A Source-Follower Based Pipeline ADC Design with Error Correction

Wenbo Li
we2070li-s@student.lu.se

Department of Electrical and Information Technology
Lund University

Supervisor: Baktash Behmanesh, Lund University
Mattias Palm A, Ericsson
Lars Sundström S, Ericsson

Examiner: Pietro Andreani

June 10, 2024

Printed in Sweden
Tryckeriet i E-huset, Lund

Abstract

The content of this thesis consists of two parts, theoretical analysis of a pipeline ADC mainly in terms of error source and correction and a source follower based pipeline ADC design prototype.

For the general pipeline ADC analysis, we discuss the influence of typical nonidealities from different building blocks, e.g. the comparator, residual amplifier, and DAC, and show the corresponding treatments for these errors. We also discuss the general background correction method using LMS algorithm to resolve gain, memory effects and nonlinearities in the pipeline ADC analog part. For a high speed and high accuracy pipeline ADC, one of the promising method is using open-loop residual amplifier with complex linear and nonlinear correction. Nonlinear correction can be especially costly in digital hardware. In this thesis, we evaluate a new source-follower based architecture for the residual amplifier which has intrinsic good linearity to avoid digital nonlinearity correction. We apply basic inter-stage gain correction and memory effect correction in the software side to correct the output data. The memory effect correction show great improvement which makes it promising for high speed designs. The design is implemented in 7nm finFET technology. In the typical process corner (tt), $80°C$, and 0.75 V power supply, the circuit runs at 4 GHz clock frequency and 2 GHz signal frequency with SNDR 22.2 dB and SFDR 24.1 dB before correction. After gain and memory correction, the performance is SNDR 58.8 dB and SFDR 77.7 dB. Especially, memory effect correction shows an additional 11 dB improvement in SFDR. The average current consumption is 20 mA, and the corresponding FoM is 167.3 dB.

Popular Science Summary

Nowadays, people are quite getting used to hear about "4G" or "5G". It actually means 4th or 5th generation of wireless communication. The revolution from 4th to 5th is basically high carrier frequency and larger bandwidth. People may wonder what carrier frequency and bandwidth mean. If we take logistics as an example, the information we transfer is the good. Higher carrier frequency is equal to faster speed of the train, it will reduce the time used. Larger bandwidth is equal to having more train carriages, it can transfer more information each time. Then the analog to digital converter(ADC) we discussed in the thesis is like loading and unloading along this good supply chain. It plays a role of interaction between transferring and destination.

In reality, the function of ADC is to convert the analog signal into digital code. It needs to be fast to keep up with speed of transferring as we mimicked above. Another consideration is the linearity, which requires the converter output to be as linear to the input as possible. We find that this linearity performance can be impacted by some "fixed" errors. The mentioned "fixed" error is to distinguish with "random" one. How to handle these errors is one of the key theory parts in this thesis. Finally, we come up with the goal to design a high speed and high linear ADC.

How to boost the speed is the first problem. Intuitively, we believe that simple circuit form can have rather fast speed. The problem accompanied with high speed benefit is the inaccuracy. This inaccuracy we encountered here is mainly for gain factors inside the system, which is one major "fixed" error as we mentioned above. To deal with this error it comes with the idea of "correction". The basic strategy is that we try to "estimate" this error, and use the estimation value to recover the correct data. Here how to estimate is a mature algorithm referred as least square error.

Another finding is the hidden bandwidth limitation inside the ADC analog circuit when the system runs at quite high speed. This limitation acts as a low pass filtering and degrades the linearity performance. Intuitively, this low pass filtering in the analog side can be compensated by having another high pass digital filter in the digital part. The coefficients of digital filter can be estimated the same way as gain error mentioned above.

Acknowledgments

I would like to give thanks to all the wonderful people during the journey of this master thesis project. Special thank to the following persons of their dedicated patience and encouragement.

Lars Sundström, for setting up the MATLAB modeling and interesting discussion about calibration.

Mattias Palm, for circuit level review and detailed help with the Cadence software.

Table of Contents

List of Figures

List of Tables

Acronyms

ADC Analog to Digital Converter.

DAC Digital to Analog Converter.

ENOB Effective Number of Bits.

FoM Figure of Merit.

IIR Infinite Impulse Response.

LSB Least Significant Bits.

MDAC Multiplying Digital-to-Analog Converter.

MSB Most Significant Bits.

RF Radio Frequency.

SAR Successive Approximation Register.

SFDR Spurious-Free Dynamic Range.

SHA Sample and Hold Amplifier.

SNDR Signal to Noise and Distortion Ratio.

SNR Signal to Noise Ratio.

Chapter 1
Introduction

1.1 Background

In wireless systems, the Analog to Digital Converter (ADC) is one of the core components for baseband processing. With the communication standard migrating from 4G to 5G, the wider channel bandwidth requires ADCs with higher speed. Architecture innovation, like Radio Frequency (RF) direct sampling, also requires high operating frequency, e.g. multiple GHz to convert a wide bandwidth without signal aliasing. Thus, there is always a need to push ADCs to be faster, more precise and more power efficient.

The ADC basically performs two kinds of functions, sampling and quantization. Sampling converts a continuous electrical signal into time-discrete representatives. Quantization will round the time-discrete signal to a sequence of digital codes with a specific number of bits. Both of the functions will definitely modify the original signal by means of introducing additional noise, which is electrical noise and quantization noise respectively, and distorting the signal due to imperfect linearity.

There are common methods to overcome the above mentioned signal corruption. Noise as a physical limitation can hardly be improved without high power and large device area. Non-linearity, can by solved by different techniques from structure, e.g. feedback, to algorithm, e.g. pre-distortion and post-correction. Communication systems has especially high requirements for linearity to handle the large dynamic range of desired and undesired signals.

1.2 Basics of ADC

1.2.1 ADC Types

There are several types of ADCs. Two main groups are Nyquist rate sampling ADCs and oversampling ADCs. The basic types for the two kinds are listed in table 1.1 [1].

Hybrid architectures can be devised based on the fundamental types. One popular example is the Successive Approximation Register (SAR) conversion combined with pipelining to achieve multi-bit conversion per stage and good power

Table 1.1: Basic ADC types

	Type	Speed	Accuracy
Nyquist Sampling	SAR	moderate	moderate
	Flash	high	low
	Pipeline	moderately high	moderate
	Dual Slope	lowest	high
	Folding	high	low
Oversampling	Delta-Sigma	low	highest

efficiency[2]. Another example of a hybrid architecture is the noise-shaping SAR, which combines SAR and Sigma-Delta Modulation and benefits from the advantages of low complexity and high SNR respectively[3].

In this thesis, a key requirement for the ADC is to achieve high speed, preferably with sampling rates up to 4 GSps, and rather high resolution, 10 bits. From the available ADC survey shown in figure 1.1 [4], we collect examples for targeted performance shown in table 1.2. From the survey, we find that pipeline is a promising architecture in these scenarios, which is also our topic.

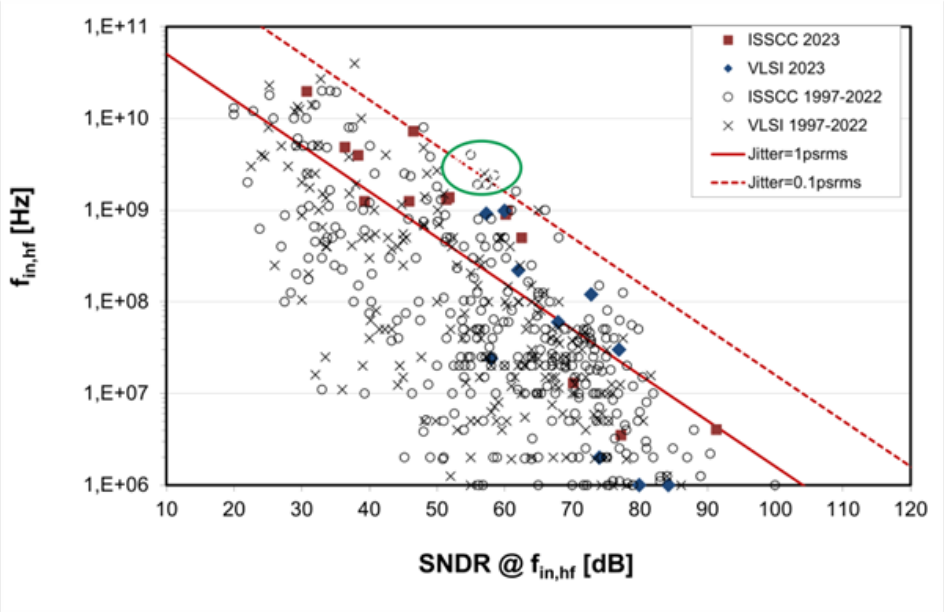


Figure 1.1: ADC survey

1.2.2 Specification of ADC

Generally, ADCs are evaluated based on its dynamic and static performance. For the dynamic specifications, the typical test is to have a sine wave signal as input,

Table 1.2: Examples of desired performance

	Technology	Architecture	SNDR(dB)	Fin(GHz)	Cite
1	28nm	Pipeline,SAR,TI	58.5	2.4	[5]
2	16nm	Pipeline, TI	61.7	1.6	[6]
3	16nm	Pipeline, SAR, TI	57.3	1.9	[7]
4	16nm	Pipeline, TI	56	1.9	[8]
5	16nm	SAR, TI	57	2.5	[9]

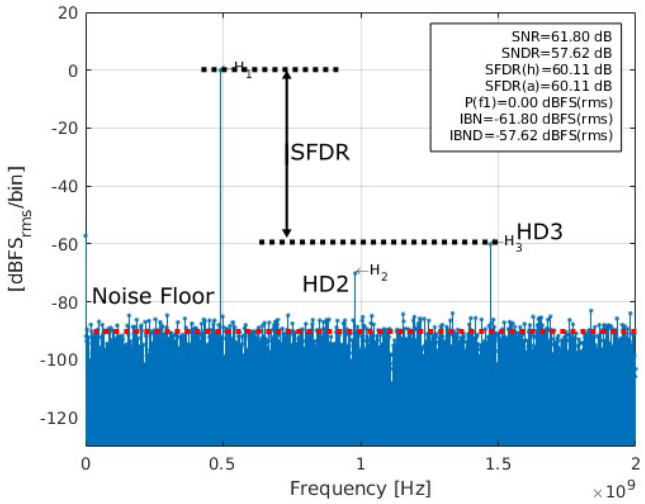


Figure 1.2: Spectrum of single tone test

and check the output spectrum by DFT. A typical spectrum is shown in figure 1.2, composed of three major parts, the desired signal tone (H1), the noise floor, and harmonic distortion (H2, H3). The dynamic performance metrics are defined as follows.

Signal to Noise Ratio (SNR)

$$SNR = \frac{P_{signal}}{P_{noise}} \tag{1.1}$$

Signal to Noise and Distortion Ratio (SNDR)

$$SNDR = \frac{P_{signal}}{P_{noise} + P_{total\ distortion}} \tag{1.2}$$

Effective Number of Bits (ENOB)

$$ENOB = \frac{SNDR(dB) - 1.76}{6.02} \tag{1.3}$$

Spurious-Free Dynamic Range (SFDR)

$$SFDR = \frac{P_{signal}}{P_{largest\ spur}} \tag{1.4}$$

One commonly used Figure of Merit (FoM) is the Schreier FoM given

$$FoM_s = SNDR(dB) + 10log\frac{f_s}{2 * P} \tag{1.5}$$

which is in dB scale. P is power in Watts, and f_s is sampling frequency. For static specifications, the typical test is to have a slow ramp signal as input, and then check the output code. However, the method could be inaccurate when the ADC is required to run at high frequency in a real-time scenario. The reason is that signal settling for the current sample maybe impacted by the previous samples due to e.g. incomplete reset.

1.3 Basics of Pipeline ADC

Figure 1.3 shows the basic operation of a pipeline ADC. Assume that we have a measurement range of $[-V_R, V_R]$. As an example, if we discretize the whole range into 4 equally sized sub-ranges (e.g. 2bit quantization), each sub-range has a "representation". It means that any value between $[-V_R, V_R]$ can be represented by one of values in $\{+\frac{3}{4}V_R, +\frac{1}{4}V_R, -\frac{1}{4}V_R, -\frac{3}{4}V_R\}$ with limited precision. Assume that we deal with a test value of $+\frac{2}{5}V_R$, it is located in the range of $[0, +\frac{1}{2}V_R]$, which is represented by $+\frac{1}{4}V_R$. The value between the test signal and its quantized "representation" is called "residue" or "quantization error", which is $\frac{2}{5}V_R - (+\frac{1}{4}V_R) = +\frac{3}{20}V_R$. Then we amplify this sub-range to be the full range again, which is 4 times in this case. The corresponding residue is amplified 4 times, which is $+\frac{3}{5}V_R$. The amplified residue is now in $[+\frac{1}{2}V_R, V_R]$, which is quantized as $+\frac{3}{4}V_R$.The final quantized result is the combination of the two quantization results, which are regarded as coarse and fine results respectively. In this case, it is $+\frac{1}{4}V_R + \frac{1}{4} * \frac{3}{4}V_R = +\frac{7}{16}V_R$.

$$total\ quantization = coarse\ quantization + \frac{fine\ quantization}{amplifacation\ gain} \tag{1.6}$$

From the above discuss, it would be interesting to investigate the residue output vs its stage input, which is regarded as stage transfer curve shown in figure 1.4. The four region is for 4 quantization output. In each region, the output can cover $-V_R$ to $+V_R$, and the slope of each linear function is 4, which is for 2bit.

