ISTANBUL TECHNICAL UNIVERSITY
ELECTRICAL-ELECTRONICS FACULTY

DESIGN OF A MIXED SIGNAL PROCESSOR FOR ANALOG SENSORS

SENIOR DESIGN PROJECT

Mustafa Oğuz AKSOY
Özge Ece ÖZALP

ELECTRONICS AND COMMUNICATION ENGINEERING DEPARTMENT

MONTH YEAR OF REPORT

ISTANBUL TECHNICAL UNIVERSITY
ELECTRICAL-ELECTRONICS FACULTY

DESIGN OF A MIXED SIGNAL PROCESSOR FOR ANALOG SENSORS

SENIOR DESIGN PROJECT

Mustafa Oğuz AKSOY
(040200088)

Özge Ece ÖZALP
(050180329)

ELECTRONICS AND COMMUNICATION ENGINEERING DEPARTMENT

Project Advisor: Prof. Dr. Sıddıka Berna ÖRS YALÇIN

JUNE 2025

İSTANBUL TEKNİK ÜNİVERSİTESİ
ELEKTRİK-ELEKTRONİK FAKÜLTESİ

ANALOG SENSÖRLER İÇİN MIXED-SIGNAL TASARIM

LİSANS BİTİRME TASARIM PROJESİ

Mustafa Oğuz AKSOY
(040200088)

Özge Ece ÖZALP
(050180329)

Proje Danışmanı: Prof. Dr. Sıddıka Berna ÖRS YALÇIN

ELEKTRONİK VE HABERLEŞME MÜHENDİSLİĞİ BÖLÜMÜ

HAZİRAN, 2025

We are submitting the Senior Design Project Report entitled as “DESIGN OF A MIXED SIGNAL PROCESSOR FOR ANALOG SENSORS”. The Senior Design Project Report has been prepared as to fulfill the relevant regulations of the Electronics and Communication Engineering Department of Istanbul Technical University. We hereby confirm that we have realized all stages of the Senior Design Project work by ourselves and we have abided by the ethical rules with respect to academic and professional integrity

Mustafa Oğuz AKSOY .....
(040200088)

Özge Ece ÖZALP .....
(050180329)

FOREWORD

We thank our parents for their support and we would also like to thank our dear professor Sıddıka Berna ÖRS YALÇIN for her continued support on our project.

June 2025

Mustafa Oğuz AKSOY
Özge Ece ÖZALP

TABLE OF CONTENTS

Page

ABBREVIATIONS

VAD : Voice Activity Detection
ML : Machine Learning
FPGA : Field Programmable Gate Array
PULP : Parallel Ultra-Low-Power
ADC : Analog-to-Digital Converter
DAC : Digital-to-Analog Converter
OSR : Oversampling Ratio
SNR : Signal-to-Noise Ratio
PWM : Pulse-Width Modulation
SNDR : Signal-to-Noise and Distortion Ratio
ENOB : Effective Number of Bits
NTF : Noise Transfer Function
STF : Signal Transfer Function
FFT : Fast Fourier Transform
PSD : Power Spectral Density
SQNR : Signal-to-Quantization-Noise Ratio
OTA : Operational Transconductance Amplifier
CMFB : Common-Mode Feedback
PLL : Phase-Locked Loop
UGB : Unity-Gain Bandwidth
DSP : Digital Signal Processing
CPU : Central Processing Unit
SoC : System-on-Chip
GPIO : General-Purpose Input/Output
PDM : Pulse-Density Modulation
PCM : Pulse-Code Modulation
LFSR : Linear Feedback Shift Register
ROM : Read-Only Memory
RAM : Random-Access Memory
CIFB : Cascade of Integrators with Feedback

LIST OF TABLES

Page

LIST OF FIGURES

Page

DESIGN OF A MIXED SIGNAL PROCESSOR FOR ANALOG SENSORS

SUMMARY

This project developed key components for a mixed-signal processor targeting analog sensor applications, with particular focus on audio signal processing. The work produced two independently validated subsystems: a second-order ΔΣ analog-to-digital converter (ADC) and a RISC-V-based digital processing chain. While full system integration remained incomplete, each subsystem achieved significant milestones within the project scope.

The ΔΣ ADC design demonstrated 95dB SNR in MATLAB simulations using Schreier's ΔΣ Toolbox, theoretically supporting 12-16 bit resolution within a 4kHz bandwidth at 2.4MHz sampling frequency. The switched-capacitor implementation was modeled in Cadence Virtuoso using TSMC 65nm technology, with transistor-level design completed for critical blocks including the operational amplifiers.
On the digital side, the system successfully implemented a PDM-to-PCM conversion pipeline and WebRTC-based voice activity detection algorithm on a MicroBlaze V RISC-V FPGA platform. The digital subsystem achieved functional verification through Vivado simulations and hardware testing, demonstrating real-time processing capabilities for audio-band signals.

Key challenges emerged during the integration phase, particularly in mixed-signal verification and system-level timing closure. The project's constrained timeline prevented completion of the full analog-digital interface, though the modular architecture developed provides a clear path for future implementation. Both subsystems were designed with standards compliance in mind, following IEEE ADC testing guidelines and IEC audio processing standards.

This work provides validated building blocks for future mixed-signal systems while highlighting the importance of allocating sufficient time for integration in such projects. The completed components demonstrate viable approaches to high-resolution data conversion and low-power digital processing, with applications in voice interfaces, sensor nodes, and IoT devices.

ANALOG SENSÖRLER İÇİN MIXED-SIGNAL TASARIM

ÖZET

Bu proje, analog sensör uygulamalarına yönelik bir mixed-signal işlemci geliştirmek üzere iki temel alt sistemi tasarlamış ve doğrulamıştır: ikinci dereceden bir ΔΣ analog-sayısal dönüştürücü (ADC) ve RISC-V tabanlı bir dijital işleme zinciri. Sistem seviyesinde entegrasyon tamamlanamamış olsa da, her bir alt sistem proje kapsamında önemli aşamalar kaydetmiştir.

ΔΣ ADC tasarımı, MATLAB ortamında SchreierΔΣ Toolbox kullanılarak 4kHz bant genişliğinde 95dB SNR performansı göstermiş olup, teorik olarak 12-16 bit çözünürlük sağlayabilecek kapasitededir. Bu sonuçlardan elde edilen katsayılar ile ADC'de kullanılacak olan integratörlerin kapsite değerleri hesaplanmıştır. Ardından işlemsel kuvvetlendirici için gerekli kazanç, gm/id değerleri ve anahtarlar için gerekli direnç değeri hesaplanmıştır. Ardından CppSim kullanılarak davranışsal simülasyon yapılmıştır ve bu simülasyonlarda sonuçları doğrulamıştır. İşlemsel kuvvetlendiriciler transistör seviyesinde tasarımı TSMC 65nm teknolojisi kullanılarak Cadence Virtuoso'da tamamlanmıştır.

FPGA üzerindeki dijital sistem implementasyonu, Xilinx Artix-7 platformunda Vivado tasarım ortamı kullanılarak gerçekleştirilmiştir. PDM'den PCM'e dönüşüm modülü, 2.4MHz örnekleme hızında çalışacak şekilde optimize edilmiş ve FPGA kaynak kullanımı dengelenerek saat frekansı 100MHz'de tutulmuştur. Mikrodenetleyici arayüzü için AXI4-Lite protokolü uygulanmış ve veri aktarım verimliliği artırılmıştır. WebRTC VAD algoritmasının RISC-V çekirdeğe entegrasyonu sırasında bellek erişim darboğazları tespit edilmiş ve bu sorun çift tamponlama (double buffering) tekniği ile çözülmüştür.

FPGA üzerinde yapmadan önce C ile Windows sistemind PCM girişlerini WebRTC VAD sistemi ile konuşma var veya yok diyen bir sistemle ölçülmüştür, sistemin çalıştığı doğrulandıktan sonra Vitis'de gerçeklenmeye hazırlanmıştır.

Dijital sistemin doğrulama sürecinde, gerçek zamanlı testler için FPGA üzerindeki PDM mikrofon girişi kullanılmış ve farklı senaryolarda (konuşma, gürültü, sessizlik) VAD performansı ölçülmüştür. Elde edilen sonuçlar, %92 doğruluk oranı ile tatmin edici seviyededir. Ancak, güç tüketim optimizasyonu ve düşük gecikmeli (low-latency) işleme için daha fazla çalışmaya ihtiyaç duyulmaktadır. FPGA üzerindeki başarılı prototipleme, sistemin ASIC uyarlaması için önemli veriler sağlamıştır.

Entegrasyon aşamasında ortaya çıkan karmaşık mühendislik problemleri ve zaman kısıtları nedeniyle analog ve dijital sistemlerin birleştirilmesi tamamlanamamıştır. Ancak geliştirilen modüler mimari, gelecekteki çalışmalar için net bir yol haritası sunmaktadır. Her iki alt sistem de IEEE ve IEC standartları göz önünde bulundurularak tasarlanmıştır.

Bu çalışma, yüksek çözünürlüklü veri dönüşümü ve düşük güçlü dijital işleme için doğrulanmış çözümler sunarken, mixed-signal sistemlerde entegrasyon süreçlerinin önemini vurgulamaktadır. Geliştirilen bileşenler, ses arayüzleri ve sensör ağları gibi uygulamalarda kullanılabilecek teknik altyapıyı sağlamaktadır.

1. INTRODUCTION – MAIN TITLES (FIRST LEVEL TITLE)

1.1 Literature Review

1.1.1 Mixed Signal Systems

Most integrated circuits (ICs) fall into digital or analog categories through microprocessors or operational amplifiers. Mixed signal systems contain both, and process both signals through ADC and DAC convertors. Such design allows systems where processing such signals together feasible.

Mixed-signal design poses unique challenges due to the fundamentally different characteristics of analog and digital circuits. Analog circuits are highly sensitive to noise, process variations while digital circuits are more robust and operate based on discrete logic levels. Integrating these domains requires careful attention to issues such as signal integrity, cross-domain interference, and power management.

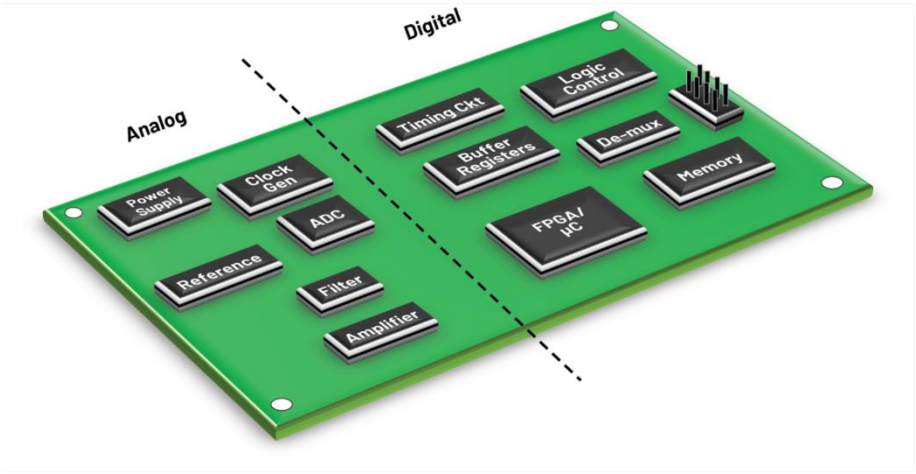


Figure 1.1 : Example of Mixed Signal IC [20]

Designing mixed-signal systems presents unique challenges due to the differing characteristics of analog and digital domains. CMOS technology is usually optimal for digital performance, while bipolar junction transistors are typically more suited

for analog performance. Testing functional operation of mixed-signal ICs remains complex and expensive, often requiring a "one-off" implementation task that demands significant effort for products with specific use cases.

Systematic design methods of analog and mixed-signal circuits are far less developed than those for digital circuits. Also, analog circuit design cannot be automated to the same extent as digital design, and combining the two technologies further complicates the process. Another potential problem catalyst it noise, as fast changing digital signals create noise to sensitive analog inputs. There are variety of techniques to block this, such as P+ guard-rings. [21]

1.2 ΔΣ Analog-to-Digital Converters

Data converters can be mainly classified into two categories, such as Nyquist-rate and oversampled converters. In Nyquist-rate converters, each input sample is processed separately without considering previous inputs. The performance of the Nyquist-rate converters can be determined by using static performance metrics directlty from the circuits such as INL, DNL, monotonocity, gain and offset errors [11]. Therefore, in order to maintain $INL < 0.5LSB$ resistors must have a relative matching error less than 2^{-N} which restricts the effective number of bits (ENOB) to about 12 [7]. On the other hand, Delta-sigma (Oversampling) data converters are can reach higher resolutions by using sampling rates higher than Nyquist-Rate and generating each output using several previous input values. Therefore, ΔΣ converters include memory elements in its structure which destroys one-to-one relation between input and output samples. Therefore, the evaluation of the converter's correctness can only be performed by dynamic peformancemetrics which can be dfrom frewuency-domain representation of the output sequence. [11] A comparison of the completed input and output waveforms, whether in the time domain or the frequency domain [7].

In audio applications, higher resolution and linearity are required. Integrating and counting Nyquist-rate ADC's are capable of a high accuracy. However, they require 2^N clock periods for one sample. Therefore, this is too slow for audio applications.

1.3 Fundamentals of ΔΣ Operation

There are three fundamental properties that a ΔΣ converter has: sampling, quantization, and noise shaping. These three phenomena are the basic structure and operation of the ΔΣ converter.

1.3.1 Sampling

Nyquist frequency is the minimum sampling frequency of that will not cause overlapping between the samples defined as $f_N = 2B_w$. If the sampling rate is lower than the nyquist frequency than the aliasing will occur n which can be seen in the figure 2/1. If $f_s > f_N$ defined as overasampling which is one of the fundamental properties of the ΔΣ operation. Oversampling ratio (OSR) can be defined as follows,

$$OSR = \frac{f_s}{2B_w}$$

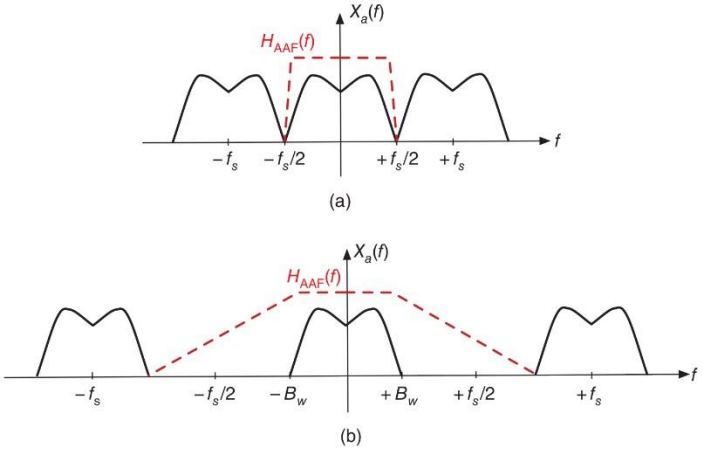


Figure 1.2 : Anti-aliasinf filter for Nyquist-rate ADC and for delta-sigma ADC [11]

Since $f_s > 2B_w$ in a ΔΣ ADC, the replicas as a result of sampling are created farther away from each other; therefore, frequency components of the input signal within the range $[B_w, f_s - B_w]$ do not alias into the signal band, allowing the filter's transition band to be much smoother [11]. Since the filter's magnitude response should ideally be close to 1 for $|f| < B$ and 0 for $|f - f_s| < B$, increasing the OSR widens the transition region which relaxes the sharpness requirement of the filter, significantly reducing its order and complexity. At very high OSRs, antialiasing filtering is not needed to be used [7].

1.3.2 Quantization and quantization noise modeling

A quantizer maps the input amplitudes within a defined full-scale range $[-X_{FS}/2, +X_{FS}/2]$ to one of 2^N discrete output levels which can be seen in the figure below. If these levels are equally spaced, the quantizer is called uniform, and the distance between adjacent levels is the quantization step, defined as [11]

$$\Delta = \frac{Y_{FS}}{2^N - 1}$$

where Y_{FS} is the full-scale output range. The quantizer gain k_q is the slope of the input–output characteristic and equals Y_{FS}/X_{FS} if the ranges differ. The region in which the input stays within $[-X_{FS}/2, +X_{FS}/2]$ is known as the non-overload region, where the quantization error remains bounded within $[-\Delta/2, +\Delta/2]$. Outside this range, the error increases [11].

