly working environment during my graduate education there.

Last but not least I would like to thank my family for their continuous and unconditional support in any endeavor I may undertake.

December 2017 Alper GİRGİN

TABLE OF CONTENTS

Page

ABBREVIATIONS

ADC : Analog to Digital Converter
CMRR : Common Mode Rejection Ratio
CMFB : Common Mode Feedback
DAC : Digital to Analog Converter
ENOB : Efficient Number of Bits
FPGA : Field Programmable Gate Array
PCB : Printed Circuit Board
PSD : Power Spectral Density
SNDR : Signal to Noise and Distortion Ratio
SNR : Signal to Noise Ratio
SoC : System on Chip
SPI : Serial Peripheral Interface
SQNR : Signal to Quantization Noise Ratio
STF : Signal Transfer Function
NTF : Noise Transfer Function
TSMC : Taiwan Semiconductor Manufacturing Company
VCVS : Voltage Controlled Voltage Source

LIST OF TABLES

Page

LIST OF FIGURES

Page

A 14-BIT 100-kHZ CONTINUOUS-TIME DELTA-SIGMA ANALOG-TO-DIGITAL CONVERTER FOR HALL EFFECT BASED CURRENT SENSOR APPLICATION

SUMMARY

Systems on chip can integrate a number of different functions in a single die and replace a PCB with many discrete components with a single integrated circuit. Therefore their popularity increased over years. Sensor systems are good examples of system-on-chip designs.

This work is also a part of a sensor system on a chip. The sensor system consists of sensor cores, preamplifiers, analog-to-digital converter, digital calibration and digital-to-analog converter. System needs an analog-to-digital converter because calibration of the sensor is in digital domain. Delta sigma ADC is preferred due to the sensor system requirements of high-resolution and low speed.

Integrated circuit design steps are modelling, schematic level design, physical design and lab measurements after production. In this work, first, Simulink and Verilog-A model of full ADC system is prepared and requirements of sensor system are satisfied. SNQR of the ADC is about 100 dB using ideal models. Next, ideal blocks are replaced with their schematic level realizations using transistors, resistors and capacitors that are provided by the foundry. TSMC 0.18um process is used in this work. In this step, non-idealities affect the system performance and SQNR of the ADC is reduces to 87 dB. Following schematic level design, physical design step starts. During physical design, Placement of each component on the chip and their interconnections are realized. There are some verification steps that ensure the physical design is ready for production (DRC, LVS, ERC, etc.). After these verifications are completed, post layout simulations are executed. Post-layout simulation adds parasitics effects to the simulation to have more realistic results. SQNR of the ADC including post-layout parasitics is around 77 dB, which meet our specification. After manufacturing the chip is charaterized in our test labs Since the chip is not only an ADC chip, but a sensor system on chip and the ADC is a sub-block of this sensor system some problems arose during measurements of the ADC. However, the ADC works mostly as expected in the system chip.

As a conclusion, a delta sigma ADC for a sensor system on chip application is designed and realized in this work with using a common integrated circuit design flow. After manufacturing the chip is characterized and the ADC is found to be functional.

HALL ETKİSİ BAZLI AKIM SENSÖRÜ UYGULAMASI İÇİN 14-BİT 100-KHZ SÜREKLİ ZAMANLI DELTA SİGMA ANALOG DİJİTAL ÇEVİRİCİ

ÖZET

Günümüzde tümdevre teknolojisinin giderek gelişmesiyle birlikte ayrık elemanlarla gerçekleştirilen elektronik sistemler hem alan hem de güç tasarrufundan dolayı system-on-chip denilen bütün bir sistemi oluşturan blokların aynı kırmık üzerinde bulunduğu sistemler olarak gerçekleştirilmektedir. Bir çok alt bloktan oluşan sensör uygulamaları ise bunun iyi bir örneğidir. Genel olarak sensor sistemleri sensor çekirdeği, kuvvetlendirici, analog sayısal dönüştürücü, sayısal işaret işleme ve sayısal analog (S/A) dönüştürücü bloklarından oluşmaktadır. Kalibrasyon gibi işlemler sayısal ortamda yapıldığı takdirde analog sayısal (A/S) dönüştürücü bloğunun olması şarttır.

Bu çalışmada da bir sensor uygulaması için analog sayısal dönüştürücü tasarlanmıştır. Sensör çekirdeği Hall etkisini kullanarak üzerine düşen manyetik alandan Hall gerilimini oluşturmaktadır. Bu gerilim küçük genlikli olduğundan kuvvetlendirici bloğu gerekmektedir. Daha sonra gerek sensor çekirdeğinden gerekse diğer elektronik işaret işleme bloklarından kaynaklı nonlineerlikleri düzeltmek için sayısal kalibrasyon gerekmektedir. Kalibre edilen işaret sayısal analog çevirici kullanılarak dışarıya analog olarak verilmektedir.

Analog sayısal dönüştürücü ise sürekli zamanlı delta sigma analog sayısal dönüştürücü olarak seçilmiştir. Bunun sebebi sensor sistemi için gerekli olan hassasiyet ve hız seviyesidir. 14-bit çözünürlük ve 100 kHz bant genişliğine sahip olması istenen analog sayısal dönüştürücü için en optimum seçenek delta sigma analog sayısal dönüştürücülerdir. Düşük hızlı ve yüksek çözünürlüklü uygulamalar için delta sigma dönüştürücüler tercih edilmektedir.

Delta sigma A/S dönüştürücüler genel olarak üç alt bloktan oluşur. Bunlar, gürültü şekillendirmenin yapıldığı filtre bloğu, yüksek hızlı örnekleme ve kuantalamanın yapıldığı kuantalayıcı bloğu ve geri besleme katsayılarının oluşturulduğu bloklardır. Bu tasarımda geri besleme katsayıları akım modlu S/A dönüştürücüler ile elde edilmiştir. Filtre ise aktif-RC integratörler kullanılarak oluşturulmuştur. Kuantalayıcı ise basit bir flash tipinde A/S dönüştürücüdür.

Tümdevre tasarımında genel olarak önce sistemin ideal modeli oluşturularak sistemin fonksiyonel bir hatası olup olmadığı kontrol edilir. Daha sonra da şematik seviyesinde tasarıma geçilerek gerçek modele sahip elemanlarla tasarıma devam edilir. Sonrasında ise fiziksel tasarım olarak devrenin serimi yapılır ve üretime hazır olduğundan emin olunduktan sonra üretime gönderilir. Üretimden geldikten sonra da çipin ölçümüne başlanır. Tümdevre tasarımının temel adımları bunlardır. Bu çalışmada da benzer tasarım adımları uygulanmıştır.

Öncelikle A/S dönüştürücü sisteminin Simulink ve Verilog-A kullanılarak devrenin ideal modelleri yapılmış ve benzetimler yapılarak sistemin herhangi bir yapısal hatası

var mı diye kontrol edilmiş, olan hatalar da düzeltilmiştir. Benzetim sonuçlarında 100 dB civarı SQNR görüldükten sonra da şematik seviyesi tasarıma geçilmiştir.

Şematik seviye tasarım ise modeldeki ideal elemanlar üreticiden gelen gerçek modele sahip elemanlarla değiştirilerek yapılmıştır. Örneğin, ideal bir anahtar elemanı yerine transistörler kullanılarak yapılan bir anahtar koyulduğunda doğal olarak o anahtarın direnci, gecikmesi, yük etkileri devreye girecektir. Bu da performansta bir düşüşe sebep olacaktır.

Gürültü ve işaret transfer fonksiyonlarını sağlayan filtre aktif-RC integratörler kullanılarak oluşturulmuştur. Bu tipteki integratörler direnç, geri besleme kapasitesi ve operasyonel kuvvetlendirici kullanılarak yapılır. Bu tasarımda da kuvvetlendirici olarak katlı kaskot yapısında giriş katı ve ortak kaynak bağlı bir çıkış katı birlikte kullanılarak iki katlı bir yapı tasarlanmıştır. Filtre üçüncü dereceden olduğu için üç tane integratör kullanılmıştır. İşaretle ilk karşılaşan blok ilk integratör olduğu için o integratör özellikle gürültü performansı daha iyi olacak şekilde tasarlanmıştır.

Kuantalayıcı katı olarak 5-bitlik basit bir flash A/S dönüştürücü kullanılmıştır. Bu tür A/S dönüştürücüler referans aralığı ve bu referansları giriş işaretiyle karşılaştıran karşılaştırıcı devrelerinden oluşmaktadır. Bu tasarımda ise direnç dizisiyle oluşturulan referanslar ve bu referansları girişle karşılaştıran 16 adet karşılaştırıcı devresi kullanılmıştır. 16 seviyelik bir flash A/S dönüştürücü olduğu için 5-bit olarak kabul edilmiştir.

Geri besleme katsayılar ise akım modlu S/A kullanılarak filtreye verilmiştir. Burada S/A dönüştürücü kullanılmasının sebebi, kuantalayıcının termometrik kodlu 16 bitlik dijital bir işaret oluşturması ve filtreye girecek geri besleme katsayılarının ise analog olması gerekliliğidir. Akım modlu S/A dönüştürücü 16 adet hücreden oluşmaktadır ve her bir bit bir hücredeki anahtarla o hücreden akım geçmesini ya da geçmemesini sağlar. Bu şekilde 16 hücrenin akımı toplanarak filtreye geri besleme olarak girer.

Burada önemli olan diğer bir blok ise dinamik eleman eşleme bloğudur. Genel olarak D/A dönüştürücülerin girişine koyulan bu tip bloklar S/A dönüştürücü hücreleri arasındaki uyuşmazlık etkilerinden kaynaklı lineerlik bozulmalarını düzeltmeye yararlar. Bu tasarımda da dinamik eleman eşleme metoduyla S/A dönüştürücünün performansı arttırılmıştır.

Dinamik eleman eşleme metodu ise şu şekilde çalışmaktadır: Kuantalayıcı çıkışından gelen termometrik kodlu bitler S/A dönüştürücü düşük bitlerin denk geldiği hücrelerin sürekli aktif olmasına sebep olur. Bu da bu hücreler arasındaki uyuşmazlık etkilerinin performansı daha çok etkilemesine sebep olur. Bu neden termometrik kodlu bitler dinamik eleman eşleme metoduyla rastgele olacak şekilde dağıtılır ve hep aynı hücrelerin aktif olması engellenir. Böylece de aktif olan S/A dönüştürücü hücrelerinin çeşitlenmesiyle uyuşmazlıkların ortalaması düşer. Bu da S/A dönüştürücünün lineerliğini arttırır.

A/S dönüştürücü çıkışı yüksek hızlı örnekleyen saat işaretiyle çalıştığı için çıkış verisini Nyquist bandına indirmek için delta sigma A/S dönüştürücülerinde seyreltme filtresi kullanılır. Sistem üçüncü dereceden olduğu için seyreltme filtresi de sinc3 fonksiyonu olacak şekilde seçilmiştir. Bu fonksiyonu gerçeklemek için ise Hogenauer filtre yapısı kullanılmıştır. Bu yapının kullanılma sebebi ise gerçekleme kolaylığıdır.

Seyreltme filtresi tamamen dijital olduğu için de Verilog koduyla tasarlanıp Cadence tasarım ortamının dijital tasarım araçları kullanılarak sentezlenip diğer bloklara eklenmiştir.

Şematik seviye tasarımda alt bloklar bu şekilde tasarlandıktan sonra tüm sistemin benzetimleri yapılmıştır. Burada da 86 dB civarında bir SNDR elde edilmiştir. Ek olarak, işaretin girdiği ilk blok olan delta sigma modulatörün termal gürültü performansına da bakılmış ve gürültü performansın -87 dB civarında olduğu görülmüştür. Bu da kuantalama gürültüsü, harmonik distorsiyon ve termal gürültü toplamının 80 dB'den yüksek olduğu anlamına gelmektedir. Bu da zaten istenilen SNDR 80 dB civarında olduğu için iyi bir değer denebilir.

Şematik tasarım bittikten sonra da devrenin fiziksel tasarımına yani serimine başlanmıştır. Serimde önce istenilen alana sığacak şekilde yer planı yapılıp sonra da alt bloklardan başlanarak hepsi için tek tek belirlenen alanlara uyacak şekilde serimleri yapılmıştır. Buradaki diğer önemli konu ise işaretin izlediği yola uygun olarak yer planının yapılmasıdır. Alt blok serimlerinde mümkün olduğunca düşük seviyeli metaller kullanılmaya çalışılmıştır.

Serimler bittikten sonra DRC, LVS ve ERC kontrolleri yapılmıştır. Sonrasında ise serim sonrası benzetimleri yapılmıştır ve bu sonuçlarda da 77 dB civarı bir SNDR elde edilmiştir. Bu değer hedeflenen değerden düşük olmasına rağmen yine de kabul edilebilir bir değerdir. Devre tamamlandıktan sonra sensör sistemindeki diğer bloklara eklenip üretime gönderilmiştir.

Çip üretilip geldikten sonra ölçümlere başlanmıştır. Çip sadece delta sigma A/S dönüştürücüyü değil de bir sürü bloğun bulunduğu sensör sistemini içerdiği için tek başına A/S dönüştürücünün performansı düzgün ölçülememiştir.

En başta seyreltilmiş delta sigma A/S dönüştürücü çıkışını çipin SPI haberleşme arayüzünü kullanarak dışarı almak planlanmıştır ancak SPI hızı özellikle 1 kHz'den yüksek hızlı A/S giriş işaretleri için yeterli gelmemeye başlamıştır.

Buna çözüm olarak A/S dönüştürücü diğer bloklarla birlikte çalıştırılmış ve tüm sistemin çıkışı olan S/A dönüştürücü çıkışının frekans spektrumu incelenmiştir. Burada daha iyi bir lineerlik performansı görülse de sebebinin yüksek ihtimalle sistemin en son bloğu olan S/A dönüştürücüden kaynaklandığı düşünülmektedir. Yine de bütün sensör sistemi incelendiğine delta sigma A/S dönüştürücünün performansına tam olarak bakılamasa da çalıştığı gözlenmiştir.

A/S dönüştürücünün benzetim performansı FoM (Figure of Merit) hesaplanarak literatürdeki diğer benzer çalışmalarla karşılaştırılmıştır. Burada da akım optimizasyonu yapılmadığı için S/A dönüştürücünün güç tüketiminin diğer çalışmalardan fazla olduğu görülmüştür.

İleri çalışma olarak ise A/S dönüştürücünün akım optimizasyonu yapılıp daha iyi bir FoM değerine sahip olması sağlanabilir. Buna ek olarak seyreltilmemiş A/S dönüştürücü çıkışı çipe yeni terminaller eklenerek direkt olarak dışarıya verilebilir. Bu sayede ölçümde karşılaşılan zorluklar çözülmüş olur ve A/S dönüştürücünün performansına net bir şekilde bakılabilir.

Sonuç olarak bu çalışmada sensör uygulaması için sürekli zamanlı delta sigma A/S dönüştürücü genel tümdevre tasarım aşamalarına uyularak tasarlanmış ve ölçümler sonucunda başarılı olduğu görülmüştür.

1. INTRODUCTION

System on chip (SoC) is a term that defines the chips that include all the blocks that are included in an electronic system such as, sensors, amplifiers, data converters, RF blocks or microcontrollers. Improvements in integrated circuit design and production technologies, popularized System on chips. Electronic systems realized as SoCs consume less area and power than system on board level designs which is their main advantage. However, harder design and verification processes and less error correction ability are some of the disadvantages of SoCs.

1.1 Motivation

The work presented in this thesis is a part of Hall Effect Based Current Sensor project which is funded by The Scientific and Technological Research Council of Turkey (TÜBİTAK) research grant 115C053.

Sensor chip consists of Hall Effect sensors, a preamplifier, an analog-to-digital converter, a digital calibration block and a digital-to-analog converter. In addition to these blocks, a serial peripheral interface (SPI) block to communicate with outside, temperature sensor block to be used in calibration and other necessary blocks like bandgap references, an oscillator, a power-on-reset and a test bus switch are in the chip. Figure 1.1 shows the block diagram of the overall sensor chip.

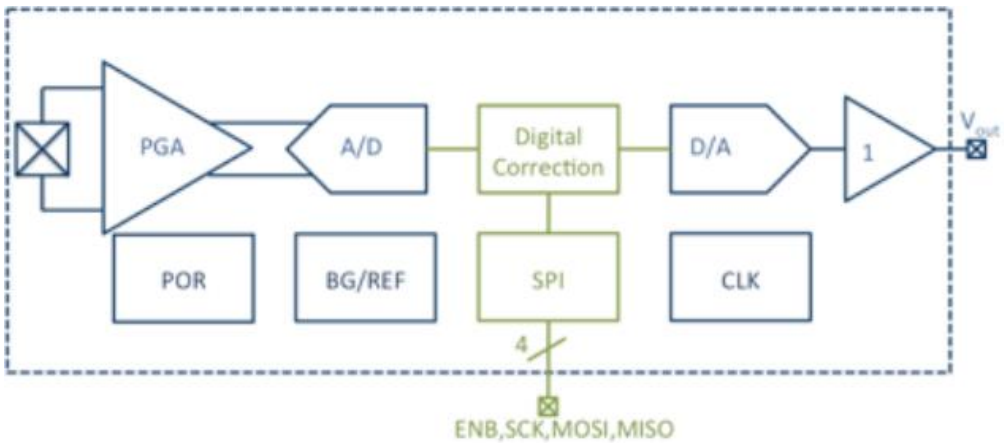


Figure 1.1: System Level Block Diagram of Hall Effect Sensor Application.

Sensors produce analog voltage output, the preamplifier amplifies this voltage and since there is a digital calibration block, an ADC is needed to convert this amplified sensor voltage to digital bits.

The reason that delta sigma ADC chosen is the low signal bandwidth (100 kHz) and mid-resolution (14-bit) requirements. Meeting these specifications with a continuous time delta sigma ADC is easier. Figure 1.2 shows the ADC survey results of the literature. It is obvious that high speed and low resolution ADCs are dominated by Nyquist rate ADCs while low speed and high resolution ADCs are dominated by delta sigma ADCs. It also can be seen that the ADCs that have similar specifications to this work are delta sigma ADCs.

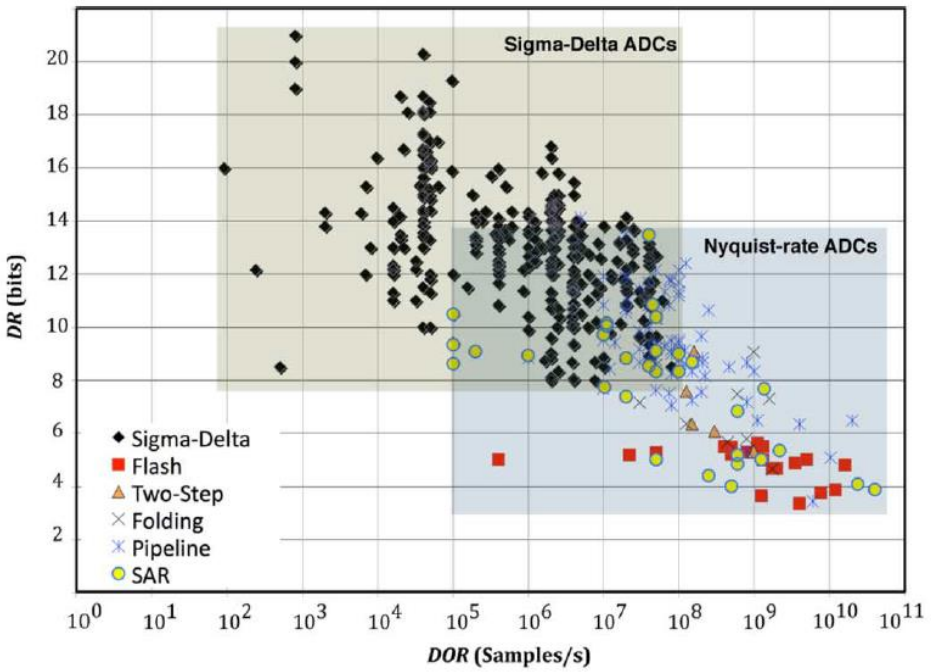


Figure 1.2: ADC Survey [1].

1.2 Design Summary

Delta sigma modulators mostly consist of three fundamental sub-blocks. In this design, first block is loop filter which basically does the noise shaping, second one is the flash ADC which samples and quantizes the signal and last block is feedback DAC which produces the feedback coefficients as currents from the output voltage to the loop filter. Additional to these, there is dynamic element matching block that reduces the mismatch based errors of feedback DAC.

Because of several advantages delta sigma ADC is planned to design in continuous mode and multi bit at output (decided as 4 bits). Different type of delta sigma ADCs are investigated and compared in next chapter which explains delta sigma ADC basics.

Since the ADC has a 100 kHz bandwidth a faster sampling clock with enough oversampling is necessary. There is a 50 MHz internal clock in the system and the sampling clock can be produced by using it, just simply dividing by 4. This makes the sampling frequency 12.5 MHz and OSR is calculated as 62.5. SNDR of ADC is expected around 82 dBc. In terms of ENOB fashion, this makes typically 13 bits (max 14, min 12). Table 1.1 shows the basic specifications of the delta sigma ADC in this work. SNDR, ENOB and current consumption values are design targets that may differ from results.

Table 1.1: Specifications of Delta Sigma ADC.

Parameter	**Value**
Power Supply	4 V
Current Consump.	5 mA
Common Mode V.	2 V
Full Scale V.	4 V
ENOB	13 Bits
SNDR	82 dBc
Bandwidth	100 kHz
Sampling Freq.	12.5 MHz
OSR	62.5
Output Bits	4 Bits

Manufacturing technology is TSMC 0.18u CMOS process with high voltage (5V) transistors. Design environment is Cadence Design Environment tools: Virtuoso Schematic and Layout XL editors, Spectre simulator and Assura layout verification. NCLaunch, RTL Compiler and Encounter tools of Cadence Design Environment are used to simulate, synthesize and implement digital blocks that are designed by using Verilog hardware description language. MATLAB and Simulink are also used for modelling and SNR calculations.

1.3 Thesis Organization

The design flow of continuous time delta sigma analog to digital converter in this thesis is given below.

- Synthesizing NTF and STF (z-domain) in MATLAB by using Delta-Sigma Toolbox
- Transforming filter coefficients from z-domain to s-domain by using Impulse Invariant Transform
- Modelling delta sigma ADC in Simulink with s-domain elements (some non-idealities also can be added)
- Calculating resistor and capacitor values of loop filter
- Modelling delta sigma ADC in Cadence with ideal blocks (important to investigate excess loop delay)
- Implementing delta sigma ADC with real elements
- Layout and measurements results

Thesis organization is planned respect to this design flow with additional theoretical sections. In this first chapter, general information about SoCs, sensor chip which includes this work's delta sigma ADC and the specifications of delta sigma ADC are given.

In second chapter, fundamentals of continuous time delta-sigma converters are explained with basic background, performance aspects and some non-idealities.

In third chapter, system level design starts by synthesizing a proper noise transfer function(NTF) and modelling this function in Simulink with other blocks of delta sigma ADC. Then, ideal model in Cadence environment is presented.

In fourth chapter, schematic level design with real components is explained and simulation results of the sub-blocks and delta sigma ADC are given.

In fifth chapter, layout of the design is explained with layout considerations and some important notes about the layout. In addition to this, measurement test setup is explained and the results are given.

2. CONTINUOUS TIME DELTA SIGMA ADC BASICS

An ideal delta sigma ADC consists of three main sub-blocks: Loop filter, quantizer and feedback DAC. Figure 2.1 shows the delta sigma ADC modulator. First, input signal are filtered by loop filter and then sampled and quantized by the quantizer block. After that the feedback DAC converts digital output to analog and feeds the input. This system is nonlinear in reality however the approximate linear equations of the system are given in (2.1) and (2.2) to understand it better.

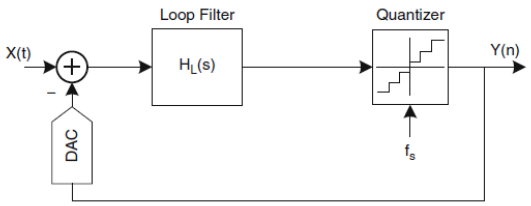


Figure 2.1: System Model of Delta Sigma ADC [2].

Here, X is the input signal, H is the transfer function of loop filter and E_Q is the quantization error. STF and NTF stands for signal transfer function and noise transfer function. Figure 2.2 shows the frequency domain plots of NTF and STF.

$$Y(s) = X(s).\frac{H_L(s)}{1+H_L(s)} + E_Q(s).\frac{1}{1+H_L(s)} \tag{2.1}$$

$$STF(s) = \frac{H_L(s)}{1+H_L(s)}\ , NTF = \frac{1}{1+H_L(s)} \tag{2.2}$$

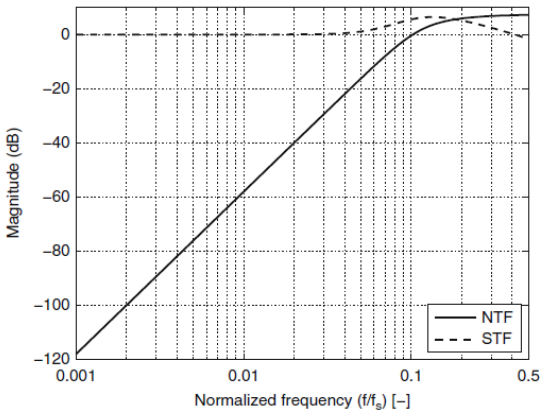


Figure 2.2: Frequency Plots of NTF and STF in 3rd Order DSM [2]

As it can be seen from Figure 2.2, noise transfer function shapes the noise, which decreases the noise in base band and increases the resolution of the ADC. This noise shaping is one of the major features of a delta sigma ADC with oversampling.

2.1 Oversampling and Noise Shaping

There might be some application specific differences between delta sigma ADCs but two basic principles are the same for all of delta sigma ADCs: oversampling and noise shaping.

Sampling with a higher frequency than required Nyquiest frequency is called oversampling. Oversampling does not change the total quantization noise power because quantization noise is not related to the sampling frequency. However, total quantization noise power spread over a wider band due to oversampling and it decreases the quantization noise power over the band of interest. Figure 2.3 shows the effect of oversampling by 2x and 4x on quantization noise power.

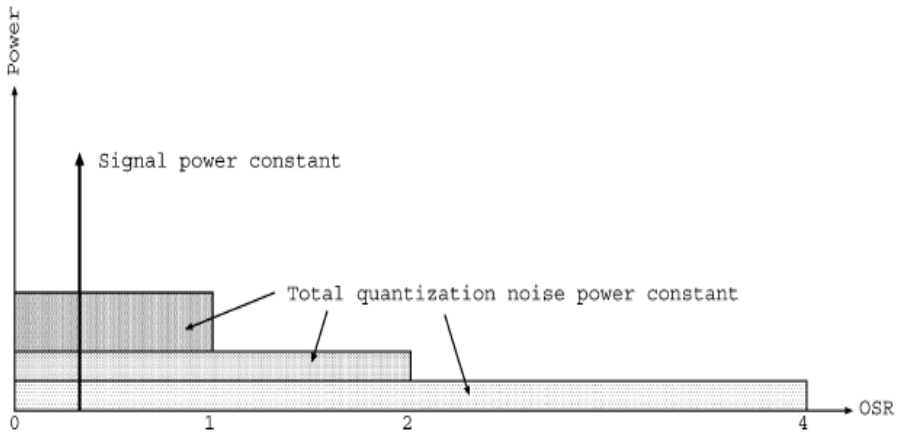


Figure 2.3: Oversampling [3].

Noise shaping is another way to decrease quantization noise power over the band of interest by pushing the quantization noise through the higher frequencies as shown in Figure 2.4. This process also does not change the total quantization noise power. Quantization error feeds the filter input and the noise transfer function of loop filter shapes the noise. Signal is not affected from this process because the input signal is only related to signal transfer function. Equation (2.1) and (2.2) shows the theory behind noise shaping.

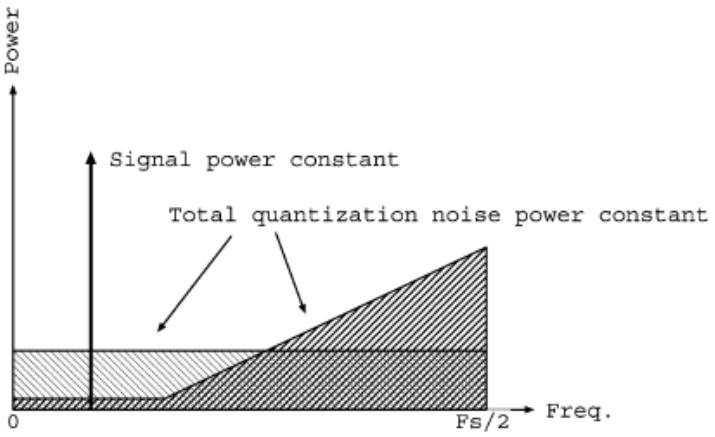


Figure 2.4: Noise Shaping [3].

These two methods are used together in delta sigma ADCs and they are the main elements that decide the performance of delta sigma ADC.

2.2 Performance Metrics

Some important performance parameters used in delta sigma ADCs are explained in this section.

Signal-to-Noise Ratio (SNR): SNR is the ratio between signal power and the total in-band noise power. For a first order delta sigma modulator, the SNR can be derived as given in (2.3) where A is input amplitude, OSR is oversampling ratio and q is quantization level.

$$\text{SNR} = 10\log\frac{18A^2\text{OSR}^3}{\pi^2 q^2} \tag{2.3}$$

This equation only covers the quantization noise. Normally, ADCs suffer from many other noise sources that decrease the SNR of delta sigma ADC. The parameter that covers all the noise and non-linearity in the design is signal to noise distortion ratio (SNDR).

Effective Number of Bits (ENOB): Efficient number of bits is another way to express the linearity of the delta sigma ADC in terms of resolution.

$$\text{ENOB} = \frac{\text{SNR}_{\text{peak}} - 1.76\text{dB}}{6.02\text{dB}} \tag{2.4}$$

Dynamic Range (DR): Dynamic range is about the input signal. It is the ratio of minimum detectable input signal by the modulator and maximum input signal which makes modulator stable.

Figure of Merit (FOM): FOM is the parameter that is a function of a few specifications of the circuit which make easier to compare the design with other designs. For delta sigma ADCs, power consumption, signal bandwidth and SNDR are taken to consideration.

$$\text{FOM} = \frac{\text{Power}}{2f_B \times 2^{\text{ENOB}}} \tag{2.5}$$

2.3 Increasing the Performance

It is possible to increase the linearity of delta sigma ADC by increasing OSR, loop filter order and number of quantizer bits. These are explained below with considering their tradeoffs or drawbacks.

As explained in Section 2.1, oversampling is applied in delta sigma ADCs to spread the quantization noise power over the out of the band frequencies. The higher the OSR is the less will be the quantization noise power in interested band. On the other hand, higher clock frequencies are needed to satisfy higher OSR. Therefore, clock frequency limits the OSR especially in high frequency applications. This also causes extra power consumption.

The order of loop filter helps to realize sharper noise filters which increase the noise shaping effect. Therefore, SNR of the delta sigma ADC can increase with higher order. However, stability becomes important when implementing third or more order loop filters which limits the order of the loop most of time.

Another method to achieve higher SNR is to change the single bit quantizer to multi bit quantizer. For this end a 3 or 4 bit flash ADC is used instead of a comparator as quantizer. Here there are more quantization levels which decrease the quantization noise power in multi bit quantizers. However, multi bit quantizers require more complex feedback DACs and Dynamic Element Matching (DEM) as well as extra circuitry to improve its linearity. These overhead become important in the multi bit quantizer case.

2.4 Delta Sigma Modulator Structures

There are three main different types of delta sigma modulators. These are feed-forward, feedback and cascaded (MASH). All of them realize the same noise transfer

function and signal transfer function in different ways and each offer some advantages and disadvantages.

2.4.1 Feedback delta sigma modulator

Figure 2.5 shows the nth order structure of feedback delta sigma modulator. I(s) blocks represent 1/s continuous time integrator blocks and E is the quantization error. In the figure "a_i" coefficients are gains of integrators.