The idea mentioned above is shown as figure1.5.A sample-and-hold is used at the input to drive an M-bit ADC which generates the Most Significant Bits (MSB) (coarse conversion). The Digital to Analog Converter (DAC) then converts the M-bits back to an analog signal which is subtracted from the held input to give the coarse quantization error (also called the residue). Next, the residue is amplified and converted into digital by a second N-bits flash which yields the Least Significant Bits (LSB) (fine conversion). The digital logic then combines coarse and fine results to obtain the complete ADC output. The combination is M-bits

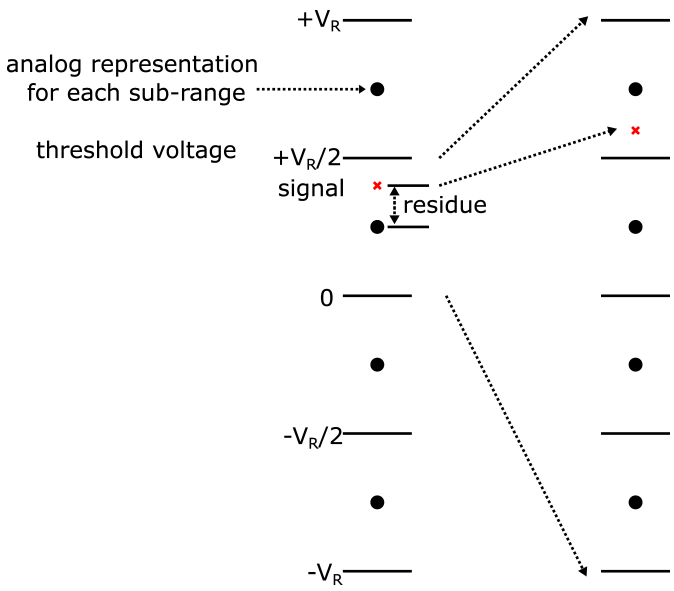


Figure 1.3: Diagram of pipeline operation

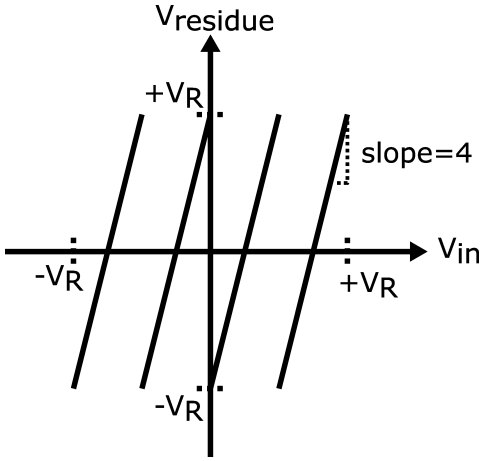


Figure 1.4: 2bit stage residue output transfer curve

plus the result of N-bits divided by K. If the gain factor K is 2^k, then the division in digital domain becomes shifting k bits. This two step conversion can be extended to multiple step conversion by cascading stages that include coarse ADC, DAC, residue generation, and amplification.

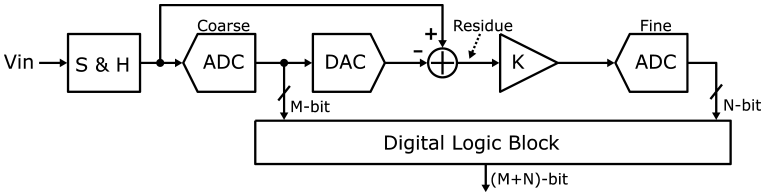


Figure 1.5: Block diagram of two step conversion

1.4 Techniques for High Speed and Resolution

1.4.1 Redundancy

Redundancy is a technique used to tolerate errors in the decisions. For a certain resolution, it require extra bits.

Figure 1.6 shows the concepts of how redundancy works. The example has a two-step conversion with same 4 times residue amplification. The first step is a 3-bit conversion with 1-bit built-in redundancy. The second step is a two-bit conversion. The left diagram is the case when no error occurs, and the right one is for the case with error. The test signal (near $+\frac{9}{32}V_R$) is labeled as red dot and is quantized as "101". During the second conversion, range "101" is amplified to cover "01" and "10", which are half of the whole range. The residue is quantized as "01" during the second conversion. In the rightmost diagram, the decision threshold is slightly wrong and has increased upwards from its ideal position, shown by the dashed line. The signal is now mistakenly quantized as "100" for the first stage conversion, since it is on opposite side of the threshold. But then the residue is amplified to be in the region of "11". Because the residue amplification is 4, the output of the second quantization is divided by 4 and added to the first step output. In the digital form, this division is done by right shifting 2 bits.

In a conclusion, redundancy provides more margin for errors in the analog domain (e.g. decision errors) and several digital representations for an identical input. This means means multiple codes map to the same value.

1.4.2 Open-Loop Amplifier

Murmann first introduced the solution of using open-loop residual amplifier to boost the speed [10]. Since the feedback path is removed in the amplifier topology, the achievable gain-bandwidth product is extended. That helps when time for small signal settling is limited, like in high speed ADCs.

However, the benefit comes with a cost. Open-loop amplifier based architecture will necessarily need correction of gain, and have larger distortion due to

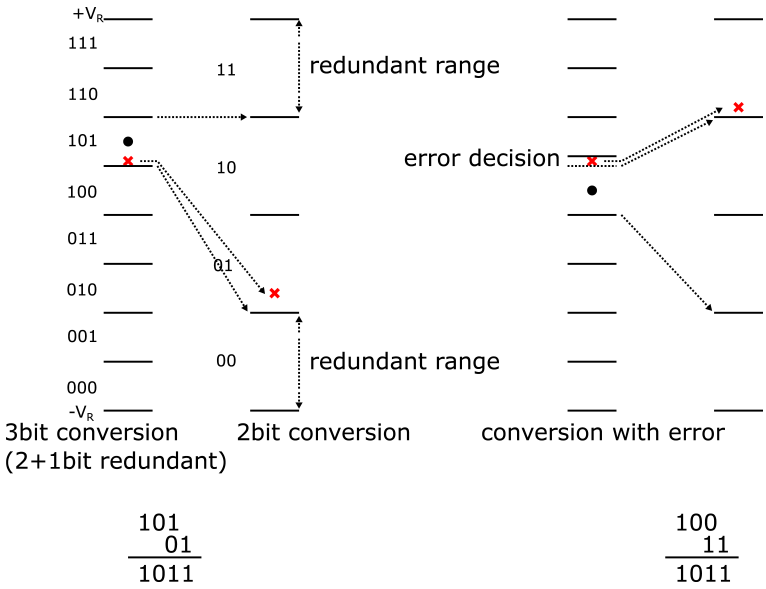


Figure 1.6: Example of redundancy

the amplifier nonlinearity which may requires complex correction algorithms to handle.

1.4.3 Correction

The correction methods can be categorized into two main types, foreground and background methods, respectively. Foreground correction needs extra timing to periodically perform measurements and trimming of the ADC itself. The correction phase may interrupt real-time processing of the input signal and the ADC cannot swiftly adapt to environment changes, like supply voltage and temperature variation. Background correction operates continuously without interfering with the ADC and thus does not have such issues. Background correction is the focus of this thesis. There are two basic methods for background correction, correlation-based and equalization-based architectures.

Figure 1.7 shows the idea of the correlation based correction method. A PRBS signal is applied to ADC along with the desired signal. The test signal can traverse the same trajectory of digitization experienced by the input. By checking the statistic property of test signal output response, the ADC parameters can be estimated.

Figure 1.8 shows the idea of equalization based correction method. A slow-but-accurate reference ADC is used to correct the fast (but erroneous) main ADC output. The correction engine usually includes a least mean square algorithm to adjust parameters for post-processing of the main ADC output.

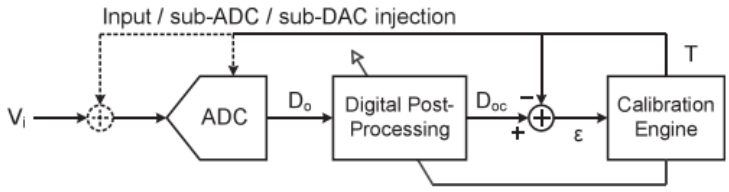


Figure 1.7: Correlation based correction[11]

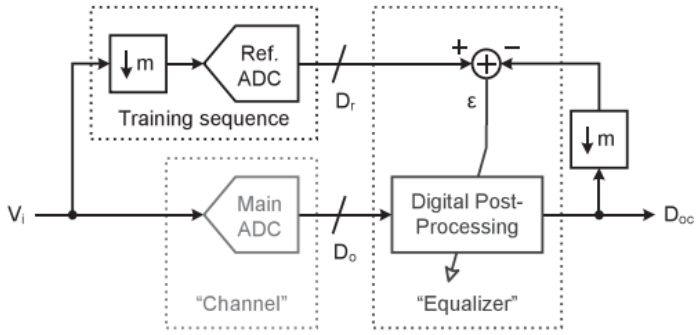


Figure 1.8: Equalization based correction[11]

1.5 Outline of the Thesis

This thesis is organized as follows

- **Chapter 1** introduces the state of art for GHz ADCs which also require high SFDR performance. We list several commonly used methods for this type of ADC.
- **Chapter 2** introduces the basics of the pipeline ADC. We mainly discuss its noise contribution and how its error sources impacts the performance.
- **Chapter 3** introduces the post background correction of its algorithm and equivalent hardware implementation.
- **Chapter 4** discusses the circuit we designed in this work. We will discuss how source follower which intrinsically has no gain is used to build "amplification" in residual amplifier.
- **Chapter 5** lists the simulation result for this design. We will mainly find the amazing SFDR improvement by applying correction methods.
- **Chapter 6** discusses future work.

Chapter 2

Pipeline ADC Analysis: Analog Part

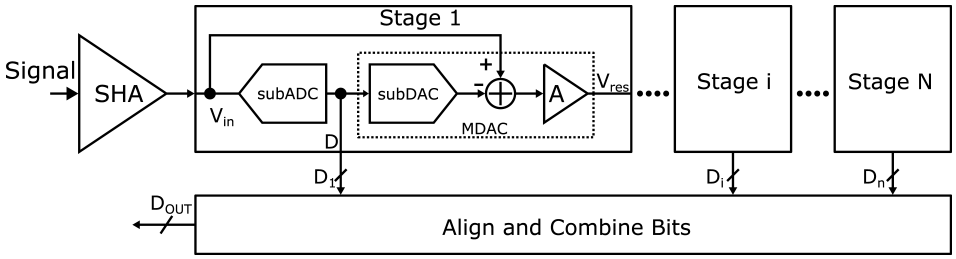
In this chapter, we will mainly investigate how circuit behavior influences the ADC performance. The discussion will cover the linear part, e.g. noise, as well as the nonlinear part. For the nonlinear part, we will reveal how the effects change the transfer curve of the stage and the whole ADC, and then look at the ADC dynamic performance. This analysis can be used as a guideline for the specification of sub-block circuits.

2.1 Introduction of Modeling

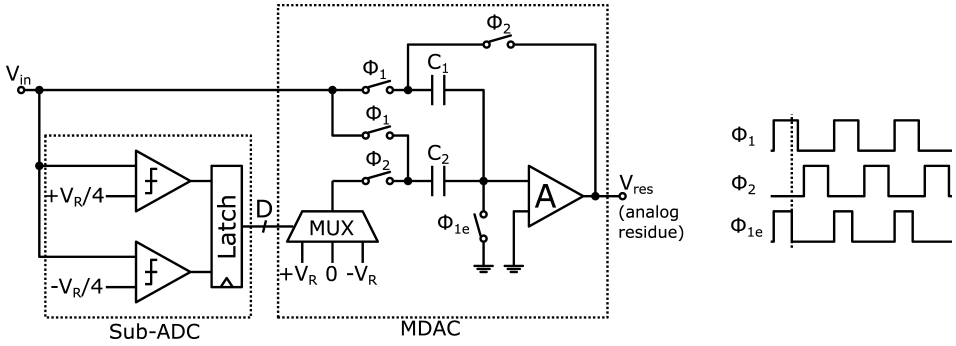
Figure 2.1 shows the diagram of a typical pipeline ADC with a 1.5bit stage. The ADC consists of both analog and digital parts. For the analog parts, the signal is sampled by the Sample and Hold Amplifier (SHA) stage and processed sequentially by each pipeline stage, each having the same function. The digital part will time align the outputs from different stages and combine them with suitable weights.