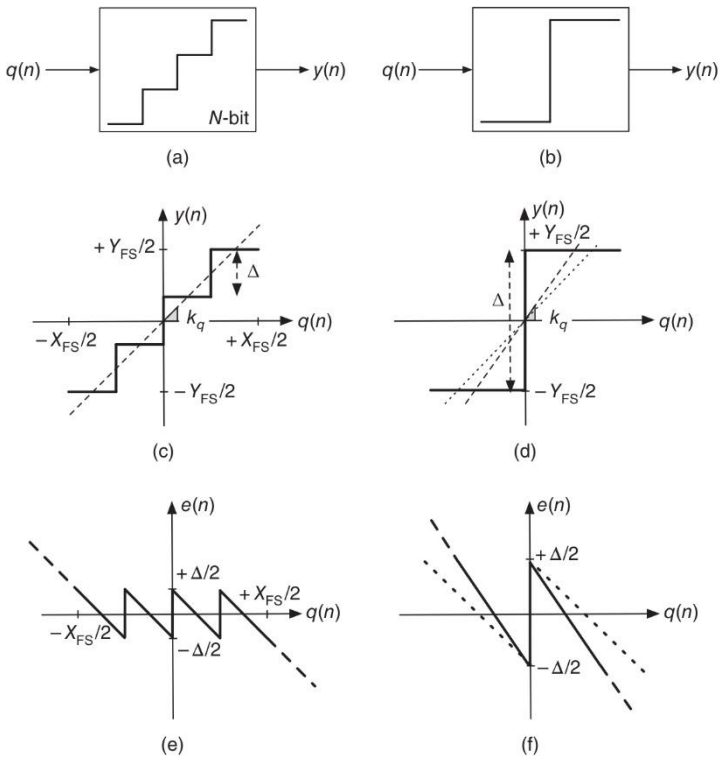


Figure 1.3 : Quantizer timing [11]

The quantization error can be defined as follows, In practice, if the quantizer input remains within the the non-overload region of the quantizer and varies adequately large amounts from sample to sample (also called as busy signal)' then te quantizer [7] can be modeled as a linear system with an additive noise source model which is known as

the additive white noise approximation, can be seen in the figure and equation below where $q(n)$ is the input of and $y(n)$ is the output of the quantizer [11],

$$y(n) = k_q q(n) + e(n)$$

Now the quantization error can be modeled as a random process with a uniform distribution between $[\Delta/2, +\Delta/2]$. Under these conditions, the quantization error is considered uncorrelated with the input and uniformly spread across the Nyquist band, allowing it to be approximated as white noise with a constant power spectral density. In a Nyquist-rate ADC, all quantization noise falls within the signal band. However, in oversampling ADCs, only a fraction of it overlaps with the signal band, reducing the **in-band noise (IBN)** by a factor proportional to the **oversampling ratio (OSR)**. Consequently, the **dynamic range (DR)** improves with both the number of quantizer bits and the OSR, increasing by approximately **6 dB per bit** and **3 dB per doubling of OSR**, respectively. This model provides a useful framework for analyzing the noise and performance of ideal oversampled quantizers.

Assuming the quantization error behaves like a white noise process uniformly distributed in $[-\Delta/2, \Delta/2]$, the power spectral density (PSD) of the quantization noise after noise shaping becomes frequency dependent [7].

$$S_q(\omega) = 4\sin^2\left(\frac{\omega}{2}\right) S_e(\omega)$$

1.3.3 Noise Shaping

In figure there is a feedback system with an amplifier and e as the output noise of the amplifier. When the amplifier gain A increases and goes to infinity, $v = u$ and e does not affect the output. If the amplifier is assumed to be noiseless but instead, the noise e defined as the quantization noise, noise can be eliminated if a negative feedback loop is embedded with a delay since amplifier can only use quantizer output in the next sample, to the system and making A is sufficiently large.

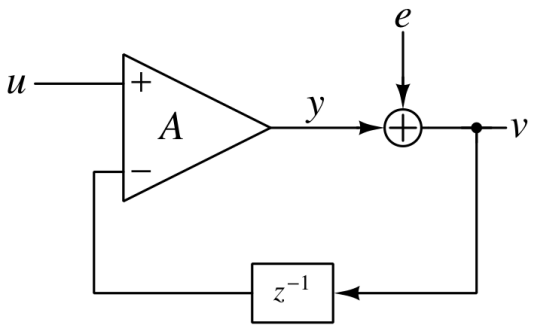


Figure 1.4 : Amplifier with quantization noise and delayed feedback [7]

From the figure above we can obtain the dollowing equation,

Tranfer function from u to v is called as STF and from e to v called as NTF. As A (k_q) approches to inifinity STF approches unity while NTF goes to 0. Hence, to make magnitude of the NTF small $|A| \gg 1$. The transfer fucntions of NTF and STF have the same denominator which is the characteristic polynomial of the system. From the denominator, the pole of the system can be found as $z = -A$. In order for this system to be stable, all of its poles must be inside of the unit circle which can be seen in the below, can onlly be achibed if $|A| < 1$. Therefore, there is confilict in the system sşbce to make NTF small, to eliminate the quantization noise, A needs to bigger; however, this will make the system unstable.

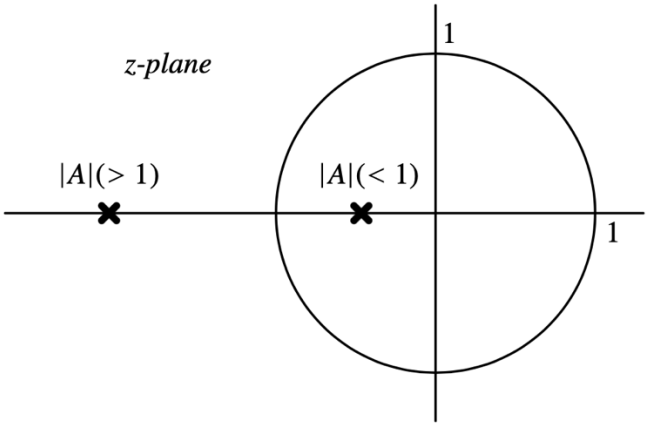


Figure 1.5 : Pole locations of the system [7]

Since the input sequence u is confined at llow frequencies due to oversampling, rather than try to surpress the quaantization noise across all frequencies, it can be elliminated only in the signal bandwindth which is $[0, \pi / OSR]$, can be acghived by making A high at only in the low frquencies. This can be achived physcially by replacing the

frequency-independent gain block A by block with a frequency -dependent gain. The lowest order system can achieve these charactheristic is an integrator which make the gain infinitw at low frequencşies so that NTF has a small magnitude at low frequencies.

The intergrator gain can be shown as $A = 1/(1 - z^1)$ in the figure below. The new system can be described as the following equation where thre STF is unity and NTF $1 - z^{-1}$, acting as a high-pass filter that completely blocks noise at DC where $(\omega = 0 \ \ or \ z = 1)$.

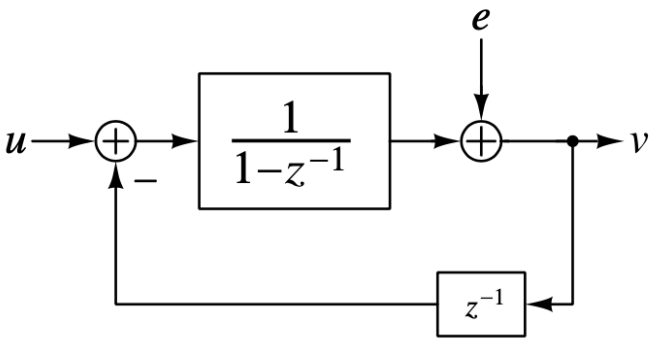


Figure 1.6 : A negative feedback system where the output quantization noise is attenuated at low frequencies[7]

Figure 2.10, shaded part shows in-band component pf the noise and extends from dc to $\omega = \pi/OSR$ which is the signal band. The NTF's magnitude response increases at 20 dB/decade, and its effect is concentrated within the signal band. This corresponds to a 20 dB/decade suppression of in-band noise, as shown in When a low-pass digital filter is applied to the modulator output followed by decimation, the in-band quantization noise is significantly reduced.

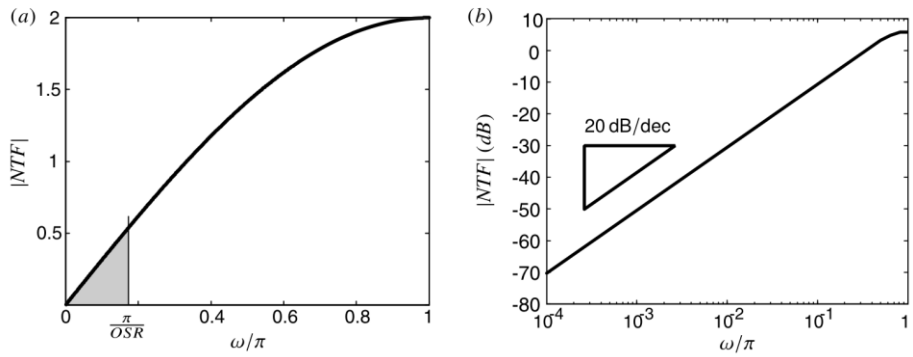


Figure 1.7 : Magnitude of the NTF of a first-order noise-shaping quantizer [7]

1.4 First Order delta-sigma Converters (MOD1)

A first-order delta-sigma (ΔΣ) modulator, commonly referred to as MOD1, consists of a feedback loop combining an integrator, a quantizer, and a subtractor which often described as the analog part of the system. Afterward, the modulator output is processed by a decimation filter, which includes a sharp digital lowpass filter followed by downsampling to yield a Nyquist-rate digital signal [7].

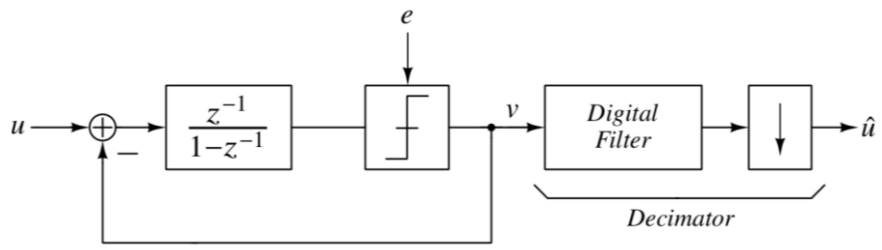


Figure 1.8 : First Order ΔΣ [7]

The MOD1 loop shapes quantization noise thorough the following operations. The integrator within the loop filter accumulates the difference between the analog input and the DAC feedback output, which, after quantization, yields a digital output sequence. This structure results in a $NTF(z) = 1 - z^{-1}$, which acts as a high-pass filter with a zero at DC. This suppresses in-band noise while allowing quantization noise to accumulate at higher frequencies. The STF (signal transfer function) remains unity, meaning the input signal passes through unaffected in the band of interest [7].

Even if the oversampling alone reduces quantization noise by spreading it over a wider frequency range, it improves SNR by only about 3 dB per OSR doubling, equivalent to a 0.5-bit resolution gain. Delta-sigma modulators, by combining oversampling with feedback, shape the noise spectrum and push most of the quantization noise out of the signal band which leads to an SNR improvement of approximately 9 dB per OSR doubling, or about 1.5 bits of additional resolution. This allows high-resolution conversion using low-precision quantizers, with digital filtering further reducing in-band distortion. The resulting performance gain stems mainly from noise shaping, not quantizer complexity. [7]

The quantizer in MOD1, is typically implemented using a combination of an ADC and DAC in a feedback configuration. The ADC digitizes the integrated difference signal,

and the DAC converts the quantized output back into an analog value for subtraction in the next cycle [7]. If a 1-bit quiaantizer is used in MOD1, due to oversampling and negative feedback, the output of quantizer is a stream of only 1s and 0s which forms a pulse-density modulated (PDM) signal, where the density of 1s over time represents the input level. A higher input results in **more 1s**, and a lower input gives **more 0s**. The feedback loop in the modulator adjusts the output so that its **average value gradually follows the input signal**, effectively encoding the signal as **pulse density** [11].

This loop is followed by a decimation filter, which removes the shaped out-of-band quantization noise and reduces the sample rate to match the desired Nyquist rate. F**i**gure 2.8 illustrates the spectrum before and after filtering, demonstrating the substantial reduction in noise within the signal band.

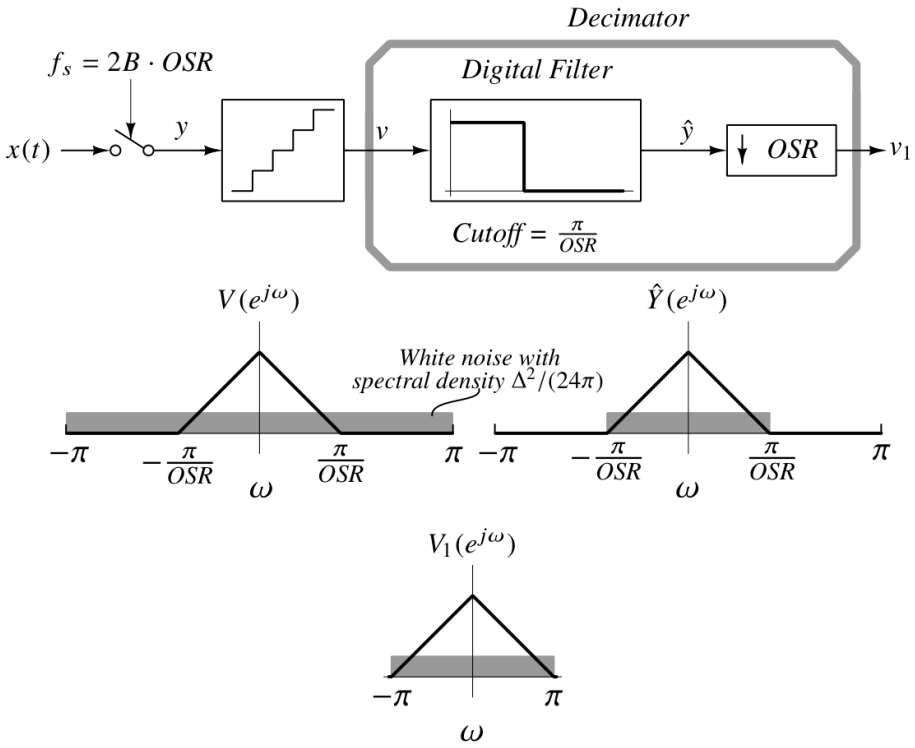


Figure 1.9 : Signal chain shows oversampling to reduce quantization noise [7]

Using a 1-bit quantizer simplifies the modulator design and avoids the non-idealities (mismatch) associated with multi-bit quantizers [7]. However, first-order ΔΣ modulators with 1-bit quantizers can suffer from idle-tones in output specturum and input-dependent noise behavior. This behevior can be prevented increasing the loop order. In second-order ΔΣ modulators, 1-bitquantizers, the quantization noise becomes

more spectrally shaped and less correlated with the input, making the white noise model a more accurate and reliable approximation. [11].

1.5 Performance Metrics of ΔΣ Converters

As it mentioned above that the performance of the ΔΣ ADC's is measured with the dynamic performance metircs which can be obtained from frequency-domain representation of the output sequence which requires fast Fourier transform (FFT) of the output sequence with a specifying windowing function. In figure xx, shows the output sequence of a sinusoidal inpiut signal with frequency f_{in} applied. The output signal has a peak at the input frequency. However, the output specturum deos not have the characteristic of a purely shaped quantization noise because of the circuit non-idealties which will be explained later. Linear errors raise the noise floor and decreases the effect of noise shaping, while nonlinear errors introduce distortion, especially at high input amplitudes, while the small input signal amplitudes will be masked under the noise floor [11].

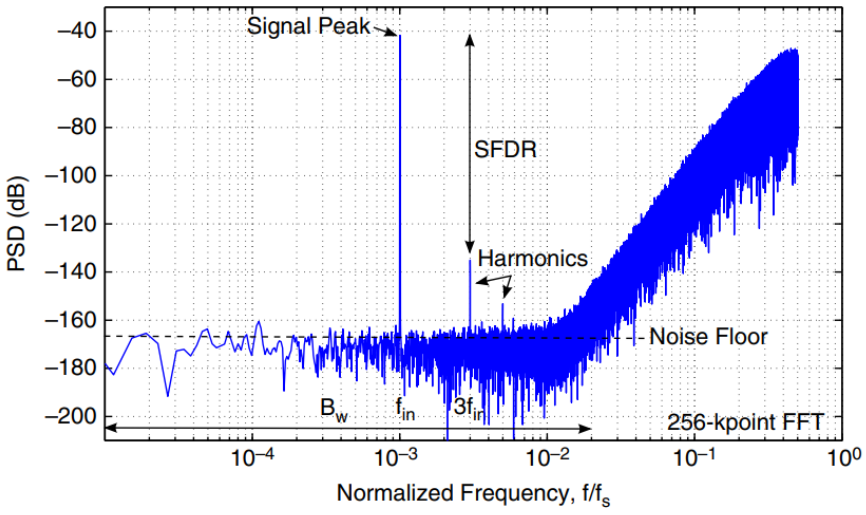


Figure 1.10 : Illustration of a typical experimental output spectrum of a ΣΔ modulator and its main characteristics. An LP ΣΔM is assumed [11].