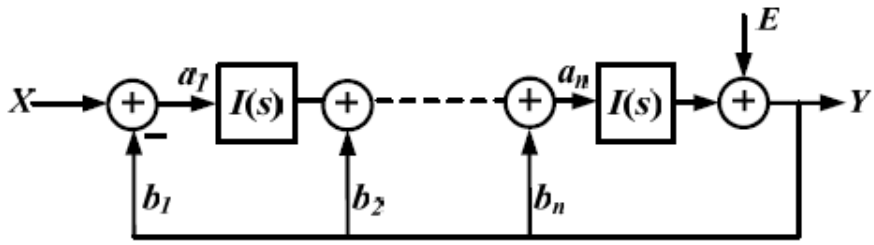


Figure 2.5: Feedback DSM Topology [4].

In feedback DSM topology, every integrator outputs are fed, which leads to large input swing. Thus, there may be some linearity problems especially for low voltage applications.

2.4.2 Feed-forward delta sigma modulator

Feed-forward system solves the problem in feedback systems by feeding the quantizer input from integrator inputs so that one feedback DAC would suffice, only to the first integrator's input. This is the main advantage of the feed-forward system. Thus, internal voltage swing requirements are decreased by this architecture. Figure 2.6 shows the nth order structure of feed-forward delta sigma modulator.

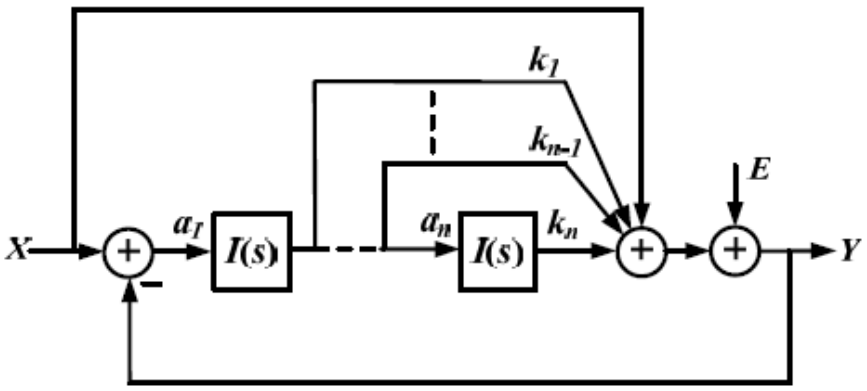


Figure 2.6: Nth Order Feed-forward DSM [4].

2.4.3 Cascaded delta sigma modulator (MASH)

Basic structure of cascaded delta sigma modulators, also called multi stage noise shaping (MASH), can be seen in Figure 2.7. It is basically two different modulators connected together with digital filters in their outputs.

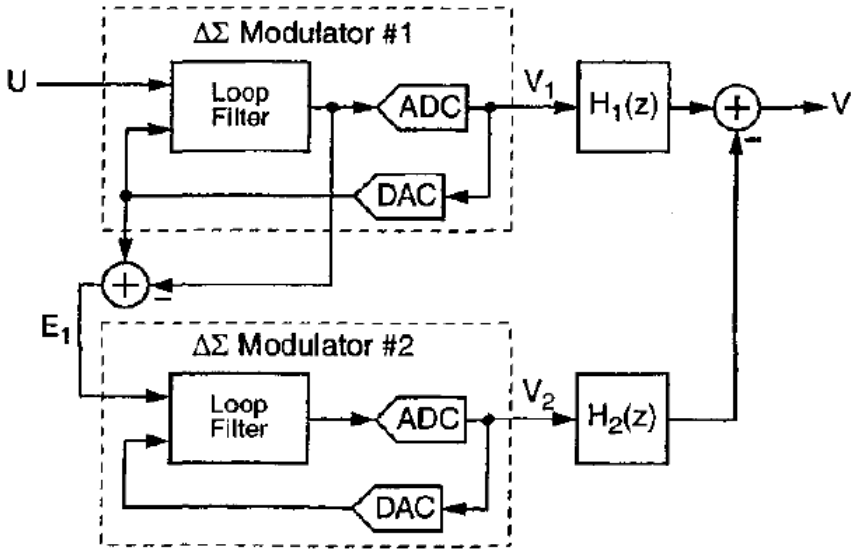


Figure 2.7: Cascaded DSM Architecture [4].

Normally in delta sigma modulators, stability becomes a big problem in high order transfer functions. On the other hand in cascaded topologies, stability is less problematic than high orders. This is the main advantages of the cascaded delta sigma modulators. The main disadvantage of MASH, it requires digital filters at the output and this increases the complexity and power consumption. Also the matching between the cascaded stages can be an issue.

2.5 Implementation Considerations

System level simulations in Simulink considers only quantization errors and shows the maximum achievable SQNR. During implementation, there are some other effects that need to be considered. SNR of the delta sigma modulator suffers from the non-ideal effects in addition to quantization error such as noise, nonlinearity, metastability, excess loop delay(ELD) and finite gain, finite GBW. This effect can be modeled in system level but normally they affect the circuit in transistor level. Therefore, noise budget must be prepared and SNDR of the modulator in system level (ideal model) must satisfy the sum of needed SNR, non-ideality caused distortions and other noise sources.

2.5.1 Noise

Thermal noise affects the delta sigma converter circuits such as all other electronic devices and it decreases SNR of system. Figure 2.8 shows the noise sources in a delta sigma ADC system.

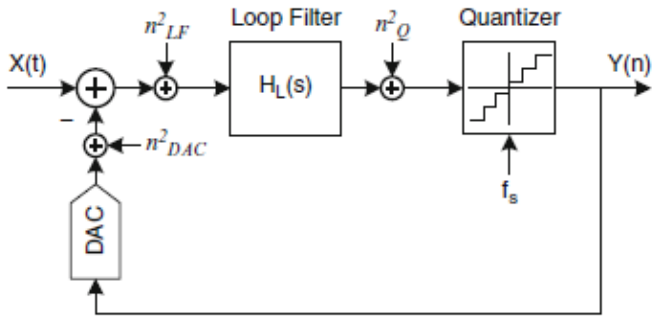


Figure 2.8: Noise Elements in DSM [5].

Another effect that can be count as noise and that reduces the SNR is clock jitter. Clock jitter is the non-idealities in the clock signal such as delays, time shifts etc. It becomes more important in high frequency designs.

As an implementation problem, it is obvious that noise effects will reduce the SNR of the circuit. Therefore margin must be buit into the SNR values in ideal system simulations. Moreover to maintain some noise margin over targeted SNR value is always good.

2.5.2 Nonlinearity

As explained before, quantizer is the only nonlinear block in delta sigma modulators. Quantization noise of the system has very important effect over SNR.

Another nonlinearity source is mismatches in circuit. To avoid these nonlinearities, quantizer output can be designed to have multi bits which increase the linearity and SNR. Although DAC mismatch can limit the multi bit system's SNR, calibration ways such as dynamic element matching (DEM) mitigate this limitation.

2.5.3 Excess loop delay

Excess loop delay (ELD) is defined as the delay between quantizer clock edge and output change of DAC. Main reason of that is the limited speed of transistors in the quantizer and DAC. ELD causes a phase shift in the signal and this can make the system unstable.

In system level, ELD can be modelled by putting a delay between quantizer and DAC. To solve the unstablity caused by ELD, a zero must be added the system to by-bass the loop filter at $F_S/2$.

Figure 2.9 shows the additional ELD model block and c coefficient to add extra zero to compensate ELD effect. There are other compensation techniques to avoid ELD but the main idea is similar.

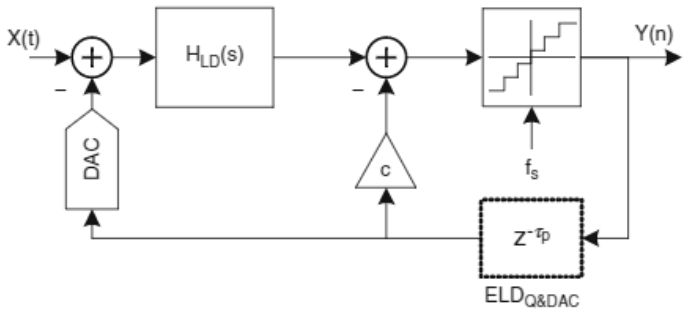


Figure 2.9: ELD Compansation [5].

2.5.4 Metastability

Quantizer output is digital and this digital output is decided by comparators or simple ADCs which also have comparators. In the high speed applications, this digital output can be resolved incorrectly by the latches of comparator due to metastability and this incorrect digital bits reduce the SQNR of the system,and increase nonlinearity.

Metastability can be modeled by adding some noise at output of the quantizer. To avoid metastability, pipeline ADCs can be used in quantizer but this also increases the latency. This delay increases the ELD and may cause unstability. Therefore, metastability must be considered in the noise budget of the system.

2.5.5 Finite amplifier gain and gain-bandwidth effect

Integrators in continuous time delta sigma modulators are mostly active-RC integrators or gm-C integrators. Active-RC integrators are based on operational amplifiers and gm-C integrators are based on transconductance amplifiers. In reality both amplifiers suffer from finite gain and gain bandwidth product(GBW). Ideally pole of the integrator is in DC but in reality it is pushed away from DC through high frequencies. Non-ideal integrators are also called lossy integrators.

Since finite gain and GBW changes the integrator's transfer function, NTF and STF of the delta sigma modulator is also affected. Non-ideal effects change the shaping of filters and this reduces the SNR of the delta sigma modulator.

Decreasing these non-ideal effects is possible by using feed-forward delta sigma modulator structure.

3. MODEL

3.1 Noise Transfer Function In Discrete Time Domain

One of the most important features of delta sigma ADC is its noise shaping behavior. Quantization noise in the in-band frequencies are suppressed through the high frequencies by the noise transfer function. Therefore, first step of the delta sigma ADC design flow is to synthesize a noise transfer function. In this design, Delta-Sigma Toolbox of Richard Schreier [5] is used to synthesize a proper NTF.

Delta Sigma Toolbox is a bunch of MATLAB codes and commands that synthesizes a NTF for given specifications, simulates the synthesized NTF, calculates its SNR and calculates the implementation coefficients for the given structure and scales those coefficients.

First, synthesizeNTF command is used with proper input parameters to get a NTF (z-domain) that has above 100 dB SNR. simulateSNR command plots the SNR vs. input amplitude graph. Therefore, it can be seen checking the max. SNR whether NTF is proper or not. H(z) is the synthesized noise transfer function. Figure 3.1 shows the root locus of NTF.

$$H(\mathrm{z}) = \frac{(z-1)(\mathrm{z}2-1.988\mathrm{z}+1)}{(z-0.6992)(z2-1.52z+0.6636)} \tag{3.1}$$

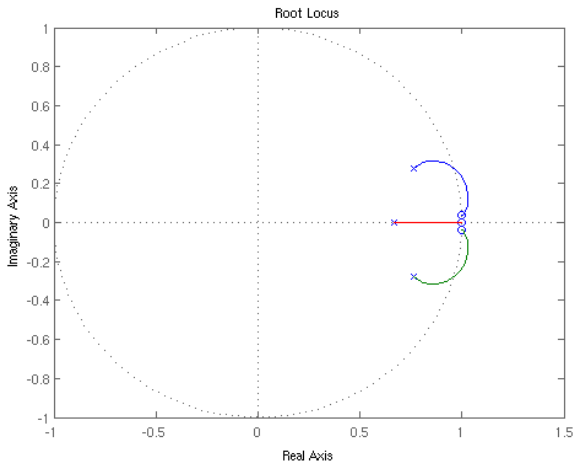


Figure 3.1: Root Locus of NTF.

Figure 3.2 shows the SNR vs. Input Amplitude Plot. Zero optimization option of Delta Sigma Toolbox optimizes the zeros to get maximum SNR from the transfer function.

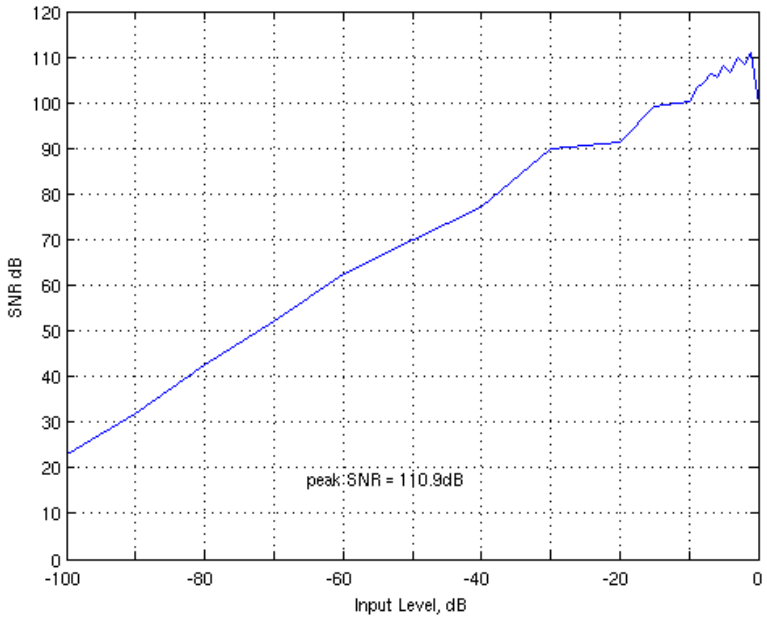


Figure 3.2: SNR vs. Input Amplitude Plot.

Last command is realizeNTF, which computes the coefficients for the CIFB structure that stands for "Cascade of Integrator, FeedBack form". z-domain coefficients of the resulting loop filter are given in Table 3.1.

Table 3.1: Discrete Time Coefficients.

Coefficient	Value
a1	0.0441
a2	0.2871
a3	0.8005
b1	0.0441
b2=b3	0
c1=c2=c3	1
g1	0.0015

Frequency responses of the resulting NTF and STF are given in Figure 3.3. MATLAB codes of this process can be seen Appendix-A.

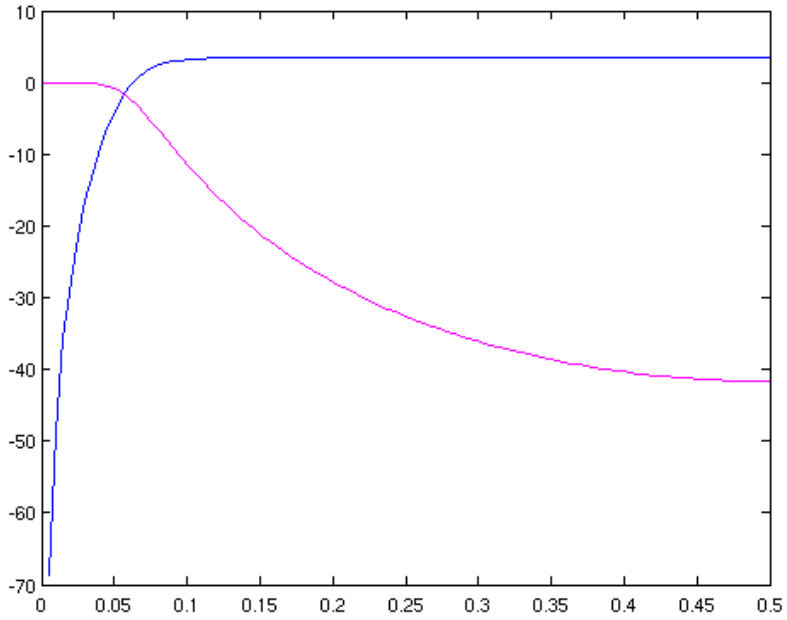


Figure 3.3: Frequency Response of NTF and STF.

3.2 DT to CT Conversation by Using Impulse Invariant Transform

After obtaining the coefficients in discrete time domain, next step is to convert them into continuous time domain. When impulse invariant transform is applied to transfer function in terms of coefficients, discrete time coefficients change by the equations given below [7].

$$k_1 = \frac{a_1 a_2 a_3}{\beta - \alpha} \tag{3.2}$$

$$k_2 = \frac{a_1 a_2 a_3}{2} \frac{\frac{2}{a_1} + \alpha + \beta - 3}{\beta - \alpha} \tag{3.3}$$

$$k_3 = \frac{a_1 a_2 a_3}{12} \frac{\alpha(\alpha-9) + \beta(\beta-9) + 4\alpha\beta + \frac{6}{a_1}(\beta+\alpha) + \frac{12}{a_1}(\frac{1}{a_2} + a_1 - 1)}{\beta - \alpha} \tag{3.4}$$

a_1, a_2, a_3 are discrete time feedback coefficients and k_1, k_2, k_3 are continuous time versions of coefficients. β and α are DAC pulse coefficients and they are selected as $\beta = 1$ and $\alpha = 0$ to satisfy the non-return to zero DAC pulse behavior. As a result the equation is simplified as seen below.

$$k_1 = a_1 a_2 a_3 \tag{3.5}$$

$$k_2 = a_1a_2a_3 + a_2a_3 \quad (3.6)$$

$$k_3 = \frac{a_1a_2a_3}{3} + \frac{a_2a_3}{2} + a_3 \quad (3.7)$$

Continuous time coefficients are calculated regarding to these equations and given in Table 3.2. b_1 coefficient sets the gain of the loop filter. To have a gain of 1, it must be equal to the a_1 coefficient. Other b and c coefficients are not changed and g_1 coefficient is ignored since it needs high resistor values to be implemented.

Table 3.2: Continuous Time Coefficients.

Coefficient	Value
k1	0.0101
k2	0.2400
k3	0.9188
b1	0.0101
b2=b3	0
c1=c2=c3	1

In the next step, Simulink model of the delta sigma ADC is prepared by using these continuous time coefficients.

3.3 Simulink Model

Studying on a Simulink model before circuit implementation is always better way to design systems. Therefore, there is a Simulink model step in the design flow. Three different models of the delta sigma ADC is prepared. First model is the loop filter with ideal quantizer elements, second one is loop filter with flash ADC instead of ideal quantizer, dynamic element matching and feedback DAC, and third model is just the similar to the second one except non-ideal integrators are used in the loop filter instead of ideal integrators.

3.3.1 Model with ideal quantizer

Simulink model of the delta sigma ADC is based on the CIFB structure in the Delta Sigma Toolbox of Schreier. Model consists of continuous time integrators, gain blocks and an ideal

quantizer. Integrators with gain blocks as coefficients are used to model the loop filter. Quantizer at the output sets the sampling frequency and quantization level which is 16 in this design, since the design is 5-bit output (multi bit).

Figure 3.4 shows the Simulink model of the design. This model is simulated by a MATLAB code which is given in Appendix-B. Values of the coefficients are given at Table 3.2 above. Additional to these coefficients, blocks just in front of the integrators has the value of F_S which is sampling frequency. Since the model is not in the normalized frequency, integrators are needed to multiply by sampling frequency.

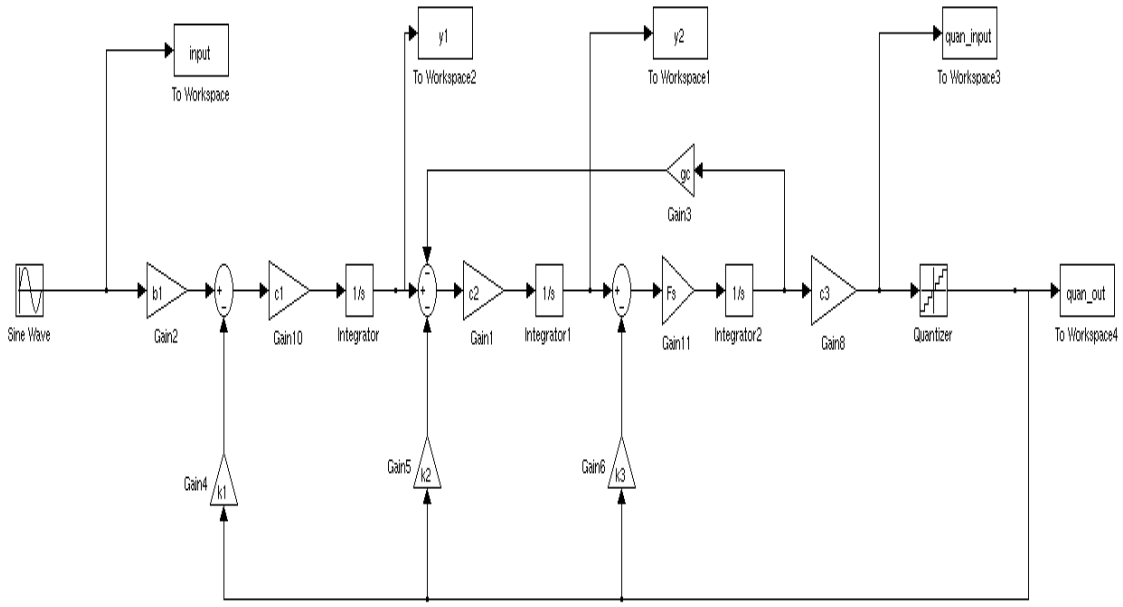


Figure 3.4: Simulink Model with Ideal Quantizer Block.

3.6 V_{PP} amplitude sine wave at 33.5 kHz with 4096 samples is applied as input and signal to quantization noise ratio (SQNR) of the system is calculated by the code that is given in Appendix-B. Power spectrum density plot of the simulation result is given in Figure 3.5. It shows that SQNR is calculated as 96.2 dB and efficient number of bits (ENOB) is around 16 bits.

These results show that coefficients of the loop filter are good enough and the noise shaping is working well. In the next model, flash ADC model and feedback DACs are used instead of ideal quantizer model.

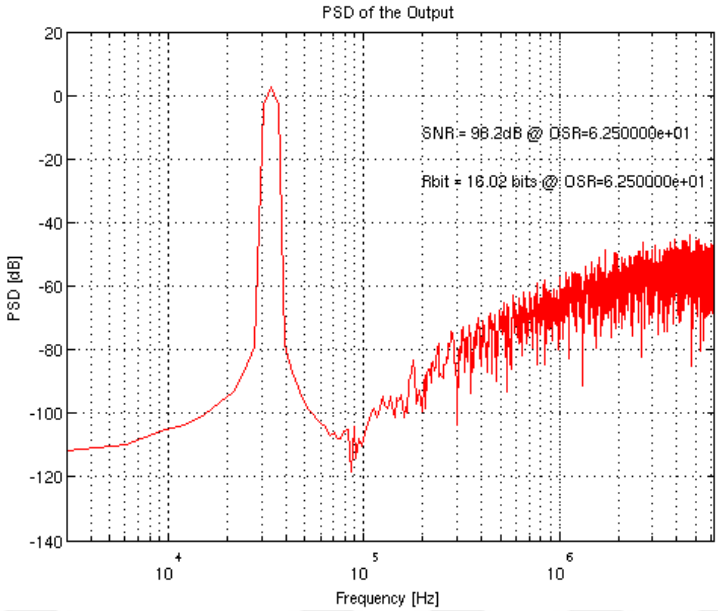


Figure 3.5: Power Spectrum Density of the First Model.

3.3.2 Model with flash ADC, dynamic element matching and feedback DAC

In this model, ideal quantizer is changed to have a more realistic approach. Figure 3.6 shows the new model with other elements that are zero-order hold for sampling in F_S, 4-bit quantizer for quantization in amplitude and digitalizing, dynamic element matching (DEM) for cancelling mismatch errors in real implemantation and DAC to have analog signal for feedback of loop filter.

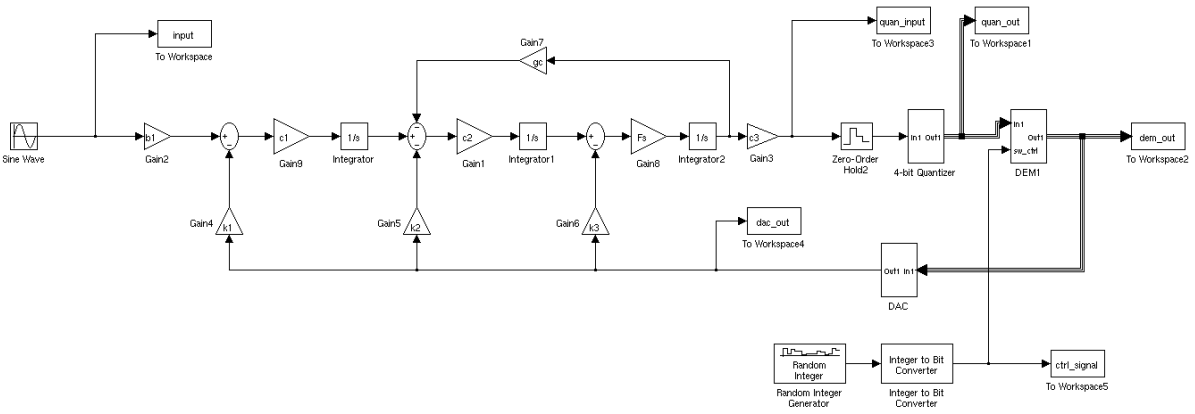


Figure 3.6: Model with Flash ADC, DEM and Feedback DACs.

In the 16-level quantizer block which is actually simple flash ADC, sampled analog signal compared by the intervals that are calculated by $\Delta = \frac{V_{FS}}{2^n - 1}$ where V_{FS} is full scale voltage and n

is the number of bits and produces a 16-bit thermometer coded digital output. Figure 3.7 shows the flash ADC model that is used as 4-bit quantizer. There are 16 levels in quantizer that can be considered as 5-bit. The additional bit is needed for DEM block which only works with 2n numbered bits.

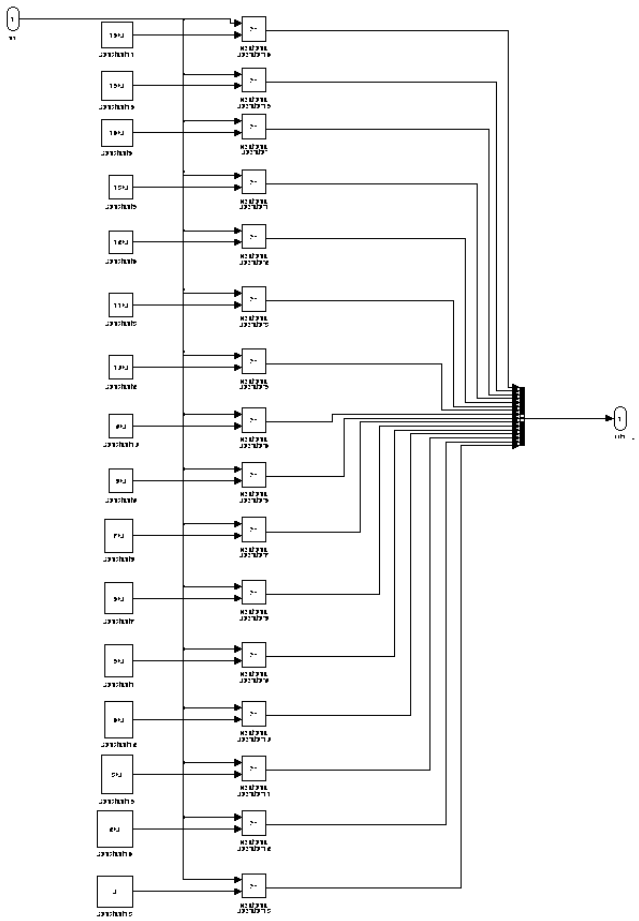


Figure 3.7: 4-bit Flash ADC Simulink Model.

Signal is converted to digital by the quantizer but the loop filter is still in analog domain and feedback signals of the loop filter depends on digital output. Therefore, there has to be feedback DAC that to produce the feedback gains properly. Since the digital output is 16-bit thermometer coded, unity weighted element DAC can do the conversion. Figure 3.8 shows the simple model of the unity weighted DAC. Digital input of DAC multiplies by unity weight which is Δ calculated above.

When implementation issues considered, mismatch in DAC elements degrades the performance of the system. To cancel the mismatch effects, dynamic element matching concept is developed. Figure 3.9 shows the model of dynamic element matching block. It

takes the thermometer coded input and changes the places of 1s and 0s. For example, 0000000011111111 is a thermometer coded signal and it may get shuffled into 0011101010011010. More detailed explanation of DEM is given in Chapter 4.

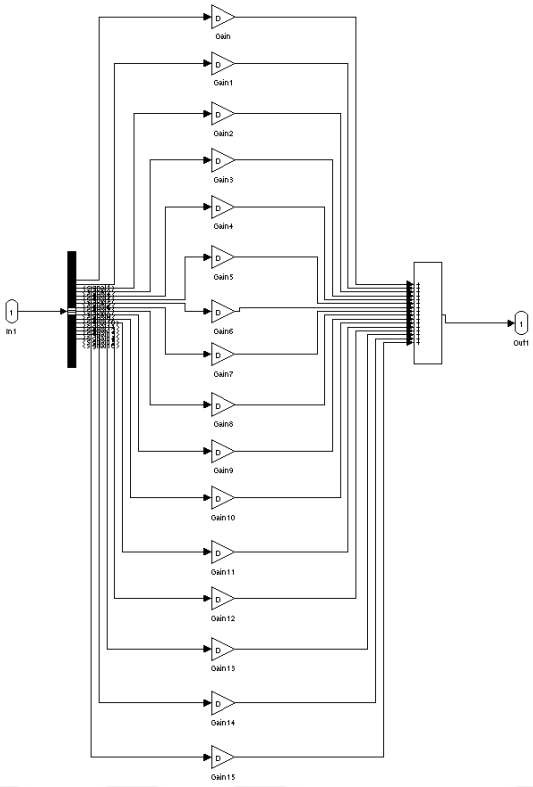


Figure 3.8: Feedback DAC Simulink Model.

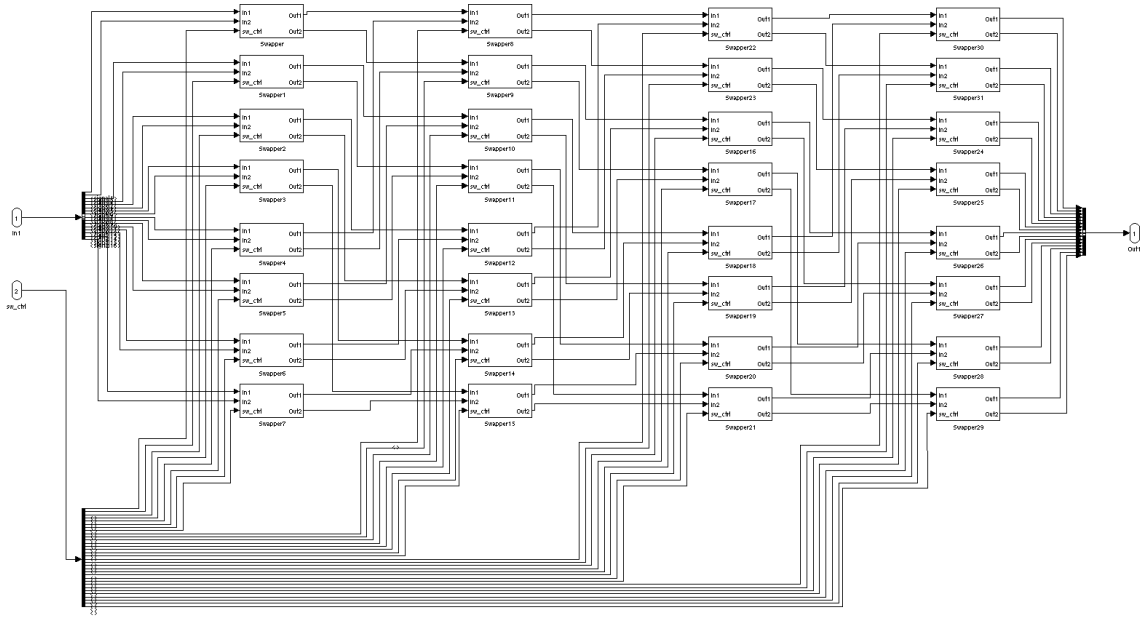


Figure 3.9: Dynamic Element Matching Block.

3.6 V_{pp} amplitude sine wave at 33.5 kHz with 4096 samples is applied as input and signal to quantization noise ratio (SQNR) of the system is calculated as 89.5 dB and ENOB is calculated as 14.58 bits. Figure 3.10 shows the PSD of the output signal. In the first model, there was no DC at input but in this model, there is a DC value at input because of the structure of Flash ADC model. Still this model shows the noise shaping is working quite well.