A typical example of a 1.5bit stage is shown in figure 2.1b. The circuit can be divided into two parts, subADC and MDAC. The subADC is a flash type ADC which has two threshold voltages connected to two comparators. The MDAC is a switched capacitor amplifier which operates between two non-overlapping phases. For Φ_1 phase, V_{in} is sampled to C_1 and C_2. C_1 is equal to C_2. In addition, the subADC generates digital output (D) at the end of Φ_1. For Φ_2 phase, C_1 forms the feedback and C_2 is connected to the DAC whose output is selected by the subADC output. The analog output of the MDAC, which is called residual voltage, is generated in Φ_2 with a certain settling behavior.

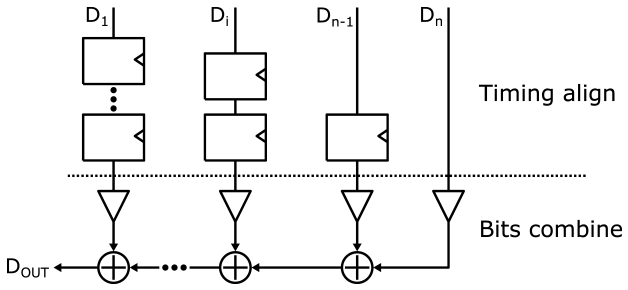
The typical circuit of the digital part is indicated in figure 2.1c. The code which relates to the same origin of input signal sample is collected from different clock cycles, so that we need to synchronize them first and then add them with different weights. Figure 2.1d shows how the aligned data is being added. For the ideal case, the MDAC has an amplification of two, so that D_2 is scaled by 0.5 compared to D_1. This binary scaling is done by shifting, and the different shifting of bits yields the weights for the different outputs of each stage.



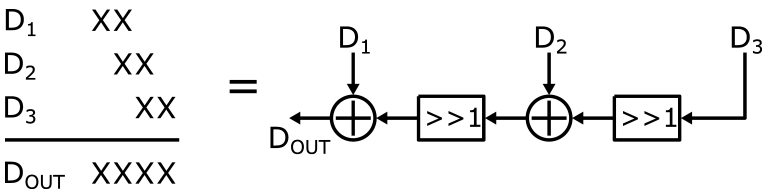
(a) pipeline adc architecture



(b) 1.5bit stage



(c) Code align and combination



(d) bit combination for 1.5bit stage output

Figure 2.1: diagram of typical pipeline ADC with 1.5bit stage

We have briefly described the pipeline ADC architecture and its corresponding circuit. Next we should explore what circuit effects may potentially impact ADC performance. The analysis focus on the stage behavior of its sub-blocks. Figure 2.2 indicates the possible error source from different circuits. For comparator side,

we mainly consider its input-referred noise and offset voltage. For the DAC, we mainly consider noise and offset from the reference voltage generation circuit. For the amplifier side, we mainly consider its input-referred offset, noise, finite gain and bandwidth. For capacitors, we consider their mismatch. Amplifier finite gain and capacitor mismatch will impact MDAC DC gain, and finite bandwidth will introduce small signal settling error, which can be categorized into MDAC gain loss.

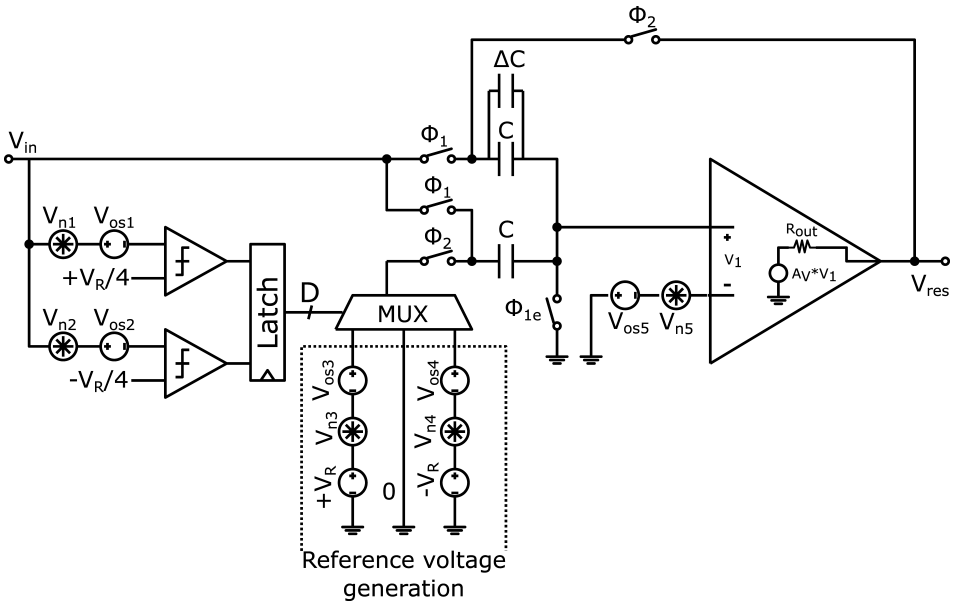


Figure 2.2: 1.5bit stage with error source

2.2 Noise Analysis

2.2.1 Quantization Noise

Figure 2.3 shows an equivalent linear model to analyze quantization noise, in which e_i indicates the quantization error for each stage from its input signal data sample and k indicates residual gain. We treat ADC as the linear block which only introduces quantization error, and DAC as a linear proportional path with gain equal to 1. It should be mentioned that the calculation in the figure is in amplitude instead of power. From the model, we find that e_1 in the first stage digital output is compensated by the second digital output after scaling. The only remaining error in output y is the last stage quantization error e_3 scaled by all the previous gain. This results in

$$SQNR = 6.02 \cdot B + 1.76 + 20log_{10}(k^n) \tag{2.1}$$

in which B is the number of bits of the last stage, and n is the number of stages with equal gain. We need to be aware that the equation above requires perfect

matching of stage gain and post-scaling. If that is not achieved, additional errors will leak to the output. Assuming that we have bit combination by shifting, this requires a high accuracy for the Multiplying Digital-to-Analog Converter (MDAC) stage. Otherwise, if we do not have precise analog gain, then we need an extra circuit to detect the actual gain and set the post-scaling weights accordingly.

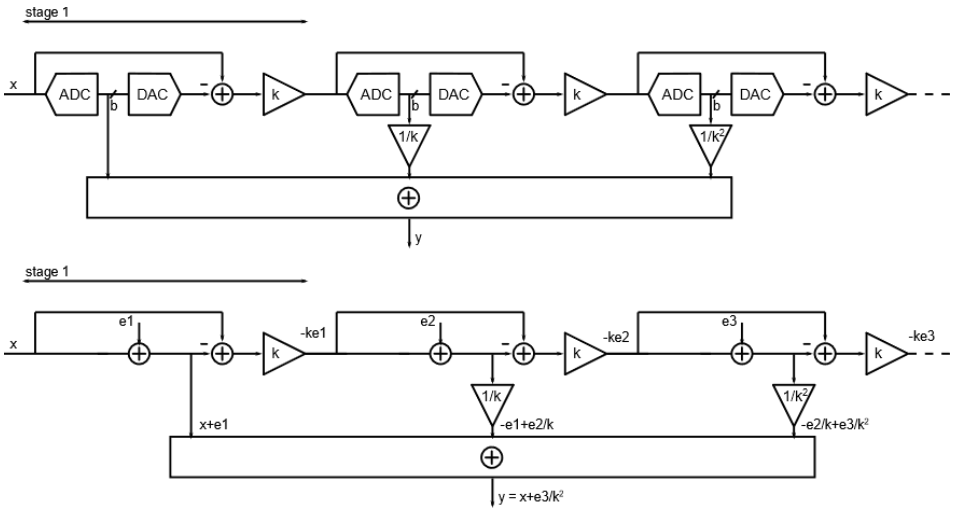


Figure 2.3: quantization noise model

2.2.2 Circuit Noise

The circuit noise analysis of a pipeline ADC is conducted in system and stage level. From the system level, figure 2.4 shows its equivalent noise. Each stage of its input-referred noise is attenuated by the gain of its previous stage, so that the total input-referred noise is

$$v_{n,tot}^2 = v_{n,1}^2 + \frac{v_{n,2}^2}{G_1^2} + \frac{v_{n,3}^2}{G_1^2 G_2^2} + \dots \tag{2.2}$$

For each stage, the input-referred noise is contributed by switches, the amplifier, and DAC reference voltage.

$$v_{n,input}^2 = \frac{kT}{2C} + (v_{n,amp}^2 + v_{n,dac}^2) \cdot \frac{1}{4} \cdot \int_0^\infty \frac{1}{1 + (\frac{\omega^2}{\omega_T^2})}\, d\omega \tag{2.3}$$

in which $\frac{1}{4}$ factor representation output-referred to input-referred conversion factor, ω_T is the closed-loop bandwidth when MDAC is in Φ_2 phase.

The comparator noise is neglected since it has almost no contribution because of the redundancy. The basic reason is that with presence of limited level of comparator noise, the final result will not change. This will be better illustrated later.

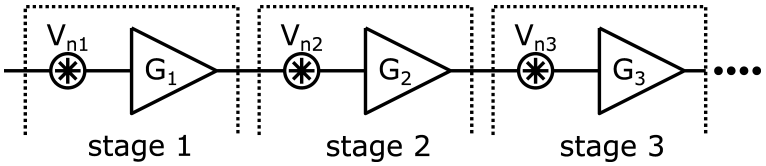


Figure 2.4: pipeline ADC system level noise equivalence

2.3 Circuit Nonidealities

2.3.1 subADC comparator offset

Figure 2.5 shows the input and output voltage function for a single stage considering comparator offset. The blue line is the standard curve without offset voltage. When the offset voltage is too large, the threshold will be horizontally shifted a lot and the output may not follow the linear relationship and cause clipping.

Figure 2.6 is the result comparison between small and large comparator offset. As long as the threshold shifting does not cause the residue output to clip the input range of the next stage, the whole ADC transfer curve will always be the same because of the 0.5bit redundancy. With large offsets, the whole ADC transfer curve shows "steps", e.g. missing codes.

We can relate the conclusion on offset to the comparator noise, mentioned before. With the existence of comparator noise, it is equal to the stage having a timing varying threshold (offset) voltage. However, the varying transfer curve of each stage will finally resemble an identical ADC transfer curve if no clipping occurs. That requires the maximum noise level to be smaller than the redundancy range. In the design, we could approximately use the 3-σ value to be the maximum tolerated noise level.

Typical comparator offset voltage is 1-10mV, which is quite much smaller than the available redundancy range 0-$V_R/4$.

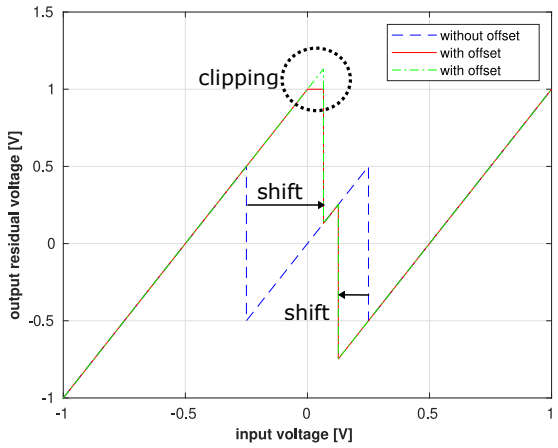
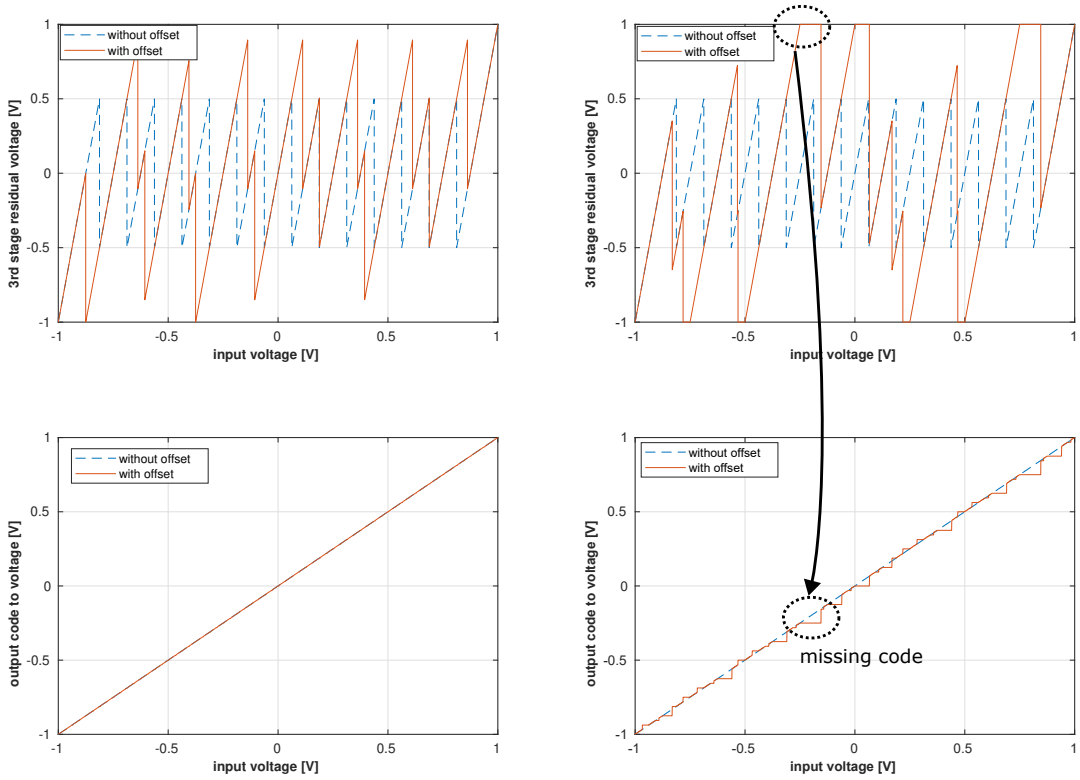