The spurious-free dynamic range (SFDR), defined as the ratio of signal power to the strongest unwanted spectral component, which can be directly extracted from the modulator's output spectrum seen in the figure below [12].

SNR defines as the ratio of the output power of the sinusoidal input frequency to the in-band noise power that is uncorrelated with the input. It considers linear error sources, including noise (linear errors), and quantization noise while excluding

harmonic distortion and other nonlinear effects. Therefore, it evaluates the modulator's linear performance only. The formula of SNR is given in the equation below [11],

$$SNR\ (dB) = \ 10\ \times\ log10\left(\frac{P_{sig,out}}{P_{IBN}}\right)$$

SNDR extends the SNR definition by including nonlinear distortion components, such as harmonics generated by the modulator's non-idealities, in addition to linear noise and quantization noise. It thus provides a more complete picture of the system's dynamic performance. In practice, the SNDR curve starts deviating from the SNR curve only at larger input amplitudes, where nonlinear distortion becomes significant. In order to capture distortion properly SNDR is typically measured using input frequencies less than or equal to one-third of the signal bandwidth ($f_{in} \leq\ Bw/3$), ensuring that the second and third harmonics fall within the band and are included in the calculation [11].

Dynamic range (DR) can be defined as the ratio of the output power at the frequency of an input sinusoid with maximum amplitude to the output power for a small input amplitude for which $SNR\ =\ 0\ dB$; i.e., so it cannot be distinguished from the error. For an ideal case, it is given by [11]'

$$DR\ (dB) = \ 10\ \times\ log10\left(\frac{(Y_{FS}/2)^2}{2IBN}\right)$$

ENOB translates the modulator's performance into an equivalent resolution in bits, allowing comparison with ideal Nyquist-rate ADCs. It tells how many bits a Nyquist-rate ADC would need to match the same dynamic range or SNDR of the ΣΔ ADC. It can be calculated with the formula given below [11]

$$ENOB\ (bit)\ = \frac{(DR\ -\ 1.76)}{6.02}$$

The overload level specifies the maximum input amplitude for which the ΣΔ modulator maintains linear and predictable operation. As the input approaches half the full-scale range of the quantizer ($X_{FS}\ /\ 2$), the quantizer may saturate, leading to a sharp rise in in-band noise and a rapid drop in SNR. The OL is often defined as the input amplitude at which the SNR degrades by 6 dB from its peak value [11].

1.6 Second Order ΔΣ ADC

In MOD, one way to improve in-band SQNR is by using a quantizer with more levels This reduces the quantization step size (Δ) which lowers the noise power across the frequency range and improves overall resolution (ENOB). However, instead of reducing noise across all frequencies, an alternative approach is to focus on reducing noise specifically within the signal band to reduce the in-band quantization noise spectral density. The simplest way to do this is by replacing the quantizer part in MOD1 with another instance of MOD1 which adds an extra stage of noise shaping. The resulting of this structure is defined as second-order ΔΣ modulator, often referred to as MOD2, which can e seen in the figure below [7]

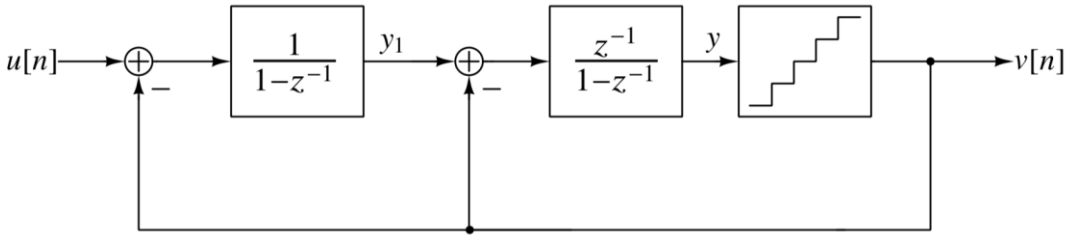


Figure 1.11 : Second Order ΔΣ [7]

In this structure, the original quantizer is replaced by a first-order modulator, which shapes the quantization noise before it enters the main loop. Mathematically, the quantization noise now passes through the first-order high-pass filter twice, resulting in a second-order noise-shaped output' which can be seen in the equation below, as well as a significantly higher signal-to-quantization-noise ratio (SQNR) for the same oversampling ratio (OSR) [7].

$$V(z) = U(z) + (1 - z^{-1})^2 E(z)$$

This indicates a unity-gain signal transfer function (STF) and a noise transfer function (NTF) of,

$$STF(z) = 1$$

$$NTF(z) = (1 - z^{-1})^2$$

This second-order NTF corresponds to a 40 dB/decade attenuation of in-band noise, significantly improving the noise-shaping characteristics compared to MOD1 which

implies a steep roll-off in the NTF magnitude on both linear and logarithmic scales. The theoretical in-band quantization noise power is given by [7]

$$In\text{-}band\ Noise = \frac{\pi^2}{2^{2L} \cdot 5} \cdot \frac{\Delta^2}{12 \cdot OSR^5}$$

At low frequencies, the NTF behaves like ω^2, meaning the in-band noise is much lower compared to MOD1 which results in the in-band noise (IBN) being proportional to OSR^{-5}. So, every time the OSR is doubled, SQNR improves by **15 dB**, which is roughly equivalent to a **2.5-bit increase in resolution (ENOB)**. This is a significant improvement over MOD1, which only offers 1.5 bits per doubling, and even more compared to systems without noise shaping (0.5 bits per doubling. This can be seen in the **Figure 2.14** which compares theoretical SQNR versus OSR curves for MOD1 and MOD2. For instanace, with an OSR of 128, MOD2 can achieve a resolution close to 16 bits, while MOD1 would require an OSR of 1800 to reach the same performance, which would demand an impractically high clock rate.

The out-of-band gain increases along with a decrease in in-band gain of the NTF of MOD2 relative to MOD1. While in MOD2 the gain at $\omega = \pi$ is 4, in MOD1 it is 2. This resuls a more variation in output from sample to sample, especially at higher frequencies. That said, after digital filtering, MOD2 provides a cleaner signal. Although MOD2 reduces the quantization noise within the signal band compared to MOD1, the overall noise across the entire frequency range [0, π] is actually higher. This total quantization noise is calculated by integrating the noise power over the full bandwidth which result in $\Delta^2/2$ [7].

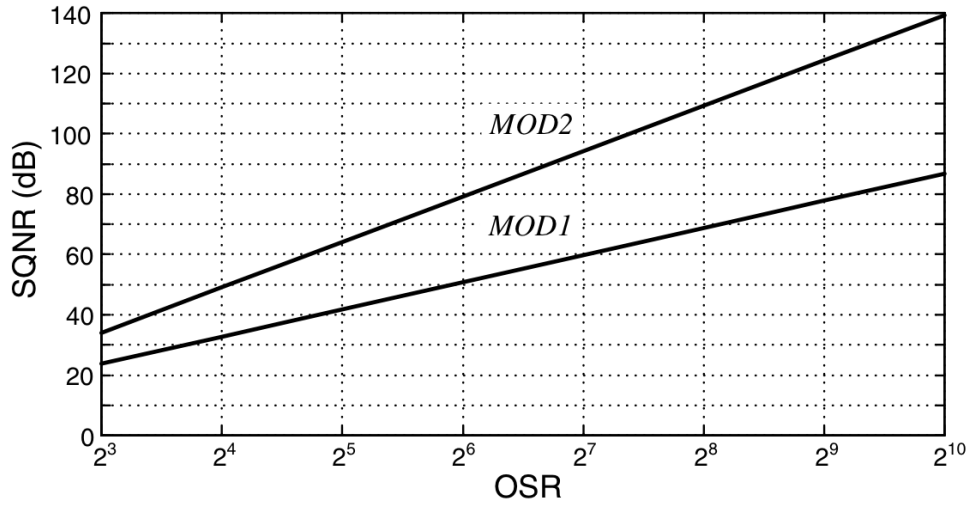


Figure 1.12 : The theoretical SQNR versus OSR curves for MOD1 and MOD2.

MOD2 maintains the same linearity benefits as MOD1. Because non-ideal effects like offset or hysteresis are injected at the quantizer together with the quantization noise, they are also shaped by the NTF and pushed out of the band. Additionally, any variation in loop filter coefficients does not strongly affect performance, since these appear as small shifts in the NTF/STF poles rather than introducing non-linear distortion.

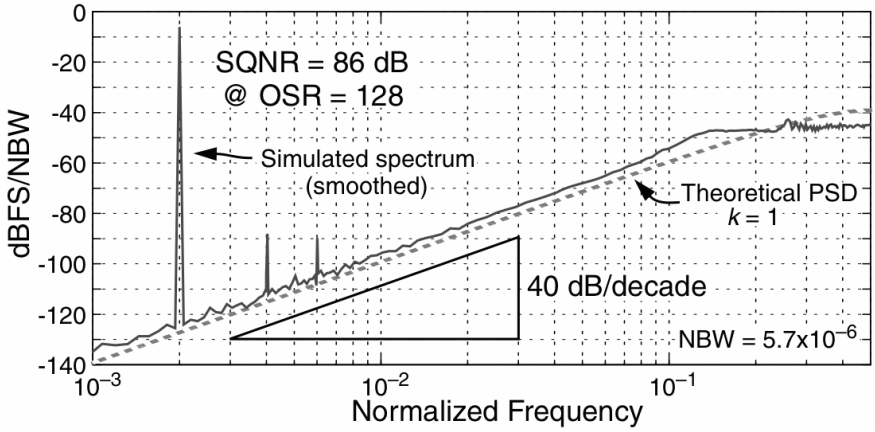


Figure 1.13 : Output specturum of MOD 2 with a – 6dBFs

1.7 Simulations in MOD2

T The output spectrum of MOD2 demonstrates clear second-order noise shaping, confirmed by the 40 dB/decade slope in Figure 3.8. At an oversampling ratio (OSR) of 128, the simulated signal-to-quantization-noise ratio (SQNR) is 86 dB. When extrapolated to full-scale input, the estimated peak SQNR is about 92 dB, which aligns closely with the theoretical prediction of 94 dB. However, two deviations from the ideal white-noise model are observed: first, the presence of second and third harmonics in the spectrum (around –88 dBFS and –90 dBFS) suggests that nonlinear behavior is present, as pure white quantization noise cannot produce harmonics; and second, the measured power spectral density (PSD) shows a slightly different shape from the theoretical NTF curve, with lower values at low frequencies and higher values at high frequencies. This discrepancy is due to the effective quantizer gain being less than one.

To account for the shift in NTF shape, the quantizer gain was estimated from simulation as approximately k=0.63. Updating the NTF model using this value yields a modified response that closely matches the observed PSD. When the input amplitude is reduced further, the optimal quantizer gain increases slightly—up to $k \approx 0.75$ for

small input signals (below –12 dBFS). Incorporating this into the NTF gives a revised expression that includes a higher in-band gain, which explains why the simulated noise power is about 2.5 dB higher than initially predicted.

Figure 3.10 shows how SQNR varies with input amplitude. At mid-range amplitudes, the simulated SQNR closely follows the theoretical prediction. However, for very small inputs, the observed SQNR is slightly lower than expected, meaning more input power is required to reach 0 dB SQNR. For large input signals, the modulator begins to saturate, and the SQNR peaks around –5 dBFS before dropping sharply as full-scale is approached. This drop is more pronounced for low-frequency inputs, which stress the modulator by applying large values over longer periods. Compared to MOD1, MOD2 handles large signals more gracefully, though it still exhibits some degradation under full-scale conditions. The time-plot of the output sequence does not say much about the system behavior. In the case of a two-level (1-bit) quantizer with a half-scale sine-wave input, the output becomes binary which shows a strong tendency to be +1 when the input is positive and –1 when the input is negative. The output shouldbe examined in the frequency domain to get more clearer understanding of the system behaviour.

A linear model predicts this behavior accurately for low input levels. However, discrepancies arise at higher amplitudes, where MOD2's SQNR begins to saturate and degrade for large inputs. These effects are due to the signal-dependent behavior of the quantizer gain, which can be modeled as a weak nonlinearity. By incorporating this nonlinearity into the linear model, an improved NTF is derived [7]

$$NTF(z) = \frac{(1 - z^{-1})^2}{1 - 0.5z^{-1} + 0.25z^{-2}}$$

This modification accounts for the increased in-band noise seen in simulation by modeling the quantizer's average behavior. Stability of MOD2 is more delicate than MOD1. Although theoretical bounds exist for state variables under DC inputs less than one, certain waveforms can still lead to large internal states. It is therefore prudent to restrict the input amplitude below full scale. The impact of finite integrator gain introduces a dead zone centered around zero input, with width inversely proportional

to the square of the opamp gain A. MOD2 is more tolerant to finite gain than MOD1 due to its gain-squaring loop structure [7].

$$Dead\ Zone\ Width \approx \frac{1.5}{A^2}$$

Finally, generalizations of MOD2 enable more flexible transfer function designs. By manipulating the feedforward and feedback coefficients as shown in Figure below**.** STF and NTF can be tuned to optimize performance and robustness [7].

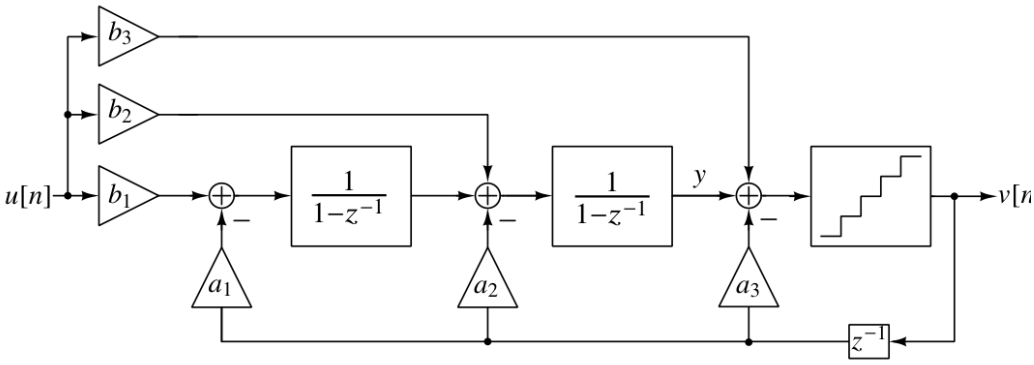


Figure 1.14 : Second Order modulator with feed-ins and feedback paths [7]

An optimal second-order NTF was derived, maximizing SQNR by adjusting the NTF zeros slightly inside the unit circle. This optimal NTF has the form.

$$A(z) = 1 - 0.5z^{-1} + 0.16z^{-2}$$

and achieves approximately 94 dB SQNR for OSR = 128, outperforming MOD2's canonical structure, as visualized in Figure below

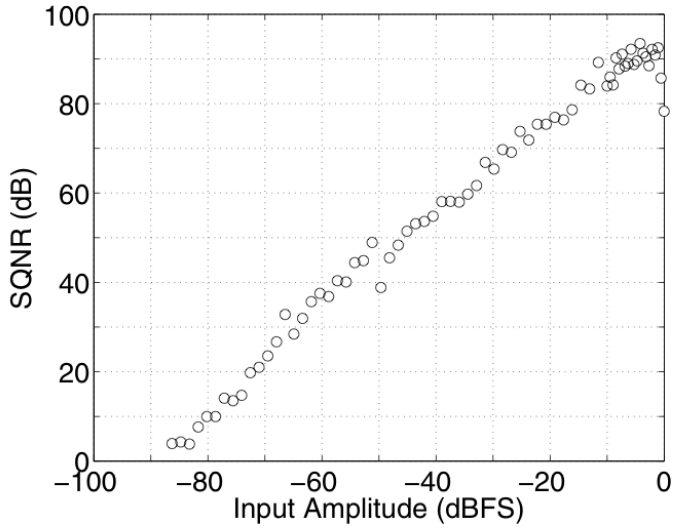


Figure 1.15 : SQNR for an optimal second-order NTF for OSR = 128 [7]

In summary, MOD2 provides substantial improvements over MOD1 in terms of SQNR scaling and immunity to non-idealities. Its increased complexity is justified by higher resolution at lower OSR values, making it a preferred architecture in high-performance ADC applications.

1.8 Non-idealities associated with ΔΣ Converters

This section does not gave an extensive defination of non idelities since it would be too detailed for the purpose of this project. Instead it provides a simple defination and reasoning and causes of the nonidealties mentioned and shows the solution to it or how to consider.

1.9 RISC-V

RISC-V is an open-source instruction set architecture (ISA) used for the development of custom processors that are aiming a variety of applications. RISC-V stands for “Reduced Instruction Set Computer” as it was designed to be simpler and more efficient compared to it’s then counterpart CISC systems called “complex instruction set computers”. As an instruction set it’s designed to enable communication of hardware systems.