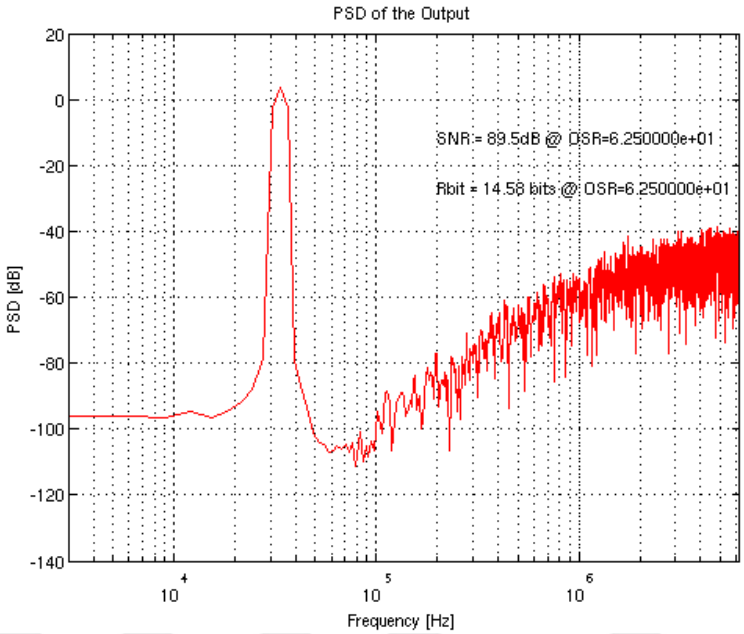


Figure 3.10: Power Spectrum Density of Second Model.

3.3.3 Model with integrator non-idealities

In the previous models integrators were ideal integrators that their gain and bandwidth were infinte. However in real circuits, integrators have finite gain, bandwidth and slew rate. These effects also must be modelled to have an accurate model. Figure 3.11 shows the integrator model with non-ideal effects. This model is taken from the Data Converter book by Franco Maloberti [8].

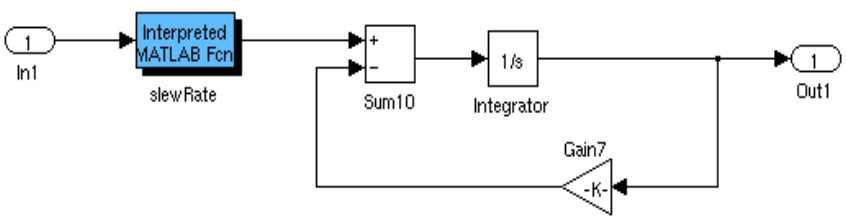


Figure 3.11: Integrator with Non-idealities.

Interepreted MATLAB Function whose code is given in Appendix-B, models the gain bandwidth product and slew rate of the integrator and the feedback part models the gain. Other blocks of the model is the same as the second model. Figure 3.12 shows the power spectral density of the modulator output with non-ideal integrator.

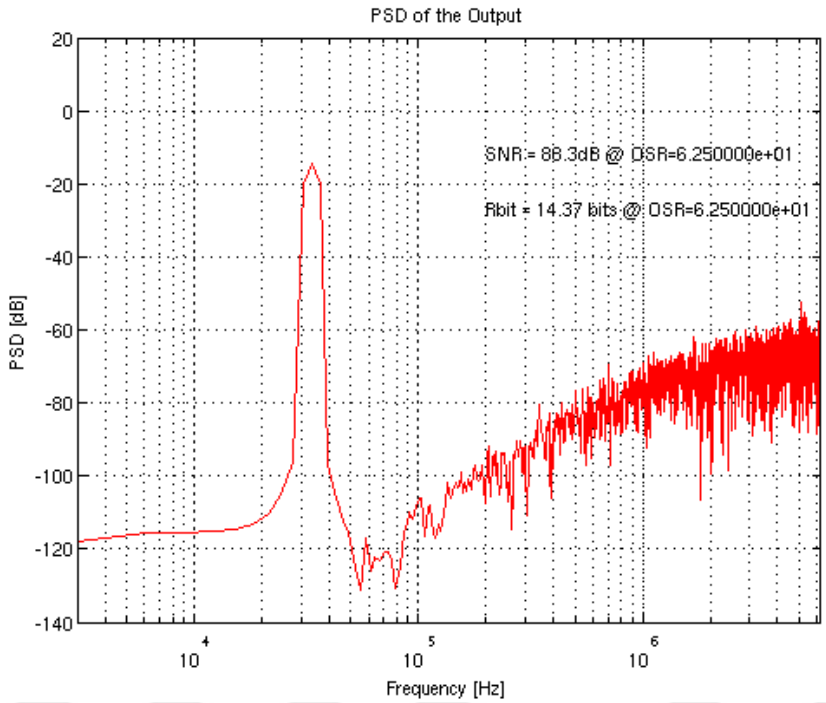


Figure 3.12: Power Spectrum Density of Third Model.

Parameters of the non-ideal integrator are selected as a gain of 1000, a GBW of 30 MHz and a slew rate of 2 V/us. These values are the expected performance specifications of the operational amplifiers in the integrators. Input amplitude is 3.6 V_{PP}. Simulation results for these parameter set are given in Figure 3.12. There is 1 dB loss in SNR and if gain and GBW values are selected lower, this loss would increase.

3.4 Calculation of Resistor and Capacitor Values of Loop Filter

In the previous sections, NTF and STF of delta sigma ADC is designed and modeled in Simulink with additional blocks. Simulation results show that the performance of the ADC satisfies the system requirements. Next step is implementation of the system. Since it affects the performance directly by noise shaping, the most important block is the loop filter as far as implementation is concerned.

Similar to the Simulink model, loop filter in schematic level is based on the integrators and feedbacks. There are many ways to realize an integrator such as active-RC, gm-C, etc. In this design, active-RC integrators are used because they are proper for resistive feedback as well as current mode feedback.

As mentioned in the first chapter, feedbacks of the loop filter are produced by current mode DACs. However, it is easy to maintain stability and other issues of the loop filter with resistive feedbacks. Therefore, first a resistive feedback structure is designed and then they are replaced with DACs that satisfy these resistor values in current mode, as shown in next section.

First of all, transfer function of third order loop filter with active-RC based integrators and feedback resistors is expressed as (3.8). Then, transfer function of Simulink model with ideal integrators and coefficients as gain blocks is determined as (3.9). After that, an equation set (3.10), (3.11) and (3.12) is found out which shows the relation between coefficients in Simulink model and resistor and capacitor values in schematic model. Lastly, resistor and capacitor values are scaled for values that suit integrated circuits better (for example, capacitors are less than 5 pF) by using this equation set.

Transfer function of the Figure 3.13 is shown in (3.8). In Figure 3.13, phases of the feedback loops are set regarding to satisfy the polarity of the equations (3.8) and (3.9). There's need to have some negativities in the equation (3.8) to get proper R and C values. This is the reason for the change in feedback polarities.

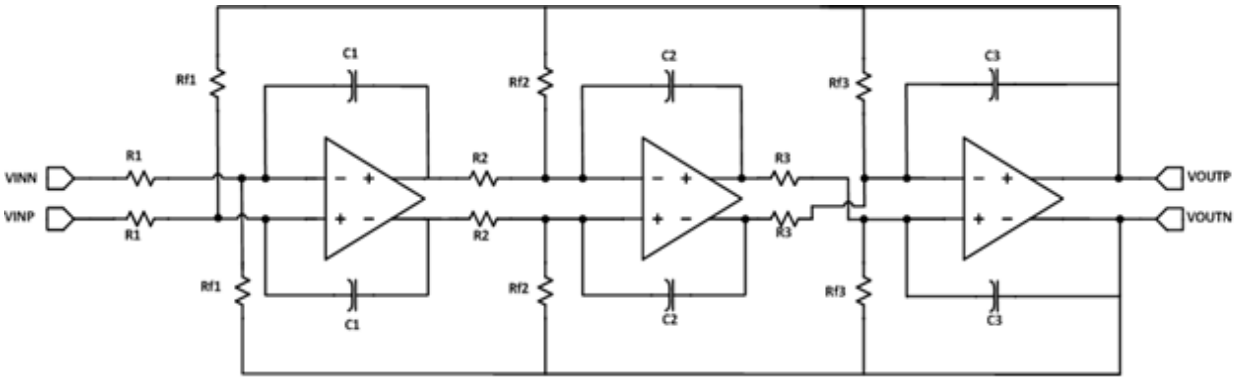


Figure 3.13: Ideal Schematic of Loop Filter.

$$\frac{V_o}{V_i} = \frac{1}{\frac{R_1}{R_{f1}} + \frac{sC_1R_1R_2}{R_{f2}} + \frac{s^2C_1C_2R_2R_3}{R_{f3}} + s^3C_1C_2C_3R_1R_2R_3} \tag{3.8}$$

Transfer function of Figure 3.14 is shown in (3.9).

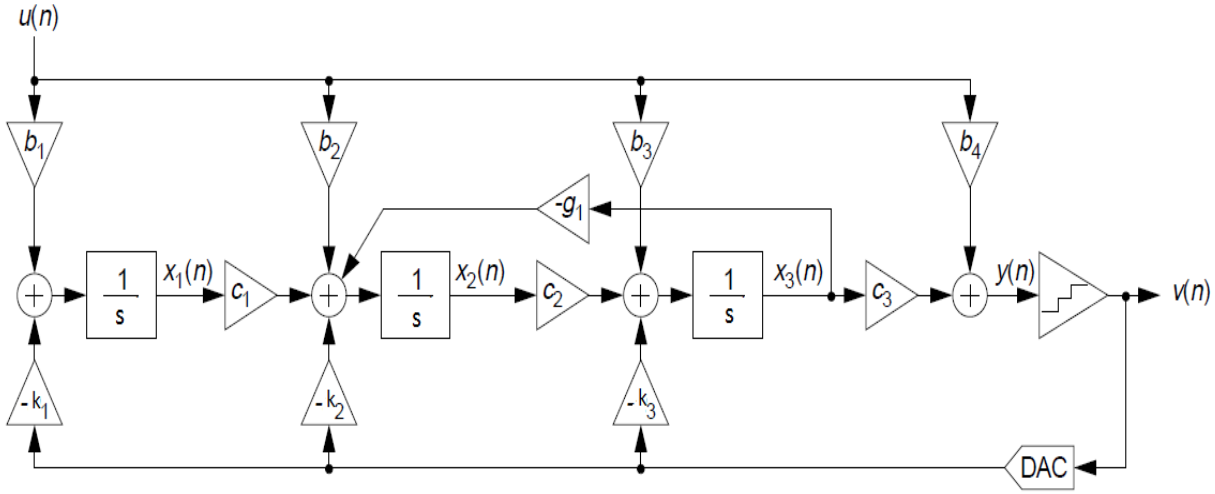


Figure 3.14: Loop Filter Structure.

$$\frac{V_o}{V_i} = \frac{\frac{b_1}{k_1}}{1 + \frac{s}{F_s}\frac{k_2}{k_1} + \frac{s^2}{F_s^2}\frac{k_3}{k_1} + \frac{s^3}{F_s^3}\frac{1}{k_1}} \tag{3.9}$$

After getting transfer functions, an equation set in (3) is obtained by equating the coefficients of the transfer functions together.

$$R_{f3}C_3 = \frac{1}{k_3 F_s} \tag{3.10}$$

$$\frac{R_{f3}}{R_{f2}}C_2R_2 = \frac{k_3}{k_1 F_s} \tag{3.11}$$

$$\frac{R_{f2}}{R_{f1}}C_1R_3 = \frac{k_2}{k_1 F_2} \tag{3.12}$$

Table 3.3 shows resistor and capacitor values that are calculated using the equation set above.

Table 3.3: Resistor and Capacitor Values.

Element	Value
R_1	125 kΩ
R_2	190 kΩ
R_3	100 kΩ
R_{f1}	125 kΩ
R_{f2}	250 kΩ
R_{f3}	250 kΩ
C_1	5 pF
C_2	3 pF
C_3	300 fF

After calculating the resistor and capacitor values, AC simulation of the filter with resistive feedback loops is performed. Figure 3.15 shows the bode plot of the filter and it can be seen that the system is stable and DC gain of the system is 1.

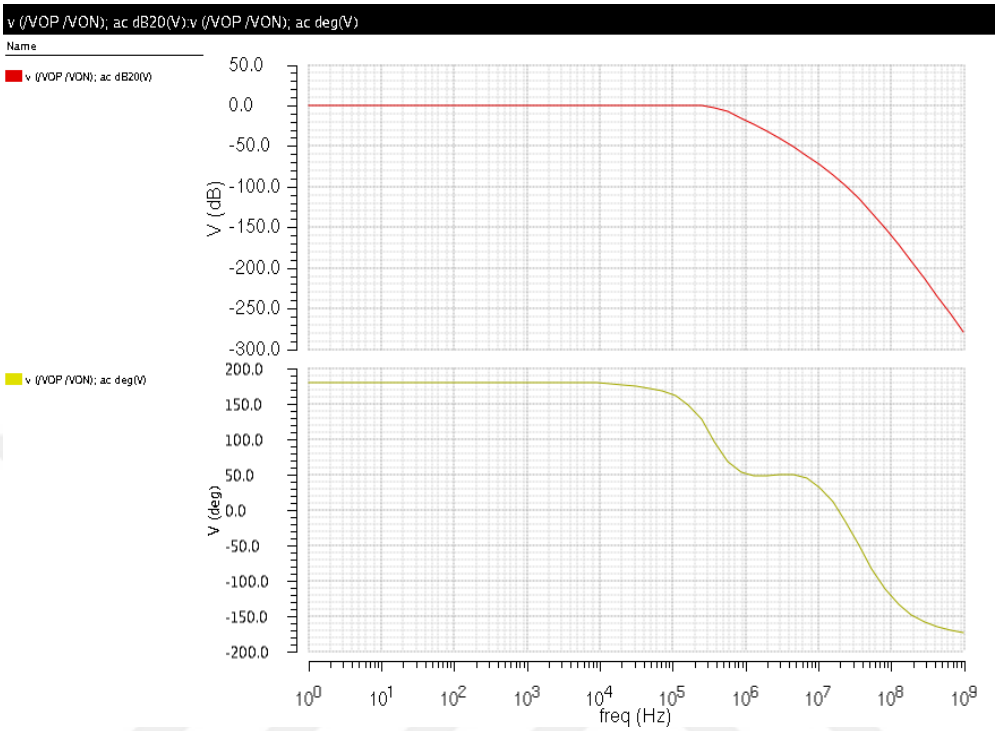


Figure 3.15: AC Response of the Loop Filter.

Transient response of the loop filter is important to see output swing of the loop filter. However, since ideal operational amplifier model does not model the output swing limitations, transient simulation of ideal model of loop filter is not performed.

3.5 Ideal Circuit Model

In previous section, resistor and capacitor values of the loop filter is calculated to realize the gain coefficients in Simulink model. In the real circuit, there will be current DACs instead of feedback resistors. Therefore, ideal schematic model of the whole system is necessary to implement these current DACs as feedbacks of the loop filter.

Feedback resistors are removed and a 4-bit flash ADC is added as quantizer to the output of the loop filter. Then, these thermometer coded bits are converted to analog currents by a Verilog-A code with voltage controlled current sources. These currents feed the loop filter instead of the calculated resistors. Of course, transconductance values of the dependent

current sources are calculated by dividing the voltage difference between full scale voltage and common mode voltage to the resistor values, in the equations given below in (3.14) and (3.15).

$$g_{m1} = \frac{V_{FS}-V_{CM}}{R_1} = \frac{4V-2V}{125k\Omega} = 2\ uS \tag{3.14}$$

$$g_{m2,3} = \frac{V_{FS}-V_{CM}}{R_{2,3}} = \frac{4V-2V}{250k\Omega} = 1\ uS \tag{3.15}$$

Figure 3.16 shows the ideal schematic model of delta sigma ADC.

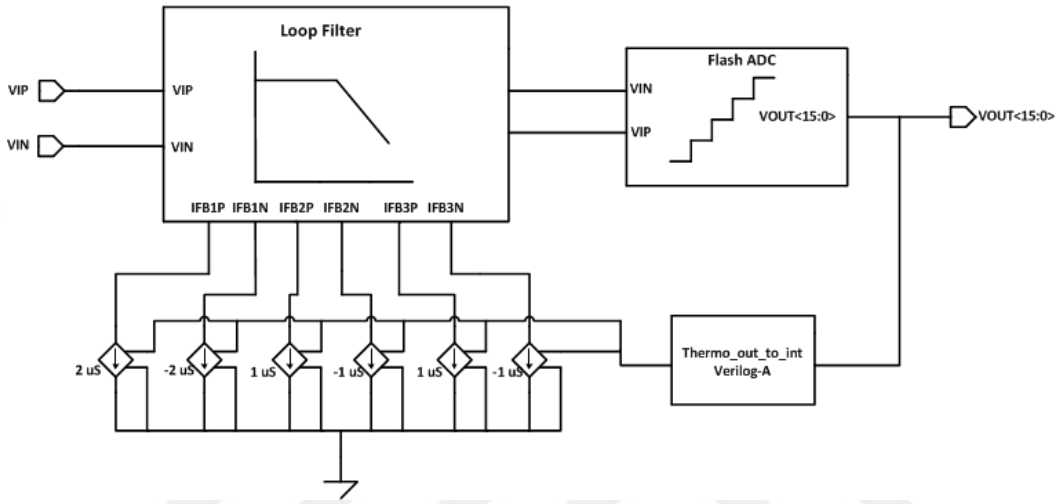


Figure 3.16: Ideal Schematic Model.

Next, structure of sub-blocks, loop filter, flash ADC and feedback DAC, are explained.

3.5.1 Loop filter

Schematic of loop filter similar to feedbacks resistor one explained in the previous section. The only difference is that feedbacks come from outside as currents. Figure 3.17 shows the schematic of the loop filter.

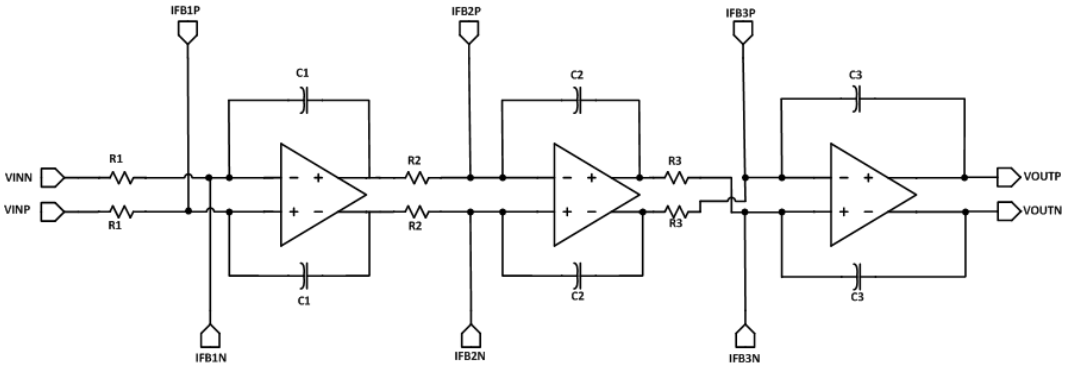
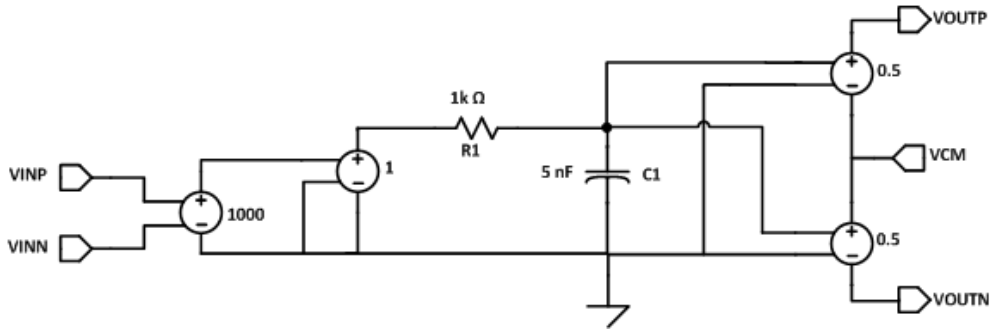


Figure 3.17: Schematic of Loop Filter.

Operational amplifiers that used in loop filter are simple ideal ones that only model gain, gain-bandwidth product and differential output. Gain and bandwidth of this amplifier is set 60 dB and 30 MHz respectively, to become close to the planned specifications of real one. Figure 3.18 shows the schematic of the ideal operational amplifier model (b) and its bode plot (a).



(b) Schematic.

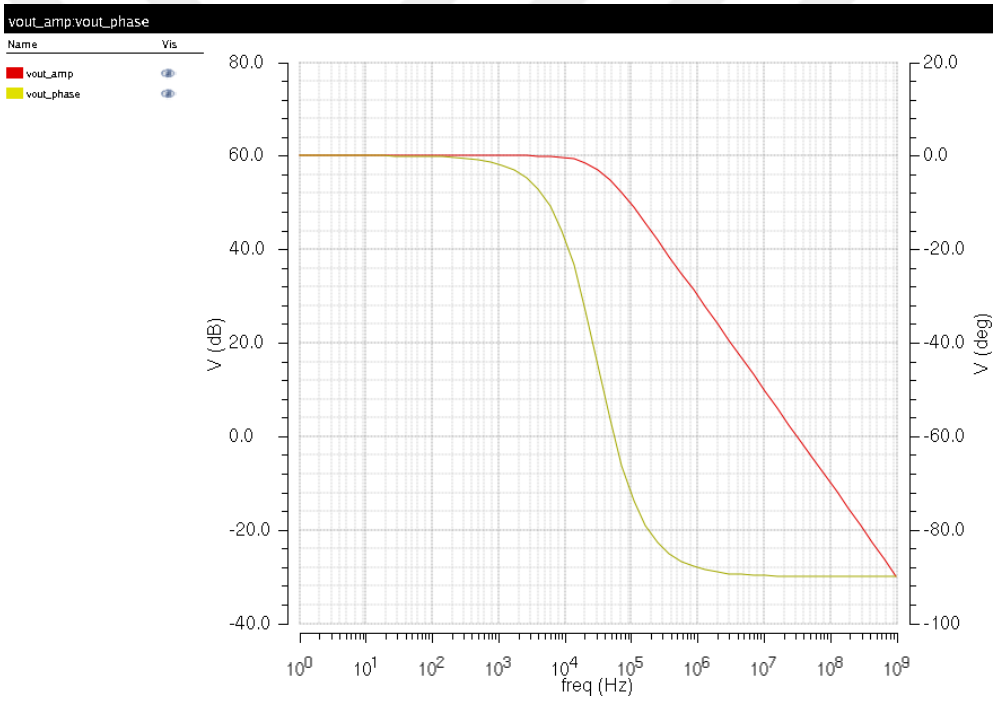


Figure 3.18: (a) Bode Plot.

3.5.2 Quantizer

A simple flash ADC consists of resistor string that produce reference voltages and comparators that generate digital output bits. In the ideal schematic model of flash ADC, ideal resistors are used as voltage dividers to produce reference voltages, just like the same way with real flash ADC, and voltage controlled voltage sources are used as comparators. Input

signal is connected to plus input terminal of voltage controlled voltage source (VCVS) and reference voltages are connected to minus input terminal of VCVS the difference between input terminals are multiplied by 1000 and the output terminals of VCVS is limited by 4V and 0V. Lastly, clocked output latches sample the outputs at the f_s rate. Figure 3.19 shows the schematic of model.

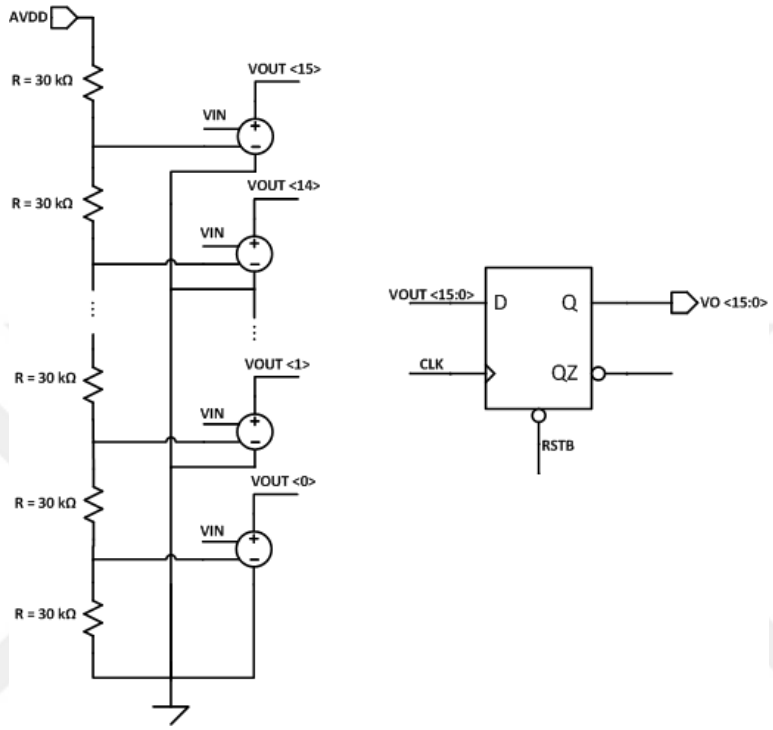


Figure 3.19: Ideal Schematic of Flash ADC.

3.5.3 Feedback DAC

Feedback currents are produced by a Verilog-A code and voltage controlled current sources. Verilog-A code basically converts the thermometer coded digital bits into an integer and that integer is multiplied by transconcutance values of the voltage controlled current sources that determine the feedback currents. This Verilog-A code is given in Appendix-C. Transconcutance values of current sources are the same as those calculated in previous section.

3.5.4 Simulation results

After full delta sigma ADC is prepared from these sub-blocks, transient simulation is run for 4096 samples, 33 kHz input sine wave that satisfies the coherent sampling with 11 period and

3.2 us for system settling time added to simulation time which is 330.88 us in total. After simulation is completed, thermometer coded output bits are summed and written into a text file. Then, this file read by MATLAB and SNR calculation is done by the MATLAB code which also used executed the Simulink model. This script is presented in Appendix-B. PSD of the simulation result is given below in Figure 3.20. More than 100 dB SQNR and 16-bit ENOB are achieved in this model. If total SNR loss due to implementation is assumed around 20 dB, it shows that this design would still satisfy the specifications.

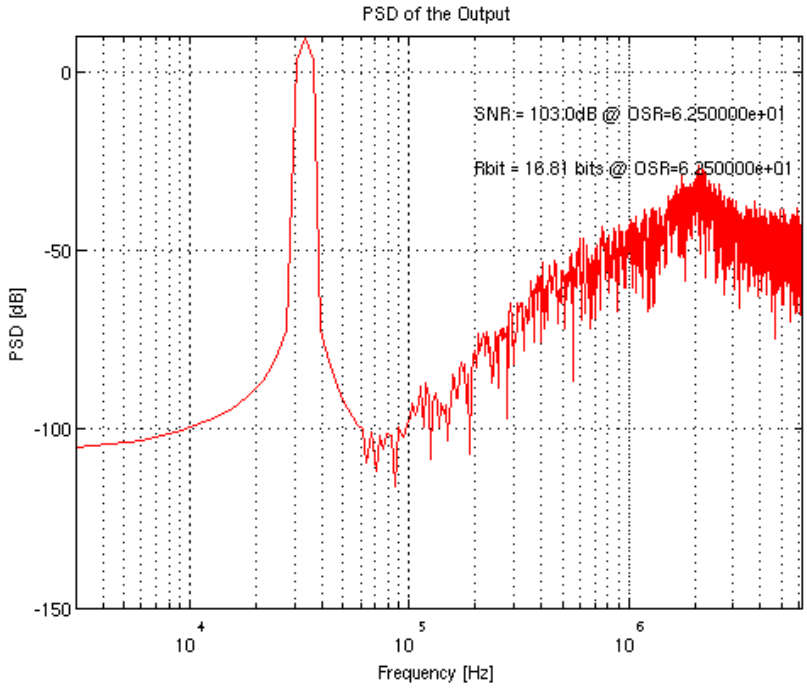


Figure 3.20: Power Spectrum Density of Ideal Schematic Model.

4. SCHEMATIC

What mentioned in this report until here are only ideal blocks and their simulations. Ideal models are important to understand the system principles and avoiding fundamental problems before they affect the next steps. In this section transistor level implementation issues are investigated and circuit level design considerations are explained.

Figure 4.1 shows the top level schematic of delta sigma ADC circuit. Loop filter and flash ADC are similar to ideal circuits, the only difference is that using real circuit elements provided by the foundry instead of ideal circuit elements. Feedback DAC is also implemented with real elements in the form of current DACs. There is also dynamic element matching block just before feedback DAC to reduce the mismatch based errors of DAC. These sub-blocks are explained briefly in next sections.

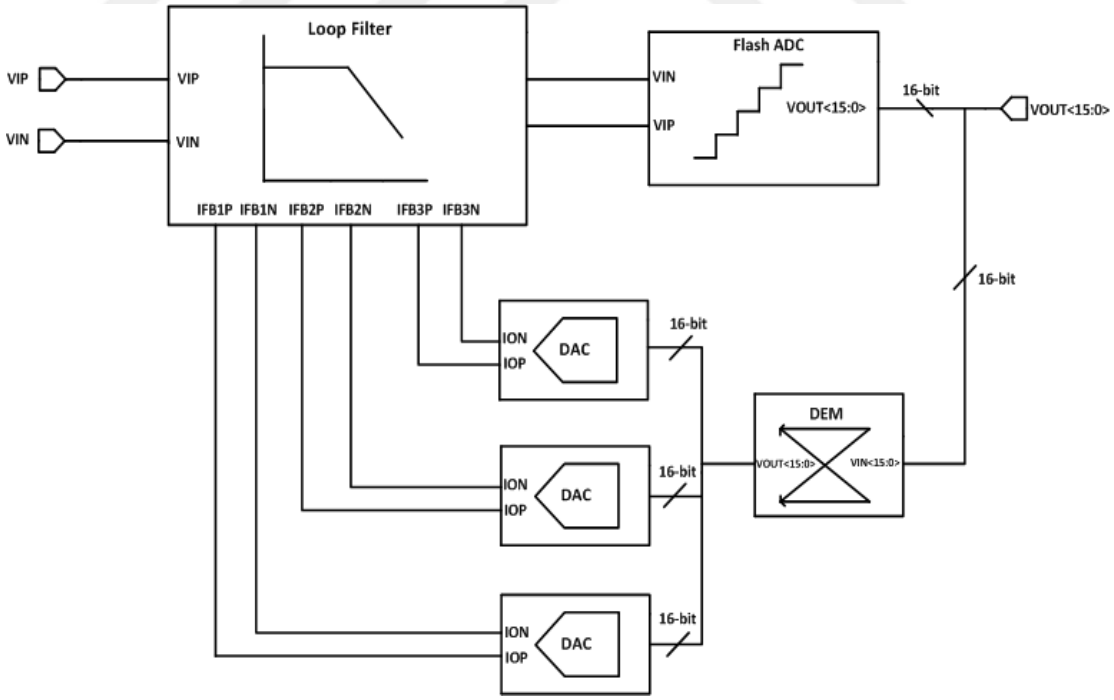


Figure 4.1: Top Level Schematic of Delta Sigma ADC.

For a 3.6 V_{PP} sine wave at 33 kHz frequency, power spectrum density of the output is given below in Figure 4.2. SNQR is calculated as 85.9 dB and 13.97 bits ENOB.

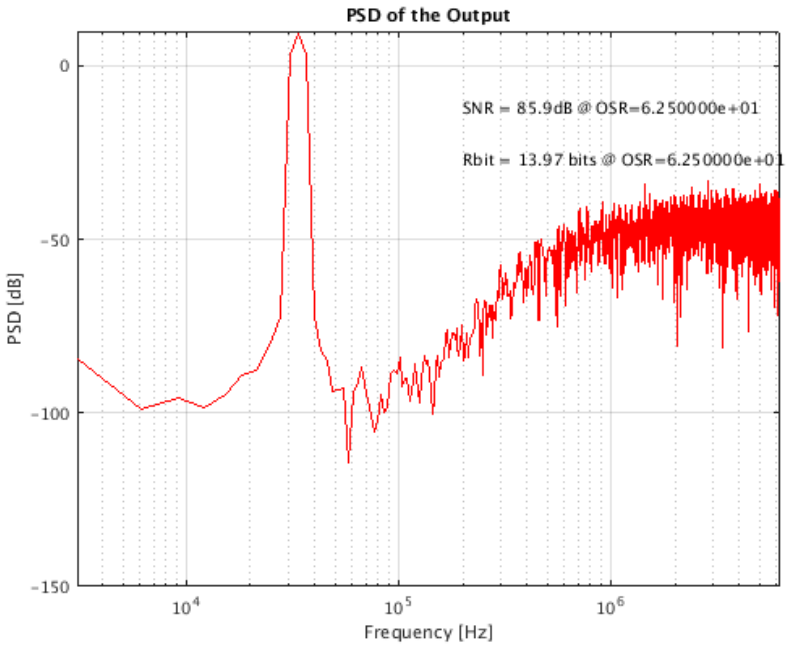


Figure 4.2: Power Spectrum Density of ADC Output.