Figure 2.5: Stage transfer curve with comparator offset



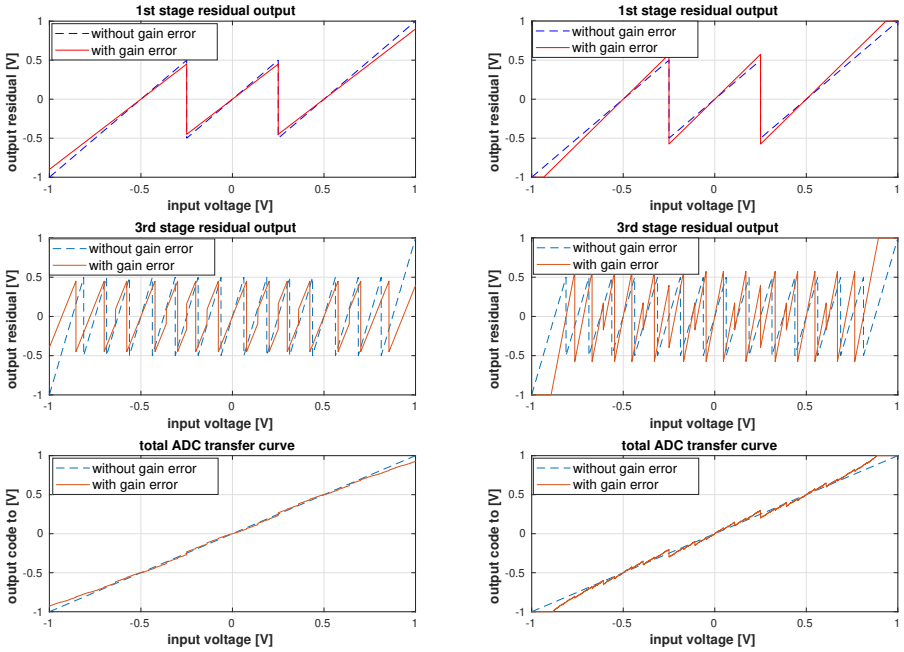
(a) small offset **(b)** large offset

Figure 2.6: ADC transfer curve with comparator offset

2.3.2 Residual Amplifier Gain Mismatch

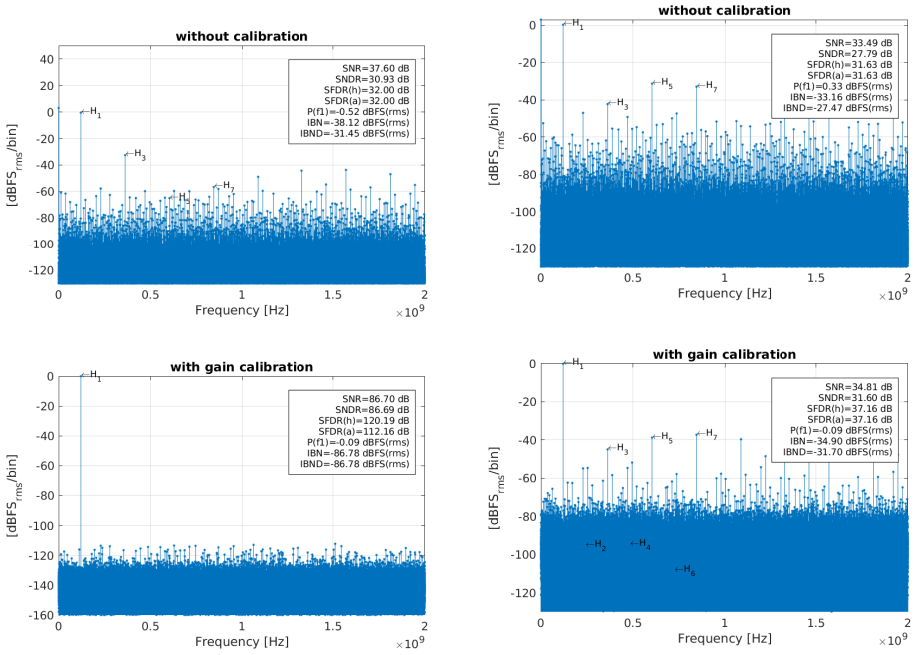
Figure 2.7 shows the transfer curve with residual amplifier gain error. The total ADC transfer curve has a sawtooth shape. With gain < 2, no clipping happens and the gain correction method can be used to optimize the results, as shown in figure 2.8. Once clipping happens, the residual voltage looses information for the signal which is not recoverable in the digital domain.

Considering the presence of digital correction, the whole ADC performance will be impacted by the mismatch between digital estimated gain and analog circuit gain. Figure 2.9 is the result when we fix the weights of post-processing and modify the amplifier gain. The SFDR quickly drops as a gain error is introduced. For this design, the accuracy in gain needs to be better than 0.1% to achieve SFDR above 70dB. This result also serves as guidance on the accuracy required for the digital part when introducing digital correction.



(a) gain < 2 **(b)** gain > 2

Figure 2.7: Transfer curve with RA gain error



(a) No clipping **(b)** With clipping

Figure 2.8: Spectrum with RA gain error

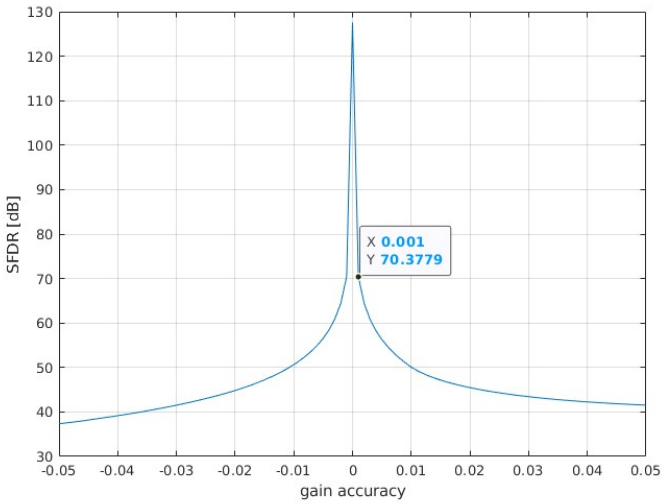


Figure 2.9: SFDR vs gain accuracy

2.3.3 Residual Amplifier Nonlinearity

Figure 2.10 shows how the RA nonlinearity impacts the ADC transfer curve. In the model, we only introduce a 3rd order component, which is usually the dominant distortion. In the modeling, we define the RA with the following polynomial form shown in figure 2.11, and its 3rd order parameter a will be used in the following sweep.

Figure 2.12 is the derived SFDR result when we sweep the 3rd order parameter a. The plot is a guide to specify the amplifier performance. In this design, since we need SFDR higher than 72dB, the desired a value should be less than 0.06.

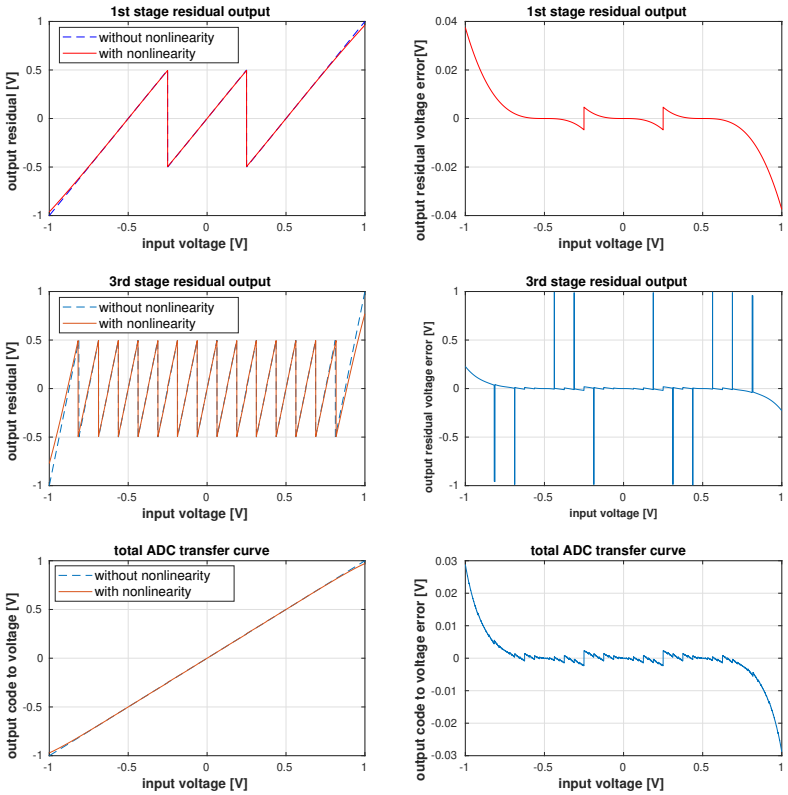


Figure 2.10: Transfer curve with RA nonlinearity

2.3.4 subDAC reference voltage mismatch

This section will discuss how dac response change impacts the ADC behavior and performance. This certain problem is shown in figure 2.13. The red line is the

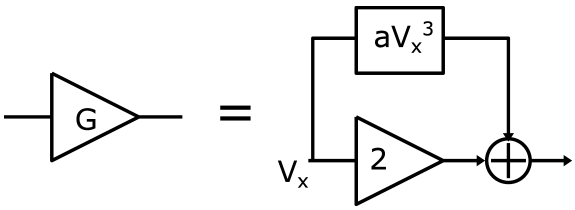


Figure 2.11: RA polynomial model

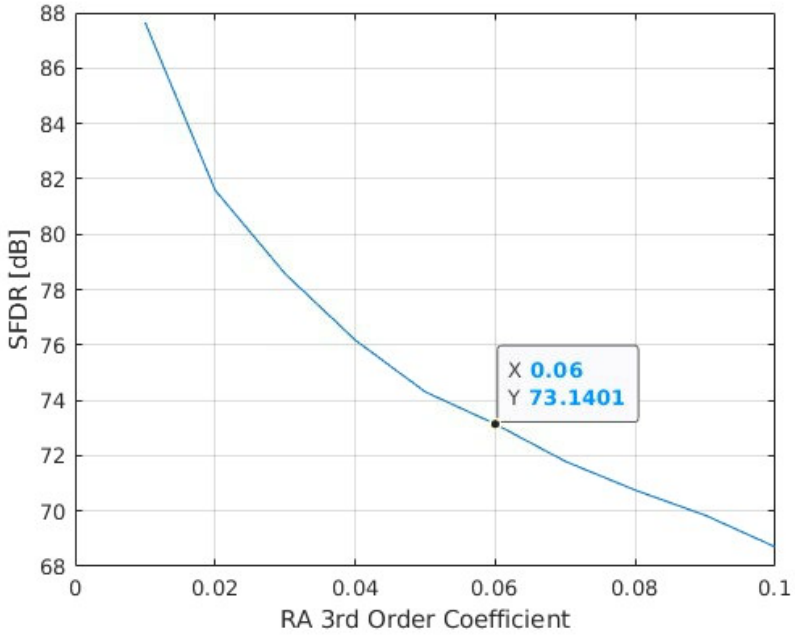


Figure 2.12: SFDR vs RA nonlinearity

ideal DAC response, in which it has $+\frac{V_R}{2}$ and $-\frac{V_R}{2}$ for input low and high case respectively. The green line shows that DAC output shift from ideal one with independent errors for input low and high case.