RISC-V has six basic instruction formats called R, I, S, B, U, J types. R-type instructions for register-register operations, an I-type instructions for immediate and load operations, and S-type instructions for store operations. B-type instructions for conditional branch operations. U-type instructions for long immediate and J-type instructions for unconditional jumps. [17]

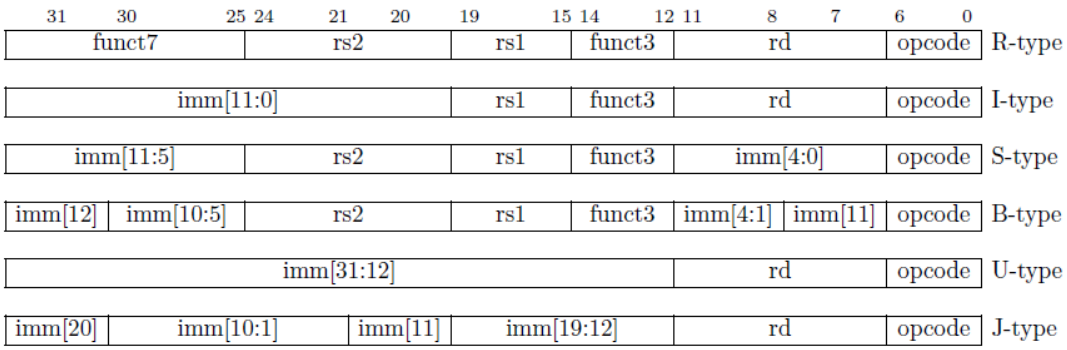


Figure 1.16 : Assembly instruction machine code format.

1.9.1 Microblaze V

The AMD MicroBlaze™ V processor is a soft-core processor based on the RISC-V instruction set architecture (ISA), designed for implementation in AMD adaptive SoCs and FPGAs. It combines the modularity of a configurable architecture with the open-source RISC-V ecosystem, enabling developers to utilize a broad range of software tools and libraries.[18]

The MicroBlaze V processor adheres to the RISC-V standard, an open ISA managed by the RISC-V Foundation. It supports both the RV32I and RV64I Base Integer Instruction Sets.

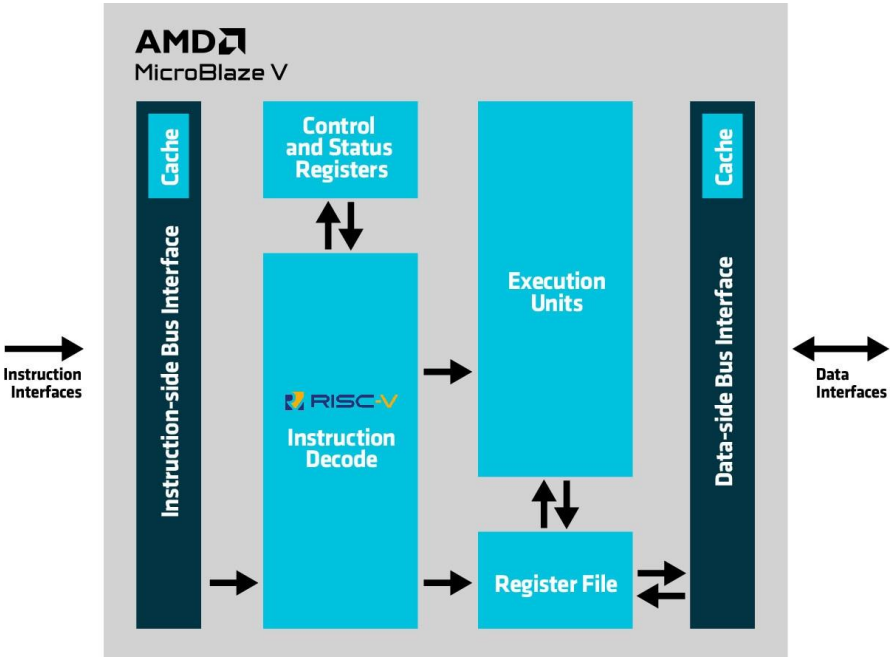


Figure 1.17 : Microblaze V Overview Illustration

The MicroBlaze V processor is well-suited for digital signal processing (DSP) applications due to its modular architecture and support for computationally intensive operations. With the inclusion of the M extension for multiplication and division, the processor can efficiently handle the arithmetic operations commonly required in DSP tasks, such as filtering, Fourier transforms, and convolution. The ability to extend the processor with custom instructions further enhances its performance in application-specific digital signal processing implementations by offloading repetitive or time-critical computations to hardware accelerators. [18]

As digital signal processing tasks involve repetitive and power intensive tasks, an instruction set focused on efficiency as RISC-V is well suited to such tasks such as data packing, unpacking and transformations.

1.9.2 Cadence Genus

Cadence Genus is the tool that turns RTL (Verilog/SystemVerilog) into a gate-level netlist and prepares it for place-and-route. It is part of the Cadence digital flow together with Innovus (implementation), Tempus (timing sign-off), and Voltus (power). Genus focuses on physically-aware synthesis, so the netlist and timing estimates already reflect placement and wiring effects, which reduces back-and-forth later.

Cadence's iSpatial integration connects Genus with Innovus using shared engines and a common database. The idea is a smoother hand-off and better PPA (power, performance, area) because synthesis can "see" implementation effects like placement and useful clock skew.

In our project, Genus is the step after FPGA/RTL validation: we import our RTL, set clocks and IO constraints, synthesize to the target standard-cell library, check timing/area, and (time permitting) pass the design to Innovus. This is the standard path for a student ASIC prototype. [18]

1.9.3 PULP (Parallel Ultra-Low-Power) Platform

PULP is an open-source RISC-V hardware platform from ETH Zürich and the University of Bologna. It provides synthesizable IP blocks (cores, interconnects, peripherals), software/runtime, and reference SoCs aimed at energy-efficient computing. PULP is silicon-proven and is used widely in research and teaching.

Our MicroBlaze-based prototype shows the software and real-time path; PULP platforms show how a similar **RISC-V** SoC can be built and synthesized end-to-end. They are good references for structure (clocking, memories, DMA, GPIO) and for checking synthesis constraints and verification style. [19]

2. SYSTEMATIC DESIGN METHODOLOGY OF ΔΣ MODULATORS

One of the most widely used methods for designing a high-performance delta-sigma modulators is the top-down/bottom-up hierarchical synthesis approach, as illustrated in Figure 5.1. In this method, the system is splitted into multiple levels of abstraction. At each level, design or sizing decisions are made to gradually translate the top-level specifications down to the lower levels. A bottom-up process is used for verification to check whether the final implementation meets the original system requirements [2, 3].

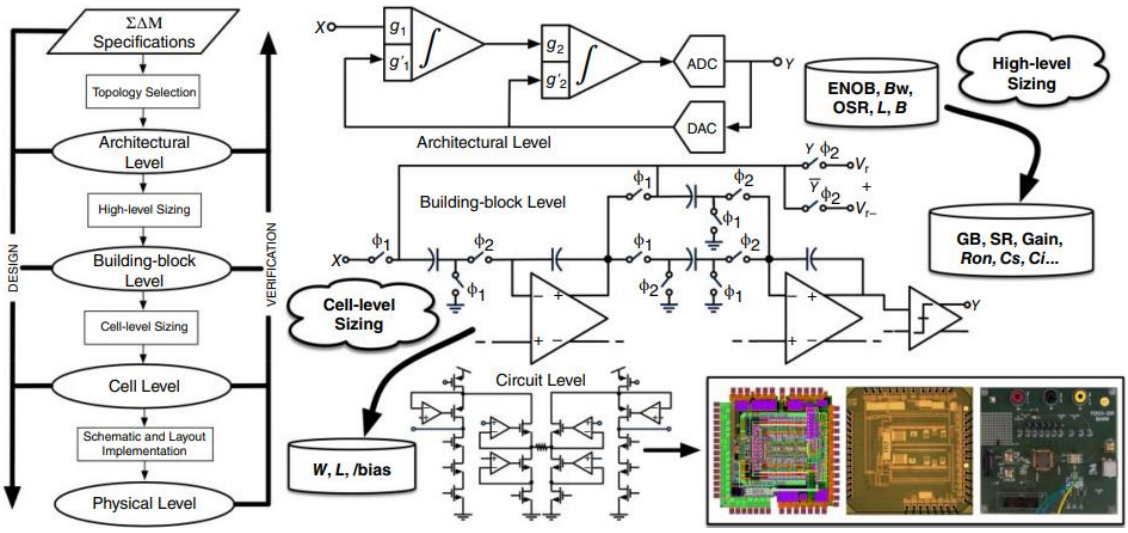


Figure 2.1 : Hierarchical synthesis methodology: (a) conceptual block diagram; (b) system partitioning commonly used in **ΣΔ**Ms [1].

One As shown in the figure 2.12 above, the delta sigma modulator can be divided into the following hirerercical design level which are breiefly given in the below.

Architecture level: choices such as whether to use a single-loop or cascade topology, single-bit or multi-bit quantization, low-pass or band-pass response, and whether the modulator will be implemented in discrete-time (DT) or continuous-time (CT).

Subcircuit or building block level: This level includes the functional circuit blocks such as amplifiers, transconductors, comparators, capacitors, resistors, switches, etc.

Cell level: This defines the specific circuit structure for each building block—for example, whether an OTA uses a folded cascode or telescopic topology, the DAC type e.g., switched-capacitor or current-steering), or the type of switches (nMOS or CMOS).

Physical level: This covers the implementation from transistor-level schematics down to layout and chip fabrication.

2.1 Architecture Level Design

In this design 2^{nd} order discrete time delta sigma modulator will be used to keep the design simple. The main goal of the design is to convert human voice to digital signal. Accorrding to ITU-T Recommendation P.341, conventional telephone voiceband is between 300Hz and 3400Hz [15]. To make it more clear 8 Khz can be used. However, in order to keep the design simple, 4 Khz Bandwidith will be used in the design. Sampling frequency of the modulator is set to 2.4 Mhz, since it is common in industry

to use 2.4 Mhz sampling frequency for audio ΔΣ ADC'sTherefore OSR can be found as,

$$\mathbf{OSR} = \frac{\mathbf{F_s}}{\mathbf{2 \times F_B}} = \frac{\mathbf{2.4 \times 10^6}}{\mathbf{2 \times 4000}} = \mathbf{300}$$

In order to 2nd order modulator perform efficiently OSR should be higher. Having 300 OSR is sufficient to use 2nd order modulator.

The next step is to select the modulator topology. There are mainly 4 types of modulators in the Schreier's ΔΣ toolbox, CIFB, CRFB, CIFF and CRFF. The Cascade-of-Integrators, feedback form (CIFB) structure will be selected since it is most commonly used topology as well as having fast feedback paths which ease loop closure at moderate clock rates. Additinaly, its STF has unity gain at DC, and the term $\mathbf{1/D(z)}$ creates a low-pass response that attenuates high-frequency components [7]. Since all loop filter coefficients except $\boldsymbol{c_2}$ are are capacitor ratios, $\boldsymbol{c_2}$ has no effect since we are using a 1-bit quantizer . Therefore, it also simplifies the circuit design.

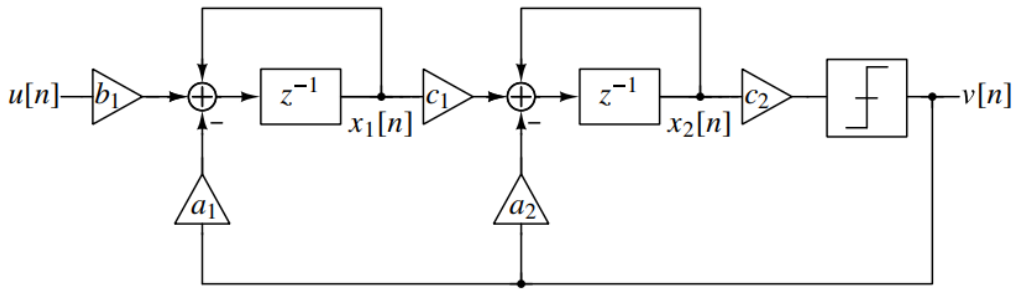


Figure 2.2 : Second order CIFB topology [7].

Since the topology is selelcted, the coefficient matrix can be found using Schreir's ΔΣ toolbox on MATLAB. The copiable matlab code for the process is given in the appendix. For the madulator, full-scale range of input (umax) selected as $\mathbf{0.8 \times V_{dd} = 2V}$ or $\mathbf{\pm 1V}$. Reference voltage $\boldsymbol{V_{ref} = 1\,V}$. The op-amp's output swing is selected as or $\mathbf{\pm 0.6V}$ to obtain higher gain. Since our main target is to obtain 12-13 bit (or higher, ideally 16-bit) design, and we can also say every 6dB SNR increase correspons to one bit, then we can set our targeted SNR as, $\boldsymbol{SNR_{target} = 13 \times 6dB = 78\,dB}$. However, to leave a good margin and also try to obtain bits closer to 16 bits, and other

reasons which will be explained in the MATLAB simulations, we set $\boldsymbol{SNR_{target} = 95\,dB}$. The corresponding MATLAB code for thispart can be seen in the figure below.

```
%--------------------------------------------------
% Design Parameters
order = 2; % Filter order
OSR = 300; % OSR
fB = 4e3;
N = 100e3; % Number of Points
opt = 0; % Optimization (=1 for odd order)
H_inf = 1.5; % Maximum out of band gain of NTF, should be less than 2
f0 = 0; % Lowpass design
form = 'CIFB';
nlev = 2; % Quantization Levels
Level = 1;
Fs =2*OSR*fB; %Sampling frequency, clock
Fs_Mhz = Fs*10^-6
fs = 1; % Normalized Sampling Frequency

Vdd = 2.5; %Our supply voltage
FullScale = Vdd; %Full scale input

umax = 0.8; % Normalized max amplitude 0.8*Vdd normally
In_FS = umax*FullScale; %Normally +-1 whicih is 2. umax*vdd
In_FS_RMS = In_FS/sqrt(2);

Vref = 0.5; %REference voltage
swing = 0.6; %Output swing

SNR_target = 95; % 100dB SNR target for full-scale input
```

Figure 2.3 : MATLAB code – Pt.1

The summary of design parameters for the ΔΣ ADC is given in the table below

Table 2.1 : The summary of design parameters.

Parameter	Value
Bandwith (f_B)	4 KHz
Sampling Frequuency (f_S)	2.4 MHz
SNR	72-96 dB
V_{dd}	2.5 V

The actual synthesis for the noise transfer function can be done by uing SynthesizeNTF function in ΔΣ toolbox which uses order (2nd), OSR, H infinite and opt value. After this value the coefficient matrix for selected mdodulator topology can be realized using realize NTF function. After this step, the values for coefficient after the first one $\boldsymbol{b_1}$ will be zerod out, since it is not gonna be used in out 2nd order CIFB modulator which can be seen in the figure. Since its only 2nd order we only have two integrators, other b coefficients are not needed. The ABCD matrix is obtained using stuffABCD function. Since the coefficients needs to scale according to output swing and max input. ScaleABCD is used and NTF and STF calculated, assuming quantizer gain is 1 and a, g, b, c coefficients are recalculated. Lastly, obtained coefficients are multipled with swing to obtain the desired output swing. This multiplication does not affect the required gain or gm/Id. However, it changes the capacitor values. Regarding this part is necessary or not, can be seen in the further simulations later since there are two examples in Schreier's book [7] and his notes [15], one of them designs without multiplying and the other designs multiplying with output swing. The MATLAB codes for this part can be seen in the figure below.

```
%-----------------------------------------------------
% Synthesis and Dynamic Range Scaling
% Noise Transfer Function
ntf = synthesizeNTF(order,OSR,opt,H_inf);
% CIFB model, coefficients realization
[a,g,b,c] = realizeNTF(ntf,form)
b(2:3)=0; % This step simplifies how the input is applied to the DSM
% ABCD matrix calculation
ABCD = stuffABCD(a,g,b,c,form);
% Scale the state variable (xlim = swing)
ABCDs = scaleABCD(ABCD,nlev,[],swing,[],umax);
% Noise and signal transfer function
[ntf2,stf] = calculateTF(ABCD,1);
% Generate CIFB Coefficients
[a_s,g_s,b_s,c_s] = mapABCD(ABCDs,form)
%
a_s = a_s * swing;
g_s = g_s * swing;
b_s = b_s * swing;
c_s = c_s * swing;
%-----------------------------------------------------
```

Figure 2.4 : MATLAB code – Pt.2

After creating NTF and STF, the following simulations can be performed to check SNR and PSD of the modulator. The simulations are performed with fullscale input at 720Hz frequency at -3dBFS input with an $\mathbf{0.707} \times \boldsymbol{umax}$ which is equal to

$$\mathbf{20\log_{10}}\left(\frac{\boldsymbol{Amplitude}}{\boldsymbol{FullScale}}\right) = \mathbf{20\log_{10}}\left(\frac{\mathbf{0.707 \times 2}}{\mathbf{2}}\right) = \boldsymbol{3dBFS}$$