4.1 Loop Filter

Resistor and capacitor values and calculations are given in previous chapter. These values are used in ideal block of loop filter and simulated. Here same capacitor and resistor values are used in real blocks with real resistor and capacitors and non-ideal operational amplifier. These capacitors, resistors and transistors are not ideal. Figure 4.3 shows the schematic of loop filter in current feedback configuration.

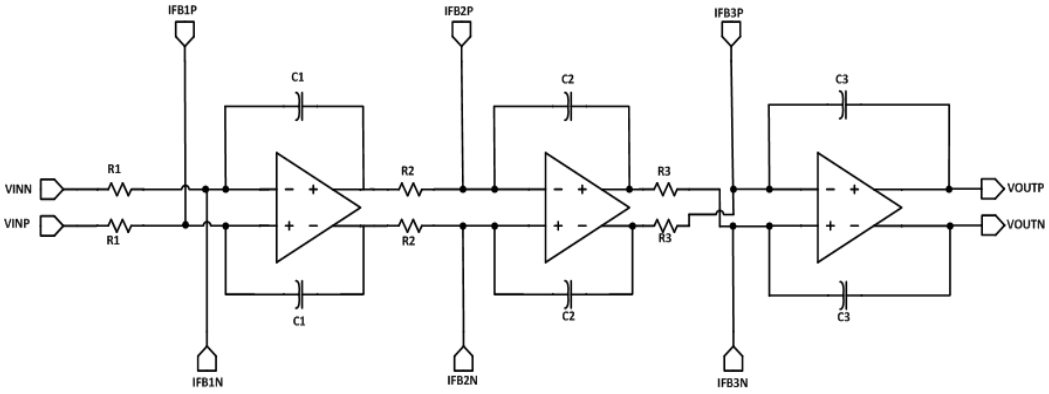


Figure 4.3: Schematic of Loop Filter.

Simulation of this block has been done in resistive feedback configuration. One reason of this is that easy test setup and the other one is that both configurations are quite similar as far as stability is concerned. AC and transient simulation results of the loop filter is given in Figure 4.4 and Figure 4.5, respectively.

This Bode plot in Figure 4.4 shows that the loop is stable and DC gain is 1 as expected. The bandwidth is also more than 100 kHz which is our design bandwidth.

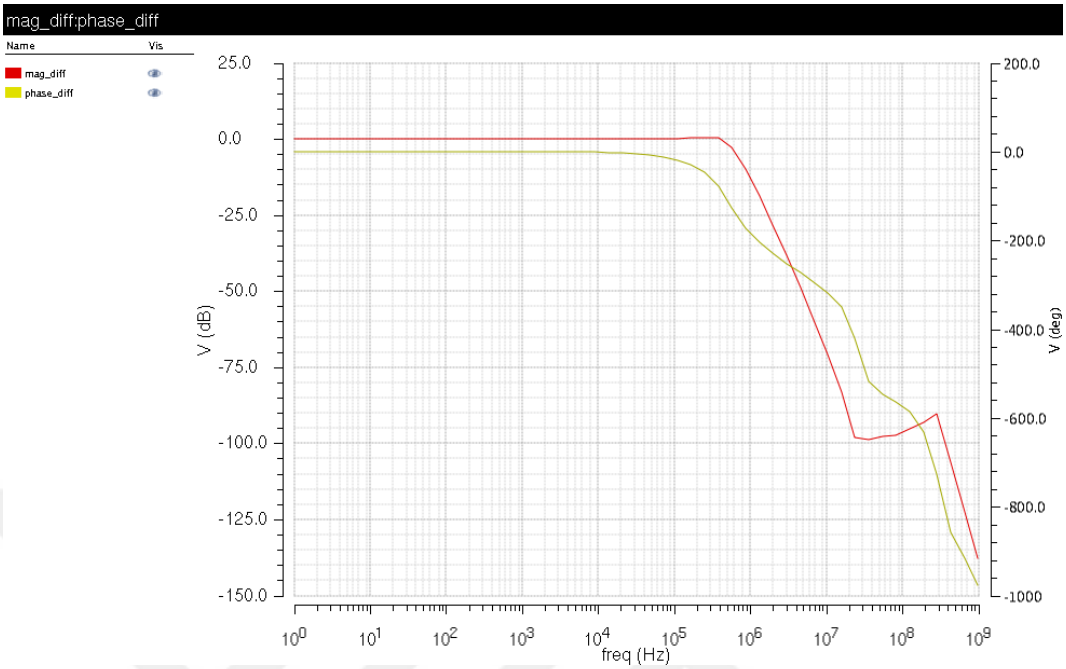


Figure 4.4: AC Simulation Result of Loop Filter.

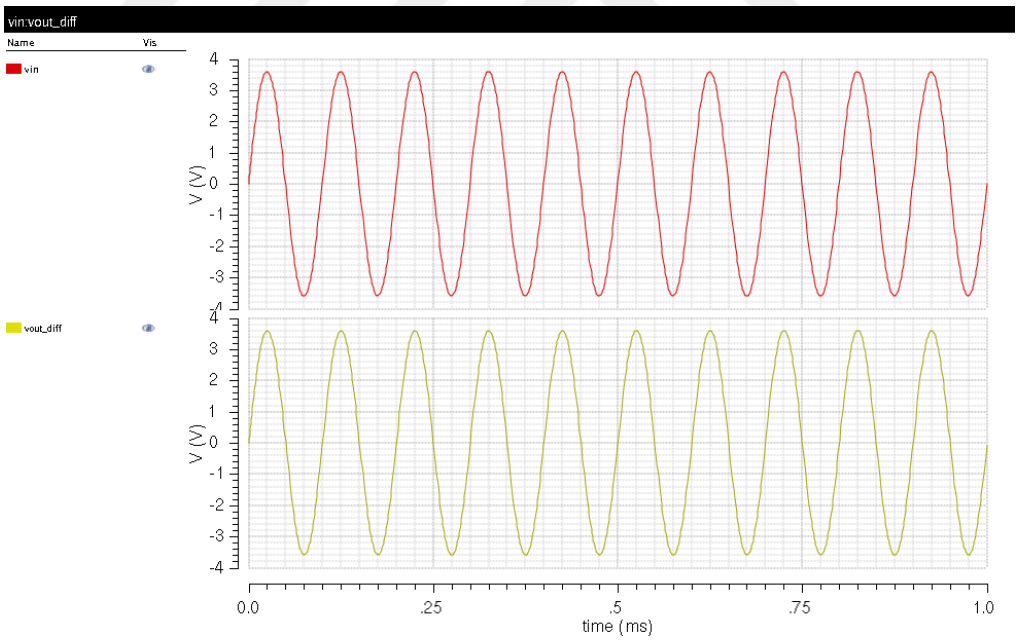


Figure 4.5: Transient Simulation Result of Loop Filter.

For transient simulation, 3.6 V_{PP} and 10 kHz frequency sine wave is applied to input differentially which doubles the amplitude. The main reason of transient simulation is to see the swing at the output. There is no clipping as it can be seen. The ratio of feedback and feed-through resistor values are important to avoid clipping at the

output. Proper values are selected to satisfy both design equations and full swing at the output.

4.1.1 Operational amplifier

Integrators of the loop filter are implemented as active-RC integrators. Active-RC integrators are based on operational amplifiers with a feedback capacitor and a resistor at the input. To satisfy more ideal integrator behavior, DC gain and bandwidth of amplifier must be as high as possible. For delta sigma ADC applications, bandwidth must be at least twice the sampling frequency. Hence for our design a target GBW of 25-30 MHz is good enough. Amplifier gain of 80-90 dB is also proper for the integrator.

Folded cascode structure in fully differential mode with common mode feedback is preferred to satisfy previously mentioned specifications. Self-biased, high swing and high output impedance biasing circiut is used to bias the necessary nodes in amplifier and also in common mode feedback.

Figure 4.6 shows schematic of folded cascode operational amplifier and Figure 4.7 shows the simulation test bench. Ideal baluns and current probes are used to break feedback loop and run stability analysis. Stability and transient simulation results are given below in Figure 4.8 and Figure 4.9, respectively.

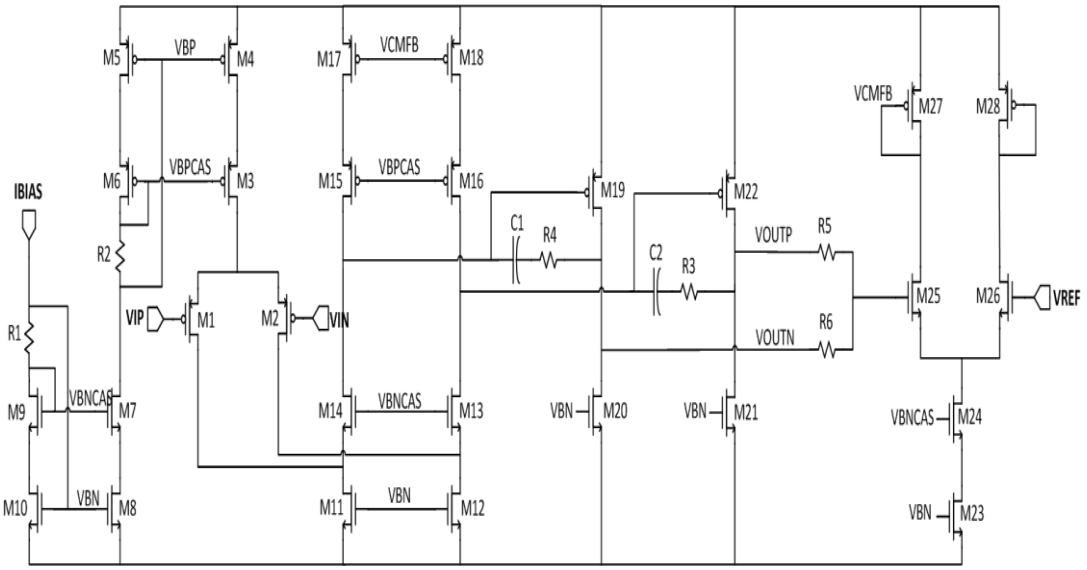


Figure 4.6: Folded Cascode Fully Differential Opamp with Common Mode Feedback.

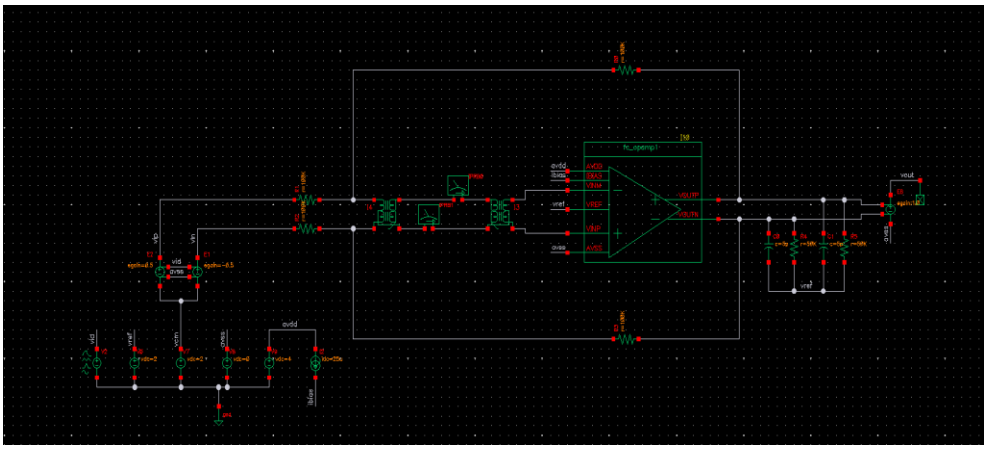


Figure 4.7: Simulation Test Bench.

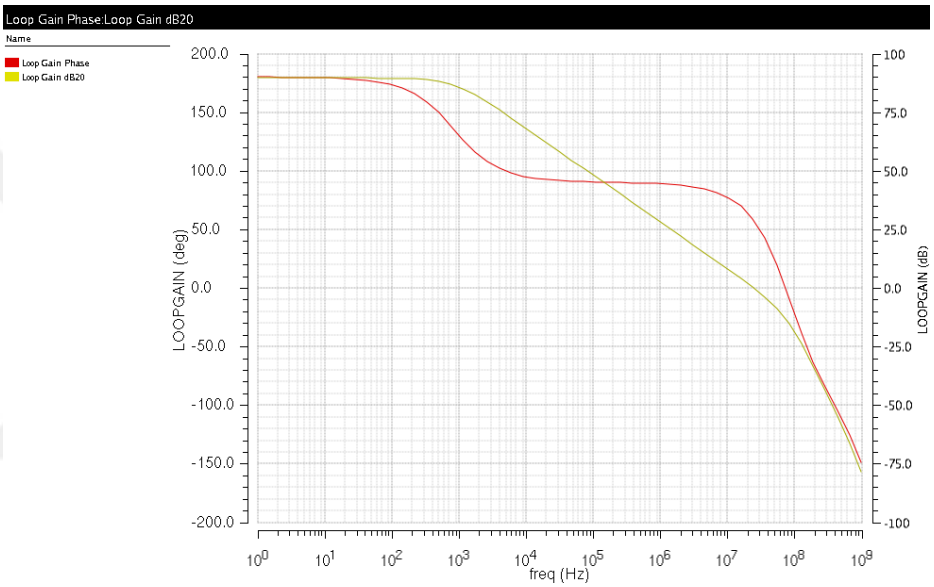


Figure 4.8: Stability Simulation Result.

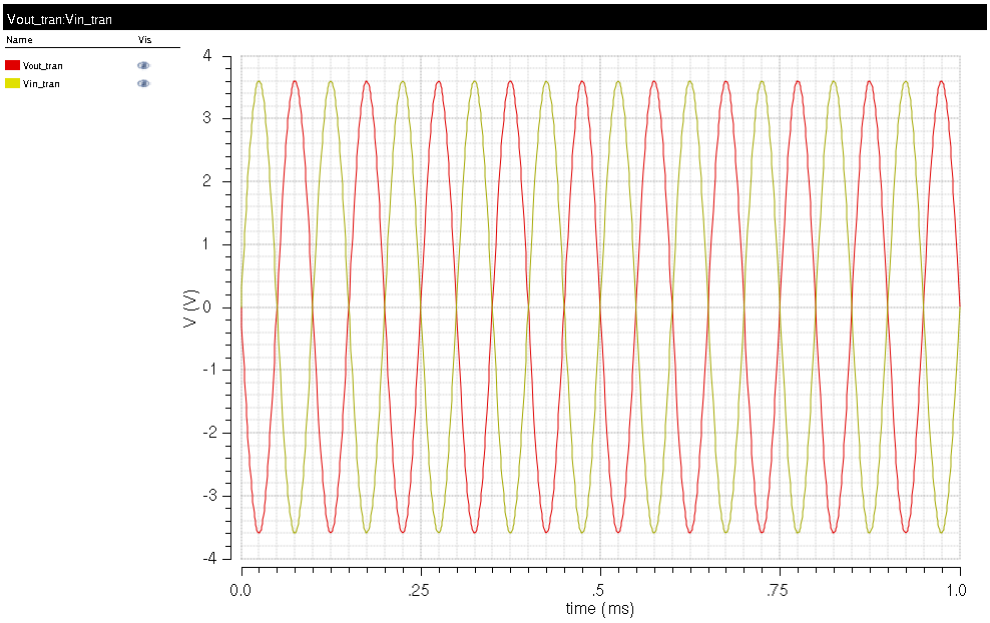


Figure 4.9: Transient Simulation Result.

4.1.2 Noise analysis

In delta sigma ADCs quantization noise is the most important element that determines the performance of the ADCs. In addition, thermal and flicker noise are also important so that they may limit the total performance of the ADC.

Since loop filter is the first block that takes the analog input signal, noise performance of this block is very important. It is mentioned before that loop filter consists of three active-RC integrators and first of these integrators are designed differently from others due to its noise performance which has to be greater than second and third integrators.

Figure 4.10 shows the input equivalent noise plot of loop filter in terms of V/sqrt(Hz). Input equivalent noise shows the noise performance of the circuit as it can be considered that there is a additional noise signal over the input signal. Simulator calculates the total output noise and divides it to transfer function of the circuit, basically.

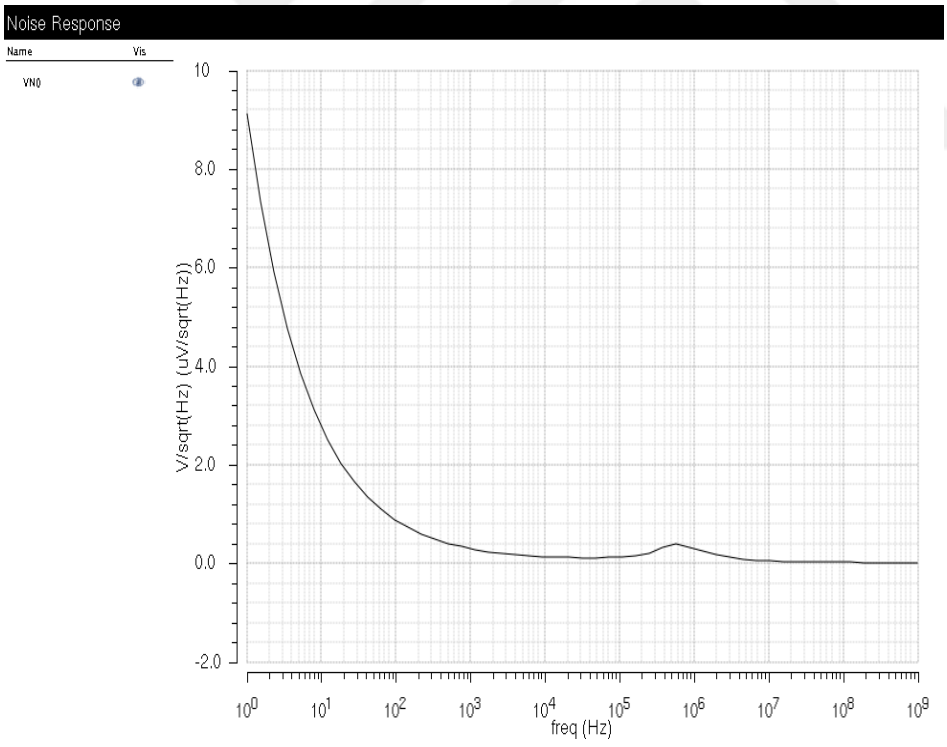


Figure 4.10: Input Noise Equivalent Plot.

The integration of this plot over the band of interest gives the total input noise power in those interval. Noise performance of loop filter in 1 Hz – 100 kHz band is given in Figure 4.11. It shows the top sixteen noise contributors in the loop filter. They are transistors in first operational amplifier and resistors in loop filter.

Device	Param	Noise Contribution	% Of Total
/IO/IO/M7	fn	1.74729e-05	14.95
/IO/IO/M6	fn	1.74729e-05	14.95
/R1	rn	1.44567e-05	10.23
/R0	rn	1.44567e-05	10.23
/IO/IO/M3	fn	1.13946e-05	6.36
/IO/IO/M2	fn	1.13946e-05	6.36
/IO/I1/M7	fn	8.01195e-06	3.14
/IO/I1/M6	fn	8.01195e-06	3.14
/IO/IO/M13	fn	6.44649e-06	2.03
/IO/IO/M11	fn	6.44649e-06	2.03
/R3	rn	3.53593e-06	0.61
/R2	rn	3.53593e-06	0.61
/IO/IO/M7	id	3.17513e-06	0.49
/IO/IO/M6	id	3.17513e-06	0.49
/IO/IO/M2	id	2.80936e-06	0.39
/IO/IO/M3	id	2.80936e-06	0.39

Integrated Noise Summary (in V) Sorted By Noise Contributors
Total Summarized Noise = 4.51966e-05
Total Input Referred Noise = 4.51051e-05
The above noise summary info is for noise data

Figure 4.11: Noise Analysis Summary.

Total input referred noise is 4.51051×10^{-5} V/sqrt(Hz) from Figure 4.11. It is converted to dB by the Equation 4.1.

$$20 \times \log(4.51051 \times 10^{-5}) = -86.9\, dB \tag{4.1}$$

This can give some idea about the noise floor of the design. Since the main goal of the design is to have total noise (quantization, thermal and flicker) and distortion level is about 80 dB, this value of thermal and flicker noise can be acceptable.

4.2 Quantizer

In a single-bit delta sigma ADCs, one comparator is enough at the output since there only need for two logic levels basically 1 and 0. However in multi-bit delta sigma ADCs, additional bits at the output increase the SQNR. Therefore if it is proper for the system, multi-bit designs are chosen. This design is also a multi-bit delta sigma ADC whose output has 16 logic level. To achieve this in an easy way, a simple flash ADC is used.

Flash ADCs produce thermometer coded bits and normally those bits are converted to binary coded bits. However, for this delta sigma ADC, there is no need to convert thermometer bits to binary coded bits, since thermometer coded bits are going to be used in feedback DACs.

Figure 4.12 shows the schematic view of the flash ADC. It is simple flash ADC with resistor string for references, comparators for digitizing and output latches to have logic values. The only difference of this flash ADC is its fully differential architecture. Fully differential architecture doubles V_{FS} and increases SNR by 3 dB, to yield better CMRR and noise immunity. However it also needs fully differential comparators and more resistors as drawback.

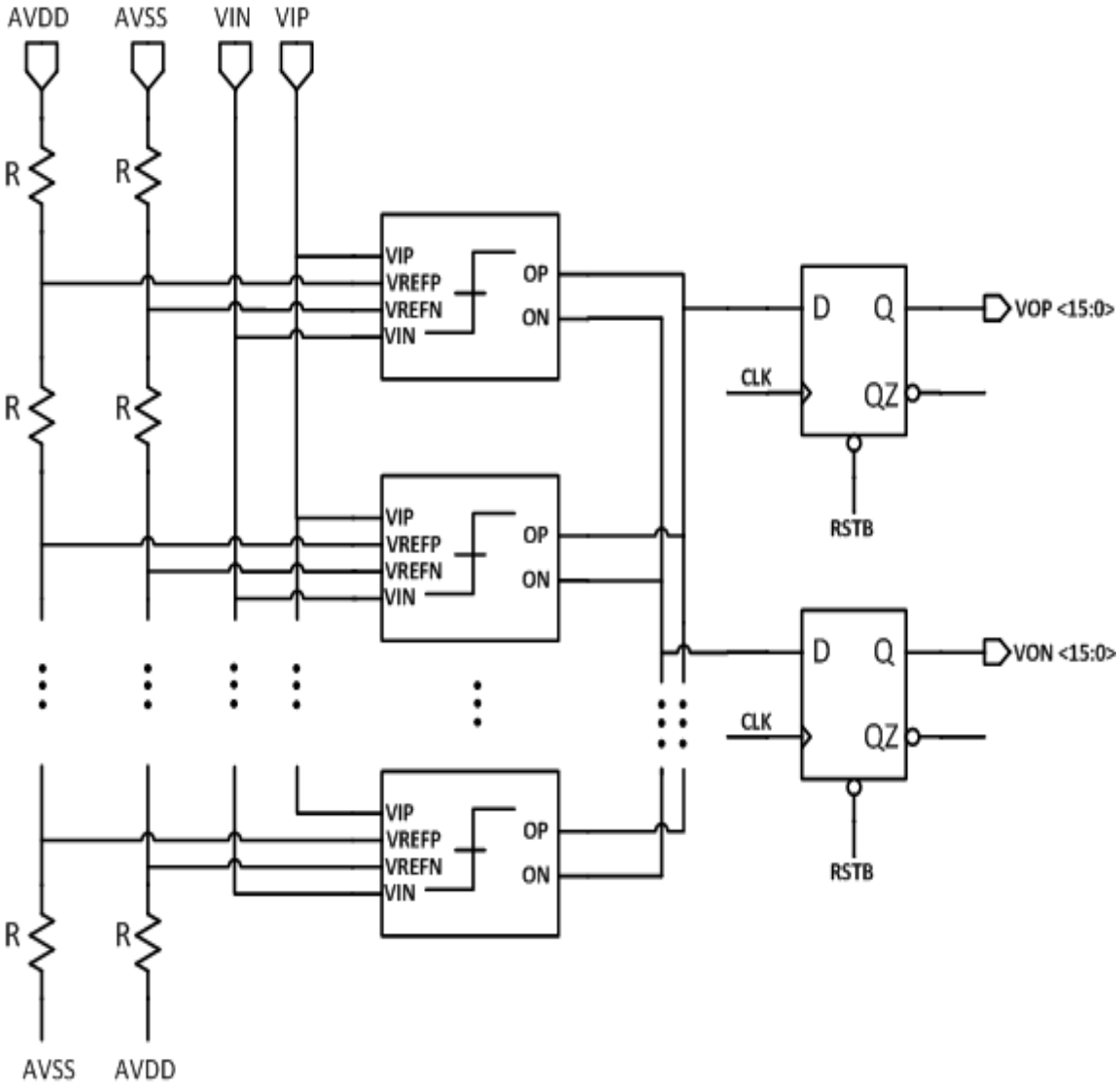


Figure 4.12: Schematic of Flash ADC.

Since the flash is fully differential there are two resistor strings for negative and positive references, resistor values are the same but the only differences is that positive string biased AVDD at the top and AVSS at the bottom and the negative one is vice versa.

For 3.8 VPP amplitude and 50 kHz frequency differential sine wave signal applied to input and flash ADC is simulated. Output bits of flash ADC is shown in Figure 4.13 as differential. Simulation results shows that the flash ADC works very well.

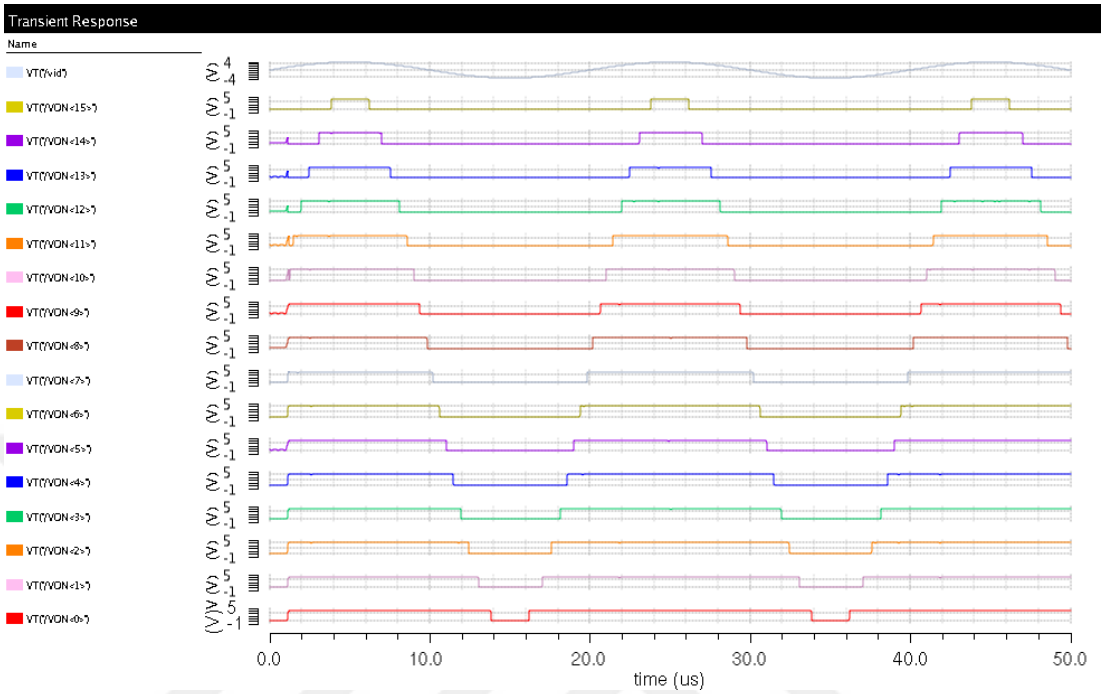


Figure 4.13: Simulation Result of Flash ADC.

4.2.1 Fully differential comparator

As mentioned previously, fully differential comparators are needed for fully differential flash ADC architectures. Schematic of fully differential comparator is given below in Figure 4.14. It is a pre-amplifier and latch based comparator with double balanced fully differential input stage.

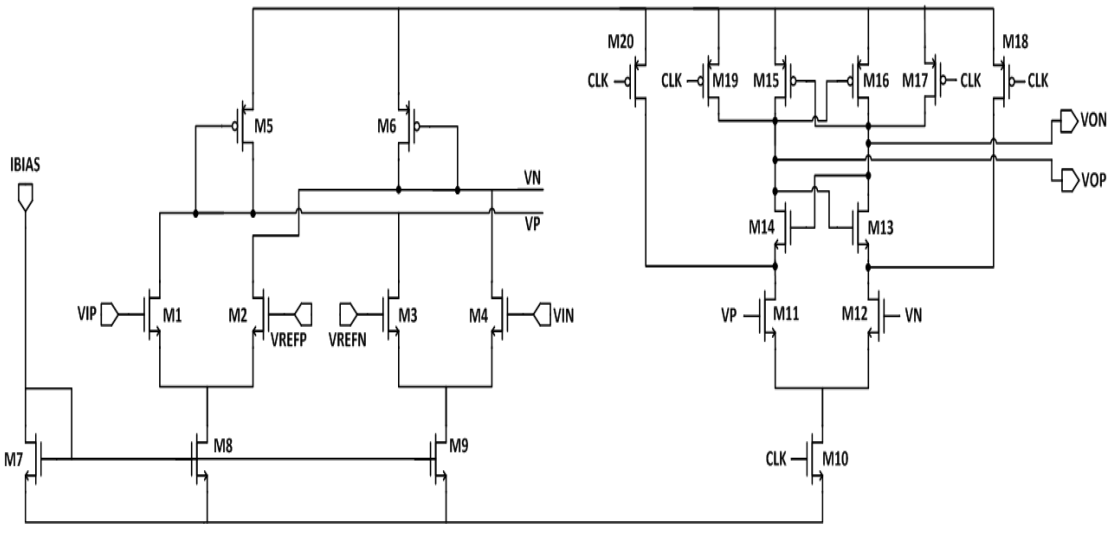


Figure 4.14: Schematic of Fully Differential Comparator.

Razavi's modified storgARM latch is used in this comparator to produce the logic values at the output. The main advantage of this latch is that it does not consume any static power. M11 and M12 switches stop the input propagation during the regeneration phase. Latch also works well with active pull up switch.

Simulations are repeated for various reference voltages which are similar to the flash ADC working principle. However, references set to 2V and 100 mV as a differential input signal at 10 kHz frequency is applied for the simulation results given above in Figure 4.15. Output signal of the comparator is also differential as it changes between 4V to -4V as it can be seen from the figure.