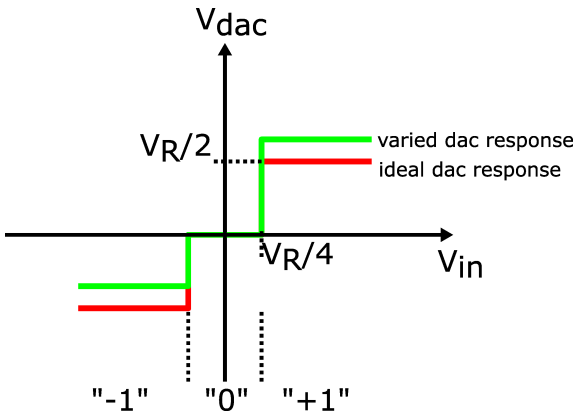


Figure 2.13: DAC response change

Figure 2.14 shows the transfer curve with the effect of offset on the DAC reference voltage, where the offset is asymmetric for positive and negative side. We denote this a PN mismatch. In the stage transfer curve, the two segments are shifted vertically. The whole ADC transfer curve has the same "segment shifting" feature.

The ADC has been simulated with a small and large DAC offset to demonstrate with and without clipping. The output is calibrated with the correct weights for the DAC, assigning different weights for code "-1" and "1", see figure 2.16. When there is no clipping, the performance fully recovered as expected, while clipping leads to a large degradation in SNDR and SFDR.

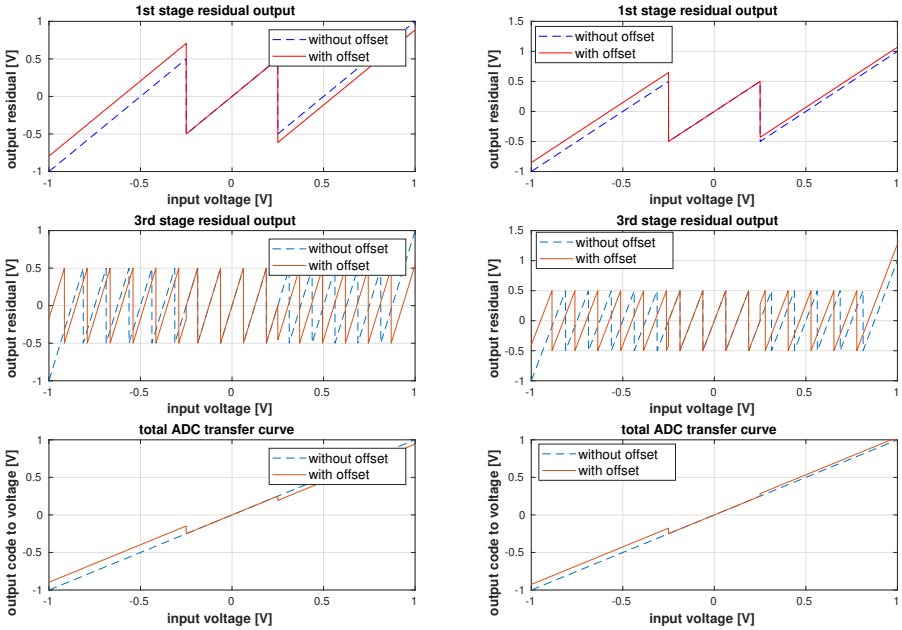
2.3.5 subDAC gain error

This section discusses the case when DAC reference voltage is attenuated along its path, which is denoted as a gain error. The attenuation may come from parasitic elements, signal settling error etc. Figure 2.17 shows the response function for DAC with gain error.

We can refer to figure 2.18. It shows a two-stage pipeline ADC with DAC gain error in the first stage. The DAC gain is denoted by K, and residue amplifier gain is G.

Ideally, the residue output will be e_1 only when $K = 1$, which means the residue output will only contain the quantization error. However, when the DAC path has an attenuation ($K < 1$), the signal will leak to the following stages.

The consequence of this effect is shown in figure 2.19. Except for the clipping at the two ends, the curve shows good linearity without distortion, and its slope deviates from the ideal one. For an intuitive solution to fix the problem, we could expect to scale the output to "rotate" the curve, this can be done by applying identical scaling factor to the combination weights in the digital domain. We



(a) no clipping **(b)** with clipping

Figure 2.14: Transfer curve with DAC offset

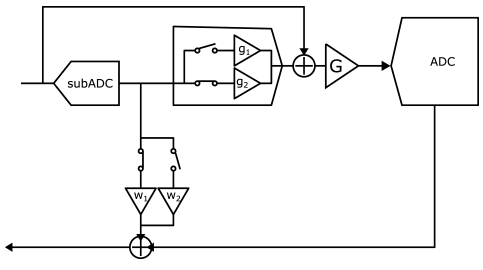
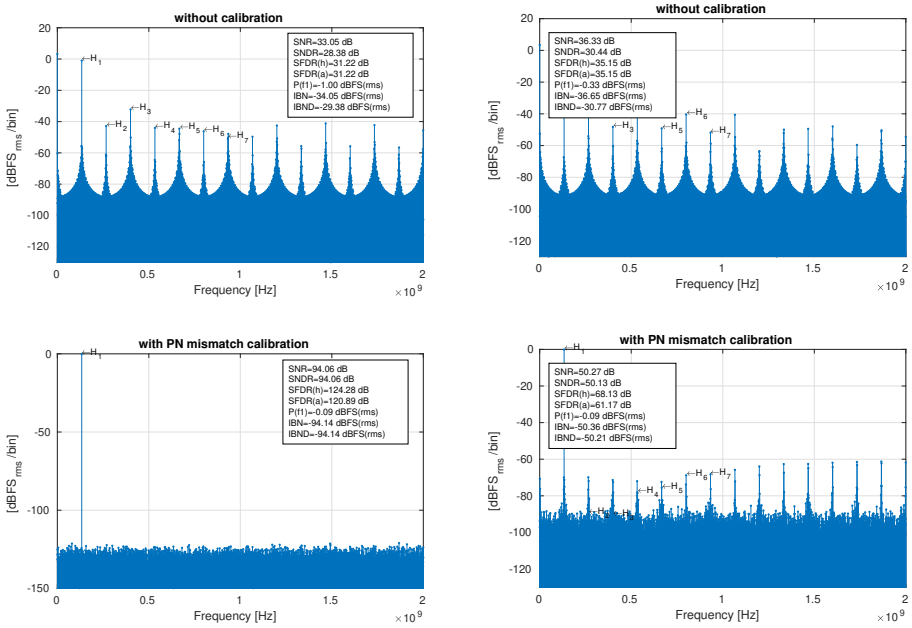
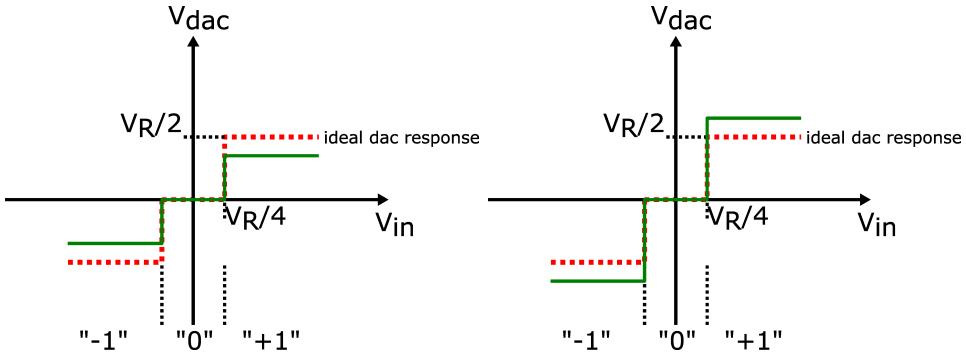


Figure 2.15: Diagram of correction with different weights



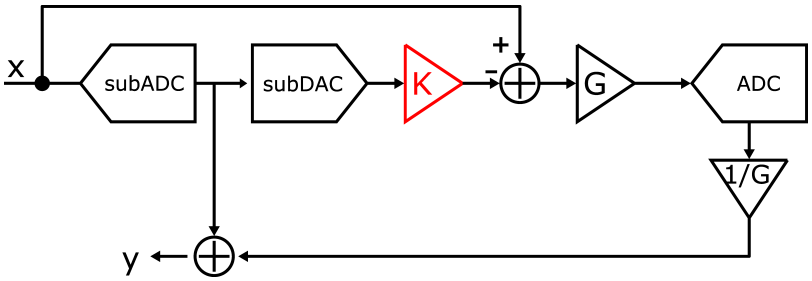
(a) no clipping **(b)** with clipping

Figure 2.16: spectrum with dac offset

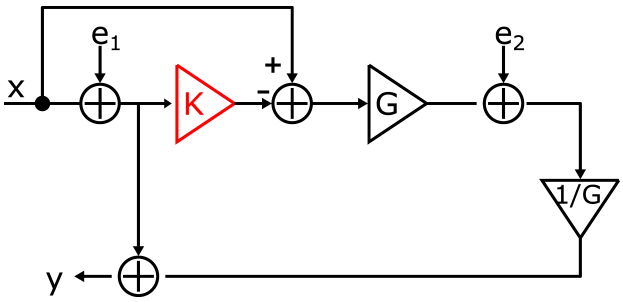


(a) small gain **(b)** large gain

Figure 2.17: DAC response with gain error



(a) Pipeline architecture with dac gain error



(b) equivalent model

Figure 2.18: DAC response with gain error

could expect that this scaling factor is almost K. A brief analysis and derivation is included at the end of thesis. For the clipping problem, we can have extra residual amplifier gain attenuation as well to avoid stage output saturating.

Figure 2.20 is a result with DAC gain error and residual amplifier gain loss. Here, we set DAC gain to be 0.95, and residual amplifier gain to be 1.9. The results show good SFDR improvement after calibration.

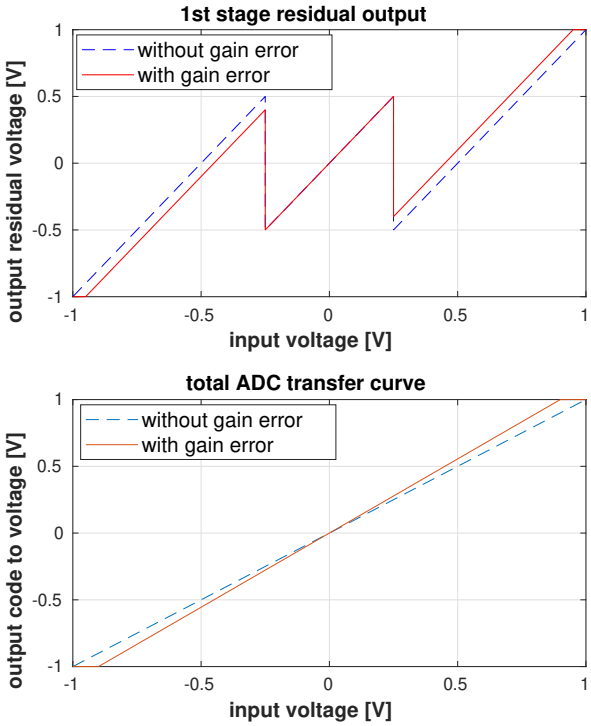


Figure 2.19: Transfer curve with DAC gain error

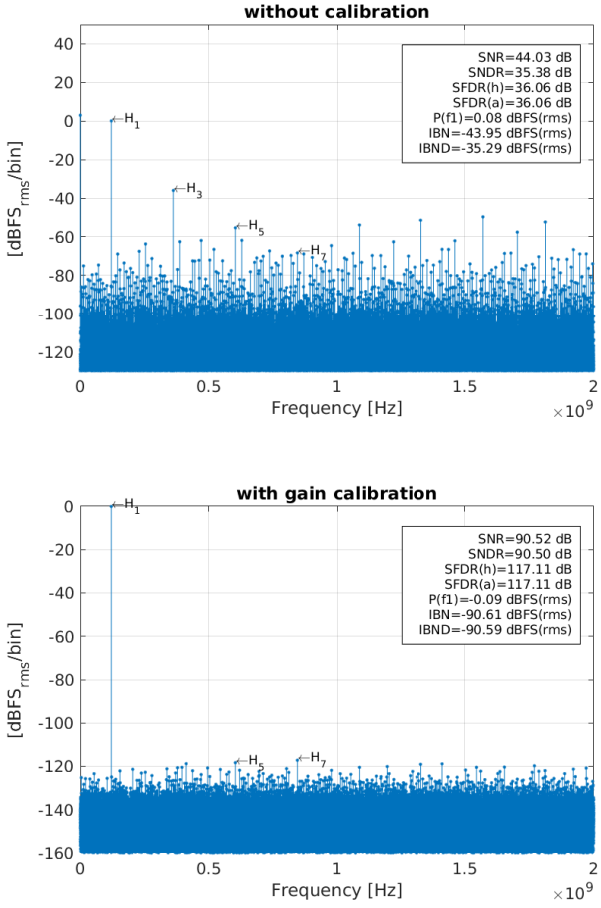


Figure 2.20: Spectrum with DAC gain error

Chapter 3

Pipeline ADC Analysis: Digital Part

3.1 Pipeline Correction Modeling

Figure 3.1 shows the modeling of the pipeline ADC stage, its analog part and corresponding digital correction part including memory effect and high order nonlinearity. In this case, we use a 3rd order nonlinearity as an example.