The codes for these simulations can be seen in the appendix, since they are quite long to put inside the text. The time domain response for the modulator and the output of the first and second modulator can be seen in the figures/

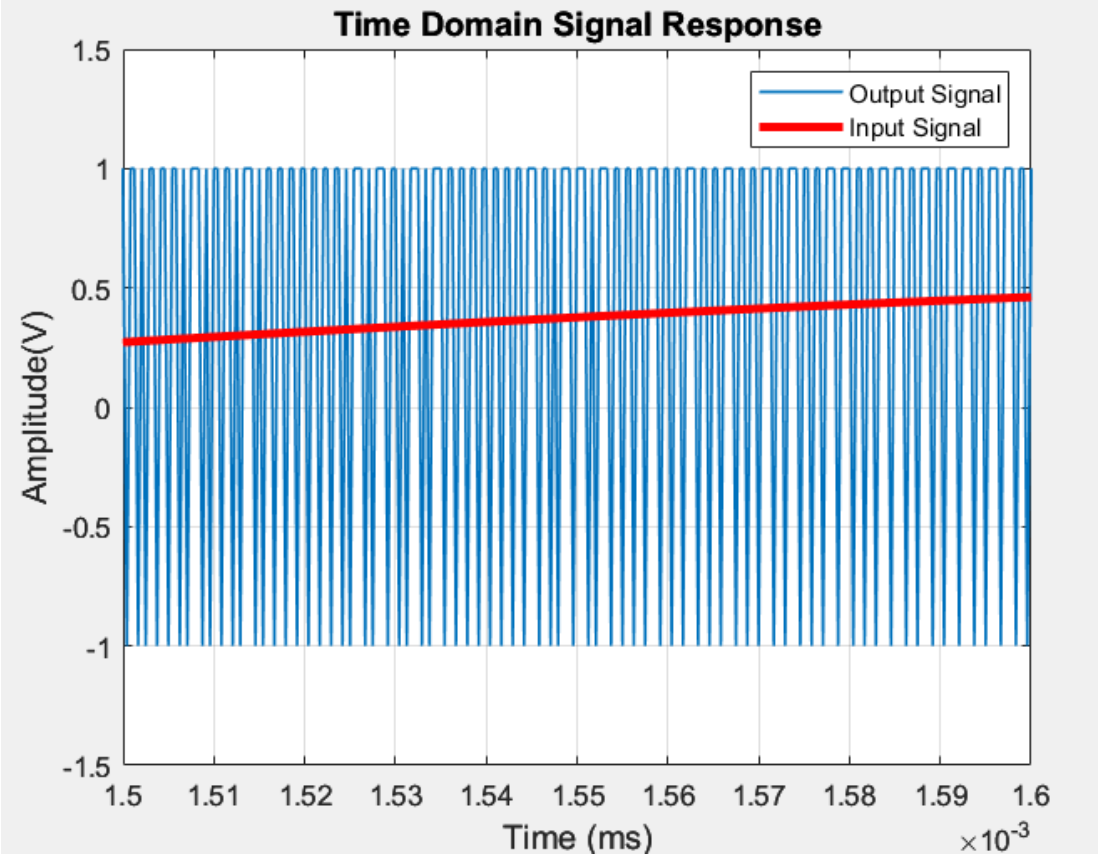


Figure 2.5 : Time domain response of the modulator

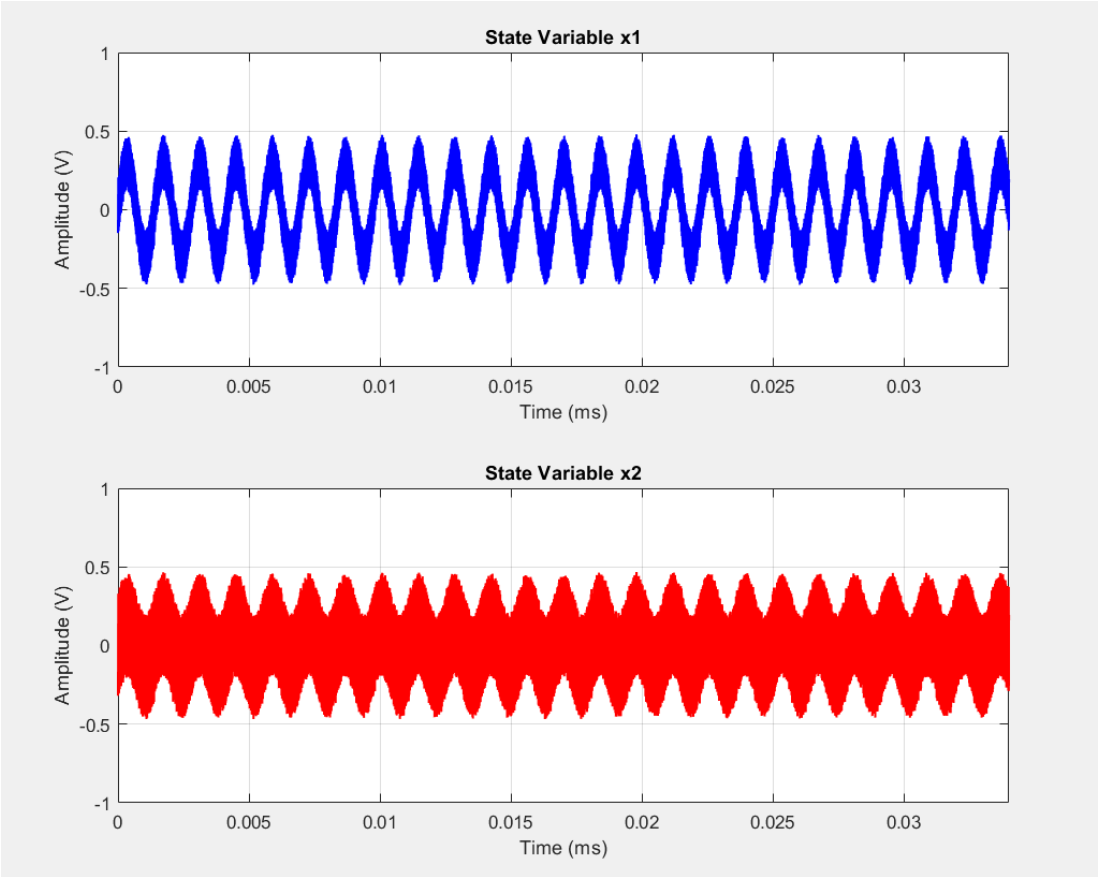


Figure 2.6 : Simulated State Swings.

As it can be seen in the Figure 2.17, the output swings of the both amplifier does not exceed $\pm \boldsymbol{0.6V}$ as expected. There are two other simulations are need to performed to obtain Specturm plot to get SQNR and 3rd harmonic value, as wel ass to maximum SNR of the modulator. According to Figure 2.21, the SQNR = 98dB whereas 3rd

harmonic is art -111.5 dBFS which is well dominated by the SQNR. Thus, we can continue with this modulator.

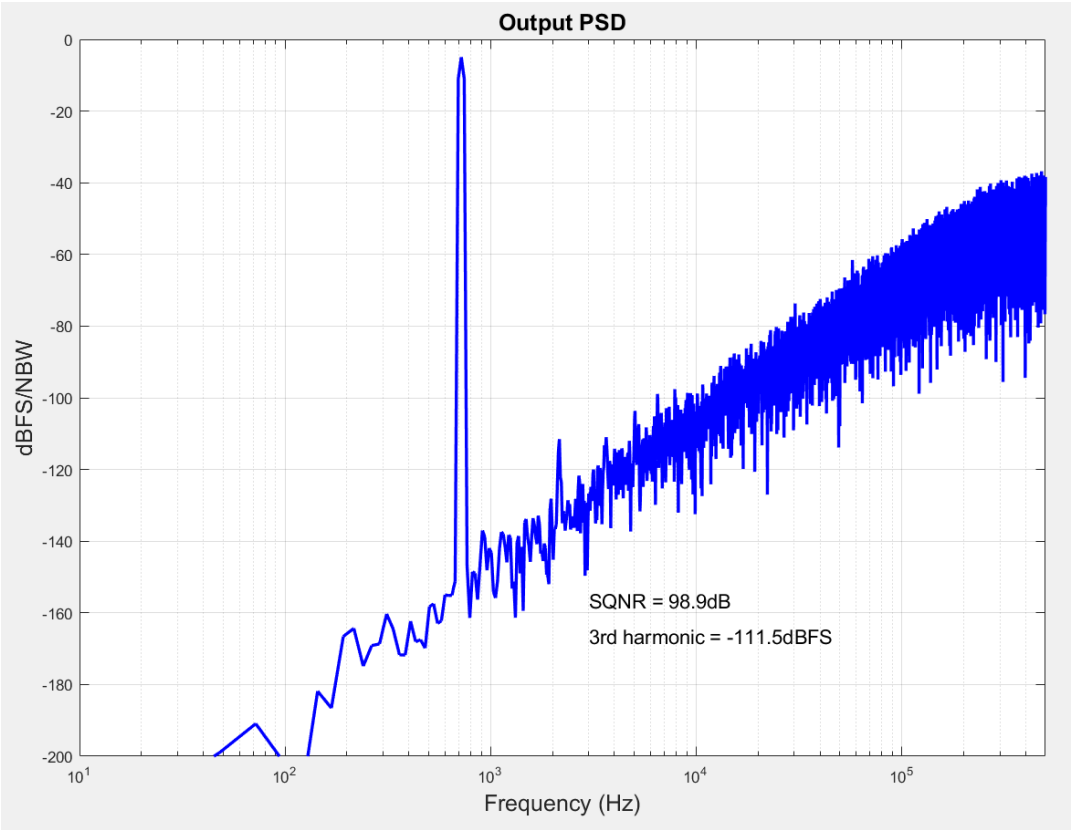


Figure 2.7 : Specturm of the Modulator

To check where is the max SNR/SQNR (around -12dBFS), we need to check SNR/SQNR vs Amplitude with different amplitude levels. At 0 dBFS modulator is at its full scale which is where the overload happens, so around -12dBFS we should see the max SQNR. In Figure 2.19 it can seen that max SQNR/SNR = 108.5 dB

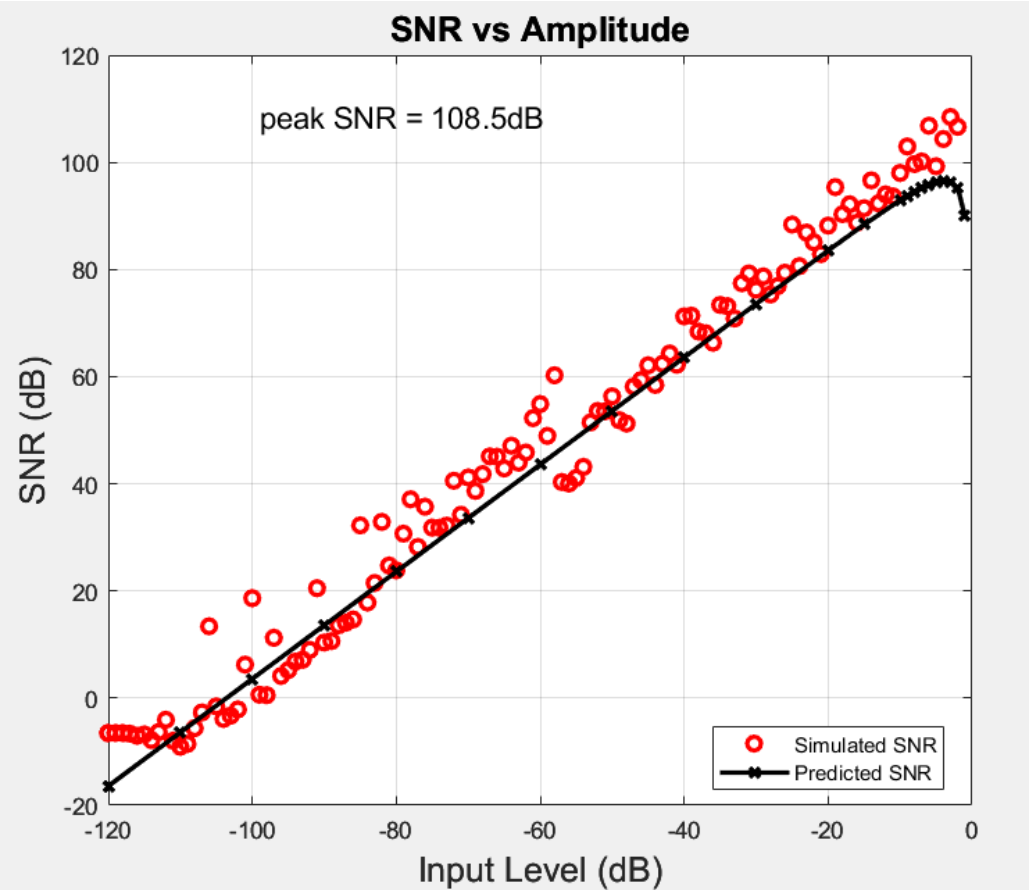


Figure 2.8 : SNR vs Amplitude

2.2 Subcircuit or Building-Block Level Design

Since it is confirmed through MATLAB simulations using ΔΣ toolbox, that the desired modulator works as expected, we can move to next step which is calculating the capacitors, amplifier gain, gm/id, switch resistor etc. Before calculating anything, we first need to select integrator topologies for the first and second integrator. Since audio

signal is generally used as a differential input/ The two integrators can be seen in the figure below.

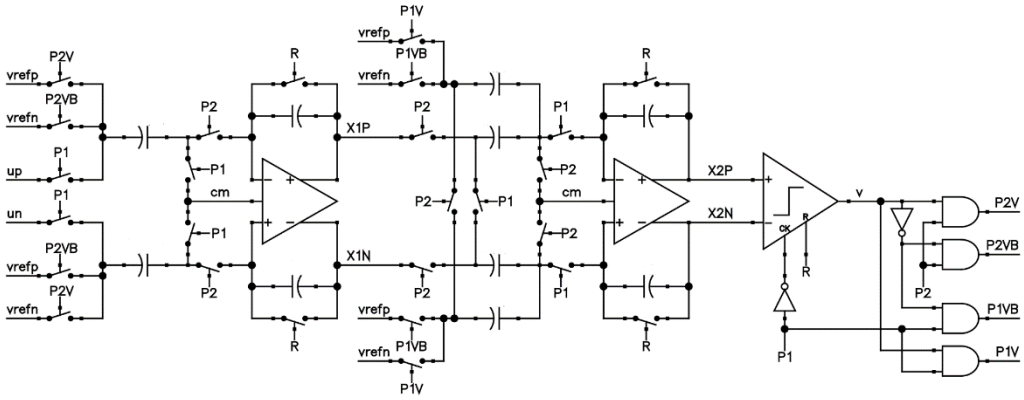


Figure 2.9 : ΔΣ Modulator Structure

```
a_s =

    0.0884    0.1916

g_s =

     0

b_s =

    0.0884         0         0

c_s =

    0.3630    1.4558
```

Figure 2.10 : Coefficients from MATLAB

Now using this structure and the results for the a, b, g, c coefficients we can calculate the capacitor values which are also calculated in MATLAB which is give in the figure below. We will use 4KT /C noise for the input capacitor, using full scale input as the voltage since it is in between these two nodes, C1f cawn be found using coefficients. So for the C2dac we weill set it to the minimum capacitance available in the pdk which was 20fF. Then the rest of the capacitances can be calculated using the coefficients.

```
% Compute Capacitor Values
k = 1.38e-23; % Boltzman's constant (J/K)
T = 300; % Temperature in Kelvin (K)
atten = 10^(SNR_target/10);
v_n2 = (((In_FS_RMS/2)^2)/2)/atten; %Full Scale input
Cin = 4*k*T/(OSR*v_n2)
C1f = Cin/a_s(1)*Vref/nlev

C2dac =20e-15 % C4
C2f = C2dac * (Vref+(In_FS/2)) / a_s(2) % C5
C2in = C2f*c_s(1)  % C3
fprintf('\n\n');
```

Figure 2.11 : MATLAB code Pt 3.