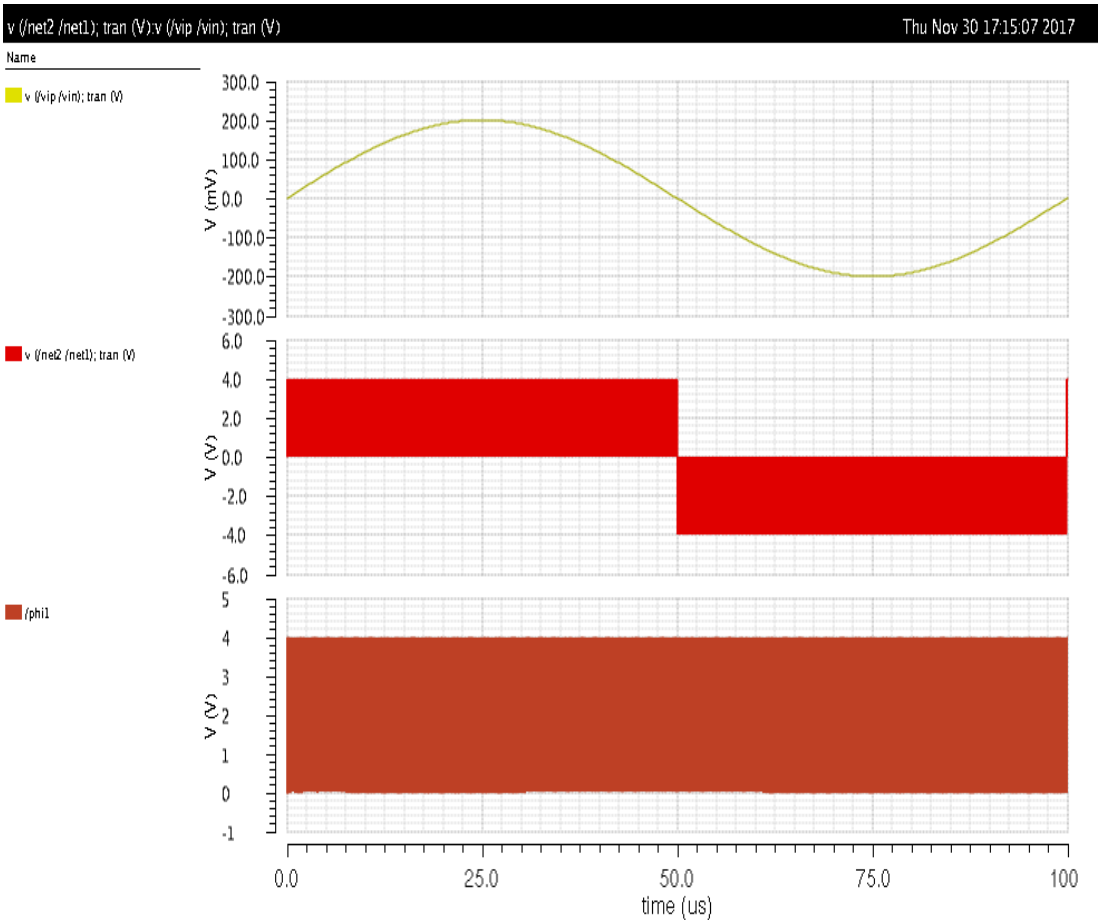


Figure 4.15: Simulation Result of Comparator.

Simulations are repeated for various reference voltages which are similar to the flash ADC working principle. However, references set to 2V and 100 mV as a differential input signal at 10 kHz frequency is applied for the simulation results given above in Figure 4.15. Output signal of the comparator is also differential as it changes between 4V to -4V as it can be seen from the figure.

The problem with this comparator is that its latch at the output does not hold the signal its value while in regeneration phase. In other words, the output signal goes to common mode voltage during regeneration phase because of the switch in the latch. Therefore another latch with opposite clock phase is needed to hold the logic bit values during regeneration phase.

4.2.2 Output latch

A simple output latch is used to solve the problem mentioned in the previous section. It is actually a NAND based D latch and its schematic is shown in Figure 4.16.

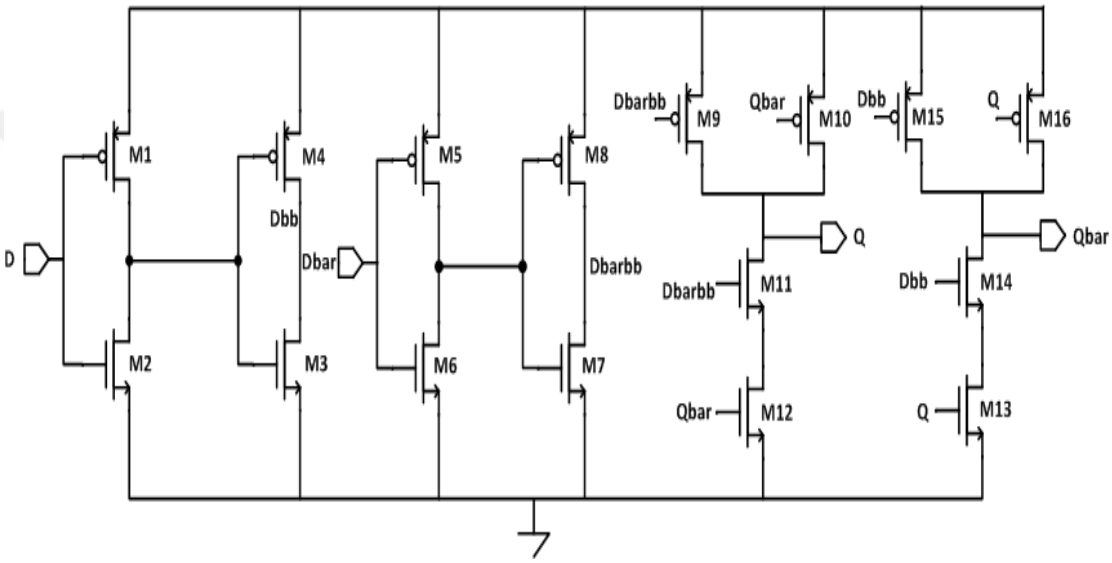


Figure 4.16: Output Latch of Flash ADC.

Output latch works with the inverted clock of the comparator. Therefore, latch in the comparator and this output latch can be considered as an edge sensitive flip flop together.

4.3 Dynamic Element Matching

In unit weighted DACs, mismatches between DAC cells decrease the linearity of the converter. There are some techinques developed to reduce or to cancel these mismatch based effects. If same DAC cells are active most of the time, their mismatch become more dominant. However, if some techniques are applied to spread the active DAC cells over time, then some mismatches can cancel each other and the average mismatch of the DAC cells reduce. These techniques are called dynamic element matching in a general way.

In delta sigma ADCs, multi-bit output designs suffers from DAC mismatch because they need unit element feedback DAC. Since this is a 4-bit design and there is a 16-bit thermometer coded flash ADC output that is fed back to DACs, there's needs for a dynamic element matching block to spread the active bits over the 16 bit signal.

Figure 4.17 shows the scrambling type of dynamic element matching systems. Butterfly connected swappers whose select signal is coming from a randomizer is used. Random bit generator produces 32-bit random bits for 32 swappers' select signals which decide that data will flow wheter one way or the other. For example, if the 16-bit thermometer coded data at the input is 0000000111111111, data at the output can be 1101011011000101 or some other randomly shuffled version of the ADC thermal encoded output. The difference between words doesn't matter for unity weighted DACs as long as the total number of 1s and 0s are equal to each other.

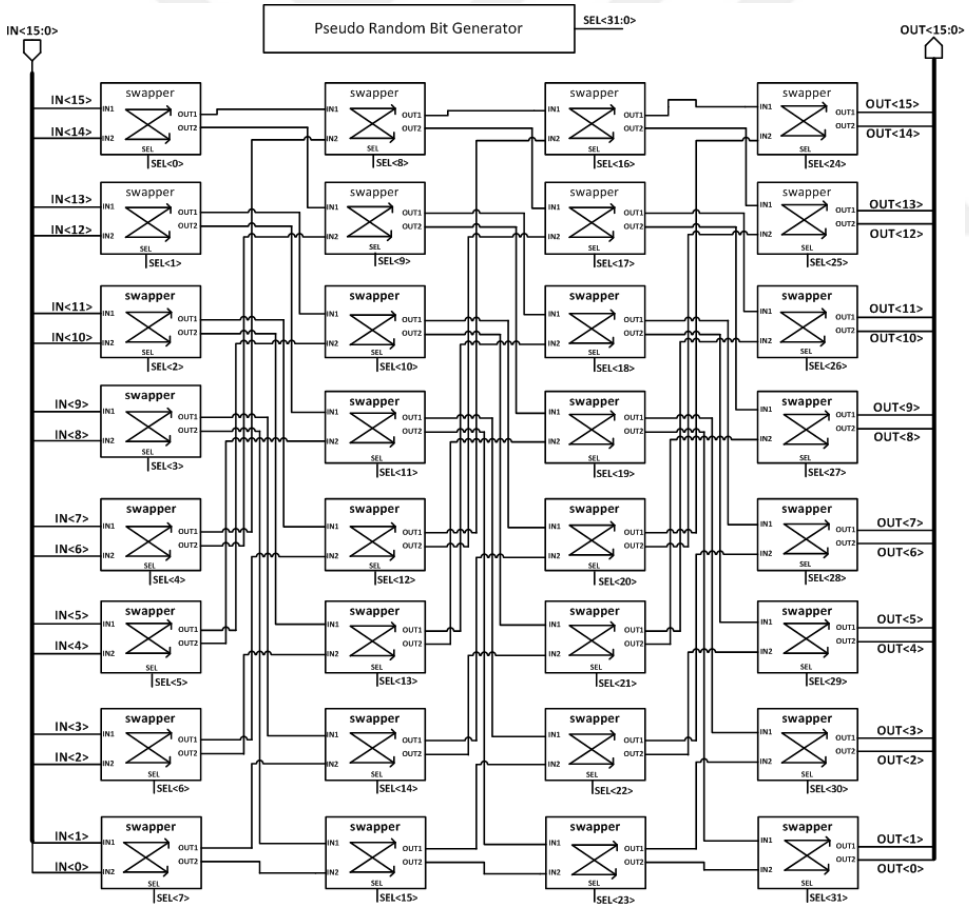


Figure 4.17: Dynamic Element Matching.

All digital cells used in dynamic element matching block are from digital core library of TSMC 0.18u process for easy and better layout.

4.3.1 Pseudo random bit generator

Randomizer is used to select input for swapper block that decides which bit goes where. Therefore, it is important for spread 1s and 0s over 16-bit. Pseudo random bit generator is used to implement randomizer in dynamic element matching block. Figure 4.18 shows the 32-bit pseudo random bit generator that is implemented with shift register and EXOR gates. This is actually called linear feedback shift register.

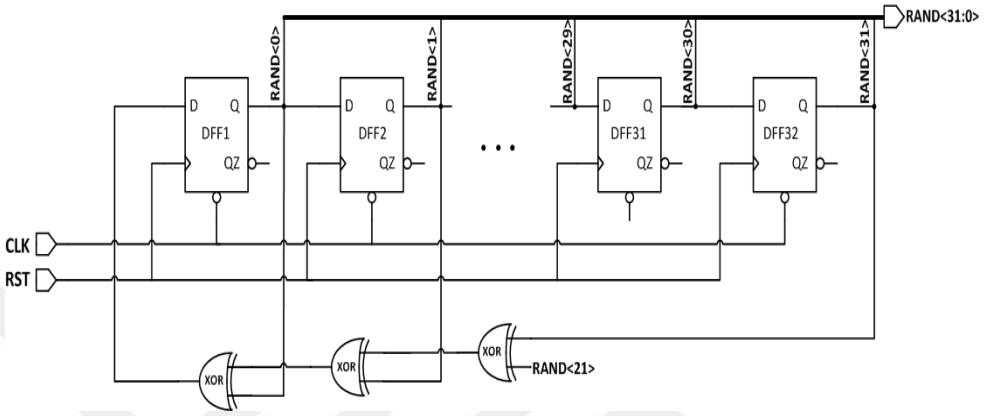


Figure 4.18: Pseudo Random Bit Generator.

The tap off nodes for the XOR inputs are important to make it a randomizer. For a 32-bit output, there has to be a shift register with 32 flip-flops and the feedbacks have to be connected to shift registers that numbered as 31, 21, 1, 0. Output of the shift registers form the 32-bit pseudo random bit.

4.3.2 Swapper

Swappers are the key blocks in dynamic element matching of this design. if SEL signal is 1, it basically does OUT1 = IN1 and OUT2 = IN2, otherwise OUT1 = IN2 and OUT2 = IN1. SEL signal is produced by randomizer and swappers are connected to each in butterfly form. Figure 4.19 shows the swapper block. It simply uses MUX gates in the digital library.

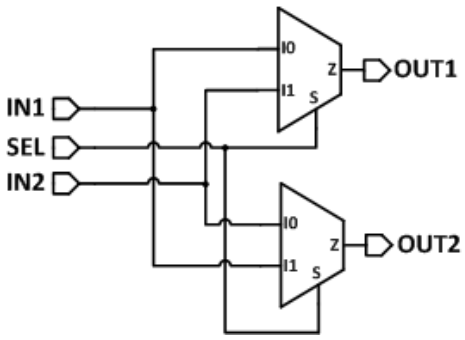


Figure 4.19: Swapper.

4.4 Feedback DAC

In multi bit delta sigma ADCs, loop filter is in analog domain and there needs to be analog feedbacks from the digital output to the loop filter. Since output bits are in digital domain, there must be a digital to analog converter to realize these feedbacks of the loop filter. These can be in current mode or in voltage mode. In this design, current mode is selected and feedbacks are realized by current mode DACs.

Figure 4.20 shows the schematic of current DAC which consist of input latches, DAC cells and a feedback. DAC cells need to get data at the same time. Otherwise, there will be nonlinearities in the output currents. Therefore, input latches are used to synchronize the 16 bit input data with system clock signal. Moreover, inverse of data used in DAC cell switches are produced by input latch.

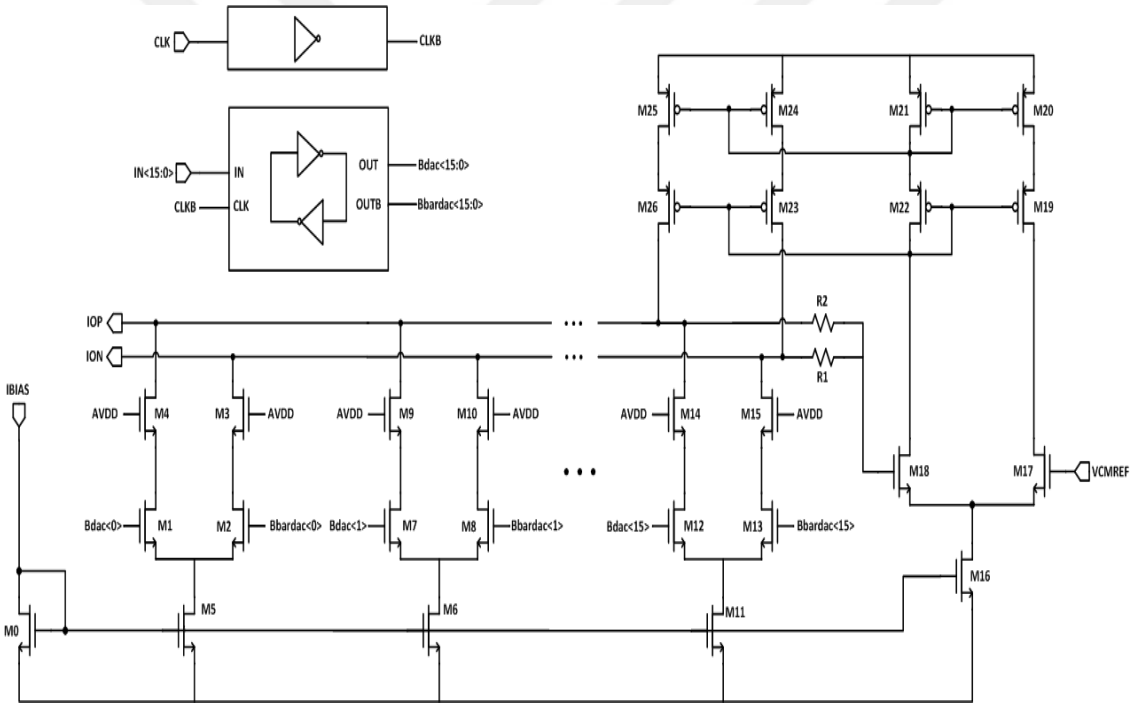


Figure 4.20: Feedback DAC Schematic View.

DAC cell switches are NMOS transistors and controlled by input data and its reverse and if a DAC cell turns on, current starts to flow through the output. Current values of each DAC cell are calculated in ideal schematic model chapter. First feedback's total maximum current has to be 16 uA, second and third ones have to be 8 uA. Feedback part of DAC senses the common mode voltage and produce source currents by PMOS current mirrors to make the total current consumption of sinks equal.

Simulation result in Figure 4.21 shows that positive current decreases and negative current increases by 2 uA every time when a DAC cell turns on. This result is similar to how feedback DAC works in ideal schematic model of ADC.

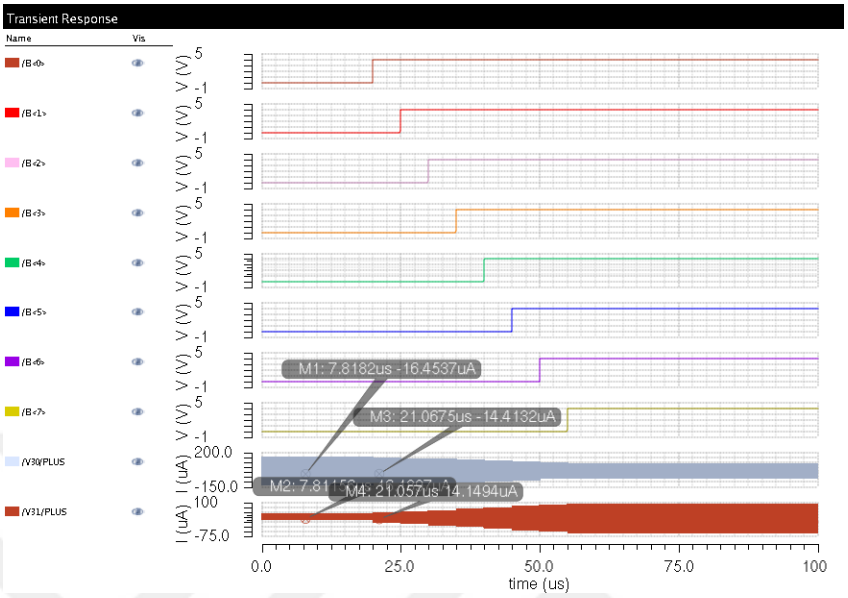


Figure 4.21: Simulation Result of Current DAC.

Since DAC cells are switch based, there are glitches in output current every time when a switch turns on. To reduce these glitches, NMOS cascodes are added to input switches and output signal crossover of input latches are set around 3 V to have smooth transition while a DAC cell switch is turning on, also known as make before break switching. (see Figure 4.25). Figure 4.22 shows that there are maximum 112 uA current glitches at the output current.

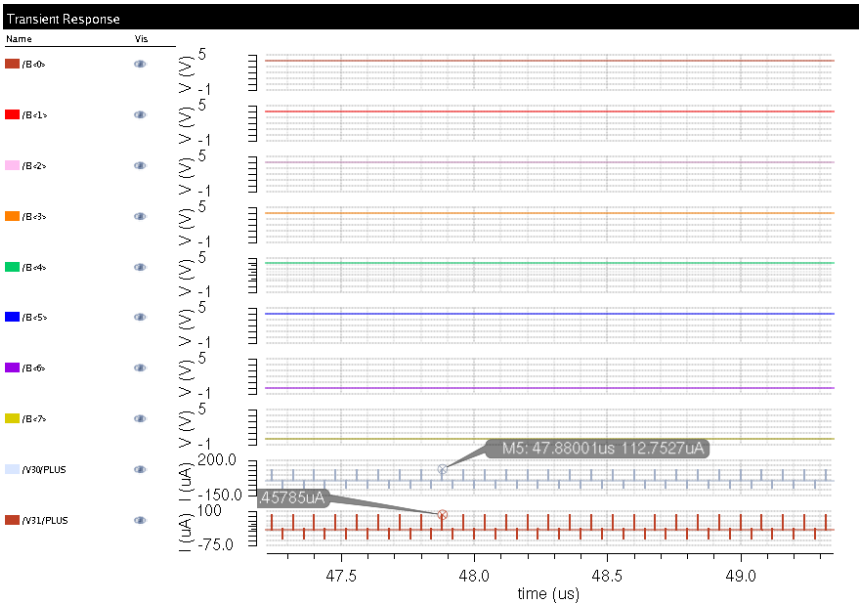


Figure 4.22: Glitches of Output Current.

4.4.1 Input latch

Input latch synchronizes the data with clock signal and avoids the sharp transition of output signals. Inverters at the input buffs the data and the inverted data for latch part. Latch is basic inverter connected latch with clock and input signal transistors. Figure 4.23 shows the latch schematic. Since transition voltage of output signal is wanted around 3V to avoid glitches in DAC output currents, clock and input signal transistors are PMOS transistors. Ratio of M2-M3 and M8 sets the transition voltage of output signals. For the ratio of 5, transition voltage is 2.7 V, which is good enough for our purposes.

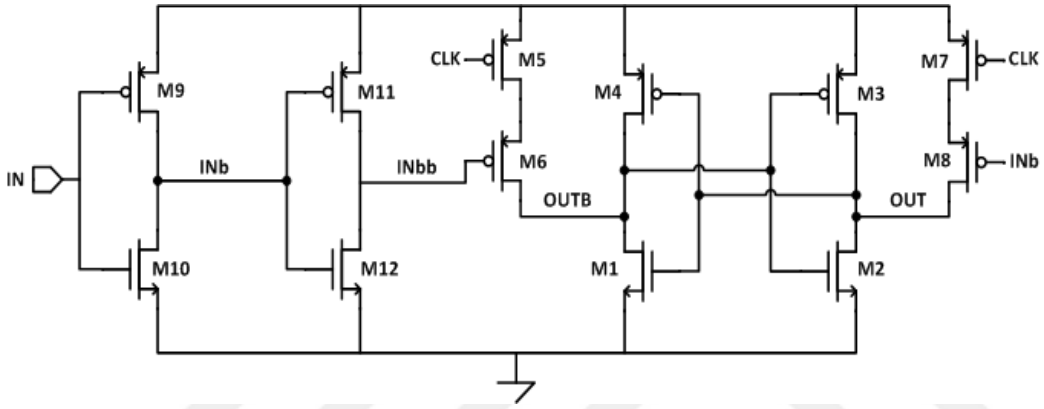


Figure 4.23: Schematic of Input Latch.

Simulation result of the input latch is shown in Figure 4.24. When the clock signal is high, positive output equals to data and negative output equals to inverted data. When the clock is low, outputs keep their values.

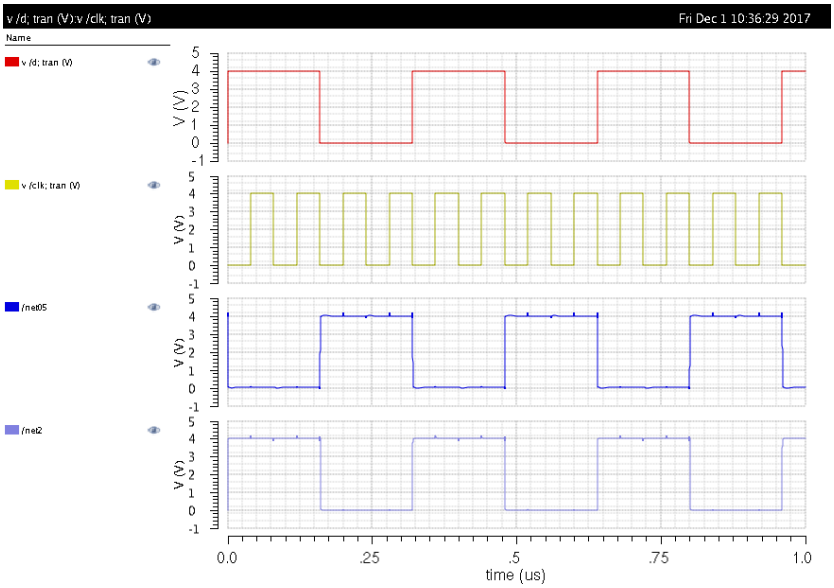


Figure 4.24: Simulation Result of Input Latch.

Output transition of the input latch is shown in Figure 4.25. Advantages of this type of transition are explained briefly in the DAC cell section which is next.

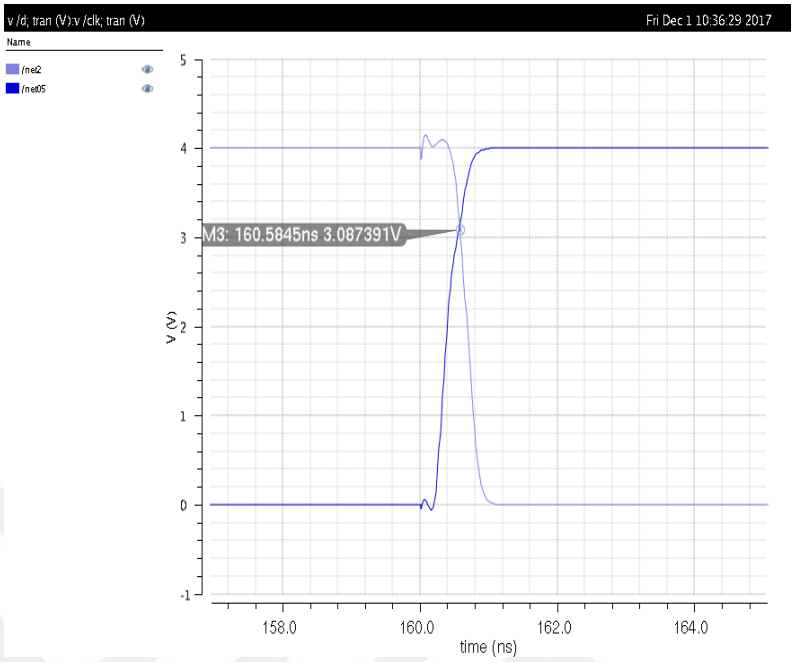


Figure 4.25: Output Transition of Input Latch.

4.4.2 DAC cells

DAC cells consist of current source and NMOS switches. If data is logic-1, then switch of the positive current output is on and biasing current flows through the positive output. If data is logic-0, current flows through negative output in a similar way. Figure 4.26 shows the schematic of a DAC cell. Since flash ADC output is 16-bits long, there are 16 DAC cells in the feedback DAC whose outputs are tied together (see Figure 4.20).

The sharp state transitions in switches causes glitches in the output current. Cascode transistors are added to NMOS switches to make output current to less sensitive to changes on the switches.

Another well known technique to reduce glitches involves providing smooth changes on switches while turning on and off, also know as make before break switching. This is done in input latch by setting output signal transitions above the NMOS transistor's threshold voltage so that both switches will be turned on and will be conducting a portion of the bias current for a short time. Therefore, as shown in Figure 4.25, output transition set around 3 V.

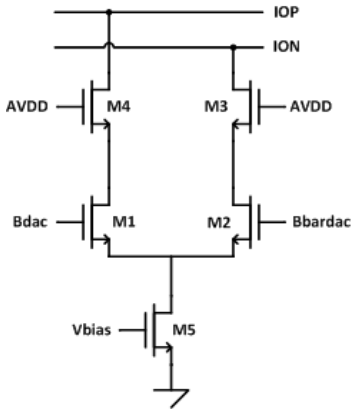


Figure 4.26: Schematic of a DAC Cell.

Dimensions of switch transistors affect the glitches on output current. The smaller dimensions help to reducethe glitch amplitude. Hence, switch transistor dimensions are selected as small as possible. Currently dimensions of current mirrors are selected as VD_{SAT} voltage becomes around 200 mV.

4.5 Decimation Filter

The raw output of the delta sigma modulator is sampled at an oversampling frequency. To down-sample this output to Nyquist frequency and generate the full resolution outputs, decimation filters are used. Having the data in Nyquist band is always preferred in applications. Decimation filter basically performs the reverse effect of the oversampling and noise shaping.

Since the delta sigma modulator of the design is third order, sinc3 filter is optimal. To implement the sinc3 filter, Hogenauer structure is used because it is effective and easy to implement. Transfer function of sinc3 filter in Hogenauer structre form is given below in (4.1).

$$H(z) = \left(\frac{1}{1-z^{-1}}\right)\left(\frac{1}{1-z^{-1}}\right)\left(\frac{1}{1-z^{-1}}\right)\left(\frac{1-z^{-N}}{N}\right)\left(\frac{1-z^{-N}}{N}\right)\left(\frac{1-z^{-N}}{N}\right) \quad (4.1)$$

Figure 4.27 and 4.28 shows the model of this transfer function, Figure 4.25 is the model accumulation part with oversampling clock and Figure 4.26 is the model of differentiation part with Nyquist frequency clock.

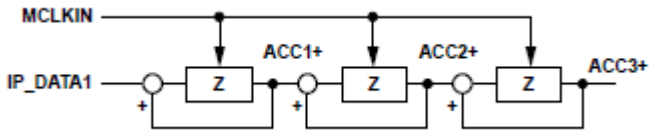


Figure 4.27: Accumulator [9].

Decimation process consists of two parts: accumulation and differentiation. Accumulation is the part that the 2's complement input signals are added to each other at the clock sampling frequency. Then in the differentiator part, data is differentiated with its previous value at the Nyquist clock rate.

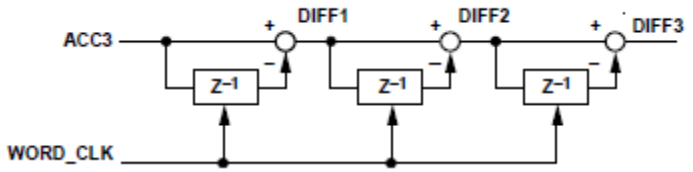


Figure 4.28: Differentiator [9].

In this design, input bandwidth is 100 kHz, OSR is 62.5 and that makes oversampling clock 12.5 MHz. Normally, OSR is selected as powers of two so that in decimation filter Nyquist band clock is generated with simple clock division from oversampling clock. However, in this chip's system clock is 50 MHz and ADC clock is provided from that clock as 12.5 MHz by simple division. It makes a small problem in decimation so that not all the 100 kHz signal bandwidth will not be covered in decimation filter. Real bandwidth can be around 97 kHz which is also acceptable.

Since the filter is in digital domain, Verilog code of decimation filter is written and implemented by using digital tools of Cadence Design Environment. Verilog code of the design and .tcl scripts of implementation tools are given in Appendix.

First, the thermometric coded flash ADC output is converted to 2's complement form since it is always better to use 2's complement bits while performing addition or subtraction operations in the design like decimation filters. Moreover, the digital blocks that will use this delta sigma output data in this chip are expecting signed bits. Therefore, converting the data to 2's complement is necessary.

Then, filter section is designed to satisfy Hogenauer filter structure that is explained above. The main problem in this structure is to decide the bitwidths of accumulato input and outputs as the accumulator outputs and differentiatior may require increased bitwidths to avoid overflow. To solve this problem, Simulink model of the filter is prepared and applied its input 1 V signal. Amplitude of the output gives a hint about how many bits have to be selected to have decent output without loss or extra unnecessary bits.

4.6 Corner and Monte Carlo Simulation Results

Corner and Monte Carlo analyses of delta sigma ADC are performed after full schematic design satisfied its specifications. Corner analysis forces the design to work in corner cases such as high and low temperatures and slow and fast transistor conditions. Corner analysis result of delta sigma ADC is given in Table 4.1.

Table 4.1: Corner Analysis Results**.**

Corner	SQNR(dB)
tt27	85.9
tt-40	93.4
tt85	79.2
ss27	82.5
ss-40	89.3
ss85	76.7
ff27	66.4
ff-40	94.1
ff85	52.0

In the table, tt stands for typical NMOS-typical PMOS process corner, ss stands for slow NMOS- slow PMOS process corner and ff stands for fast NMOS - fast PMOS process corner. The numbers represen the temperature. For example, ss-40 represent the corner with slow-slow process in -40 °C tempereture.

Figure 4.29 shows the histogram of Monte Carlo result.

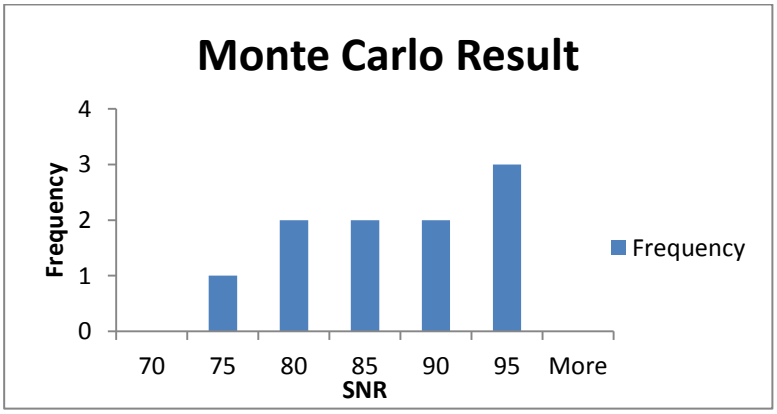


Figure 4.29: Monte Carlo Result.