The memory effect is modeled as an Infinite Impulse Response (IIR) filter in the time-discrete form, and the nonlinearity is modeled directly using a forward polynomial for the residue output. The respective analog and digital part is regarded as a cascade of a linear and a nonlinear system.

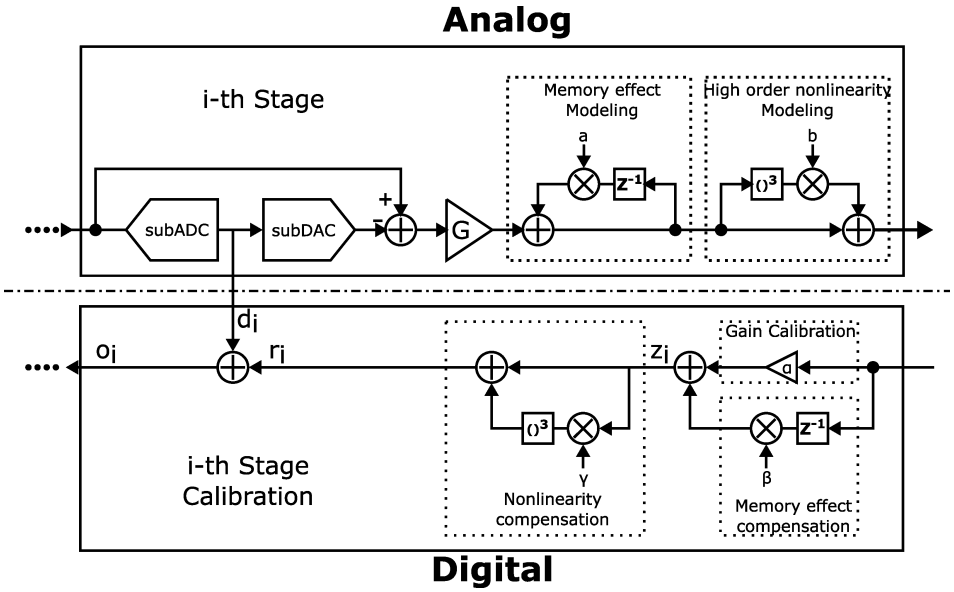


Figure 3.1: Model of pipeline ADC stage and associated correction

The equation below formulate the background correction architecture for each stage, and the meaning of different variables is indicated in table 3.1. In (3.4), o_1 is the digital output of the first stage correction result, and α_0 is the factor to

scale the whole digital range.

$$\begin{cases} o_i(n) = d_i(n) + r_i(n) & (3.1) \\ r_i(n) = z_i(n) + \gamma_i z_i(n)^3 & (3.2) \\ z_i(n) = \alpha_i o_{i+1}(n) + \beta_i o_{i+1}(n-1) & (3.3) \\ y(n) = \alpha_0 o_1(n) & (3.4) \end{cases}$$

symbol	meaning
i	index of i-th stage
n	index of n-th data sample
α	estimation of stage gain
β	correction of memory effect
γ	correction of amplifier nonlinearity
d	subADC digital output
r	residue value recovered in digital domain
z	linear corrected output
o	corrected value of each stage

Table 3.1: meaning of symbols in equation

3.2 LMS Algorithm

3.2.1 LMS Basics

The LMS engine is a commonly used method to adjust filter weights in the real-time scenarios, like identification or correction[12]. We take the correction architecture shown in figure 3.2 as an example. $H_1(s)$ is an unknown system that will distort the input signal $d(n)$ to some extent. We use another filter, $H_2(s)$, to recover the distorted signal $u(n)$ until the average of error between them is minimal. In our case, the unknown system is the ADC with linear and nonlinear transfer functions, while correction filter represents the background correction equations which are listed above, the LMS engine is used to update the parameters in the equation.

Assuming $H_1(s)$ is an FIR filter with M taps, and $H_2(s)$ an FIR filter with N taps

$$u(n) = \sum_{k=0}^{M-1} w_{1,k} d(n-k) \tag{3.5}$$

$$y(n) = \sum_{k=0}^{N-1} w_{2,k} u(n-k) \tag{3.6}$$

The goal is to have minimal expectation of error power

$$min: \ E[e(n)^2] = min: \ E[(d(n) - y(n))^2] \tag{3.7}$$

where e(n) is the error difference between desired reference signal d(n) and output of the final processing y(n)

The way to get the minimal value is by the steepest descent method, and to track the optimal value in the real case, we have the following equation, which is the main body of the LMS algorithm. For detailed derivation, refer to [12].

$$\frac{\partial e^2}{\partial w_{2,k}} = 2e\frac{\partial y}{\partial w_{2,k}} \tag{3.8}$$

$$w_{2,k}(n) = w_{2,k}(n-1) + \mu\frac{\partial y}{\partial w_{2,k}}e(n) \tag{3.9}$$

where μ is the parameter to configure the step size for each update. Using equation (3.6) to derive $\frac{\partial y}{\partial w_{2,k}}$ in equation 3.9 yields

$$w_{2,k}(n) = w_{2,k}(n-1) + \mu u(n-k)e(n) \tag{3.10}$$

This is the basic LMS algorithm. From the discussion above, the key part of the LMS algorithm is to get the proper partial derivation for each parameter (e.g. $\frac{\partial y}{\partial w_{2,k}}$). In the FIR case, it is obviously the corresponding delayed input from correction filter, which is $u(n-k)$.

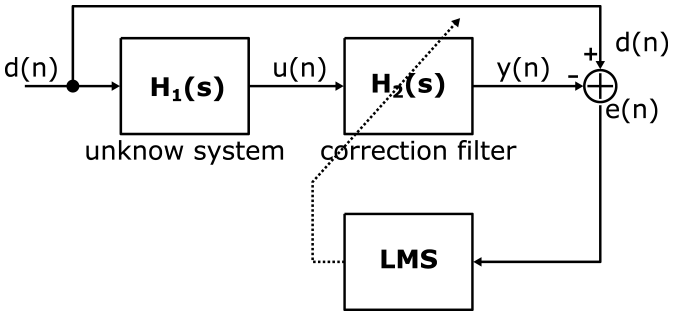


Figure 3.2: Correction architecture for adaptive filter

3.2.2 Algorithm Design

We apply the above theory in the correction model described in (3.1)-(3.4) following (3.9) to estimate α_i, β_i, γ_i in the model

$$\begin{cases} e(n) = u(n) - y(n) & (3.11) \\ \alpha_i(n) = \alpha_i(n-1) + \mu\dfrac{\partial y}{\partial \alpha_i}e(n) & (3.12) \\ \beta_i(n) = \beta_i(n-1) + \mu\dfrac{\partial y}{\partial \beta_i}e(n) & (3.13) \\ \gamma_i(n) = \gamma_i(n-1) + \mu\dfrac{\partial y}{\partial \gamma_i}e(n) & (3.14) \end{cases}$$

The difficult part is to get corresponding partial derivative, for instance $\frac{\partial y}{\partial \alpha_i}$. When we turn to figure 3.1, we realize that each stage is a double input and signal output function, as $o_i = f(d_i, o_{i+1})$. If we take α factor of stage i as an example. It has the "flow" shown as figure 3.3. We apply chain rule to calculate each derivative

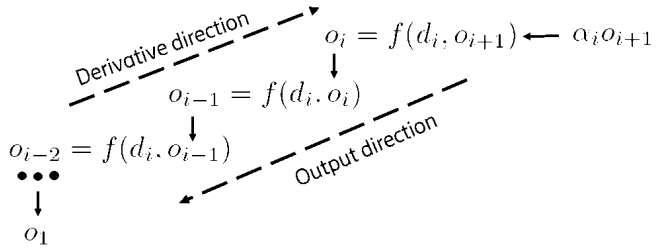


Figure 3.3: Derivative calculation

$$\begin{cases} \dfrac{\partial y}{\partial \alpha_i} = \dfrac{\partial y}{\partial o_1}\dfrac{\partial o_1}{\partial o_i}\dfrac{\partial o_i}{\partial z_i}\dfrac{\partial z_i}{\partial \alpha_i} & (3.15) \\ \dfrac{\partial y}{\partial \beta_i} = \dfrac{\partial y}{\partial o_1}\dfrac{\partial o_1}{\partial o_i}\dfrac{\partial o_i}{\partial z_i}\dfrac{\partial z_i}{\partial \beta_i} & (3.16) \\ \dfrac{\partial y}{\partial \gamma_i} = \dfrac{\partial y}{\partial o_1}\dfrac{\partial o_1}{\partial o_i}\dfrac{\partial o_i}{\partial \gamma_i} & (3.17) \end{cases}$$

$$\frac{\partial y}{\partial \alpha_i} = \frac{\partial y}{\partial o_1}\frac{\partial o_1}{\partial o_2}\frac{\partial o_2}{\partial o_3}\cdots\frac{\partial o_{i-1}}{\partial o_i}\frac{\partial o_i}{\partial \alpha_i} \tag{3.18}$$

Different parts in equation (3.15)-(3.17) are calculated as the following

$$\frac{\partial o_1}{\partial o_i} = \frac{\partial o_1}{\partial o_{i-1}}\frac{\partial o_{i-1}}{\partial o_i} \tag{3.19}$$

$$\frac{\partial y}{\partial o_1} = \alpha_0 \tag{3.20}$$

$$\begin{cases} \dfrac{\partial o_i}{\partial z_i} = 1 + 3\gamma_i z_i(n)^2 \approx 1 & (3.21) \\ \dfrac{\partial z_i}{\partial \alpha_i} = o_{i+1}(n) & (3.22) \\ \dfrac{\partial z_i}{\partial \beta_i} = o_{i+1}(n-1) & (3.23) \\ \dfrac{\partial z_i}{\partial \gamma_i} = z_i(n)^3 & (3.24) \\ \dfrac{\partial o_{i-1}}{\partial o_i} = \dfrac{\partial o_{i-1}}{\partial z_{i-1}}\dfrac{\partial z_{i-1}}{\partial o_i} = \alpha_{i-1}\left[1 + 3\gamma_{i-1} z_{i-1}(n)^2\right] \approx \alpha_{i-1} & (3.25) \end{cases}$$

Approximation in equation (3.21) and (3.25) is useful to reduce the computation complexity and is reasonable since higher order nonlinearity terms are much smaller than the lower order dittos.

Applying calculation (3.19)-(3.25) to the basic LMS equation (3.11)-(3.14), we finally derive the update equation for each parameter

$$\alpha_0(n) = \alpha_0(n-1) + \mu o_1(n)e(n) \tag{3.26}$$

$$\alpha_i(n) = \alpha_i(n-1) + \mu o_{i+1}(n)e(n)\prod_{k=0}^{i-1}\alpha_k \tag{3.27}$$

$$\beta_i(n) = \beta_i(n-1) + \mu o_{i+1}(n-1)e(n)\prod_{k=0}^{i-1}\alpha_k \tag{3.28}$$

$$\gamma_i(n) = \gamma_i(n-1) + \mu z_i(n)^3 e(n)\prod_{k=0}^{i-1}\alpha_k \tag{3.29}$$

3.2.3 Algorithm Test

A 10bit pipeline ADC model with identical stages including mentioned gain, memory, and nonlinearity effects is used to verify the algorithm. The input is a single tone sine waveform, and output corrected data is valid after the algorithm converges.

The parameters for ADC model of each stage are gain G=1.92, memory effect coefficient a=0.01, and 3rd order coefficient b=-0.04. The 3rd order coefficient corresponds to 40 dB HD3 in an amplifier, which is reasonable for an open-loop amplifier. The test is a proof of the theoretical feasibility of the algorithm. We mainly focus on its convergence and performance improvement. The 3rd order correction is only applied for the first two stages. The μ factor for different stages is set differently to boost the convergence speed. The first stages have a small μ while the later stages have a large one. The result is shown in figure 3.4. Compared to linear correction, correction for memory effects improve SFDR by 19 dB. The nonlinearity correction further improves SFDR by at least 6 dB, pushing the spurs to below the noise floor.