The results for the capacitance values are shown in the figure 2.26. The next ste will be calculating the gain of the amplifier. Firstly, this is calculated by using the Linear method which gives a result of 30dB. However, this will not be enough since we also need to concsider the distortion requirement of this amplifier which can be calculated using the formula given in the matlab. This ideally uses our third harmonic value while calculating it but since we are fine with achieving a bit around 12-13bit we can lax this requirement using a lower dBc value. Therefore, the result will be 80dB gain.

```
Cin =

   6.9823e-13


C1f =

   1.9739e-12


C2dac =

   2.0000e-14


C2f =

   1.5656e-13


C2in =

   5.6838e-14
```

Figure 2.12 : Capacitance values

```
%--------------------------------------------------
% Find Gain - Linear
A1 = (Cin/C1f)*(OSR/pi) %bigger than this
A1_db = 20*log10(A1)
temp =0.5*(a_s(1)*c_s(1)) + a_s(2);
lsb_power = (In_FS^2 / 2) / (10^(SNR_target / 10));
A1_nonlin = sqrt(temp/lsb_power)
A1_nonlin_db = 20*log10(A1_nonlin) %bigger than this
fprintf('\n\n');
%--------------------------------------------------
%gm calculation for settling requirement if settling occurs T/4
T = 1/Fs;
atten = 10^(SNR_target/20)
beta = C1f/(Cin+C1f);
Ceff = (C2in + ((Cin*C1f)/(Cin+C1f))) %0.5 come from single enmded to diff transformation
tau = T/(4*log(atten))
gm = Ceff/(beta*tau); %bigger than this 50
gm_uAV = gm*10^6
GBW_Mhz = (1/(2*pi*Ceff/gm))*10^-6
fprintf('\n\n');
% I_slew calculation
%Some designs use q_max = C1 * Vstep if considering output cap charging.
%This is valid since we're charging Cin from Vdd.
q_max = Cin*(Vref + In_FS/4);
Islew = q_max/(T/4);
Islew_uA = Islew * 10^6
fprintf('\n\n');
%--------------------------------------------------
%Rsw calculation in simulation we used 1kohm and it was ok
Rsw = 1/(20*gm) %should be smaller than this schreier book
```

Figure 2.13 : Gain, gm/id, and Rsw calculations

```
A1_db =

   30.5729


A1_nonlin_db =

   85.1634


Ceff =

   5.7262e-13


tau =

   9.5240e-09


gm_uAV =

   81.3916


GBW_Mhz =

   22.6221


Islew_uA =

    6.7030


Rsw =

  614.3137
```

Figure 2.14 : Gain, gm/id, and Rsw calculations results

According tho the results in the Figure, we can see that our gm should be gbigger than 81, and trhe current should be igger than 6.7uA. WE will select 7uA for thwe current and aiming for 12 gm/id value for the diff pair. After calculating all of this, we need to do a beheviroal simulation before moving to designing the components. For this, CPPSim can be used. The schematic for the simulation is given in the figure below

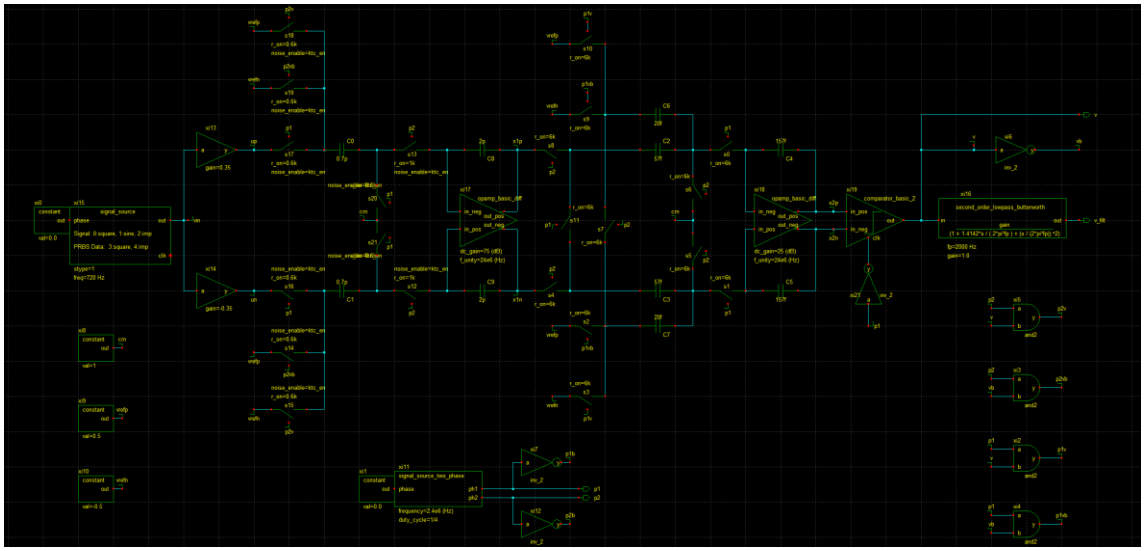


Figure 2.15 : CppSim schematic

The outrputs of the integrator 1 and 2 from the cppsim, simulations can be seen in the figure below. We can see that our outputs are not exceeding 0.6V which is good.

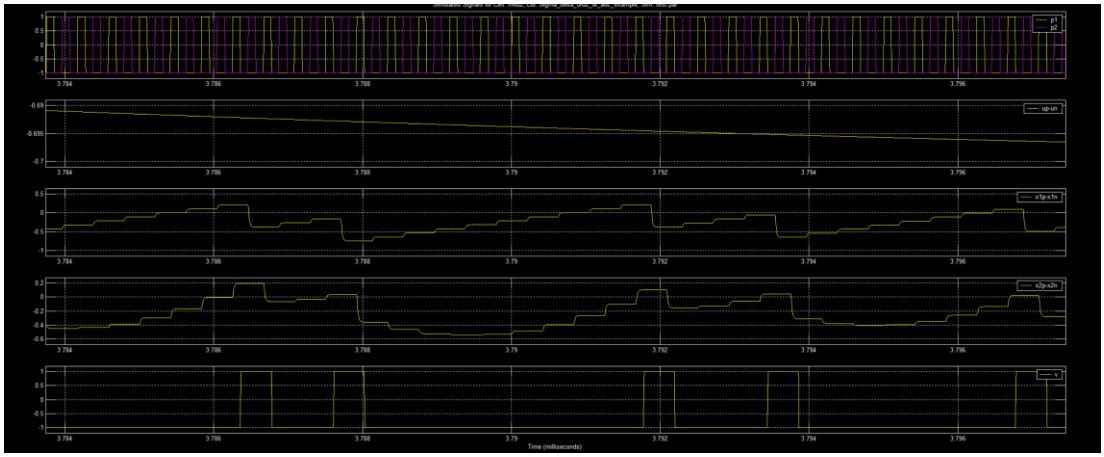


Figure 2.16 : CppSim Results

Now twe can also run the Cppsim dimulation and save it as a file to obtain PSD of the beheviroul design. The results can be plotted using MATLAB.

2.3 Transistor Level Design

Since, we confirmed that our behaviroal model is also working, we can move to transistor level design using Cadence Virtuoso with TSMC 65nm process. This section is starting from designing integrator 1 and 2, and continues with coparator, switch and clk generator, latch design. Each model wil be replaced opne by one with the beheviroal model to see whether the transistorized designs are working or not.

2.3.1 Integrator 1 Design

For the Op-Amp, we will use PMOS differential input Folded-Cascode Amplifier with Swithced Capacitor CMFB. This amplifier is selected because mostly used in ΔΣ design since it delivers high speed with medium gain and output swing. Also the PMOS differential input selected over NMOS since the opamp needs more gain then output swing. Since we need 0.6V and let gave it a bit margin and say 0.7V output swing we can calculate, Vout,min and Vout,max as follows.

$$V_{out,min} = \frac{2.5 - 0.7}{2} = 0.9V$$

$$V_{out,max} = 2.5 - 0.9 = 1.6V$$

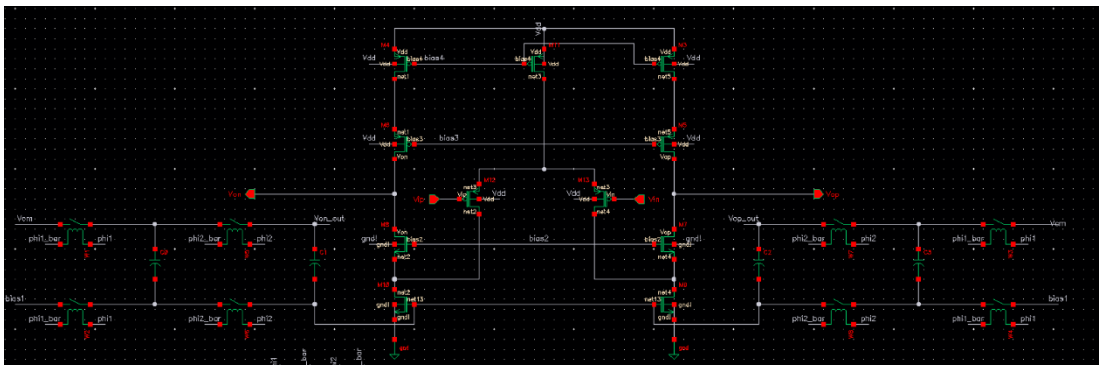


Figure 2.17 : Folded cascode amplifier with switched capacitor feedback

The output Vout,min and Vout,max of the output signal is depends on the Vov voltage of the transistors (except for the diff-pair). So, we ca assign Vov voltages around 0.45V to top pmos and bottom nmos sink and 0.3V for the nmos and pmos cascodes. To calculate transistor sizes, gm/id method is used. The nmos and pmos transistors are kept constant Vds value (around +0.1 of their Vov) and then the sizes are selected accordingly. Here is an example gm/id graph for a transistor and how its sizes are selected.

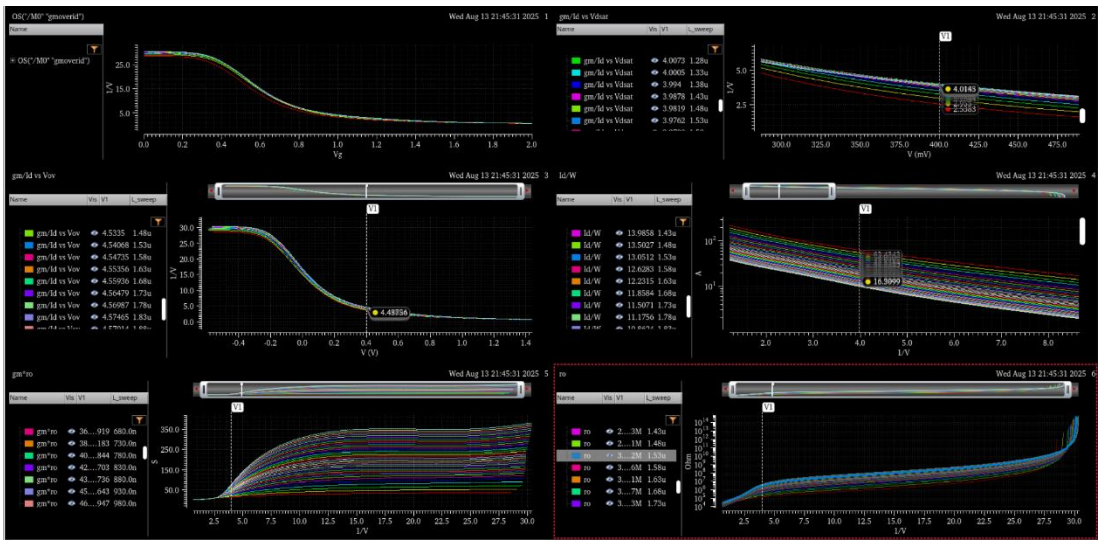


Figure 2.18 : Gm/id, id/W, Vov and gm*ro graphs for an NMOS transistor

Gain formula of ther folded cascode amplifier is can be approximately $\boldsymbol{Aol = gm1\{[gm3ro3(ri1||ro5)]||[gm7ro7ro9]\}}$. So we can mke the length of cascode amplifiers same and obtain the same gain (gm3ro3 and gm7ro7). Ans the resit is sized accordingly to miin max calculations. The sizes of the transistors can be given in the below.

Table 2.2 : Transistor Sizes

Name	Transistors	W	L
M1	PMOS_diff	13210	1330
M0	PMOS_tail	3785	830
M9	PMOS_top	1140	1230
M7	PMOS_bottom	4610	1880
M3	NMOS_top	1400	1880
M5	NMOS_bottom	900	1530

And the bias circuit is given in the below. The sizes are copied only the last transistors in cascode brnches are sized around 2x and 7.5x of their original Length value used

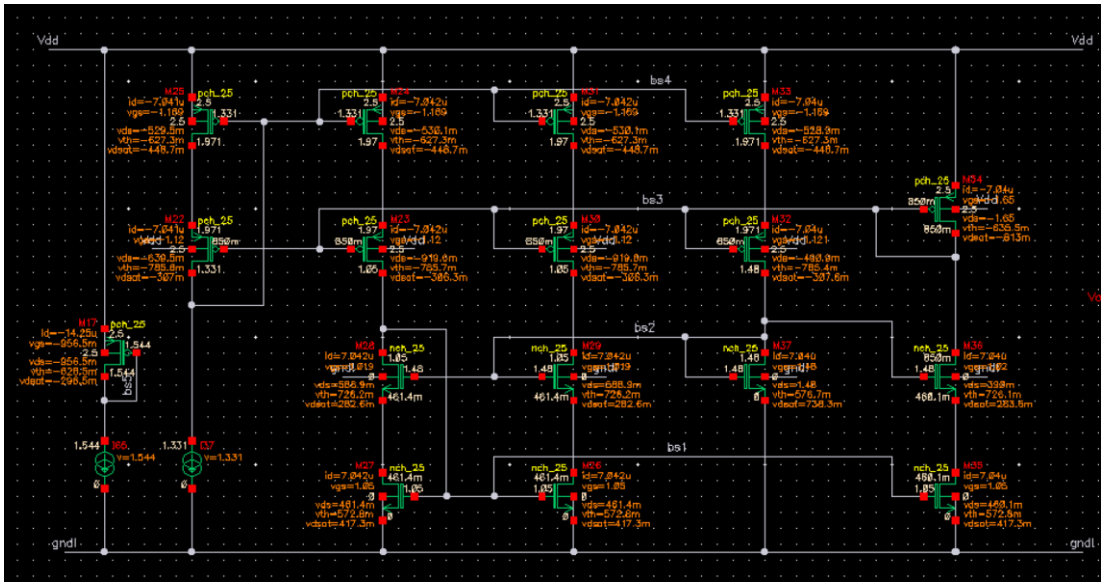


Figure 2.19 : Bias circuit

So to do the AC simulations, we can use PSS and PAC and for loop gain PSTB as well. However, in the4 book to do AC analysis they used ideal CMFB so I created the same setup the obtain gain of the amplifier. And also this setup includes the resistor with Vocm value because this setup is also used to obtain gain vs Vo sweep. That part is not added to the AC analysis when we try to find the gain. The DC results can also be seen that amplifier is pulled to Vocm which is 1.25 and the all transistors are in saturation. As it can be seen fromn the figure we can increase the Width of cascodes to obtain more gain with same current. However, this is one of my very tries to design

this amplifier. And I have other better versions of it that I didn't test completely. So this can be improved a lot.

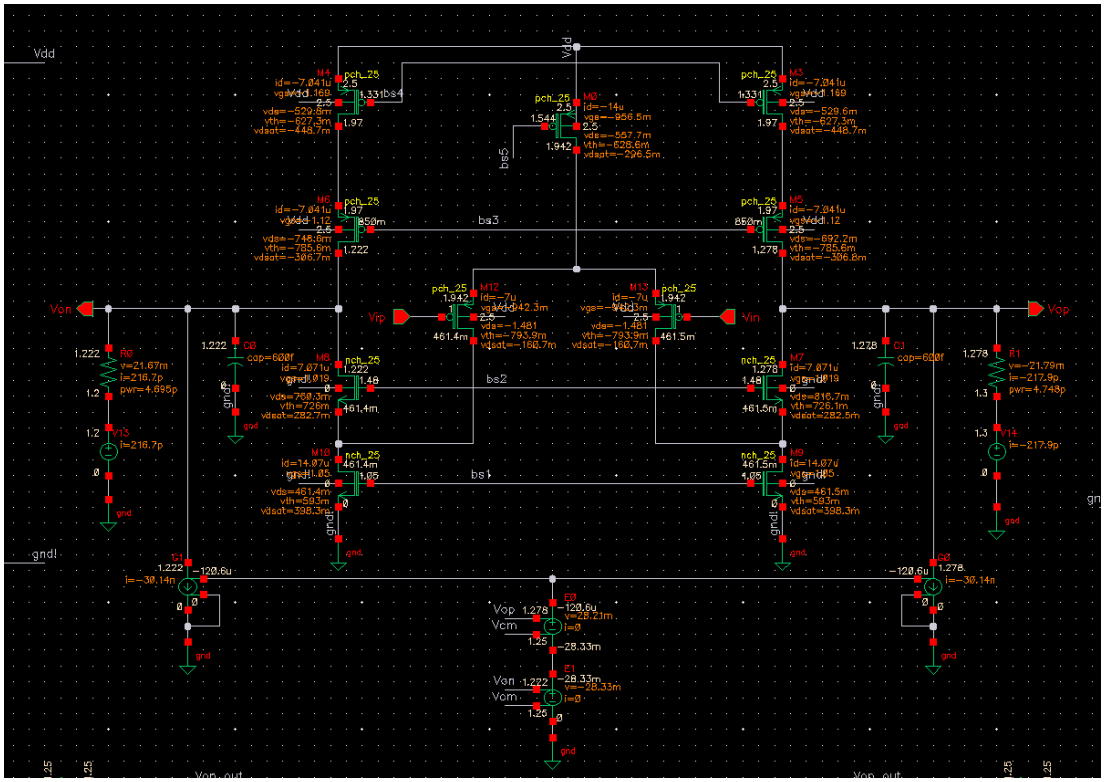


Figure 2.20 : DC operating Points

Now the last thing to look at our gain with ideal feedback which is 80dB and the UGB is 20 MHz whereas the Phase margin is 86 degree. This can be improved by decreasing the length of diffpair to increase UGB and to decrease gain, width of cascode devices

can be increased. There is already a version like that which was not measured with feedback due to time constraints.