5. LAYOUT AND MEASUREMENTS

In this chapter, layouts of sub-blocks are given, the layout perspectives that are used in the designs are explained and some important points on the layouts are highlighted.

First, floorplan of the design is done respect to the expected areas of sub-blocks. Then, layouts of sub-blocks are completed and finally sub-blocks are integrated regarding to the floorplan.

5.1 Floorplan

Floorplan basically means deciding the exact placement of each block on the chip. There are several considerations while floorplanning a block. These considerations are explained in this section and floorplan of the design is also given.

Total area and its shape are probably the most fundamental considerations of floorplanning. Area and shape of the design are planned to fit in the full sensor chip floorplan. It is expected to have 700um x 700um area with square-like shape.

Another important consideration is the signal flow. It means that sub-blocks should be placed respect to signal flow in the design. This makes signal routing easier and avoids the performance loss due to routing.

Keeping digital and analog parts separate is also important since there are two separate power domains in the design: DVDD, DVSS for digitals and AVDD, AVSS for analogs.

Floorplan of the design is completed regarding these considerations as shown in Figure 5.1. Every sub-block’s estimated area is determined and their dimensions are changed to satisfy these considerations given above. Arrows shows the signal flow.

All the layouts given in this chapter are DRC, LVS clean, post-layout simulations are done and are taped out.

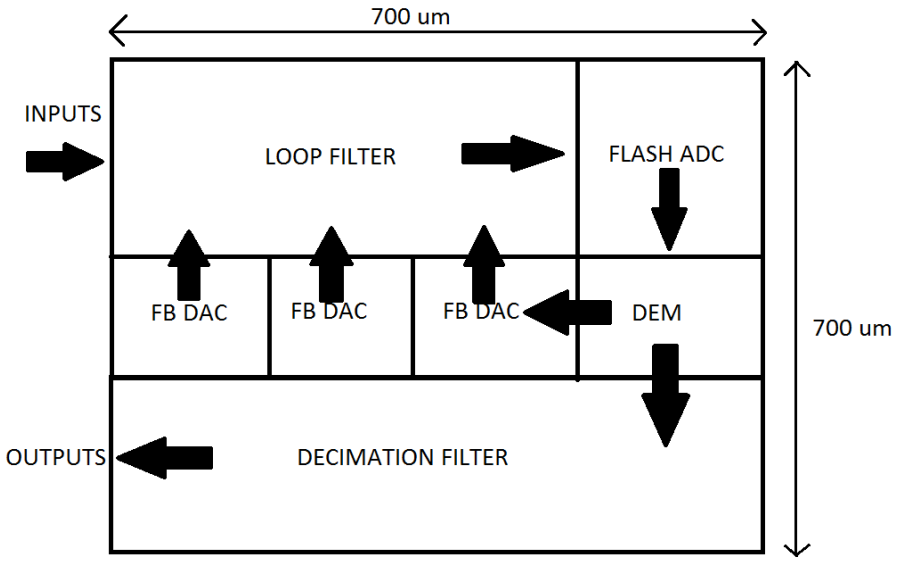


Figure 5.1: Floorplan of the Design.

5.2 Loop Filter

Loop filter consists of three operational amplifiers, resistors and capacitors. First operational amplifier is different and two others are the same. Figure 5.2 shows the first operational amplifier.

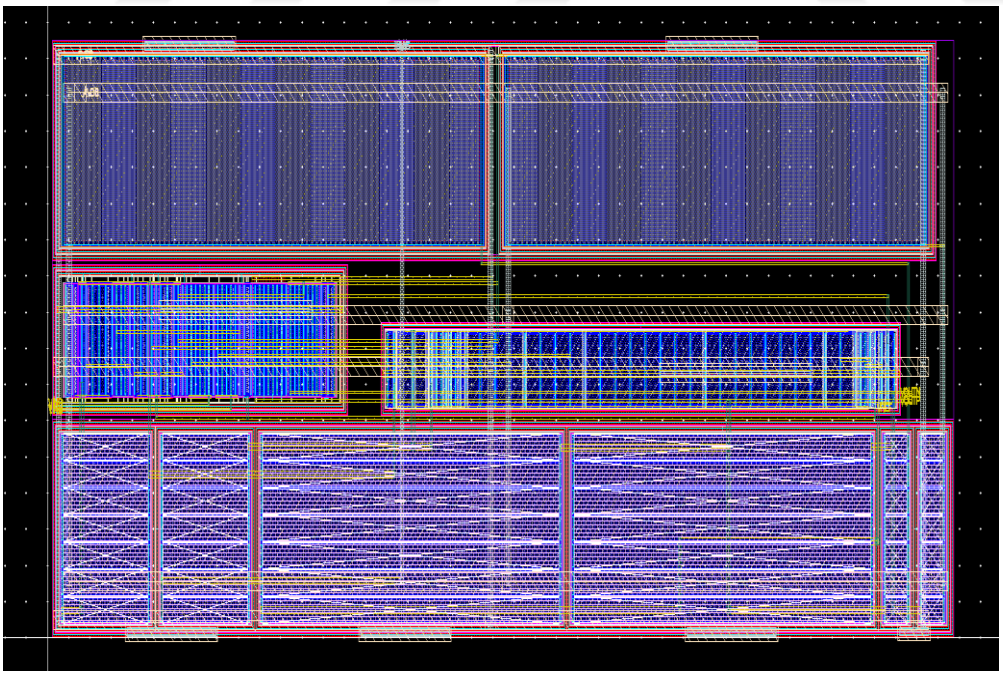


Figure 5.2: Layout of First Operational Amplifier.

Dimensions of first operational amplifier is determined as 200 um x 100 um in floorplan.

Some important notes about the operational amplifier layouts:

- All resistors are placed close to each other at the lower half.
- All capacitors are placed close to each other at the upper half.
- Capacitor and resistor dimensions are adjusted to fit the expected area in layout.
- NMOS transistors are connected to each other with dummy transistors when needed, to have steady and continuous line of diffusion, same as PMOS transistors.
- Signal are routed using only M2 (horizontal) and M3 (vertical) layers and power signals are routed using M5 (vertical) and M6 (horizontal) layers. Basically, odd number layers are vertical, even numbers are horizontal.
- Pins are placed so that inputs are in left side and outputs are in right side of the layout.

Most of these important guidelines given above are followed in not only the operational amplifier layout but in all layout cells. Figure 5.3 shows the operational amplifier that is used in second and third stages.

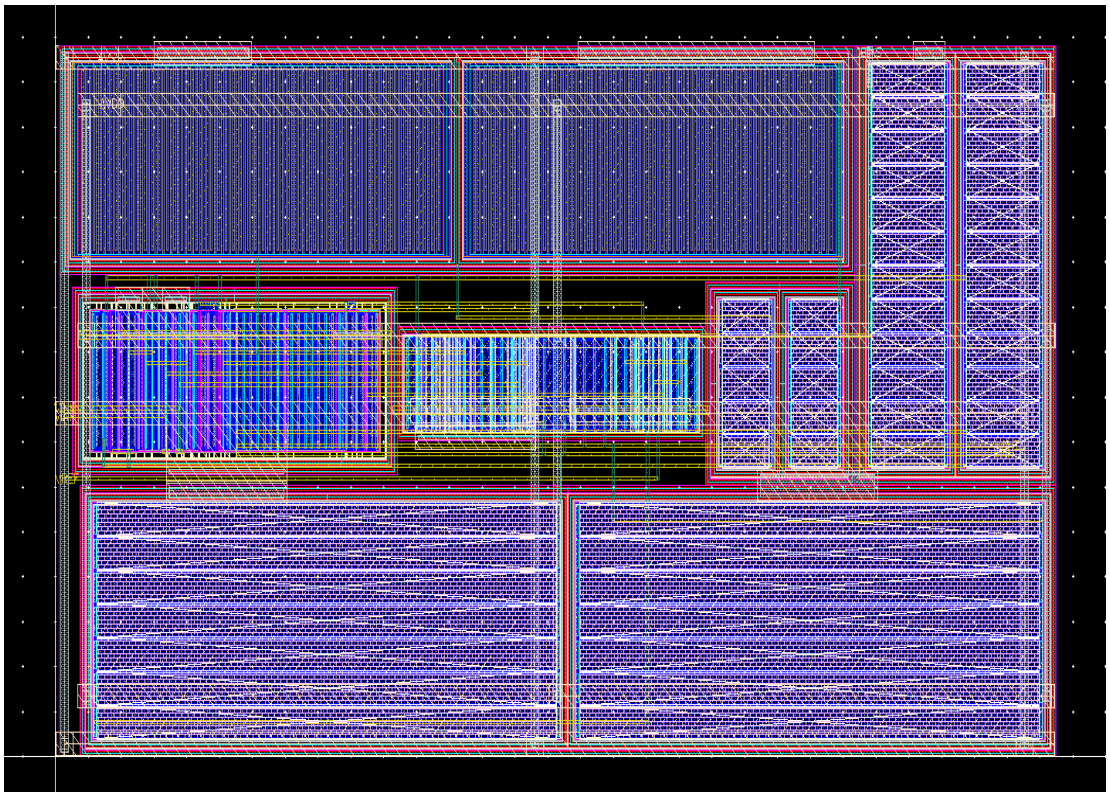


Figure 5.3: Layout of Second and Third Operational Amplifier.

Second and third operational amplifiers are similar to the first one. The only difference is that their widths are smaller due to small capacitor and transistor dimensions.

Figure 5.4 shows the layout of the loop filter. Operational amplifiers are placed in sequence. Resistors of the loop filter are placed below the amplifiers near the resistors of the operational amplifiers. Capacitors of the loop filter are placed above the operational amplifier capacitors. Routes are similar to the amplifier routes.

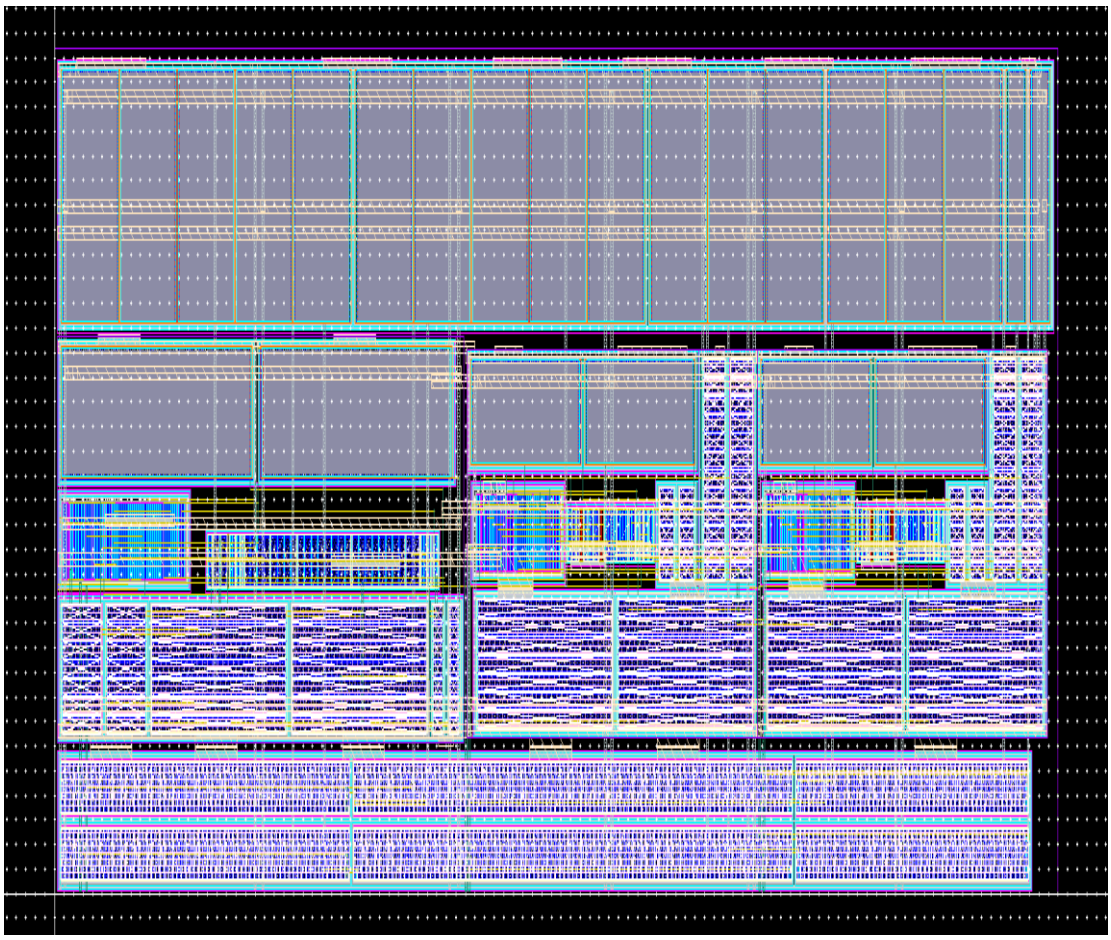


Figure 5.4: Loop Filter Layout.

5.3 Quantizer

The quantizer block which is actually a flash ADC, consists of resistor strings, comparators, output latches and a biasing circuit for the comparator's bias currents.

Resistors, biasing circuit, comparators and output latches are placed from top to bottom, respectively, as shown in Figure 5.5.

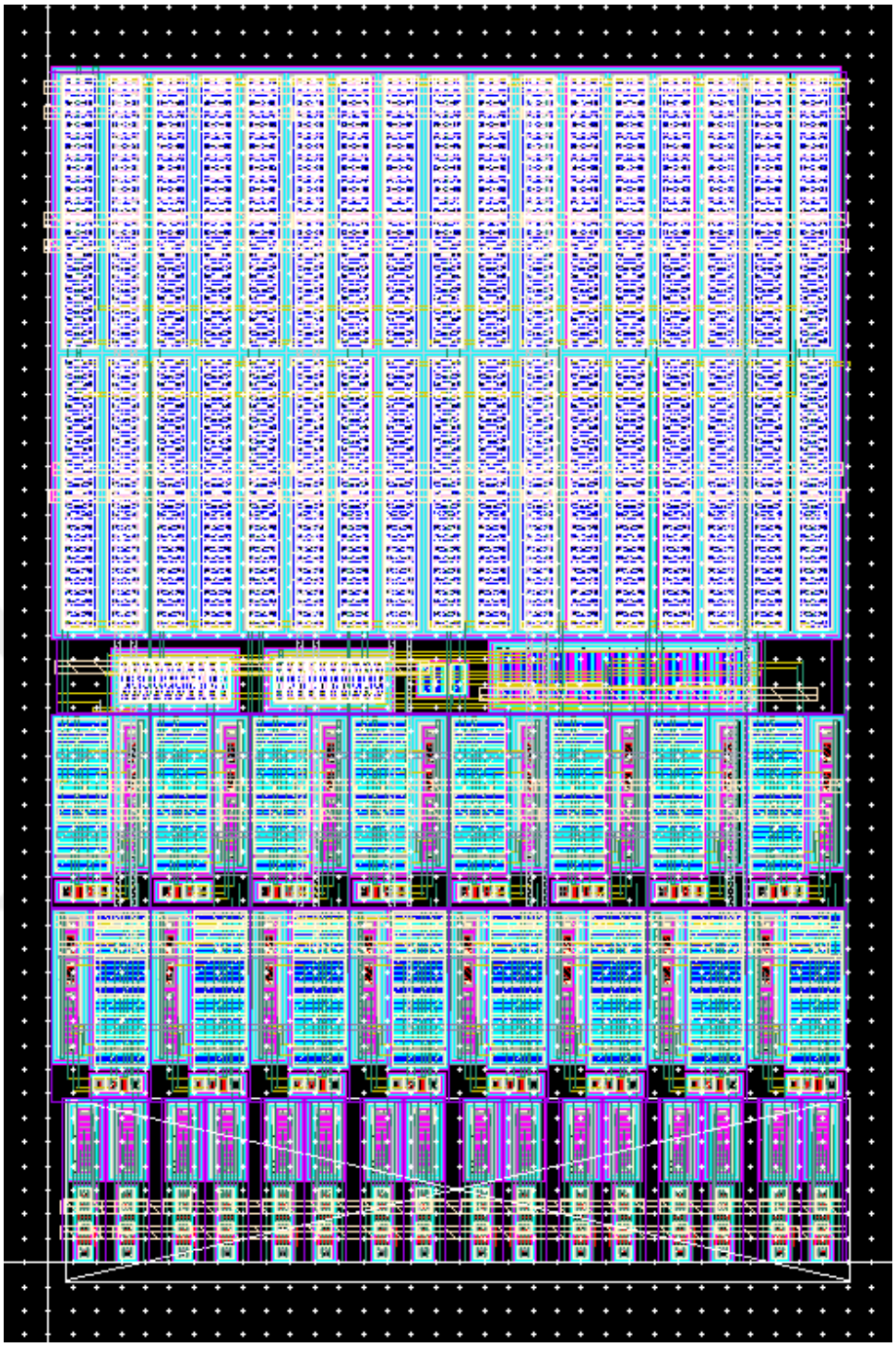


Figure 5.5: Layout of Flash ADC.

5.4 Dynamic Element Matching

Dynamic element matching is a digital block and is designed with using the logic gates in standard cell library in schematic level. Layout views of those logic gates are used in the layout of dynamic element matching cell.

The block consists of D flip-flops and XNOR gates used in 32-bit pseudo-random bit generator and MUX gates used in swappers. Bit generator is in the right side of the block and and swappers with complex routes are in the left side. Figure 5.6 shows the layout of dynamic element matching block whose dimensions are 150 um x 90 um.

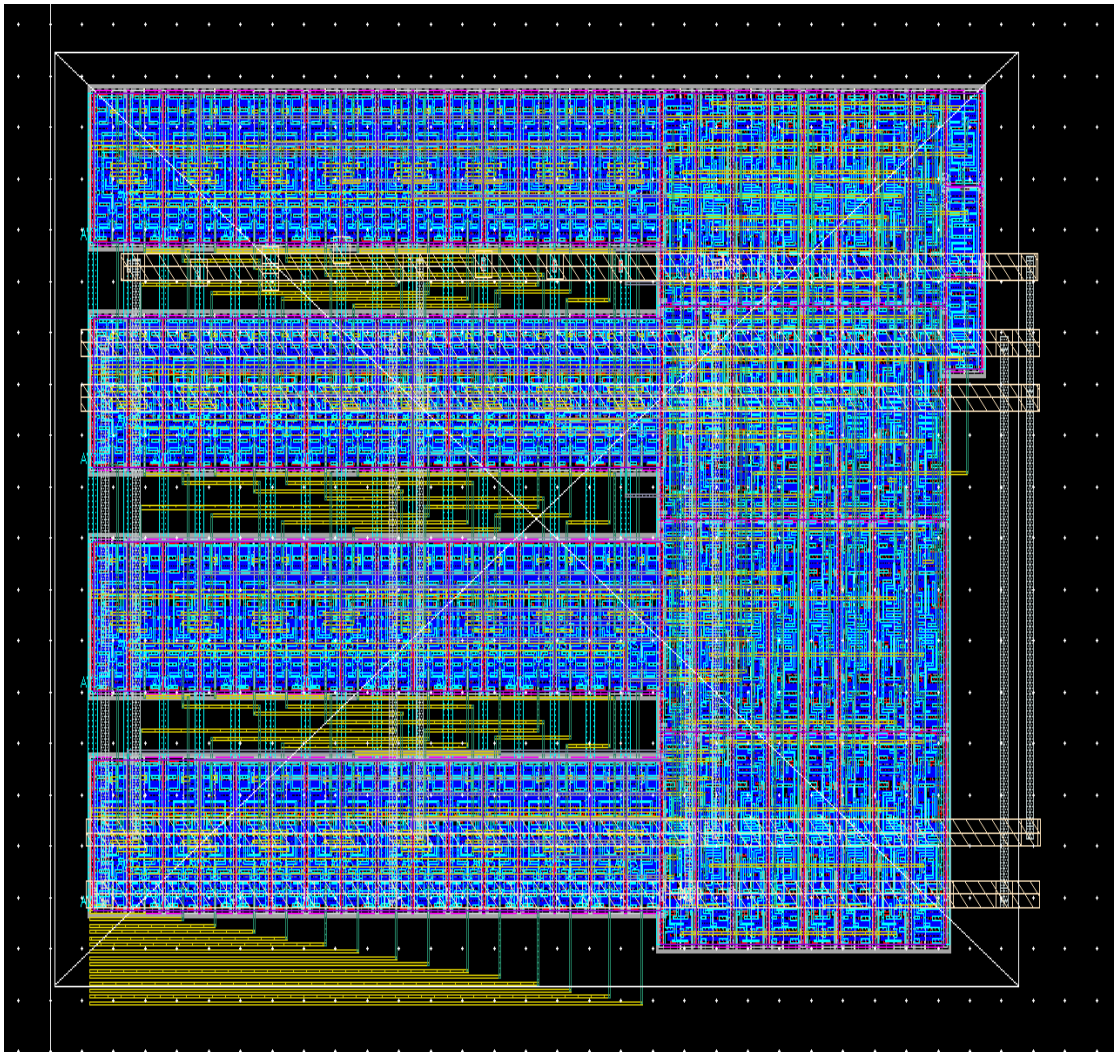


Figure 5.6: Layout of Dynamic Element Matching.

5.5 Feedback DAC

Even though dynamic element matching reduces the effect of mismatches in DAC cells, mismatch reduction techniques are also applied in layout. Digital inputs of DAC are on bottom side, therefore latches are placed accordingly. Outputs of latches are connected to the DAC cells which are placed near the top of the block. Having a symmetry and using dummies where needed helps reducing the mismatch effects.

Layout of feedback DAC is shown in Figure 5.7. Dimensions are 190 um x 160 um.

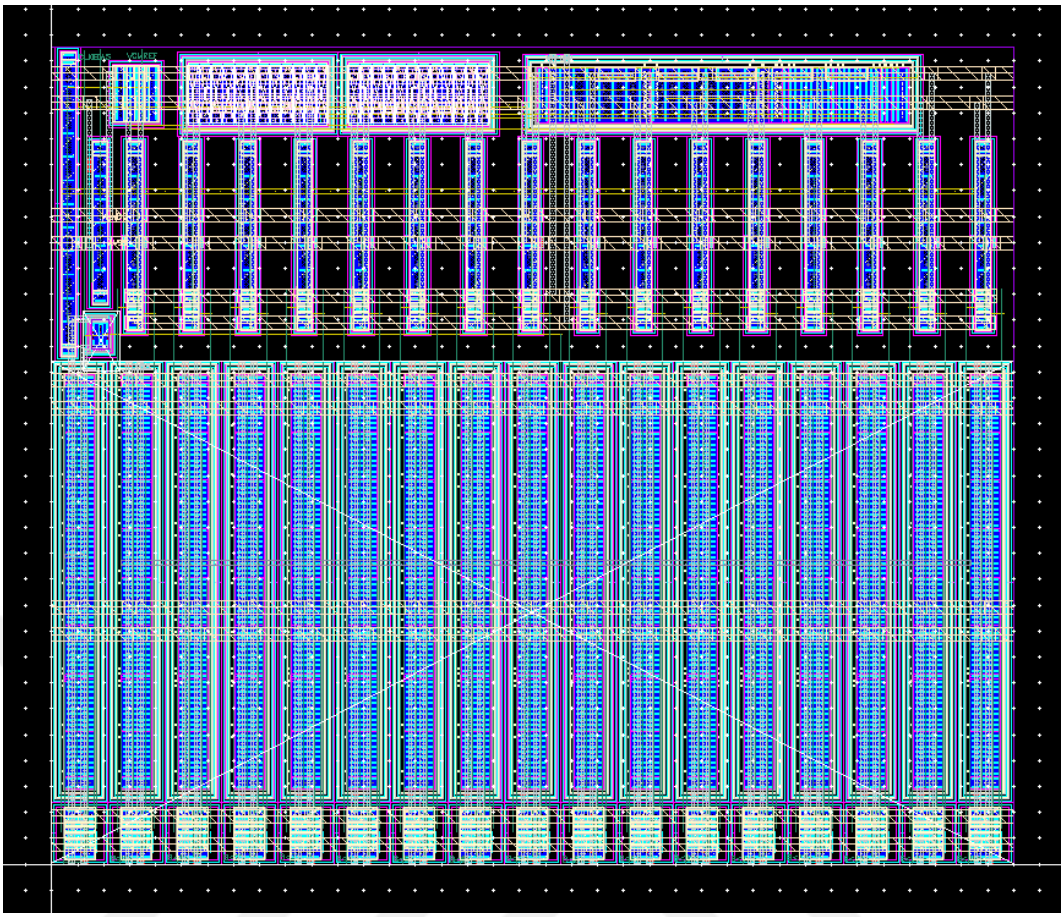


Figure 5.7: Layout of Feedback DAC.

5.6 Decimation Filter

Decimation filter block is designed in Verilog and synthesized using digital synthesis tool of Cadence, RTL Compiler. This is explained in previous chapter. In this layout chapter, place and route of this digital block is explained.

The script used in place and route is given in Appendix- E. M5 and M6 layers are reserved for only power routing and all other signals are routed using M1-M4 layers. Empty areas are filled with filler cells and metal fillings are also completed in digital environment automatically. However, DRC and LVS checks are completed in analog environment. First, post-PAR Verilog netlist and .gds file is imported to Virtuoso and then DRC and LVS checks are run by Assura.

Figure 5.8 shows the layout of decimation filter. Its dimensions are a bit different from the floorplan to have a place for LDO of the design. Its width is decreased and height is increased, final dimension is 500 um x 125 um.

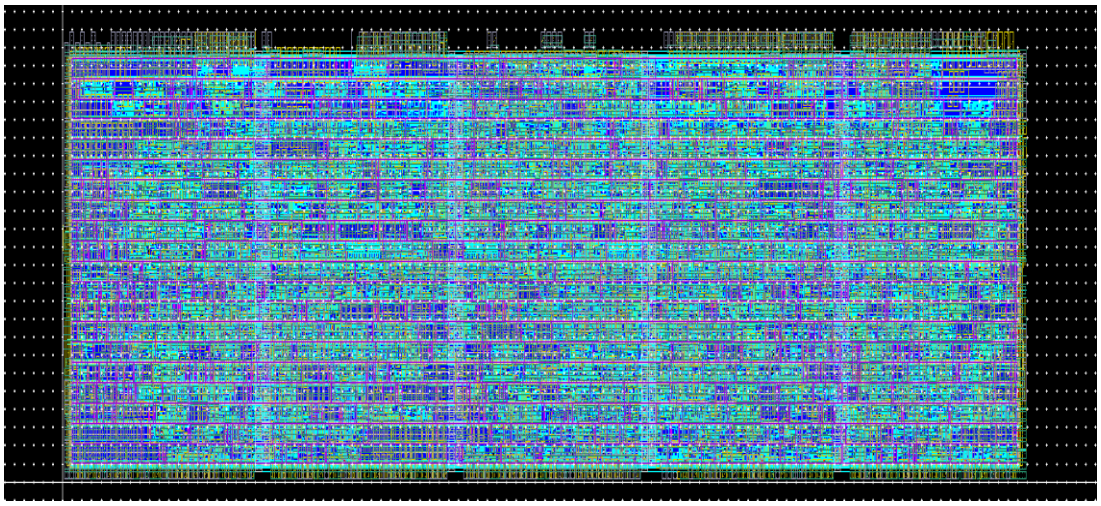


Figure 5.8: Layout of Decimation Filter.

5.7 Top level Delta-Sigma ADC Layout

After all sub-block layouts are completed regarding to floorplan, now it is time to bring them together. Although dimensions of some blocks are changed after layout is completed, still combining the blocks is easier due to floorplan.

In addition to the blocks that are explained in previous sections, there are a few more blocks that have to be mentioned. First one is biasing circuit which produces the currents that are needed for the sub-blocks. At the very top of the full layout, there is biasing circuit block which is basically a few number of current mirrors.

Another additional block is the low dropout regulator (LDO). It is a voltage regulator that feeds the analog and digital power lines of the design. It is in the bottom left corner of the layout, and is fit in that corner by modifying the digital filter dimensions in digital PAR.

Other empty areas are filled with MOSFET capacitors (MOSCAP) which are bypass coupling capacitors of the LDO output, make the power lines less noisy.

Last additional block is the input switch. It is placed in the top left corner of the layout where inputs of ADC exist. It selects where the inputs will come from whether preamplifier outputs, input pads or temperature sensor outputs.

Figure 5.9 shows the top level layout of delta sigma ADC. It is DRC and LVS clean with no important ERC errors. Delta sigma ADC layout integrated to the full sensor

chip layout and sent to production. Micrograph of full chip is given in Figure 5.10. ADC is in bottom left part of full chip.

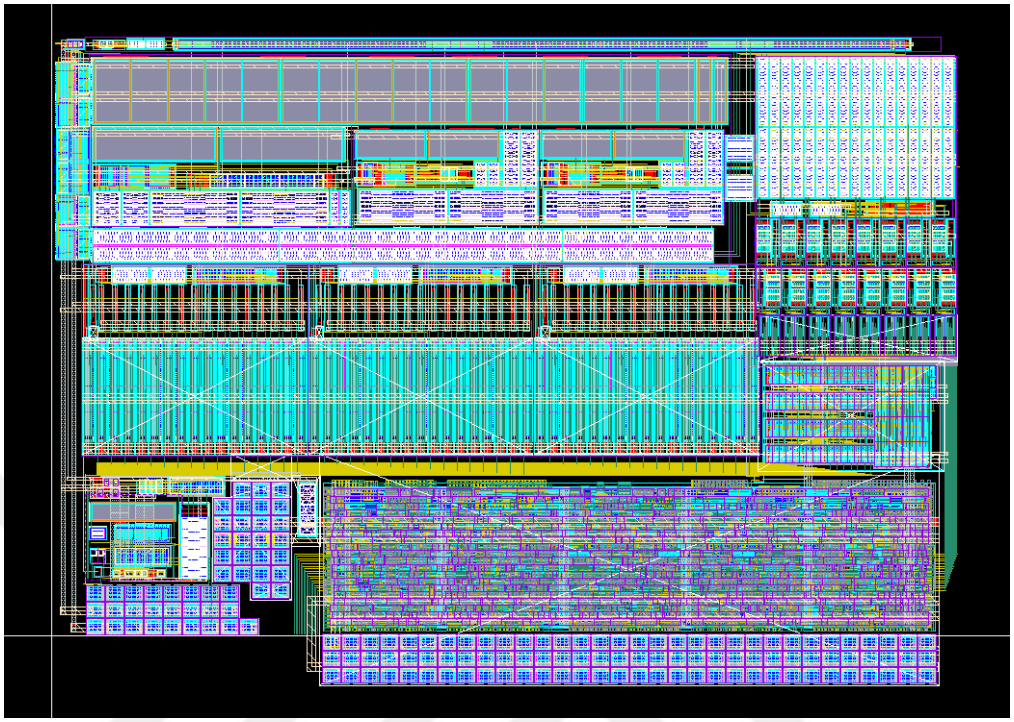


Figure 5.9: Layout of Delta Sigma ADC.

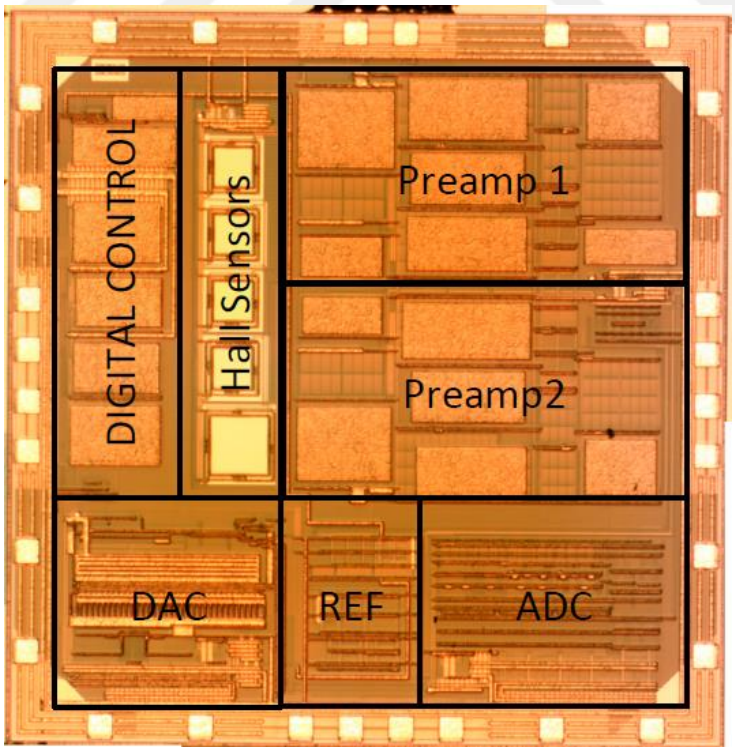


Figure 5.10: Micrograph of Sensor Chip.