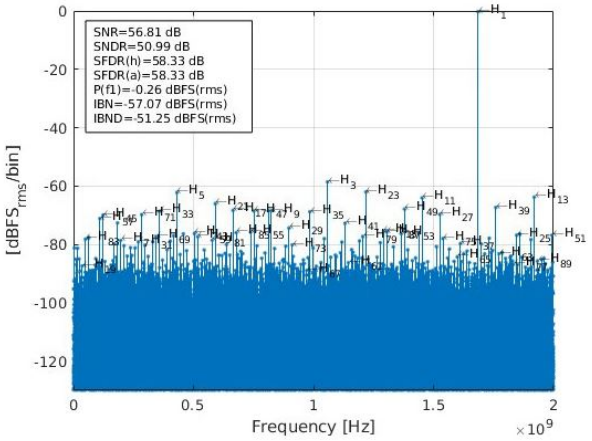
3.3 LMS Implementation

To implement the algorithm in hardware and make it efficient, we have to follow several steps :

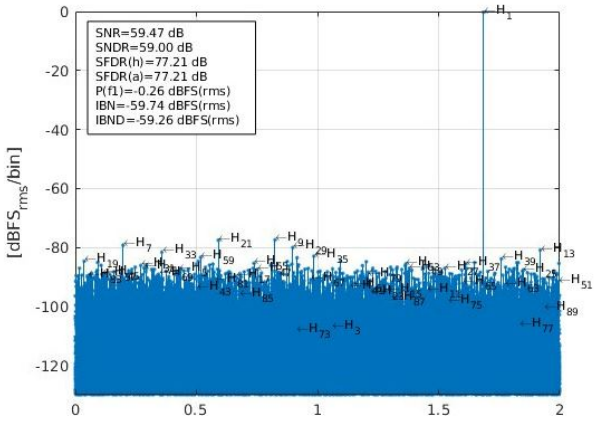
- convert normal LMS algorithm to Sign-Sign LMS.
- replace coefficients which are constant and inactive to change with binary coded coefficients.
- use fixed points representation.

We apply the above steps to equation (3.27)-(3.29). For sign-sign LMS, we replace $e(n)$ with $sign(e(n))$ and $o_{i+1}(n)$ with $sign(o_{i+1}(n))$, which reduces computation complexity by avoiding fixed point number multiplication. $\prod_{k=0}^{i-1}\alpha_k$ is merged into step factor μ

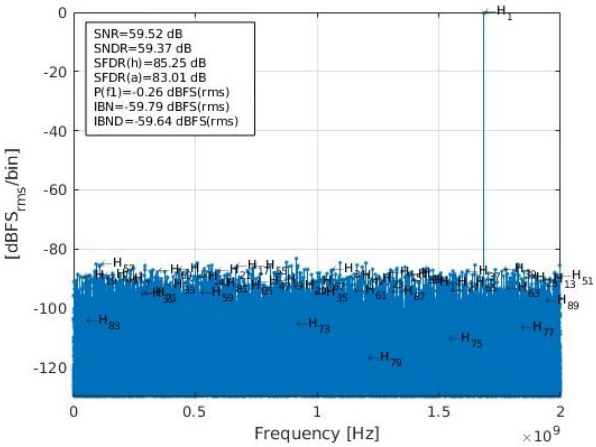
$$\alpha_i(n) = \alpha_i(n-1) + \mu_1 \cdot sign(o_{i+1}(n)) \cdot sign(e(n)) \tag{3.30}$$



(a) with gain correction only



(b) gain and memory correction



(c) gain memory and nonlinearity correction

Figure 3.4: Pipeline ADC corrected with LMS algorithm

$$\beta_i(n) = \beta_i(n-1) + \mu_2 \cdot sign(o_{i+1}(n-1)) \cdot sign(e(n)) \tag{3.31}$$

$$\gamma_i(n) = \gamma_i(n-1) + \mu_3 z_i(n)^3 sign(e(n)) \tag{3.32}$$

Since the γ parameter update equation is not linear, we apply separate Sign LMS to γ instead of Sign-Sign LMS. What remains to be done is to determine the bits for parameters and for every calculation node. The precision increases after we introduce memory correction and nonlinearity correction, which in turn requires more bits compared to gain only case. We will reveal different hardware usage for different cases.

3.4 Modeling Results

3.4.1 case with gain correction only

Figure 3.5 is the diagram of the digital implementation for gain correction only. In the figure, the error signal e is derived by $d(n) - y(n)$ as shown in 3.2. In this case, the desired signal $d(n)$ is from the 10bit reference ADC, which is also the setting for the following simulation. The final output $y(n)$ is shown in (3.4). We configure the gain estimation α variable to be a 16-bit fractional number. 16bit resolution comes from the basic requirement of SFDR vs RA gain error shown in figure 2.9. In order to have larger than 70dB SFDR, we need at least 0.001 precision, which is equal to 10bit. Then we need finer resolution for sign-sign LMS. By trial and error, we determine 10+6 = 16 bit for α factor. Generally, α is around 0.5. The first stage increment μ_1 is $\frac{1}{2^{16}}$, while the second stage is $\frac{1}{2^{15}}$. $\mu_1 = \frac{1}{2^{16}}$ in stage 1 is the finest LSB step in 16bit. The remaining stage μ_1 parameters will be scaled up following the same rule. The reason is that back stages do not require high accuracy and large step size improves the convergence speed. A test result is shown in figure 3.6 which has good SFDR performance.

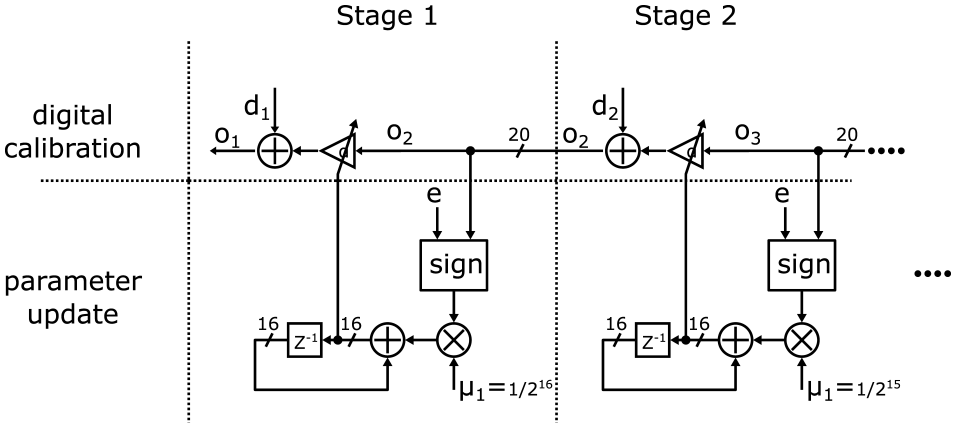


Figure 3.5: Digital implementation for gain correction only

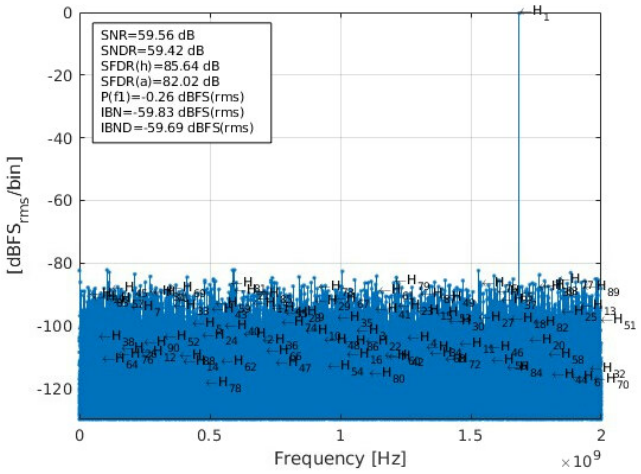


Figure 3.6: Simulation result for gain correction only

3.4.2 with gain and memory correction

When we introduce memory correction, we may need more bits to cover the finer precision. Figure 3.7 shows the resulting digital implementation. We increase node bits from 20 to 24 bits, and memory effect parameter β is an 18-bit fractional number. A test result with the same configuration as section 3.2.3 excluding nonlinear modeling is shown in figure 3.8. It shows SFDR performance of more than 80 dB.

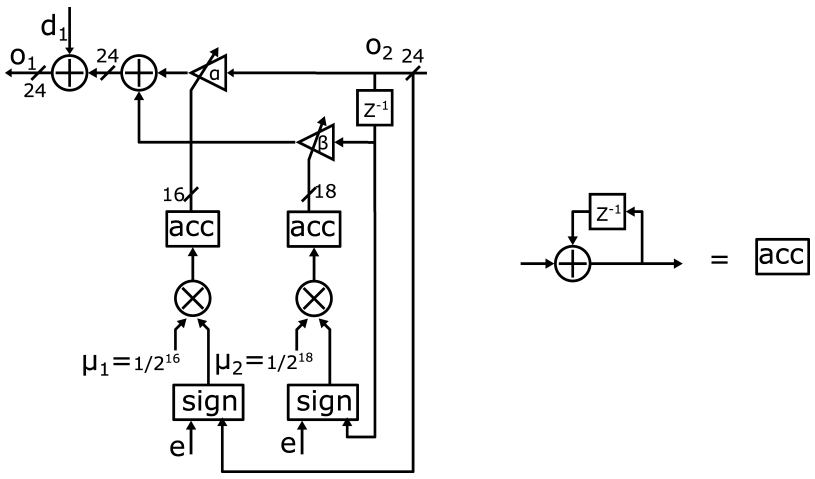


Figure 3.7: Digital implementation for each stage of gain and memory correction

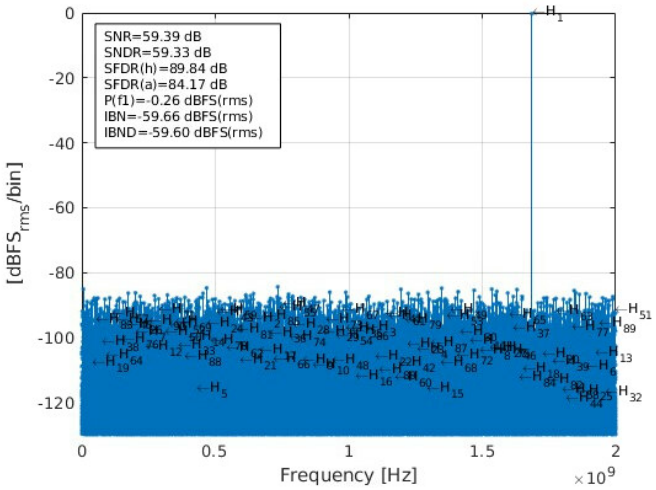


Figure 3.8: Simulation result for gain and memory correction

3.4.3 with gain memory and nonlinearity correction

We introduce nonlinearity correction using sign-LMS instead of sign-sign LMS. Since cubic operation will expand data bits a lot. Then we need to think the really necessary bit for nonlinearity correction. We get 13bit setting by simulation trial and error. We test the performance while decreasing the number of bit until it becomes undesired. The corresponding simulation result is shown in figure 3.10. With nonlinearity appearing in analog model, we finally get 82.8 dB SFDR.

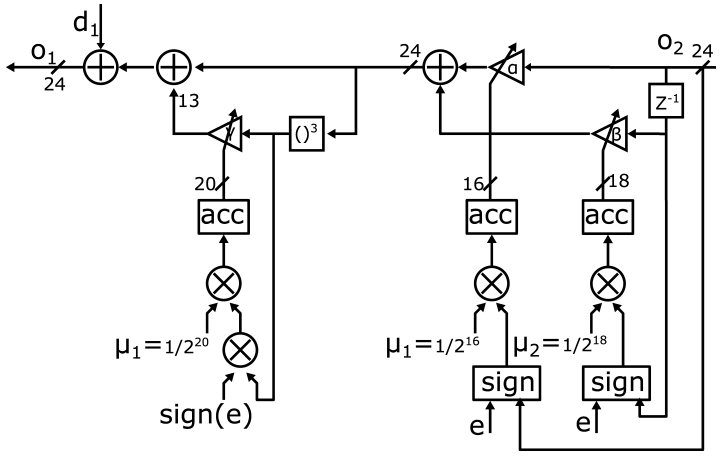


Figure 3.9: Digital implementation for each stage of gain memory and nonlinearity correction

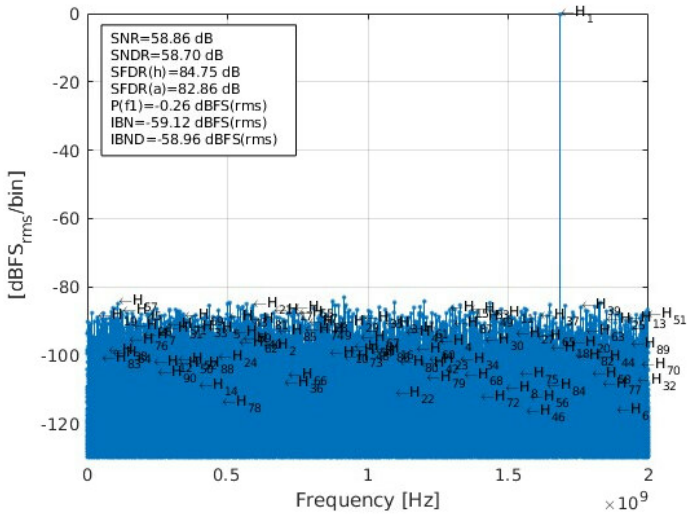


Figure 3.10: Simulation result for gain memory and nonlinearity correction

Chapter 4
Design

This chapter describes the design, operation, and dimensioning aspects of the various building blocks used in the pipeline ADC. Throughout the design phase, the supply voltage is chosen 750mV and the common-mode voltage is mid supply, e.g. 375mV.