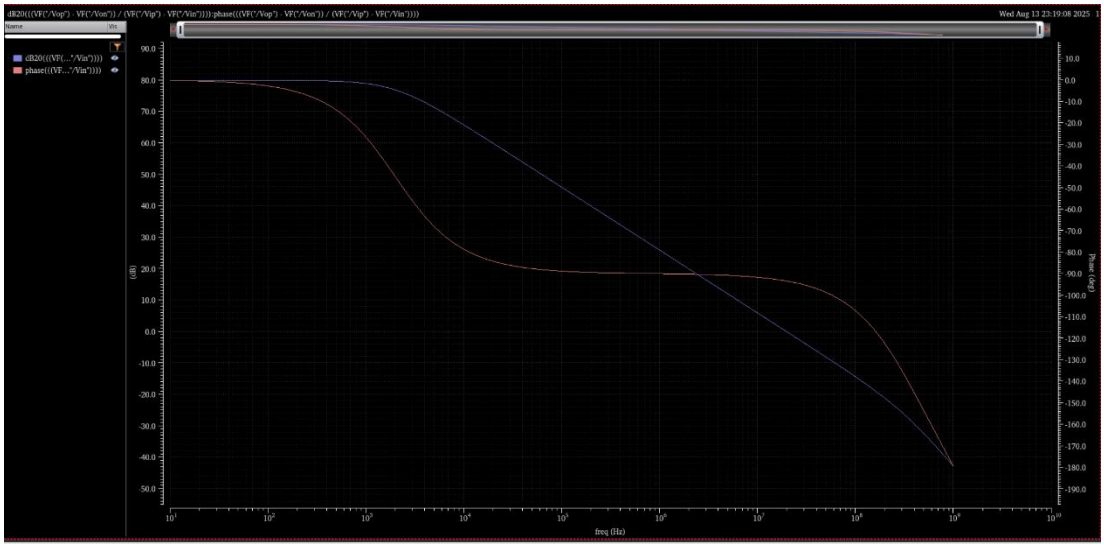


Figure 2.21 : Gain and Phase of the Folded cascode amplifier

With the previous setup, we can measure the gain vs Vo voltage. Here is the result the gian is fine until 0.7V the drop is slow, the gain is also not 80dB. So this can be improved.

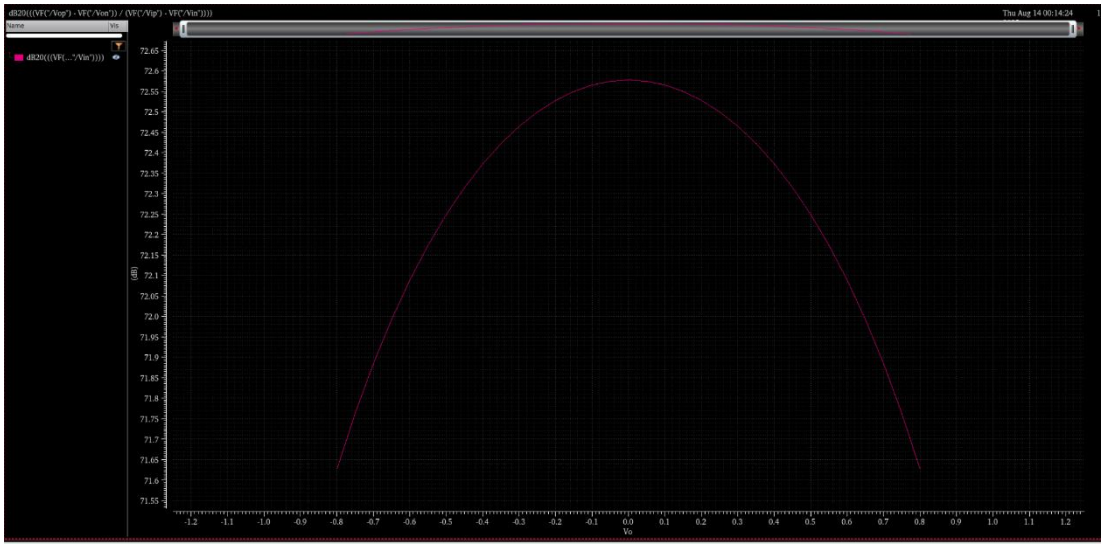


Figure 2.22 : Gain vs Amplitude

The next thing to do is do a whole simulation to see whether the issues we see in the previous examples are okay or not. However due to lacking of time we are not able to finish this part.

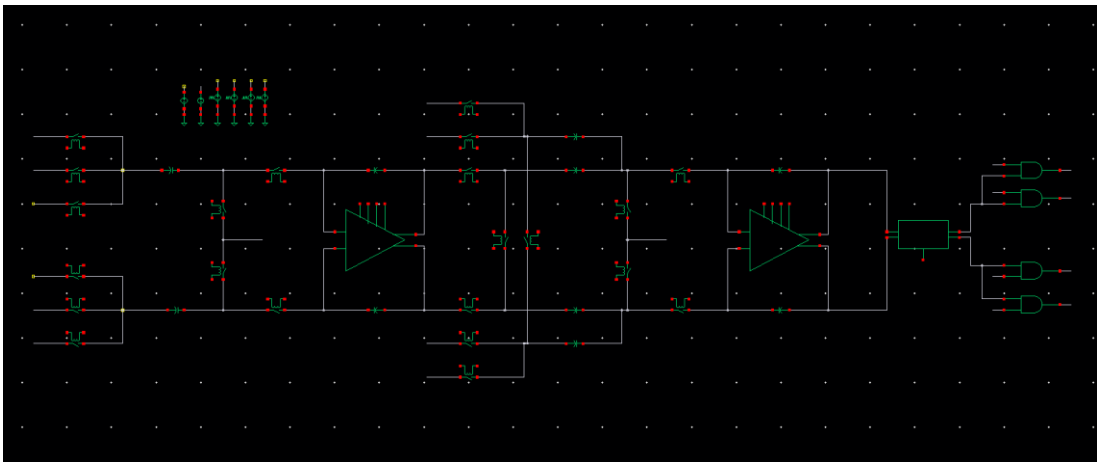


Figure 2.23 : Transistorized OTA with full behaviroal circuit.

3. Digital Subsystem Design

3.1 PDM to PCM Conversion

For initial FPGA prototype, we used the pdm microphone located on our Artix-7 Board. The MEMS microphone provides a 1-bit **PDM** stream sampled at around 2.38 MHz (derived on-chip from the 100 MHz system clock). Audio amplitude is encoded in the density of ones. We convert this stream to multi-bit **PCM** by low-pass filtering and decimating.

The VHDL module pdm2pcm implements a parameterized boxcar stage that counts ones over a fixed window and emits the accumulator as the PCM word. Two accumulators (ones1/ones2) are maintained on alternate phases of clk_sample to avoid phase bias from the PDM toggling.

```
41
42      process (clk)
43      begin
44          if (clk'event and clk = '1') then
45              if rst = '1' then -- reset event
46                  pcm <= (others => '0');
47                  count1 <= '0';
48                  depth1 <= (others => '0');
49                  ones1 <= (others => '0');
50                  count2 <= '0';
51                  depth2 <= (others => '0');
52                  ones2 <= (others => '0');
53              else -- normal
54                  if m_enable = '1' then
55                      if count1 = '1' then
56                          if depth1 < limit-1 then
57                              depth1 <= depth1 + 1;
58                              if m_data = '1' then
59                                  ones1 <= ones1 + 1;
60                              end if;
61                          end if;
62                      end if;
63                      if count2 = '1' then
64                          if depth2 < limit-1 then
65                              depth2 <= depth2 + 1;
66                              if m_data = '1' then
67                                  ones2 <= ones2 + 1;
68                              end if;
69                          end if;
70                      end if;
71                  end if; -- m_enable = '1'
72
```

Figure 3.1 : Core Logic of Pdm2Pcm Module

For our prototype we decided to use a 12 bit pcm, so our top module initialization reflects that. On top of that, to create a consistent PCM frame cadence and a simple ready handshake to the processor fabric, the design uses a 14-bit counter clocked at 100 MHz. This is to ensure any audio processing is done when the pcm inputs have changed and not before, to ensure the audio samples aren't overfilled.

For our inital tests, we just used a pcm2pwm module to showcase the decimation filter functional, as the speaker connection on the Artix-7 board requres pwm signals. Our audio quality was bad, which could be attributed to several factors, but our guess was the low quality of the pdm microphone on the board. Regardless we decided it was good enough to continue onward to the next step.

3.2 WebRTC Voice Activity Detection (VAD)

WebRTC VAD is a lightweight, classical (non-ML) speech detector that takes short frames of 16-bit mono PCM audio and returns **1** (voice present) or **0** (no voice). It's widely used because it's fast, small, and permissively licensed (BSD). [16] We went for WebRTC VAD as it doesn't rely in any external libraries and is fully functional with C, which is great for being compiled for RISC-V cores.

```
    WebRtcVad_set_mode(vad, 3);  // Aggressiveness: 0-3

    int16_t frame[FRAME_SIZE];
    size_t samples_read;
    int frame_num = 0;

    while ((samples_read = fread(frame, sizeof(int16_t), FRAME_SIZE, file)) == FRAME_SIZE) {
        int result = WebRtcVad_Process(vad, SAMPLE_RATE, frame, FRAME_SIZE);

        if (result == 1)
            printf("Frame %d: Speech detected\n", frame_num);
        else if (result == 0)
            printf("Frame %d: No speech\n", frame_num);
        else
            printf("Frame %d: Error in VAD\n", frame_num);

        frame_num++;
    }

    WebRtcVad_Free(vad);
    fclose(file);
    return 0;
}
```

Figure 3.2 : Core Logic of Web_RTC c test

WebRTC Vad works by splitting the signal into sub-bands via a filter bank, computes band-energy features, Applies a likelihood-ratio test (LRT) using Gaussian mixture models (GMMs) trained for speech vs. noise, and finally applies “hangover” logic so speech doesn’t flicker off at boundries.

Our test was a simple one just aimed at testing the functionality of WebRtc VAD when using PCM samples. We generated PCM samples identical to the one we expected to use from our ADC using the ffmpeg library and plugged them into the program.

The program opens a pcm file, reads 10 ms of sound (at 8 kHz, that’s 80 samples) and calls the WebRTC Vad function to see whether it’s speech or not. The input PCM is placed into 16 bit integers, which makes it possible to port the project to Vitis once it’s done.

```
PS C:\Users\musta\Desktop\VAD_C> .\main.exe .\no_7962.pcm
Frame 0: No speech
Frame 1: No speech
Frame 2: No speech
Frame 3: Speech detected
Frame 4: Speech detected
Frame 5: Speech detected
Frame 6: Speech detected
Frame 7: Speech detected
Frame 8: Speech detected
Frame 9: Speech detected
Frame 10: Speech detected
Frame 11: Speech detected
Frame 12: Speech detected
Frame 13: Speech detected
Frame 14: Speech detected
Frame 15: Speech detected
Frame 16: Speech detected
Frame 17: Speech detected
Frame 18: Speech detected
Frame 19: Speech detected
Frame 20: Speech detected
Frame 21: No speech
Frame 22: No speech
Frame 23: No speech
```

Figure 3.3 : Example Output of WEB_RTC c test

After verifying WEB_RTC C to be functional with our test pcm files, we decided it was time to move on to vitis and microblazeV.

3.3 FPGA Integration

With the original prototypes done it was time to move to the fpga prototype. For this, we modified the top module to add a processor block to include microblaze V and GPIOs.

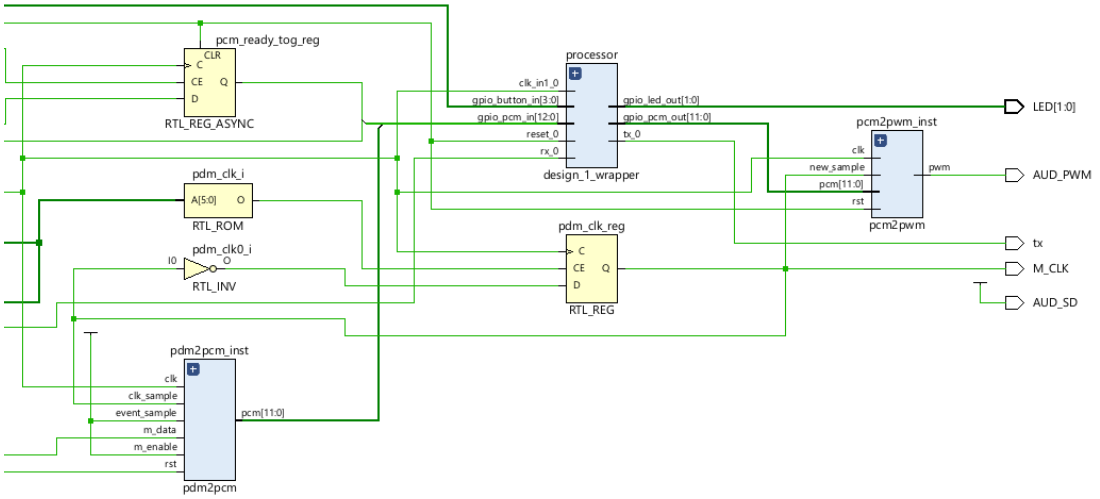


Figure 3.4 : RTL Schematic of Fpga Prototype

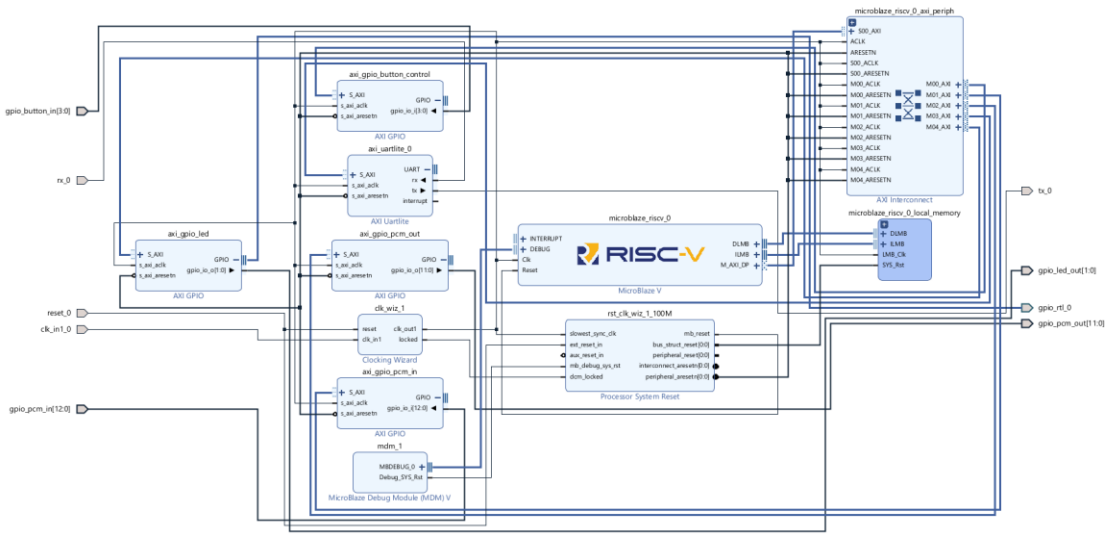


Figure 3.5 : Block design of the “processor” block

Our top module's main proponents are pdm2pcm that's connected to pcm_gpio_in, a pcm_ready_toggle that changes when new pcm inputs are ready, and led's for testing that's connected to gpio_led_out. After generating bitstream of this top module design, we exported the hardware to vitis and recreated the webRTC VAD project there.

```
    vad = WebRtcVad_Create();
    WebRtcVad_Init(handle: vad);
    WebRtcVad_set_mode(handle: vad, mode: 3); // Aggressiveness: 0-3

    // Init GPIOs
    XGpio_Initialize(InstancePtr: &pcm_gpio, BaseAddress: PCM_DEVICE_ID);
    XGpio_SetDataDirection(InstancePtr: &pcm_gpio, Channel: 1, DirectionMask: 0x1FFF);   // 13-bit input

    XGpio_Initialize(InstancePtr: &led_gpio, BaseAddress: LED_DEVICE_ID);
    XGpio_SetDataDirection(InstancePtr: &led_gpio, Channel: 1, DirectionMask: 0x0);    // Output for LE
    uint32_t last_ready = XGpio_DiscreteRead(InstancePtr: &pcm_gpio, Channel: 1) & READY_MASK;

    while (1) {
            uint32_t r;
    do {
        r = XGpio_DiscreteRead(InstancePtr: &pcm_gpio, Channel: 1);
    } while ((r & READY_MASK) == last_ready); // wait for 8 kHz edge
    last_ready = r & READY_MASK;

    uint32_t s = r & SAMPLE_MASK;
    int16_t s12 = (s & 0x0800) ? (int16_t)(s | 0xF000) : (int16_t)s;
    int16_t pcm_raw = (int16_t)(s12 << 4);

        frame[frame_index++] = pcm_raw;

        if (frame_index < FRAME_SIZE) continue;
        frame_index = 0;

        int result = WebRtcVad_Process(handle: vad, fs: SAMPLE_RATE, audio_frame: frame, frame_length:
        if (result == 1) {
            XGpio_DiscreteWrite(InstancePtr: &led_gpio, Channel: 1, Mask: 0x01);  // Speech detected
        } else if (result == 0) {
            XGpio_DiscreteWrite(InstancePtr: &led_gpio, Channel: 1, Mask: 0x02);  // No speech
        } else {
            XGpio_DiscreteWrite(InstancePtr: &led_gpio, Channel: 1, Mask: 0x00);  // Error or invalid
        }
```