5.8 Post-Layout Simulation Results

Parasitic extraction of design is done in RC mode which includes parasitic capacitors and resistors together. Figure 5.11 shows the post layout simulation result of delta sigma ADC. Results show that SNR is around 76 dB and ENOB is 12.32 bits. It is less than expected but still good enough.

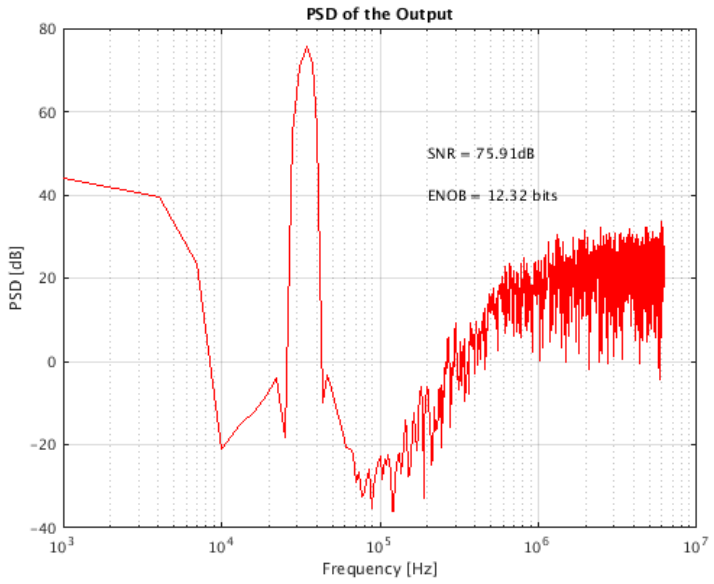


Figure 5.11: Post-Layout PSD of Output.

5.9 Measurements Results

In this section, some properties of HES chip are explained to have a better understanding about the test setup of delta sigma ADC.

HES chip communicates with outside by using serial peripheral interface (SPI). SPI is a commonly used serial communication protocol that consists of two blocks; master and slave and four wires; Chip Enable (CE), Master In Slave Out (MISO), Master Out Slave In (MOSI) and SPI Clock (SCK). In this sensor application, sensor chip has its SPI slave block which is designed by using Verilog and implemented by using digital implementation tools of Cadence. Master block of SPI can be selected anything such as microcontroller or USB-to SPI converters to configure sensor chip from outside by using a computer or get the digital data from chip to computer. ADC output bits are one of these data that are wanted outside the chip to measure the

performance of the ADC itself. In this measurement setup an Arduino UNO module is used as the SPI master block. Arduino code that used as SPI master in this design is given in Appendix-F.

Sensor chip Dice are wire bonded to the PCB to be used in test setup. This is called Chip-On-Board method. Figure 5.12 shows the prepared PCB and wire bonded chip on it.

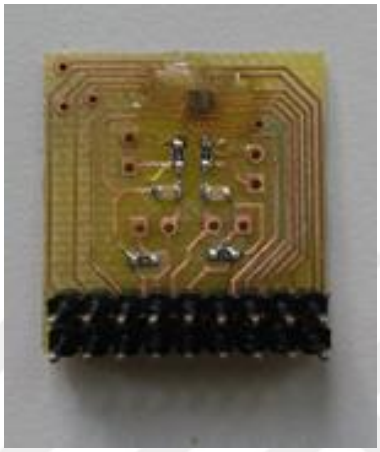


Figure 5.12: Test Board.

After ADC is turned on, outputs are written to the adc_out register of SPI register table. SPI can carry maximum 8-bit data in one SPI cycle. Therefore the adc outputs are carried out through MISO line to the Arduino host 8 bits at a time..This transfer is executed as first MSB byte then LSB byte of the 14-bit ADC output. After that, Arduino will pass the bits to computer via serial port.

Since there is a decimation filter in the end of ADC, frequency of output bits will be in 200 kHz which is Nyqiuest frequency of the system. The problem starts here. There is 14-bit data sampled in 200 kHz and it takes two SPI cycles to carry the data out. From these facts, minimum required SPI clock can be calculated as 200 kHz x 16 = 12.5 MHz without considering delays of CE signal of SPI. Arduino has maximum 8 MHz SPI clock that makes Arduino useless to take the data out.

To solve this problem, an FPGA is used as the SPI master and some of the Arduino overhead can be reduced. On FPGA Master module of SPI would be designed in Verilog. Moreover, block RAMs of FPGA is used to store the data. After data read is completed, data in the RAM taken by ChipScope feature of FPGA. SPI master Verilog code including RAM integration is given in Appendix – F.

Frequency spectrum of the data that is taken by FPGA is plotted by using MATLAB. Results of various frequencies are given in Figure 5.13, 5.14, 5.15 and 5.16.

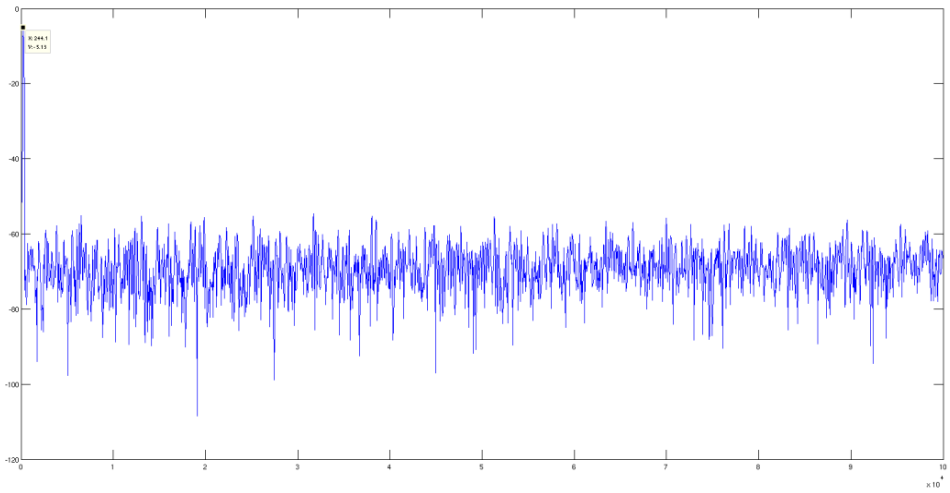


Figure 5.13: FFT Plot of ADC Output (244 Hz).

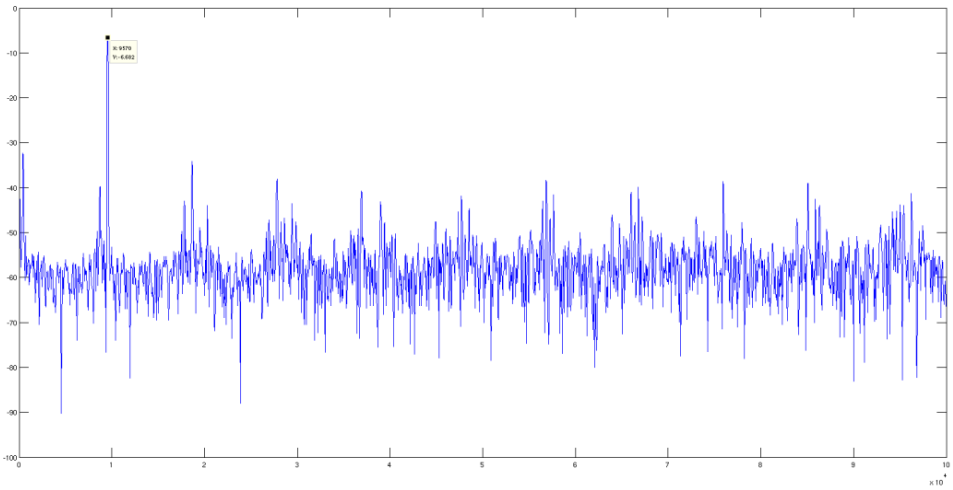


Figure 5.14: FFT Plot of ADC Output (10 kHz).

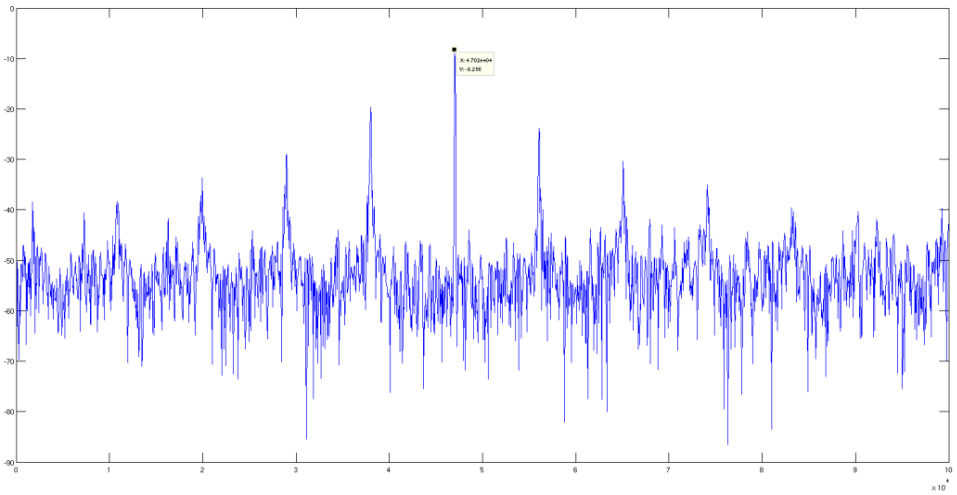


Figure 5.15: FFT Plot of ADC Output (47 kHz).

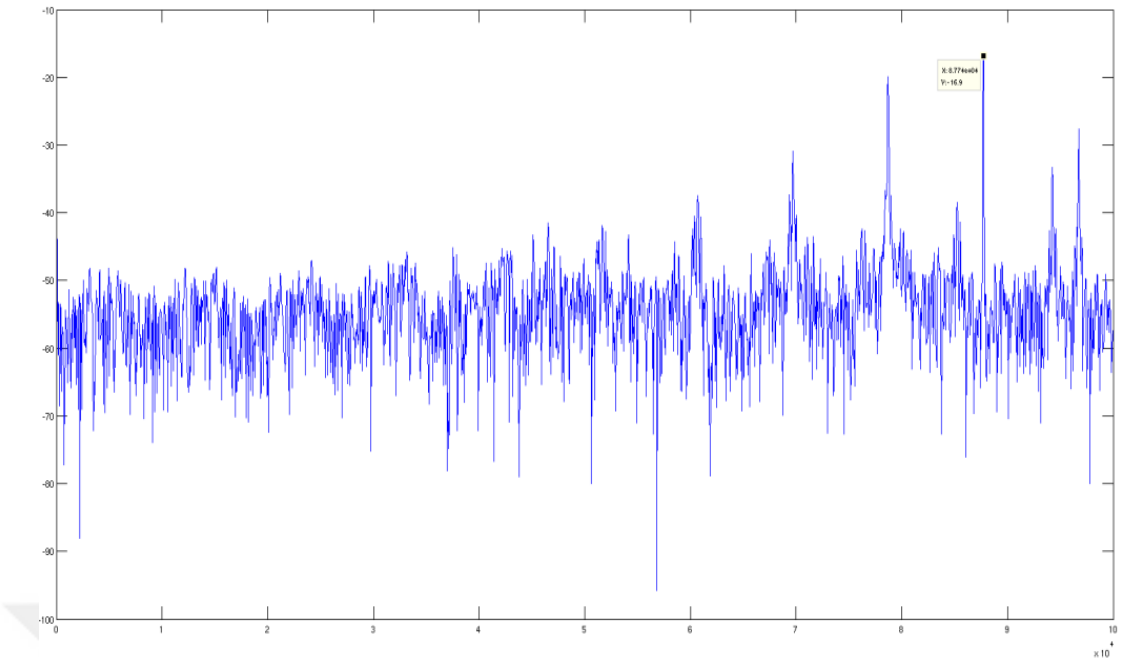


Figure 5.16: FFT Plot of ADC Output (88 kHz).

Figures show that linearity of 244 Hz is good enough however linearity becomes problem when ADC input frequency increase. The reason of these harmonics might be that the serial data transfer is still a problem in the kHz frequencies.

Another measurement to confirm whether the serial transfer is a problem or not, is to measure ADC and DAC together. Hence the linearity of analog output of DAC can show what causes the nonlinearity.

The chip is configured in preamplifier + ADC + DAC mode. Preamplifier gains are set to 1. DAC gain is set to maximum. A 1.5 Vpp signal with different offsets in positive and negative terminals is applied to preamplifier input (Yellow and green ones in plots). DAC outputs are blue and pink waveforms shown in osciloscope window in Figure 5.17. White one is differential DAC output that is used in FFT calculation.

Figure 5.17 and 5.19 show that signal in time domain become distorted when input signal frequency is higher. Figure 5.18 and 5.20 show that linearity of DAC output is a lot better than Figure 5.13, 5.14, 5.15 and 5.16. This proves that the nonlinearity in those plots comes from the speed and misalignment problems of serial data transfer.

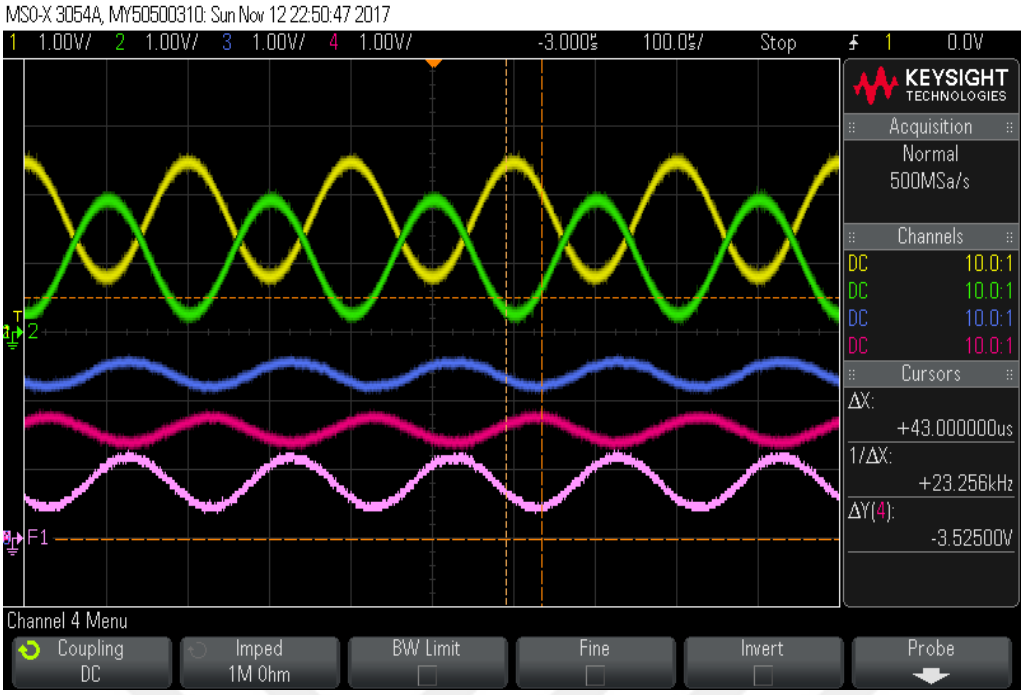


Figure 5.17: Time Domain Plot of DAC Output (5 kHz).

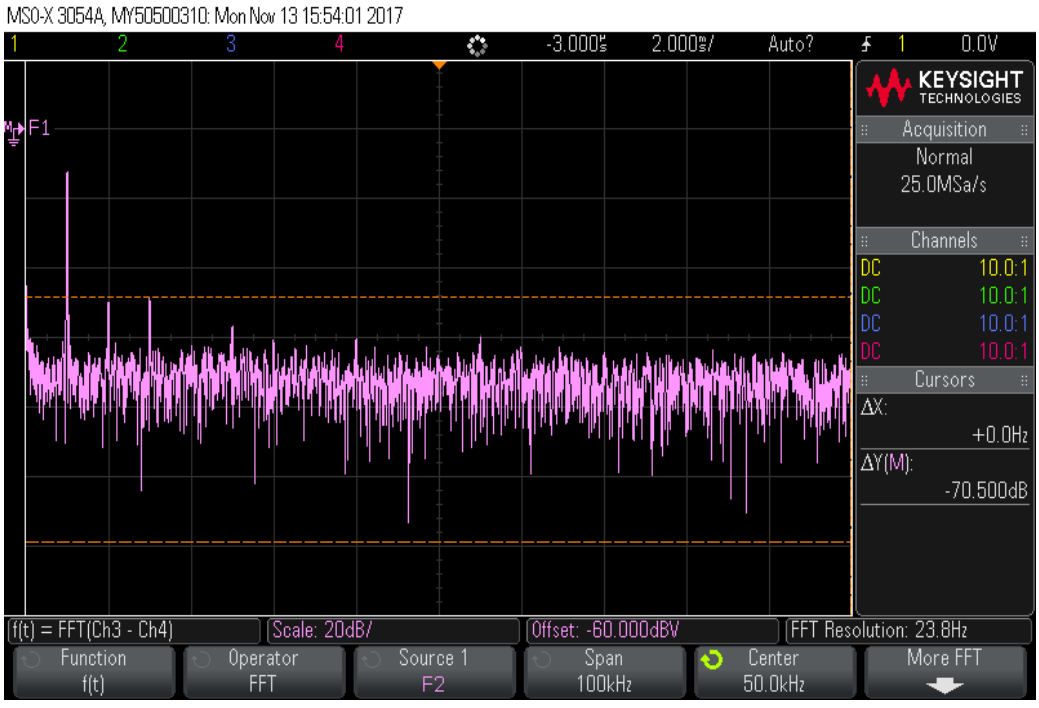


Figure 5.18: FFT Plot of DAC Output (5 kHz).

Figure 5.20 shows that there is around 40 dB linearity in DAC output. It is actually less than expected and the reason of this nonlinearity may be ADC, DAC or both. Because the linearity of preamplifier output is around 60 dB.

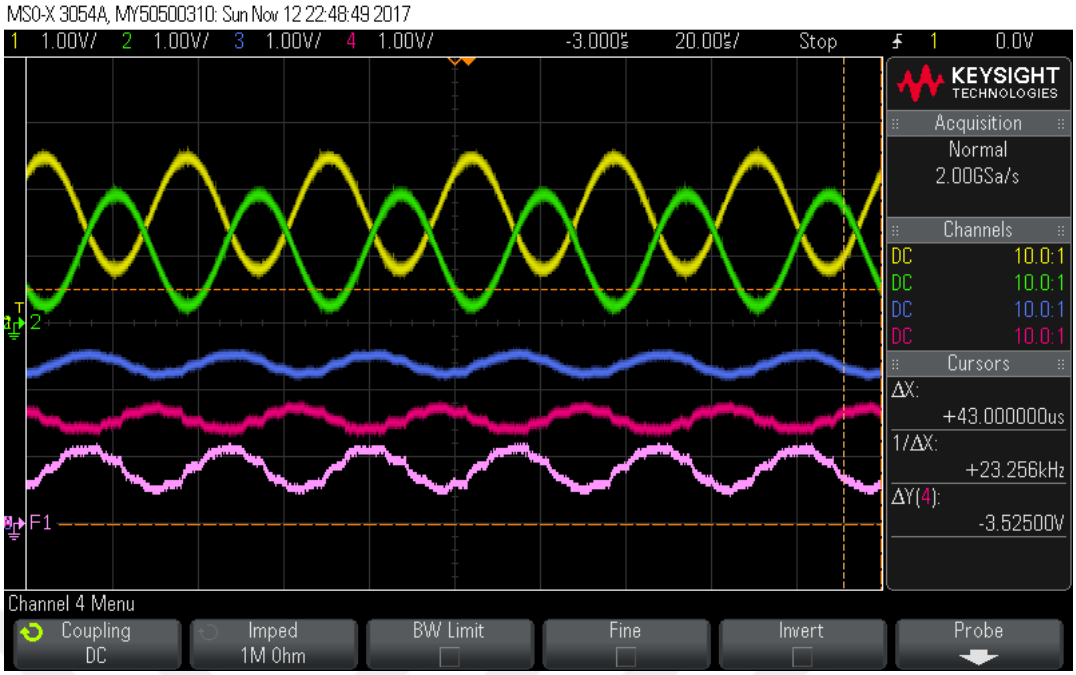


Figure 5.19: Time Domain Plot of DAC Output (30 kHz).

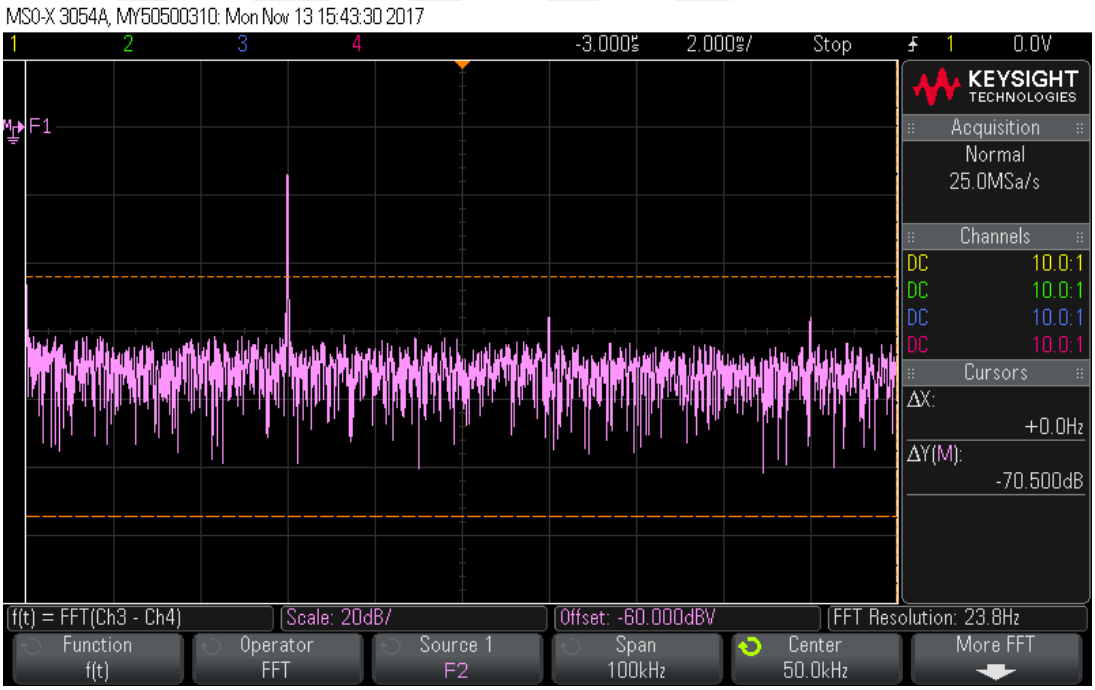


Figure 5.20: FFT Plot of DAC Output (30 kHz).

6. CONCLUSIONS

Continuous time delta sigma ADC that used in a Hall Effect based current sensor application is presented. In the IC design, modelling the design with ideal elements in Simulink or Verilog-A helps the designer to avoid the systematic errors that can be critical in circuit level design phases. Therefore, first ideal model of the design was prepared in Simulink and also in Cadence with Verilog-A. After models met the specifications, transistor level design of the design was started. All simulation results of the models and transistor level designs of sub-blocks and full ADC were given. Next step after transistor level design was layout. All layout techniques that were used in this work were explained and the layout views of sub-blocks and full chip were presented. Since the chip that was taped out was a system on chip sensor and did not include the ADC alone, some problems arose during the measurements of the ADC. These problems and measerement setup of the design were explained briefly. Additionally, measurement results are also presented and discussed. As a conclusion, although there are some problems in measurement, it can be said that this work achieved its aim by performing its function in the sensor chip.

Figure of merit of the delta sigma ADC is calculated as 12.5 pJ/conv. from equation given in (2.5). Comparison of this work with other similar delta sigma ADCs in the literature is given in the Table 6.1.

In Table 6.1, this work, [16] and [17] are simulation results while [18] and [19] are measurement results. Since current optimization is not applied to this design, FOM of the design is higher than other works in the literature.

In the next version of the sensor chip, it is planned to have the raw data (undecimated) of the delta sigma modulator be available through output pads This way we can have a better characterization of the DS Modulator performance. Additionally, current optimization will be performed so that the FOM of the design will be lower in the next version. These can be considered as future work of the design.

Table 6.1. Comparison with Other Works.

Parameters	**This Work**	**[16]**	**[17]**	**[18]**	**[19]**
Bandwidth	100 kHz	41 kHz	24 kHz	25 kHz	20 kHz
SNDR	86 dB	74 dB	76.1 dB	74 dB	79 dB
Power Cons.	20 mW	0.75 mW	0.86 mW	0.3 mW	0.028 mW
FOM(pj/conv.)	12.2	2.2	4.37	1.46	0.097
Supply (V)	4	0.75	-	0.5	0.6
Technology	0.18 um	0.13 um	0.18 um	0.18 um	0.13 um

REFERENCES

[1] **de la Rosa, J.** (2011). Sigma-delta modulators: Tutorial overview, design guide, and state-of-the-art survey. *Circuits and Systems I: Regular Papers, IEEE Transactions on, vol. 58, pp. 1 _21*.

[2] **Cherry, J. A. and Snelgrove, W. M.** (1999). *Continuous-time Delta-sigma Modulators for High-speed A/D Conversion: Theory, Practice, and Fundamental Performance Limits*. Kluwer Academic Publishers, Hingham, MA, USA, ISBN 978-0792386254.

[3] **Janssen, E. and van Roermund, A.** (2011). *Look-Ahead Based Sigma-Delta Modulation. Springer Netherlands*, ISBN: 978-94-007-1386-4.

[4] **Schreier, R. and Temes, G. C.** (2005). *Understanding Delta-Sigma Data Converters*. New York, IEEE press, Wiley-Interscience, ISBN: 0-471-46585-2.

[5] **Bolatkale, M., Breems, L. J. and Makinwa, K. A. A.** (2014). High Speed and Wide Bandwidth Delta-Sigma ADCs, *Springer International Publishing*.

[6] **Schreier, R.** (2010). Delta-Sigma Toolbox, [Online]. Available: http://www. mathworks.com/matlabcentral/fileexchange/19.

[7] **Ortmanns, M. and Gerfers, F.** (2006). *Continuous-Time Sigma-Delta A/D Conversion Fundamentals, Performance Limits and Robust Implementations*. Secaucus, NJ, USA: Springer-Verlag New York, Inc., ISBN: 978-3-540-28406-2. Springer Series in Advanced Microelectronics Volume 21.

[8] **Maloberti, F.** (2007). *Data Converters*. Secaucus, NJ, USA: Springer-Verlag New York, Inc., ISBN: 0-387-32485-2.

[9] **Analog Devices** AD7401A: Isolated Sigma-Delta Modulator. Product Datasheet. Norwood, MA.

[10] **Candy, J.** (1985). A use of double integration in sigma delta modulation, *Communications, IEEE Transactions on, vol. 33, pp. 249 _ 258*.

[11] **Geerts, M. S. Y. and Sansen, W.** (2003). Design of Multi-Bit Delta-Sigma A/D Converters. Springer-Verlag New York, Inc., ISBN 978-1-4020-7078-5. The Kluwer International Series in Engineering and Computer Science Volume 686.

[12] **Razavi, B.** (2015). The StrongARM Latch. *IEEE Solid-State Circuits Magazine*.

[13] **Carley, L. R.** (1989). A Noise Shaping Coder Topology for 15+ Bit Converters. *IEEE Journal of Solid-State Circuits, Vol. 24, No. 2*.

[14] **Cypress Semiconductor Company** PRS32: 32-Bit Pseudo Random Sequence Generator, Datasheet. San Jose, CA.

[15] **Bastor, J., Marques, A. M., Michel, S. J. S. and Sansen, W.** (1998). A 12-Bit Intrinsic Accuracy High-Speed CMOS DAC. *IEEE Journal of Solid-State Circuits, Vol. 33, No. 12*.

[16] **Essawy, A., Ismail, A. and Dessuoky, M.** (2014). A 0.75V Continuous-Time Sigma-Delta Analog-to-Digital Converter in CMOS Technology. *International Caribbean Conference on Devices, Circuits and Systems (ICCDCS)*, Playa Del Carmen, Mexico, April 2-4.

[17] **Saravana, K. and Chatterjee, S.** (2012). A 110-dB Dynamic Range, 76-dB Peak SNR Companding Continuous-Time ΔΣ Modulator for Audio Applications. *25th International Conference on VLSI Design (VLSID)*, Hydarabad, India, January 7-11.

[18] **Pun, K. P., Chatterjee, S., and Kinget, P. R.** (2007). A 0.5-V 74-dB SNDR 25-kHz continuous-time delta-sigma modulator with a return-to open DAC. *IEEE Journal of Solid-State Circuits, Vol. 42, No. 3*.

[19] **Zhang, J., Lian, Y., Yao, L. and Shi, B**. (2010). A 0.6-V 82-dB 28.6-W Continuous-Time Audio Delta-Sigma Modulator. *IEEE Journal of Solid-State Circuits, Vol. 46, No.10.*

APPENDICES

APPENDIX A: Delta-Sigma Toolbox MATLAB Code

```
% Synthesize Noise Transfer Function (NTF)
order = 3;      % Third order
OSR = 62.5;     % Oversampling Ratio
opt = 1;        % Optimizing zeros
nlev = 16;      % Number of quantizer levels
f0 = 0;         % Center frequency
H_inf = 1.5;    % Out of Band Gain
H = synthesizeNTF(order,OSR, opt, H_inf, f0);
figure(1);
rlocus(H);
 [snr,amp] = simulateSNR(H,OSR,[],[],nlev);
% SNR Change with Input Amplitude
figure(2);
plot(amp,snr,'b');
grid on;
figureMagic([-100 0], 10, 2, ...
[0 120], 10, 1);
xlabel('Input Level, dB');
ylabel('SNR dB');
s=sprintf('peak SNR = %4.1fdB\n',...
max(snr));
text(-65,15,s);
k=1;
% Implementing Noise Transfer Function
form = 'CIFB';
xLim = 0.6;
[a,g,b,c] = realizeNTF(H,form);
b(2:end) = 0;           %only single input feed-forward path
ABCD = stuffABCD(a,g,b,c,form);
[ntf,stf] = calculateTF(ABCD,k);
figure(4);
f = linspace(0,0.5,100);
z = exp(2i*pi*f);
magNTF = dbv(evalTF(ntf,z));
magSTF = dbv(evalTF(stf,z));
plot(f,magNTF, 'b', f,magSTF, 'm', 'Linewidth' ,1);
```

APPENDIX B: SNR Calculation MATLAB Code

```
clear all;
clc;
%Modulator Parameters
bw=100e3;            % Base-band
R=62.5;              % Oversampling
Fs=bw*R*2;                              % Oversampling frequency - 16Mhz
Ts=1/Fs;
Ao=10000;            % Op-Amp Gain
GBW=16e6;            % GBW
Vfs=1;               % Full-scale voltage
n=4;                 % Number of output bits
D=Vfs/((2^n)-1);     % Quantization interval, delta
%Modulator Coefficients
a1=0.0441;           % Discrete coefficients
a2=0.2871;
a3=0.8005;
c1=1*Fs;
c2=1*Fs;
c3=1;
k1=a1*a2*a3;         % Impulse-Invariant Transform coefficients
k2=a1*a2*a3 + a2*a3;
k3=(a1*a2*a3/3)+(a2*a3/2)+a3;
b1=k1;
g1=(9.25e-4)*a2*a3;
gc=g1/(k2*k3);
%Non-ideal integrator
Gain1=1000;
alpha1=1-1/Gain1;
Amax1=100;
sr1=2e6;
GBW1=30e6;

%Simulation Parameters
N=4096;              % Number of Sample
nper=11;             % Number of Period
Fin=nper*Fs/N;                  % Input signal frequency (Fin = nper*Fs/N) - Coherent Sampling
finrad=Fin*2*pi;           % Input signal frequency in radians
Ampl=0.5;
```

```
bias=0.5;
Ntransient=0;
%  Open Simulink diagram first
options=simset('RelTol', 1e-3, 'MaxStep', 1/Fs);
sim('CTDS_ADC_realint', (N+Ntransient)/Fs, options);          % Starts Simulink simulation
% Calculates SNR and PSD of the bit-stream and of the signal
w=hann(N);
f=Fin/Fs;                                  % Normalized signal frequency
fB=N*(bw/Fs);                % Base-band frequency bins
yy1=zeros(1,N);
yy1=dac_out(2+Ntransient:N+Ntransient+1)'-bias;
ptot=zeros(1,N);
[snr,ptot]=calcSNR(yy1(1:N),f,fB,w',N);
Rbit=(snr-1.76)/6.02;        % Equivalent resolution in bits
%  Plots
figure(1);
clf;
semilogx(linspace(0,Fs/2,N/2), ptot(1:N/2), 'r');
grid on;
title('PSD of the Output')
xlabel('Frequency [Hz]')
ylabel('PSD [dB]')
axis([0 Fs/2 -140 20]);
text_handle = text(floor(Fs/R),-15, sprintf('SNR = %4.1fdB @ OSR=%d\n',snr,R));
text_handle = text(floor(Fs/R),-30, sprintf('Rbit = %2.2f bits @ OSR=%d\n',Rbit,R));



function [snrdB,ptotdB,psigdB,pnoisedB] = calcSNR(vout,f,fB,w,N)
% SNR calculation in the time domain (P. Malcovati, S. Brigati)
% function [snrdB,ptotdB,psigdB,pnoisedB] = calcSNR(vout,f,fB,w,N)
% vout: Sigma-Delta bit-stream taken at the modulator output
% f:    Normalized signal frequency (fs -> 1)
% fB:     Base-band frequency bins
% w:      windowing vector
% N:    samples number
% snrdB:     SNR in dB
% ptotdB:    Bit-stream power spectral density (vector)
% psigdB:    Extracted signal power spectral density (vector)
% pnoisedB:  Noise power spectral density (vector)
```

```
fB=ceil(fB);
signal=(N/sum(w))*sinusx(vout(1:N).*w,f,N);          % Extracts sinusoidal signal
noise=vout(1:N)-signal;                               % Extracts noise components
stot=((abs(fft((vout(1:N).*w)'))).^2);                % Bit-stream PSD
ssignal=(abs(fft((signal(1:N).*w)'))).^2;       % Signal PSD
snoise=(abs(fft((noise(1:N).*w)'))).^2;               % Noise PSD
pwsignal=sum(ssignal(1:fB));              % Signal power
pwnoise=sum(snoise(1:fB));                            % Noise power
snr=pwsignal/pwnoise;
snrdB=dbp(snr);
norm=sum(stot)/sum(vout(1:N).^2)*N;
        % PSD normalization
if nargout > 1
        ptot=stot/norm;
        ptotdB=dbp(ptot);
end
if nargout > 2
        psig=ssignal/norm;
        psigdB=dbp(psig);
end
if nargout > 3
        pnoise=snoise/norm;
        pnoisedB=dbp(pnoise);
end

function outx = sinusx(in,f,n)
% Extraction of a sinusoidal signal
sinx=sin(2*pi*f*[1:n]);
cosx=cos(2*pi*f*[1:n]);
in=in(1:n);
a1=2*sinx.*in;
a=sum(a1)/n;
b1=2*cosx.*in;
b=sum(b1)/n;
outx=a.*sinx + b.*cosx;

function out = slew(in,alfa,sr,GBW,Ts)
% Models the op-amp slew rate for a discrete time integrator
% in:   input signal amplitude
% alfa: effect of finite gain (ideal op-amp alfa=1)
```

```
% sr:   slew rate in V/s
% GBW:  gain-bandwidth product of the integrator in Hz
% Ts:   sample time
% out:  output signal amplitude
tau=1/(2*pi*GBW);  % Time constant of the integrator
Tmax = Ts/2;
slope=alfa*abs(in)/tau;
if slope > sr                          % Op-amp in slewing
        tsl = abs(in)*alfa/sr - tau;  % Slewing time
        if tsl >= Tmax
                error = abs(in) - sr*Tmax;
        else
                texp = Tmax - tsl;
                error = abs(in)*(1-alfa) + (alfa*abs(in) - sr*tsl) * exp(-texp/tau);
        end

else                                   % Op-amp in linear region
        texp = Tmax;
        error = abs(in)*(1-alfa) + alfa*abs(in) * exp(-texp/tau);
end
out = in - sign(in)*error;

function y=dbv(x)
% dbv(x) = 20*log10(abs(x)); the dB equivalent of the voltage x
y = -Inf*ones(size(x));
nonzero = x~=0;
y(nonzero) = 20*log10(abs(x(nonzero)));

function y=dbp(x)
% dbp(x) = 10*log10(x): the dB equivalent of the power x
y = -Inf*ones(size(x));
nonzero = x~=0;
y(nonzero) = 10*log10(abs(x(nonzero)));
```

APPENDIX C: Ideal Feedback DAC Verilog-A Code

```
`include "discipline.h"
`include "constants.h"

// flash output to dacs  by tufan karalar
// -  16bit thermometer flash output to integer. clocked input.
// intention to be used in the delta sigma ADC ideal model.
//
// vin[15:0]:                    [V,A]
// vclk:   [V,A]
// vo:      data output terminals         [V,A]
//
// INSTANCE parameters
//    mismatch_fact = maximum mismatch as a percentage of the average value []
//    vtrans_clk    = clk high to low transition voltage [V]
//    vref          = voltage that voltage is done with respect to [V]
//    tdel, trise, tfall = {usual} [s]
// MODEL parameters
//    {none}
// Essentially converts an integer to a digital word.

module flash_out_dac(vo,vin,vclk,vdd,vss);
input vclk,vdd,vss;
output vo;
input [15:0] vin;

electrical vo,vclk, vdd, vss;
electrical [15:0] vin;

  real vsupply;
  integer vb[15:0];
  integer int_in,dacsum;
  genvar ii,iii;

  analog begin
    @ ( initial_step ) begin
      vsupply = V(vdd) - V(vss);
    end
          vsupply = V(vdd) - V(vss);
```

```
    @(cross((V(vclk)-vsupply/2), +1)) begin
       dacsum = -8; // this was 0 but instead of shifting the result by 8 at the end it's shifted upfront
       for (iii=15; iii>=0; iii=iii-1) begin
            if(V(vin[iii])>(vsupply/2)) begin
                      vb[iii]=1;
                      dacsum = dacsum + 1;
                   end
            else begin
                      vb[iii]=0;
                end
            end
    end

    V(vo) <+ dacsum;
  end
endmodule
```

APPENDIX D: Decimation Filter Verilog Code and Test Code

Verilog Code:

```
`timescale 1ns / 1ps
module filter_top(filter_in, clk, rst, filter_out);
	input [15:0] filter_in;
	input clk, rst;
	output [15:0] filter_out;
	wire [4:0] thermo_out;
	thermo_to_bin thermo1(.thermo_in(filter_in), .bin_out(thermo_out), .rst(rst));
	dec_filter filter1(.dec_in(thermo_out), .dec_out(filter_out), .clk(clk), .rst(rst));
endmodule
module thermo_to_bin(thermo_in, bin_out, rst);
	input [15:0] thermo_in;
	input rst;
	output reg [4:0] bin_out;
	always @(negedge rst) begin
		if(!rst) begin
			bin_out <= 5'b00000;
		end
		else begin
			bin_out <= bin_out;
		end
	end
	always @(thermo_in) begin
		case(thermo_in)
			16'b0000000000000000 : bin_out <= 5'b00000;
			16'b0000000000000001 : bin_out <= 5'b00001;
			16'b0000000000000011 : bin_out <= 5'b00010;
			16'b0000000000000111 : bin_out <= 5'b00011;
			16'b0000000000001111 : bin_out <= 5'b00100;
			16'b0000000000011111 : bin_out <= 5'b00101;
			16'b0000000000111111 : bin_out <= 5'b00110;
			16'b0000000001111111 : bin_out <= 5'b00111;
			16'b0000000011111111 : bin_out <= 5'b01000;
			16'b0000000111111111 : bin_out <= 5'b01001;
			16'b0000001111111111 : bin_out <= 5'b01010;
			16'b0000011111111111 : bin_out <= 5'b01011;
			16'b0000111111111111 : bin_out <= 5'b01100;
			16'b0001111111111111 : bin_out <= 5'b01101;
			16'b0011111111111111 : bin_out <= 5'b01110;
			16'b0111111111111111 : bin_out <= 5'b01111;
			16'b1111111111111111 : bin_out <= 5'b10000;
			default : bin_out <= 5'b00000;
		endcase
	end
endmodule
module dec_filter(dec_in, dec_out, clk, rst);
	input [4:0] dec_in;
	input clk, rst;
	output reg [15:0] dec_out;
	reg [23:0] acc1;
	reg [23:0] acc2;
	reg [23:0] acc3;
	reg [23:0] diff1;
	reg [23:0] diff2;
	reg [23:0] diff3;
	reg [23:0] acc3_d;
	reg [23:0] diff1_d;
```

```
	reg [23:0] diff2_d;
	reg [5:0] clk_counter;
	reg clk_low;
	always @(posedge clk or negedge rst) begin
		if(!rst) begin
			acc1 <= 0;
			acc2 <= 0;
			acc3 <= 0;
		end
		else begin
			acc1 <= acc1 + dec_in;
			acc2 <= acc2 + acc1;
			acc3 <= acc3 + acc2;
		end
	end
	always @(negedge clk or negedge rst) begin
		if(!rst) begin
			clk_counter <= 0;
		end
		else begin
			clk_counter <= clk_counter + 1;
			clk_low <= clk_counter[5];
		end
	end
always @(posedge clk_low or negedge rst) begin
		if(!rst) begin
			diff1 <= 0;
			diff2 <= 0;
			diff3 <= 0;
			acc3_d <= 0;
			diff1_d <= 0;
			diff2_d <= 0;
		end
		else begin
			diff1 <= acc3 - acc3_d;
			diff2 <= diff1 - diff1_d;
			diff3 <= diff2 - diff2_d;
			acc3_d <= acc3;
			diff1_d <= diff1;
			diff2_d <= diff2;
		end
	end
	always @(posedge clk_low or negedge rst) begin
		if(!rst)begin
			dec_out[15:0] <= 16'h0000;
		end
		else begin
			dec_out[15] <= diff3[23];
			dec_out[14] <= diff3[22];
			dec_out[13] <= diff3[21];
			dec_out[12] <= diff3[20];
			dec_out[11] <= diff3[19];
			dec_out[10] <= diff3[18];
			dec_out[9] <= diff3[17];
			dec_out[8] <= diff3[16];
			dec_out[7] <= diff3[15];
			dec_out[6] <= diff3[14];
			dec_out[5] <= diff3[13];
			dec_out[4] <= diff3[12];
			dec_out[3] <= diff3[11];
```

```
                        dec_out[2] <= diff3[10];
                        dec_out[1] <= diff3[9];
                        dec_out[0] <= diff3[8];
                end
        end
endmodule
```

Verilog Test Code:

```
`timescale 1ns / 1ps
module dec_filter_test;
        reg [15:0] filter_in;
        reg clk, rst;
        wire [15:0] filter_out;
        filter_top UUT(filter_in, clk, rst, filter_out);
        reg clk1;
        integer         data_read    ; // file handler
        integer         data_write    ; // file handler
        integer         scan_file    ; // file handler
        integer i;
        reg [15:0] captured_data;
        `define NULL 0
        initial begin
                data_read = $fopen("filter_in.csv", "r");
                if (data_read == `NULL) begin
                        $display("data_file handle was NULL");
                        $finish;
                end
                data_write = $fopen("filter_output.txt", "w");
        rst = 0;
        clk = 0;
        clk1 = 0;
        #1200 rst = 1;
        end
        always @(posedge clk or negedge rst) begin
                if (!rst)
                        filter_in <= 0;
                else begin
                        scan_file = $fscanf(data_read, "%b\n", captured_data);
                        if (!$feof(data_read)) begin
                                filter_in <= captured_data;
                        end
                        else #400000 $finish;
                end
        end
        always @(posedge clk1)begin
                $fdisplay(data_write, "%b", filter_out);
        end
        always #500 clk = ~clk;
        always #32000 clk1 = ~clk1;
endmodule
```

APPENDIX E: Decimation Filter Synthesis and Place & Route Tcl Scripts

Synthesis Tcl Script:

```
set_attribute lib_search_path
/work/kits/tsmc/lib/180bcd/TSMCHOME/digital/Front_End/timing_power_noise/CCS/tcb018bcdgp2
a_110a
set_attribute library tcb018bcdgp2atc_ccs.lib
set_attribute use_scan_seqs_for_non_dft false
set_attribute optimize_constant_latches false
set_attribute optimize_constant_1_flops false
set_attribute optimize_constant_0_flops false
read_hdl -v2001 {dec_filter.v}
elaborate filter_top
define_clock -period 80000 -fall 80 -rise 80 -name clkin1 -domain domain_1 [find /des* -port
ports_in/clk]
set_attribute slew {100 100 200 200} [find -clock clkin1]
external_delay -input 500 -clock [find -clock clkin1] -edge_rise [find /des* -port ports_in/*]
external_delay -output 500 -clock [find -clock clkin1] -edge_rise [find /des* -port ports_out/*]
set_attribute external_pin_cap 5 [find /des* -port ports_out/*]
set_attribute operating_conditions NCCOM
synthesize -to_mapped -effort high
report timing > report_time.txt
report gates > report_gates.txt
report area > report_area.txt
write_encounter design filter_top -basename encounter/dec_filter -lef
/work/kits/tsmc/lib/180bcd/TSMCHOME/digital/Back_End/lef/tcb018bcdgp2a_110a/lef/tcb018bcdgp
2a_6lm.lef
write_sdf -edges check_edge > filter_top.sdf
```

Place & Route Tcl Script

```
#########################################################
#  Encounter Command Logging File                       #
#  Created on Thu Sep 22 16:27:16 2016                  #
#########################################################
#@(#)CDS: First Encounter v08.10-s273_1 (32bit) 06/16/2009 02:26 (Linux 2.6)
#@(#)CDS: NanoRoute v08.10-s155 NR090610-1622/USR60-UB (database version 2.30, 78.1.1)
{superthreading v1.11}
#@(#)CDS: CeltIC v08.12-s254_1 (32bit) 06/11/2009 13:55:16 (Linux 2.6.9-67.0.10.ELsmp)
#@(#)CDS: CTE v08.10-s204_1 (32bit) Jun 10 2009 13:59:08 (Linux 2.6.9-67.0.10.ELsmp)
#@(#)CDS: CPE v08.12-s010
loadConfig
/home/alpergirgin/projects/BASAK/alper_digital/DEC_FILTER/par/encounter/dec_filter.conf 0
commitConfig
fit
setDrawView fplan
setDrawView place
getIoFlowFlag
setIoFlowFlag 0
floorPlan -site core10T -d 500 125 5.0 5.0 5.0 5.0
uiSetTool select
getIoFlowFlag
fit
loadIoFile /home/alpergirgin/projects/BASAK/alper_digital/DEC_FILTER/par/filter_top.save.io
windowSelect 27.185 88.593 -18.134 -23.331
windowSelect 54.651 102.325 -54.526 -33.630
windowSelect 306.648 197.769 26.498 22.675
windowSelect 233.864 -78.948 244.163 -76.889
windowSelect 515.387 85.846 463.889 -11.658
deselectAll
globalNetConnect VDD -type pgpin -pin VDD -inst * -module {}
```

```
globalNetConnect VSS -type pgpin -pin VSS -inst * -module {}
globalNetConnect VDD -type tiehi
globalNetConnect VSS -type tielo
addRing -spacing_bottom 0.46 -width_left 1 -width_bottom 1 -width_top 1 -spacing_top 0.46 -
layer_bottom METAL1 -stacked_via_top_layer METAL6 -width_right 1 -around core -jog_distance
0.28 -offset_bottom 0.28 -layer_top METAL1 -threshold 0.28 -offset_left 0.28 -spacing_right 0.46 -
spacing_left 0.46 -offset_right 0.28 -offset_top 0.28 -layer_right METAL2 -nets {VSS VDD } -
stacked_via_bottom_layer METAL1 -layer_left METAL2
addStripe -block_ring_top_layer_limit METAL6 -max_same_layer_jog_length 0.88 -
padcore_ring_bottom_layer_limit METAL4 -set_to_set_distance 100 -stacked_via_top_layer
METAL6 -padcore_ring_top_layer_limit METAL6 -spacing 1 -merge_stripes_value 0.28 -layer
METAL5 -block_ring_bottom_layer_limit METAL4 -stop_x 110 -width 3 -nets {VSS VDD } -start_x
100 -stacked_via_bottom_layer METAL1
addStripe -block_ring_top_layer_limit METAL6 -max_same_layer_jog_length 0.88 -
padcore_ring_bottom_layer_limit METAL4 -set_to_set_distance 100 -stacked_via_top_layer
METAL6 -padcore_ring_top_layer_limit METAL6 -spacing 1 -merge_stripes_value 0.28 -layer
METAL5 -block_ring_bottom_layer_limit METAL4 -stop_x 210 -width 3 -nets {VSS VDD } -start_x
200 -stacked_via_bottom_layer METAL1
addStripe -block_ring_top_layer_limit METAL6 -max_same_layer_jog_length 0.88 -
padcore_ring_bottom_layer_limit METAL4 -set_to_set_distance 100 -stacked_via_top_layer
METAL6 -padcore_ring_top_layer_limit METAL6 -spacing 1 -merge_stripes_value 0.28 -layer
METAL5 -block_ring_bottom_layer_limit METAL4 -stop_x 310 -width 3 -nets {VSS VDD } -start_x
300 -stacked_via_bottom_layer METAL1
addStripe -block_ring_top_layer_limit METAL6 -max_same_layer_jog_length 0.88 -
padcore_ring_bottom_layer_limit METAL4 -set_to_set_distance 100 -stacked_via_top_layer
METAL6 -padcore_ring_top_layer_limit METAL6 -spacing 1 -merge_stripes_value 0.28 -layer
METAL5 -block_ring_bottom_layer_limit METAL4 -stop_x 410 -width 3 -nets {VSS VDD } -start_x
400 -stacked_via_bottom_layer METAL1
addStripe -block_ring_top_layer_limit METAL6 -max_same_layer_jog_length 0.88 -
padcore_ring_bottom_layer_limit METAL4 -set_to_set_distance 100 -stacked_via_top_layer
METAL6 -padcore_ring_top_layer_limit METAL6 -spacing 1 -merge_stripes_value 0.28 -layer
METAL5 -block_ring_bottom_layer_limit METAL4 -stop_x 410 -width 3 -nets {VSS VDD } -start_x
400 -stacked_via_bottom_layer METAL1
sroute -allowJogging true
zoomBox 75.936 165.496 -21.567 72.113
zoomBox 16.168 129.151 -3.065 99.207
selectWire 3.7600 117.3200 495.7600 118.3200 1 VSS
uiSetTool moveWire
editMove y 4.0
deselectAll
selectWire 2.3000 118.7800 497.2200 119.7800 1 VDD
uiSetTool moveWire
editMove y -2.19
deselectAll
selectWire 3.7600 121.3200 495.7600 122.3200 1 VSS
uiSetTool moveWire
editMove y -3.23
zoomBox 49.157 -35.690 -27.060 61.127
zoomBox 10.239 -3.335 -2.952 26.407
deselectAll
selectWire 3.7600 3.7600 495.7600 4.7600 1 VSS
uiSetTool moveWire
editMove y -3.63
deselectAll
selectWire 2.3000 2.3000 497.2200 3.3000 1 VDD
uiSetTool moveWire
editMove y 2.255
deselectAll
selectWire 3.7600 0.1300 495.7600 1.1300 1 VSS
uiSetTool moveWire
```

```
editMove y 2.865
deselectAll
setPlaceMode -timingDriven true
placeDesign -noPrePlaceOpt
addCTSCellList {CKBD12 CKBD16 CKBD4 CKBD6 CKBD8 CKND0 CKND1 CKND12 CKND16
CKND8}
clockDesign -genSpecOnly Clock.ctstch
clockDesign -specFile Clock.ctstch -outDir clock_report -fixedInstBeforeCTS
addFiller -cell FILL1 FILL16 FILL2 FILL32 FILL4 FILL64 FILL8 -prefix FILLER
setNanoRouteMode -quiet -routeTopRoutingLayer 4
setNanoRouteMode -quiet -routeBottomRoutingLayer 1
setNanoRouteMode -quiet -drouteEndIteration default
setNanoRouteMode -quiet -routeWithTimingDriven false
setNanoRouteMode -quiet -routeWithSiDriven false
routeDesign -globalDetail
addMetalFill -layer { 1 2 3 4 } -nets { VSS VDD }
verifyGeometry
verifyConnectivity -type all -error 1000 -warning 50
setMetalFill -layer 1 -windowSize 100.000 100.000 -windowStep 50.000 50.000 -minDensity 30.000
-maxDensity 80.000
setMetalFill -layer 2 -windowSize 100.000 100.000 -windowStep 50.000 50.000 -minDensity 30.000
-maxDensity 80.000
setMetalFill -layer 3 -windowSize 100.000 100.000 -windowStep 50.000 50.000 -minDensity 30.000
-maxDensity 80.000
setMetalFill -layer 4 -windowSize 100.000 100.000 -windowStep 50.000 50.000 -minDensity 30.000
-maxDensity 80.000
setMetalFill -layer 5 -windowSize 100.000 100.000 -windowStep 50.000 50.000 -minDensity 30.000
-maxDensity 80.000
setMetalFill -layer 6 -windowSize 100.000 100.000 -windowStep 50.000 50.000 -minDensity 30.000
-maxDensity 80.000
verifyMetalDensity -reportfile filter_top.density.rpt
write_sdf -edges check_edge filter.sdf
saveNetlist filter.v
streamOut dec_filter.gds -mapFile streamOut.map -libName DesignLib -units 2000 -mode ALL
saveDesign /home/alpergirgin/projects/BASAK/alper_digital/DEC_FILTER/par/filter.enc
```

APPENDIX – F: Codes Used In Measurement Setup

Arduino Code of Test Setup:

```
#include <SPI.h>
const int chipSelectPin = 10;
int charToBinary(char charIn);
void writeRegister(byte thisRegister, byte thisValue);
void writeRegister(byte thisRegister, byte thisValue);
String Buffer;
char char1;
char char2;
char char3;
char char4;
char char5;
char char6;
int reg1 = 0;
int Address = 0;
int Address_read = 0;
int Data1 = 0;
int flag = 0;

void setup() {
 Serial.begin(9600);
  // start the SPI library:
  SPI.begin();
  SPI.beginTransaction(SPISettings(1000000, MSBFIRST, SPI_MODE0));
  // initalize the  data ready and chip select pins:
  pinMode(chipSelectPin, OUTPUT);
}
void loop() {
  while(Serial.available() > 0) {
    Buffer = Serial.readStringUntil(';');
   // Serial.println(Buffer);
    flag = 1;
   // Serial.println(flag);
    }
    if(flag != 0) {
      char1 = Buffer.charAt(0);
      char2 = Buffer.charAt(1);
      char3 = Buffer.charAt(2);
      char4 = Buffer.charAt(3);
      char5 = Buffer.charAt(4);
      char6 = Buffer.charAt(5);
      Address = {(charToBinary(char1)<<4) + charToBinary(char2)};
      Data1 = {(charToBinary(char3)<<4) + charToBinary(char4)};
      Address_read = {(charToBinary(char5)<<4) + charToBinary(char6)};
      writeRegister(Address, Data1);   // First byte address + command, second byte data
      Serial.println(readRegister(Address_read), HEX);
      flag = 0;
     }
  }
//Converts value of char to binary 0-F.
int charToBinary(char charIn) {
       switch(charIn) {
      case '0' : reg1 = 0x0;
      break;
      case '1' : reg1 = 0x1;
      break;
      case '2' : reg1 = 0x2;
```

```
      break;
      case '3' : reg1 = 0x3;
      break;
      case '4' : reg1 = 0x4;
      break;
      case '5' : reg1 = 0x5;
      break;
      case '6' : reg1 = 0x6;
      break;
      case '7' : reg1 = 0x7;
      break;
      case '8' : reg1 = 0x8;
      break;
      case '9' : reg1 = 0x9;
      break;
      case 'A' : reg1 = 0xA;
      break;
      case 'B' : reg1 = 0xB;
      break;
      case 'C' : reg1 = 0xC;
      break;
      case 'D' : reg1 = 0xD;
      break;
      case 'E' : reg1 = 0xE;
      break;
      case 'F' : reg1 = 0xF;
      break;
    }
    return (reg1);
}
//Read from or write to register from the HES_SPI:
unsigned int readRegister(byte thisRegister) {
  unsigned int result = 0;   // result to return
  digitalWrite(chipSelectPin, LOW);
  // send the device the register you want to read:
  SPI.transfer(thisRegister);
  // send a value of 0 to read the first byte returned:
  result = SPI.transfer(0x00);
  digitalWrite(chipSelectPin, HIGH);
  // return the result:
  return (result);
}
//Sends a write command to HES_SPI
void writeRegister(byte thisRegister, byte thisValue) {
  // take the chip select low to select the device:
  digitalWrite(chipSelectPin, LOW);
  delayMicroseconds(20);
  SPI.transfer(thisRegister); //Send address and write command
  SPI.transfer(thisValue);  //Send value to record into register
  delayMicroseconds(20);
  // take the chip select high to de-select:
  digitalWrite(chipSelectPin, HIGH);
}
```

Verilog Code (FPGA) of Test Setup:

```
`timescale 1ns / 1ps
module master_ram(miso, clk, rst, ce, sck, mosi, douta, enb);
        input clk;                              // 100 Mhz System Clock
        input rst;
        input miso;
```

```
        output [15:0] douta;
        output sck, mosi, ce;
        output reg enb;
        reg [11:0] addra;
        reg [11:0] addrb;
        reg [7:0] count;
        reg [7:0] cnt_ram1;
        reg [1:0] cnt_addra;
        wire [7:0] data_out;
  wire [15:0] doutb;
        wire [15:0] dina;
        wire clkb;
        reg wea;
        reg clk_cs;
        reg [7:0] cnt_clk;
        wire miso, ce, sck, mosi;
        wire clk_100;
        wire clk_50;
        wire clk_25;
        wire clk_55;
        wire [35:0] ILAControl0;
        always @(posedge clk_50 or negedge rst) begin                   // sets the CE signal of spi
master
                if(!rst) begin
                        enb <= 0;
                        count <= 0;
                end
                else begin
                        count <= count + 1;
                        if(count == 0) enb <= 1;
                        else if (count == 123) begin
                                count <= 0;
                                enb <= 0;
                        end
                        else enb <= 0;
                end
        end
        always @(posedge clk_50 or negedge rst) begin                   // sets the slow clock of
chipscope
                if(!rst) begin
                        clk_cs <= 0;
                        cnt_clk <= 0;
                end
                else begin
                        if(cnt_clk == 61) begin
                                cnt_clk <= 0;
                                clk_cs <= ~clk_cs;
                        end
                        else begin
                                cnt_clk <= cnt_clk + 1;
                                clk_cs <= clk_cs;
                        end
                end
        end
        chipscope_ila_32 ila2(.CONTROL(ILAControl0), .CLK(clk_cs),
.DATA({4'b0000,addrb,doutb[15:8],data_out}), .TRIG0({3'b000,rst})) /* synthesis syn_noprune =
true */;
        chipscope_icon icon1(.CONTROL0(ILAControl0)) /* synthesis syn_noprune = true */;
        clk_division clk_div(.CLK_IN1(clk), .CLK_OUT1(clk_100), .CLK_OUT2(clk_50),
.CLK_OUT3(clk_25), .CLK_OUT4(clk_55), .RESET(~rst));
```

```
	bram bram1(.rst(rst), .clka(clk_50), .wea(wea), .addra(addra), .dina(data_out), .douta(douta),
.clkb(clk_50), .addrb(addrb), .doutb(doutb));
	spi_master spi_master1(.SRIN_16bit(doutb), .SROUT_8bit(data_out), .MISO(miso),
.RST(rst), .ENB(enb), .SYS_CLK_MS(clk_50), .SCK(sck), .MOSI(mosi), .CE(ce));
	// Sets the ram addresses, Port-B for read commands to send master, Port-A for read and
write adc_out into ram.
	always @(posedge clk_50 or negedge rst) begin
		if(!rst) begin
			addrb <= 0;
			addra <= 7;
			wea <= 1;
			cnt_ram1 <= 0;
			cnt_addra <= 0;
		end
		else begin
			if(cnt_ram1 == 123) begin
				if(addrb == 6) begin
					addrb <= 5;
				end
				else begin
					addrb <= addrb + 1;
				end
				if(addra == 12'hFFF) addra <= 7;
				else addra <= addra + 1;
				cnt_ram1 <= 0;
			end
			else cnt_ram1 <= cnt_ram1 + 1;
		end
	end
endmodule
module spi_master(SRIN_16bit, MISO, RST, ENB, SYS_CLK_MS, SCK, MOSI, CE, SROUT_8bit);
	input [15:0] SRIN_16bit;
	input MISO;
	input SYS_CLK_MS, RST, ENB;
	output reg SCK, CE;
	output MOSI;
	output reg [7:0] SROUT_8bit;
	reg [15:0] SR_master;
	reg [5:0] counter;
	reg [3:0] cnt;
	reg [2:0] SCKr;
	wire SCK_re, SCK_fe;
	assign SCK_re = (SCKr[2:0] == 3'b011);
	assign SCK_fe = (SCKr[2:0] == 3'b100);
	always @(posedge SYS_CLK_MS) begin
		SCKr <= {SCKr[1:0], SCK};
	end
	assign MOSI = CE ? 0 : SR_master[15];
	always @(posedge SYS_CLK_MS or negedge RST or posedge ENB)begin
		if(!RST)begin
			SR_master <= 16'h0000;
			counter <= 0;
		//	MOSI <= 0;
			CE <= 1;
			SCK <= 0;
			cnt <= 0;
		end
		else begin
			if(ENB)begin
				CE <= 0;
```

```
                SCK <= 0;
        //      MOSI <= 0;
                counter <= 0;
                SR_master[15:0] <= SRIN_16bit;
        end
        else begin
                if(CE == 1) begin
                        SCK <= 0;
        //              MOSI <= 0;
                end
                else begin
                        if(cnt == 2) begin
                                SCK <= ~SCK;
                                cnt <= 0;
                        end
                        else
                                cnt <= cnt + 1;
                end
                if(SCK_re)begin
                        SR_master[15] <= SR_master[14];
                        SR_master[14] <= SR_master[13];
                        SR_master[13] <= SR_master[12];
                        SR_master[12] <= SR_master[11];
                        SR_master[11] <= SR_master[10];
                        SR_master[10] <= SR_master[9];
                        SR_master[9] <= SR_master[8];
                        SR_master[8] <= SR_master[7];
                        SR_master[7] <= SR_master[6];
                        SR_master[6] <= SR_master[5];
                        SR_master[5] <= SR_master[4];
                        SR_master[4] <= SR_master[3];
                        SR_master[3] <= SR_master[2];
                        SR_master[2] <= SR_master[1];
                        SR_master[1] <= SR_master[0];
                        SR_master[0] <= MISO;
                        counter <= counter + 1;
                        if(counter == 15) begin
                        #20     CE <= 1;
                                counter <= 0;
                                SROUT_8bit <= {SR_master[6:0], MISO};
                        end
                end
        end
    end
  end
endmodule
```

CURRICULUM VITAE

Name Surname **:** Alper Girgin

Place and Date of Birth **:** Bartın, 24.08.1990

E-Mail **:** alpergirgin16@hotmail.com

Address **:** Bozkurt district, Bilezikçi street, 92/2, Şişli, İstanbul

EDUCATION

- **B.Sc.** **:** Electronics Engineering, ITU (2014)
- **High School** **:** Kocaeli Körfez Science High School (2008)

PROFESSIONAL EXPERIENCE:

- IC Design Engineer, OATMAKE Elektronik (2015-2017)
- Reserchер, TUBITAK Project -High Speed DAC Design (2014-2015)