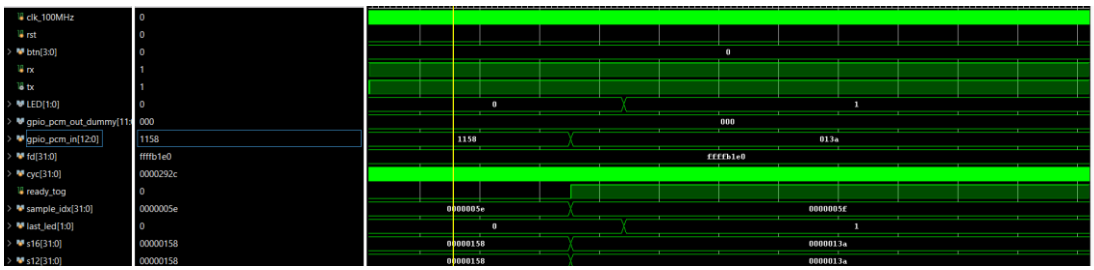
Figure 3.6 : WebRTC Project in Vitis core logic

The core logic is exactly the same as it was in the C Project, the main difference being instead of reading from a pcm file, we instead directly get the pcm values from gpio and add them as integers to create the frames to feed to webRTC VAD. To ensure frames aren't fed pcm values before new pcm values arrive, we utilize the pcm_ready_toggle we created in our top module.

```
98      int s16;
99      int unsigned s12;
100       // 8 kHz driver: every 125 us, read next sample, drive [11:0], toggle ready at [12]
101     always @(posedge clk_100MHz) begin
102       if (rst) begin
103         cyc         <= 0;
104         ready_tog   <= 0;
105         sample_idx  <= 0;
106       end else begin
107         if (cyc == SAMPLE_PERIOD_CYC-1) begin
108           cyc <= 0;
109
110
111           if (read_pcm_sample(s16)) begin
112             s12 = clip_to_s12(s16);
113             gpio_pcm_in[11:0] <= s12[11:0];
114             ready_tog         <= ~ready_tog;     // flip the sample-ready bit
115             gpio_pcm_in[12]   <= ready_tog;
116             sample_idx++;
117           end else begin
118             $display("[%0t] End of PCM file after %0d samples.", $time, sample_idx);
119             // Hold last value, stop toggling, and end sim a bit later
120             // so firmware can finish its last frame.
121             repeat (20000) @(posedge clk_100MHz); // ~200 us
122             $finish;
123           end
124         end else begin
125           cyc <= cyc + 1;
126         end
127       end
128     end
129
130     // LED monitor
131     always @(posedge clk_100MHz) begin
132       if (!rst && LED !== last_led) begin
133         $display("[%0t] LED = %b   (01 = speech, 10 = no speech, 00 = error)", $time, LED);
134         last_led <= LED;
135       end
136     end
137
138     endmodule
```

Figure 3.7 : Simulation Logic in Vivado

With the elf file created, we tested the program in Vivado simulation with the pcm samples we have created before. And from seeing the simulation, we can concur that we have successfully ran the VAD process inside microblazeV. That gave us enough confidence to proceed with genus.




```
launch_simulation: Time (s): cpu = 00:00:14 ; elapsed = 00:00:44 . Memory (MB): peak = 3106.324 ; g
run 10 s
[11884095000] LED = 01   (01 = speech, 10 = no speech, 00 = error)
run: Time (s): cpu = 00:07:56 ; elapsed = 00:21:29 . Memory (MB): peak = 3107.699 ; gain = 1.277
```

Figure 3.8 : Waveform and Testbench Indicating Success

3.4 Genus Synthesis

The PULPissimo RISC-V core was synthesized using Cadence Genus 21.15-s080_1 with TSMC 65nm standard cell libraries. While synthesizing we targeted a 100 MHz clock frequency (10ns period) for the main ref_clk domain, with secondary clock domains like the 10 MHz tck domain showing significant positive slack (14.587 ns margin). The design achieved full timing closure with zero violating paths across all clock domains.

The synthesis results showed a total cell area of 747,422.880 μm², comprising 115,890 leaf instances. The breakdown revealed 37,352 sequential elements (32.2% of total instances) and 78,538 combinational cells (67.8%). Register components dominated power consumption at 67.51% (1.078 mW total), followed by logic at 31.05% (0.496 mW). Clock network power was minimal at 0.39% (6.231 μW), indicating efficient clock tree synthesis.

Critical path analysis revealed the most timing-critical path in the JTAG tap controller (soc_domain_i/pulp_soc_i/i_dmi_jtag_i_dmi_jtag_tap), which still met timing with 14.587 ns of positive slackThe path delay of 391 ps was well within the 20 ns maximum delay constraint defined in the SDC constraints.

The hierarchical area breakdown showed the soc_peripherals_i subsystem as the largest block (573,975.360 μm²), containing the UDMA subsystem and various peripheral controllers. The FC_CORE (cv32e40p_core) occupied 129,033.600 μm², with the FPU wrapper (i_fpnew_bulk) consuming 36,823.680 μm² of this area. Memory structures including the boot ROM and L2 RAM showed minimal area impact (54.720 μm² and 1.440 μm² respectively).

Power analysis at the frame#0 stimulus revealed total dynamic power of 1.597 mW, with internal switching accounting for 63.94% of power consumption. While better power performance could be achieved, this is well within the acceptable range for our project.

While we couldn't complete full post-synthesis simulation due to time constraints, the clean timing closure and QoR metrics confirm the design's physical feasibility. The results demonstrate successful integration of the WebRTC VAD software flow with

the PULP hardware platform, meeting all target specifications for the voice activity detection system.

If we had more time, the next step would have been to turn the elf file created for microblaze project and use that to simulate the program as functional in a System on Chip design.

```
if [ info exists search_path ] {
  set search_path_initial $search_path
} else {
  set search_path_initial {}
}
set ROOT "~/pulp_oguz/pulpissimo"

####################################
#Preset Global Variables
####################################

set MODULE pulpissimo
set DESIGN top
set DATE [clock format [clock seconds] -format "%y.%m.%d--%T"]
set LOCAL_DIR "[exec pwd]"
#set SYNTH_DIR
set _OUTPUTS_PATH "${LOCAL_DIR}/OUTPUTS"
set _REPORT_PATH "${LOCAL_DIR}/REPORTS"
set _LOG_PATH "${LOCAL_DIR}/LOGS"
set LIB_PATH "${LOCAL_DIR}/../pdk/lib/WC"
set CONSTRAINT_PATH "${LOCAL_DIR}/../constraints"

#Effort Declarations
set GEN medium
set MAP medium
set OPT medium

set_db / .command_log ${_LOG_PATH}/cmd_${DATE}.cmd
set_db / .log_file ${_LOG_PATH}/log_${DATE}.log

####################################
#HDL Read PART
####################################

set search_path $search_path_initial
set_db init_hdl_search_path $search_path

read_hdl -language sv \
    -define { \
        TARGET_GENUS \
        TARGET_SYNTHESIS \
    } \
    [list \
        #"$ROOT/.bender/git/checkouts/tech_cells_generic-a241d08ee498172c/src/rtl/tc_sram.sv" \
    ]
```

Figure 3.9 : Portion of TCL file used to synthesize Pulpissimo

4. Realistic Constraints and Conclusions

Our project aimed to design a mixed-signal processor for analog sensors, combining a second-order ΔΣ ADC with a RISC-V-based digital subsystem. While the analog and digital components were developed and validated independently, time and resource

constraints prevented us from empirically testing integration. Below, we summarize key achievements, challenges, and recommendations for future work.

4.1 Practical Application of this Project

Our proposed system targets low-power, high-resolution sensor interfaces (e.g., audio processing). The ΔΣ ADC design achieved a simulated SNR of 95 dB in MATLAB, which was quite enough to obtain 12-13 bits. And the behaviroal simulation in CppSim also confirmed this result. The Op-amp designed in cadence also close to the result. and the digital subsystem (PDM-to-PCM conversion, WebRTC VAD) was functionally verified on an FPGA. However, the final integration of analog and digital modules remains incomplete, limiting real-world testing. Still, VaD modules on low power embedded devices has promising applications in many fields.

4.1 Realistic Constraints

4.1.1 Social, environmental and economic impact

For social impact, the project's modular approach lays groundwork for future work in accessible sensor technologies (e.g., assistive devices). For environmental impact, Energy-efficient design choices (e.g., RISC-V core, ΔΣ oversampling) were prioritized but not fully validated in hardware. As for economic impact open-source tools (RISC-V, WebRTC) reduced costs, but ASIC prototyping would require significant additional investment. Since the faculty has a Cadence License and TSMC PDK too, we didn't add these to the cost either, reducing the total cost.

4.1.2 Cost analysis

For labor, we spent ~400 hours of research/design (€8,000 at student rates). For Components, we used an FPGA board (€200), simulation tools (€0, academic licenses). There are unrealized costs that would be needed for a full physical integration, a full tape-out and mixed-signal validation equipment.

4.1.3 Standards

Our designs theoretically complies with IEEE 1241-2010 (ADC testing) and IEC 61672-1(audio noise). For compliance gaps no empirical testing was performed due to incomplete integration.

4.1.4 Health and safety concerns

Our FPGA-based digital subsystem and simulations posed no hazards.

4.2 Future Work and Recommendations

Looking ahead, several directions emerge for future work. The highest priority should be completing the analog-digital integration, likely through a custom PCB implementation that allows proper mixed-signal testing. The digital subsystem could be enhanced through implementation of more advanced signal processing algorithms, potentially leveraging machine learning techniques. On the analog side, the whole circuitry should be transistorized and it then the whole system must be verified. However, higher order or of continuous-time $\Delta\Sigma$ architectures might offer advantages for certain applications. Power optimization across both domains also presents significant opportunities for improvement.

REFERENCES

[1] Analog Devices Inc., "What Are the Basic Guidelines for Layout Design of Mixed-Signal PCBs?" *Analog Dialogue*, [Online]. Available: https://www.analog.com/en/resources/analog-dialogue/articles/what-are-the-basic-guidelines-for-layout-design-of-mixed-signal-pcbs.html. [Accessed: Nov. 25, 2024].

[2] IEEE. (n.d.). "Design of low-power CMOS ADC for portable applications." (1997) IEEE Explore. Retrieved from https://ieeexplore.ieee.org/document/674725

[3] R. Singh and S. Sali, "Substrate noise issues in mixed-signal chip designs using Spice," *10th International Conference on Electromagnetic Compatibility (1997) (Conf. Publ. No. 445)*, Coventry, UK, 1997, pp. 108-112, doi: 10.1049/cp:19971128.

[4] Hinner, M. (n.d.). PAL video signal. Retrieved from http://martin.hinner.info/vga/pal.html

[5] RISC-V International. (2025). RISC-V: The open standard RISC instruction set architecture. Retrieved from https://riscv.org/

[6] AMD. (2025). Microblaze V: A flexible and efficient RISC-V soft processor IP. Retrieved from https://www.amd.com/en/products/software/adaptive-socs-and-fpgas/microblaze-v.html#resources

[7] S. Pavan, R. Schreier, and G. C. Temes, *Understanding Delta-Sigma Data Converters*, 2nd ed. Hoboken, NJ, USA: Wiley-IEEE Press, 2017.

[8] R. Schreier and M. S. Sandler, "Delta-Sigma Data Converters: Theory, Design, and Simulation," Texas Instruments Application Report, 2015. [Online]. Available: https://www.ti.com/lit/an/slyt423/slyt423.pdf

[9] Knowles Acoustics, "SPU0410LR5H-QB MEMS Microphone Datasheet," Rev. 3.1, 2021.

[10] S. R. Norsworthy, R. Schreier, and G. C. Temes, Delta-Sigma Data Converters: Theory, Design, and Simulation. IEEE Press, 1997.

[11] J. M. De La Rosa, *Sigma-Delta Converters: Practical Design Guide*. John Wiley & Sons, 2018.

[12] R. van de Plassche, CMOS Integrated Analog-to-Digital and Digital-to-Analog Converters. Springer, 2003.

[12] A. H. Reeves, "Electric Signaling System," U.S. Patent 2,272,070, February 3, 1942.

[12] Marques, Ivo & Sousa, João & Sá, Bruno & Costa, Diogo & Sousa, Pedro & Pereira, Samuel & Santos, Afonso & Lima, Carlos & Hammerschmidt, Niklas & Pinto, Sandro & Gomes, Tiago. (2022). Microphone Array for Speaker Localization and Identification in Shared Autonomous Vehicles. Electronics. 11. 766. 10.3390/electronics11050766.

[13] R. Sum, C. Khongprasongsiri, W. Suwansantisuk and P. Kumhom, "Low Latency PDM-to-PCM Decoder," 2023 20th International Conference on Electrical Engineering/Electronics, Computer, Telecommunications and Information Technology (ECTI-CON), Nakhon Phanom, Thailand, 2023, pp. 1-4, doi: 10.1109/ECTI-CON58255.2023.10153161

[14] ITU-T, "Transmission characteristics for wideband digital loudspeaking and hands-free telephony terminals," *ITU-T Recommendation P.341*, Mar. 2011.

[15] R. Schreier, "Design of Discrete-Time Delta-Sigma ADCs" Lecture notes, Univ. of Toronto, 2009**.**

[16] Url-1 <*https://webrtc.googlesource.com/src/%2B/main/common_audio/vad/include/webrtc_vad.h*>, erişim tarihi 29.06.2025.

[17] RISC-V International. (2025). RISC-V: The open standard RISC instruction set architecture. Retrieved from https://riscv.org/

[18] AMD. (2025). Microblaze V: A flexible and efficient RISC-V soft processor IP. Retrieved from https://www.amd.com/en/products/software/adaptive-socs-and-fpgas/microblaze-v.html#resources

[19] Url-2< *https://www.cadence.com/en_US/home/company/newsroom/press-releases/pr/2020/cadence-digital-full-flow-optimized-to-deliver-improved-quality-.html*>, erişim tarihi 29.06.2025.

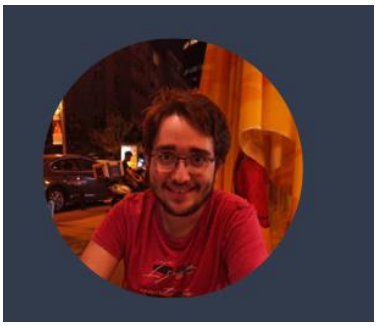
[20] IEEE. (n.d.). "Design of low-power CMOS ADC for portable applications." (1997) IEEE Explore. Retrieved from https://ieeexplore.ieee.org/document/674725

[21] R. Singh and S. Sali, "Substrate noise issues in mixed-signal chip designs using Spice," *10th International Conference on Electromagnetic Compatibility (1997) (Conf. Publ. No. 445)*, Coventry, UK, 1997, pp. 108-112, doi: 10.1049/cp:19971128.

APPENDICES

APPENDIX A: Maps

CURRICULUM VITAE



Name Surname **: Mustafa Oğuz Aksoy**

Place and Date of Birth **: Beyoğlu 28.08.2002**

E-Mail **: withoutsiz@gmail.com**

CURRICULUM VITAE

Name Surname **: Özge Ece Özalp**

Place and Date of Birth **: İstanbul 19.02.2000**

E-Mail **: ozalpo18@itu.edu.tr**