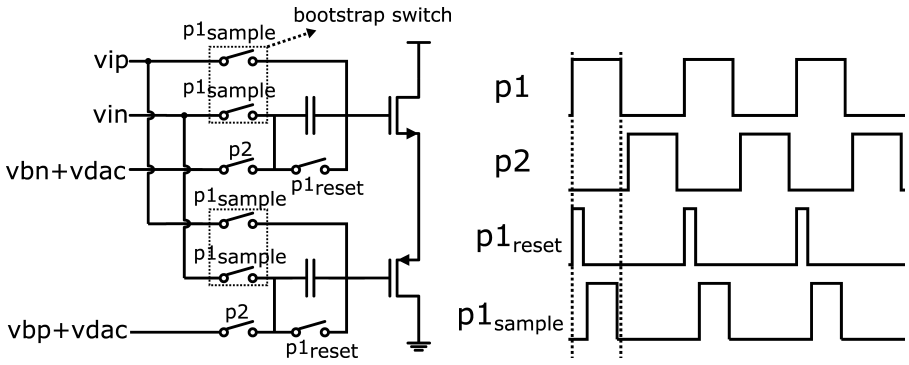
4.1 Residual Amplifier

4.1.1 Functionality

The circuit shown in figure 4.1 resembles the residue amplifier. P1 and P2 are two non-overlapping clock phases. During the P1 phase, the differential input signal is sampled and stored on respective capacitor. During the P2 phase, the signal is buffered to the output by a push-pull source follower converting the differential input to a single-ended output. The complementary source follower has bias voltages vbn and vbp. As will be described later, the way by which the bias is applied allows also for the DAC output to be applied such that the residue will be formed before amplification. Some properties of the amplifier circuit are listed as the following

- The switches controlled by P1 are bootstrap switches since they will forward the signal. Signal dependent resistance of the switch will introduce undesired nonlinearity.
- P1 reset is a short pulse at the beginning of P1 phase to quickly reset the capacitor’s voltage. It is beneficial for large signal settling especially when the signal toggles between extremes of its range, which is the case when the signal is close to the Nyquist frequency.
- The circuit has no static power consumption during the P1 phase, since two gates of the source follower are connected together to vip and there is not enough voltage drop between gate and source to turn on the transistor.
- The voltage difference between vbn and vbp determines the over-drive voltage of mos transistors. The larger (vbn-vbp) value, the larger over-drive voltage, and thus the larger gm and bandwidth for the source follower.

- With a large (vbn-vbp) value, the available signal swing shrinks since the maximum gate voltage sets a hard limit. It is the trade-off between speed and signal swing.



(a) amplifier x1 **(b)** timing

Figure 4.1: diagram of amplifier

Conceptually, the residual amplifier is supposed to have the signal operation expressed as equation 4.1

$$\begin{aligned} V_{residue} &= 2V_{signal} + V_{DAC},\ V_{DAC} \in \{-V_{ref}, 0, +V_{ref}\} \\ &= (V_{signal} + \frac{1}{2}V_{DAC}) - (-V_{signal} - \frac{1}{2}V_{DAC}) \end{aligned} \tag{4.1}$$

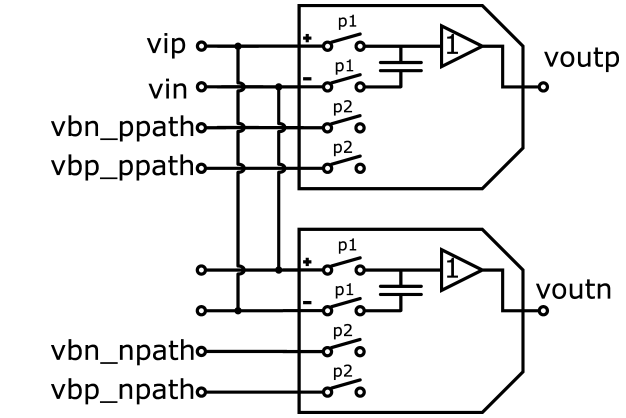
where V_{signal} and V_{DAC} stands for effective signal voltage and DAC output voltage respectively, both are differential. The following will describe how this equation inspires the design of the residual amplifier.

The circuit discussed above has unity signal gain. To achieve amplification by two, we propose to have another replica circuit but sample the signal with inverse sign, so that two single-ended outputs form a differential output signal with amplification. The corresponding connection is shown in figure 4.2a.

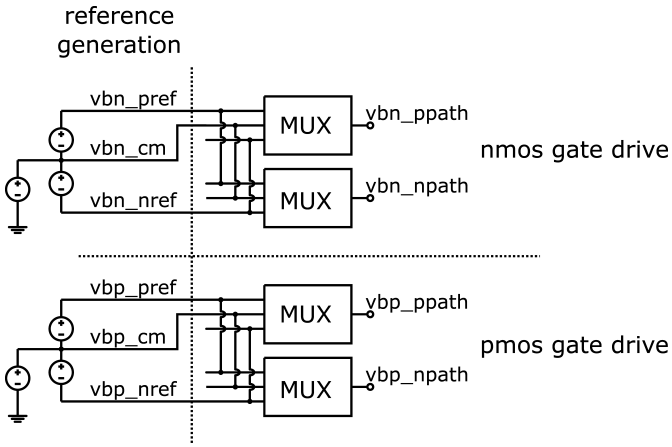
In figure 4.2a, two identical sub-amplifiers form two paths, and each has an individual biasing voltage, named vbx_zpath,in which x is replaced by p or n, representing pmos or nmos and z is replaced by p or n, representing the positive or negative path. In addition, vbn_ppath is the same termination of the one labeled as "vbn+vdac" in figure 4.1.

To introduce the DAC signal, we will need to shift the biasing voltages according to the DAC output. But they are shifted toward different directions. The corresponding scheme is shown in figure 4.2b. For reference generation, the desired DAC shifting is applied on top of the original biasing. Table 4.1 lists how selection is done with multiplexers.

Here is an example to explain the operation. Assuming V_{signal} = +140mV and its reference range is between -200mV to 200mV, V_{signal} is qualified as "1" and encoded with "10". The sub-amplifiers will sample +140mV and -140mV respectively. The vbn voltage for positive path will shift 100mV downwards, and 100mV upwards for the other path. The final output is (140mV-100mV)-(-140mV+100mV)=80mV.



(a) residual amplifier



(b) biasing

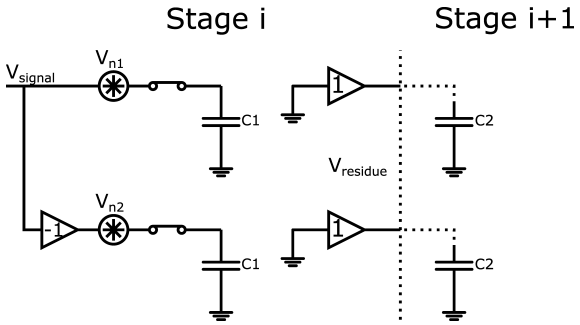
Figure 4.2: Diagram of residual amplifier

ADC code	00	01	10
code meaning	-1	0	+1
vbn_ppath	vbn_nref	vbn_cm	vbn_pref
vbp_ppath	vbp_nref	vbp_cm	vbp_pref
vbn_npath	vbn_pref	vbn_cm	vbn_nref
vbp_npath	vbp_pref	vbp_cm	vbp_nref

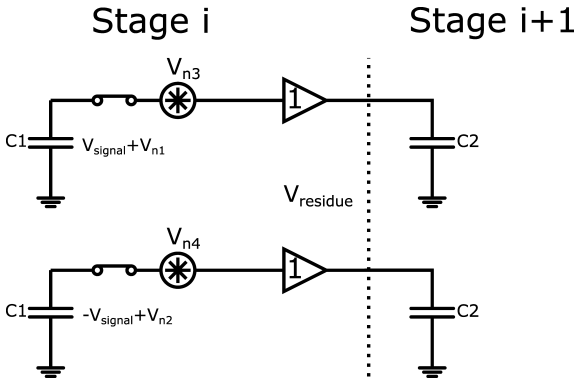
Table 4.1: Gate voltage selection

4.1.2 Noise

The noise analysis is divided into two different phases. Figure 4.3 shows the equivalent model for respective clock phase.



(a) p1



(b) p2

Figure 4.3: noise equivalent circuit

During phase P1, the signal and noise from switches are sampled to the C_1. C_1 is the total sampling capacitor in each source follower unit. For two source follower units in MDAC, the total noise is

$$v_{n,p1}^2 = 2\frac{kT}{C_1} \tag{4.2}$$

Please note that signal has been converted to in phase and reverse phase forms at this time, which means amplified. So that the equivalent input refer noise is

$$v_{n,p1,input-refer}^2 = \frac{kT}{2C_1} \tag{4.3}$$

Equation (4.3) tells that total capacitance $2C_1$ contributes the noise in a simple way, which meets a general intuitive expectation.

During phase p2, the signal and stored noise from the p1 phase are directly delivered to the output. The source follower input referred noise is delivered to

output by a first order network which is constructed by the source follower and next stage sampling capacitor. Actually, p2 is the same behavior with p1, and this sampling noise in p2 should be categorized into second stage when considering second stage input refer noise.

$$v_{n,p2}^2 = 2\int_0^\infty \frac{4kT\gamma}{g_m}\left|\frac{1}{1+\frac{j\omega}{g_m/C_2}}\right|^2 d\omega = 2\frac{kT\gamma}{C_2} \tag{4.4}$$

Finally, each stage of its input refer noise is basically simple form of equation (4.3). We can approximately ignore γ factor in (4.4) when amplifier noise is not dominant. This means that bandwidth is mainly determined by switch resistance instead of amplifier g_m.

4.1.3 Gain

In the residual amplifier, the signal and the DAC output have different signal paths. The signal only passes the push pull source follower, while the DAC output passes the sampling capacitor first, which forms attenuation together with source follower input equivalent capacitance, and then passes through the push pull source follower.

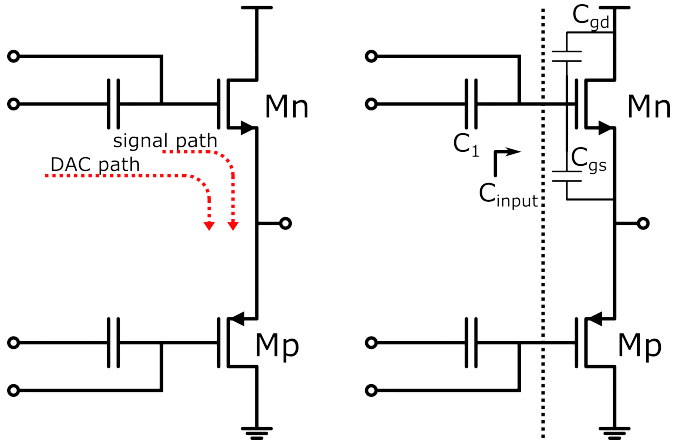


Figure 4.4: diagram of gain for different path

The gain of the source follower is limited to be less than 1.

$$G_{signal} = G_{source\ follower} = \frac{(g_{mn}+g_{mp})(r_n||r_p)}{1+(g_{mn}+g_{mp})(r_n||r_p)} \approx 1 \tag{4.5}$$

neglecting the body effect in the analysis.

The input equivalent capacitance is contributed by C_{gs} and C_{gd}. For C_{gs}, its equivalence can be treated using miller theorem

$$C_{input\ equivalent} = C_{gd} + (1 - G_{source\ follower})C_{gs} \approx C_{gd} \tag{4.6}$$

From another intuitive view to understand the result, C_{gs} is bootstrapped, e.g. the source node follows the gate node. Thus, the voltage across the capacitor is kept to be almost the same, so looking into the gate, we almost cannot "see" this capacitor in the dynamic behavior.

The gain for DAC path is given by

$$\begin{aligned} G_{DAC} &= \frac{C_{sampling}}{C_{sampling} + C_{input\ equivalent}} * G_{source\ follower} \\ &= \frac{C_1}{C_{gd} + C_1} * G_{source\ follower} \end{aligned} \tag{4.7}$$

4.2 Bootstrap Switch

4.2.1 Function

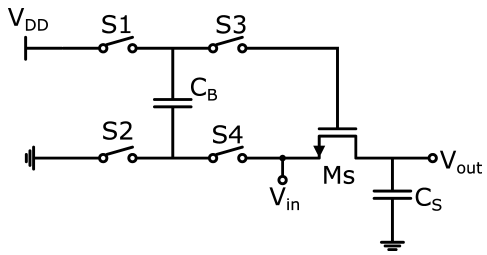
The bootstrap switch used in the amplifier is shown in figure 4.5. Figure 4.5a is its functional equivalent. The circuit operates in two different phases, pre-charge and track phase. When in pre-charge phase, only S1 and S2 switches are closed, C_B would be charged to have the voltage of V_{DD}. When S1, S2 are open and S3, S4 are closed, M_S is turned on with constant gate-to-source voltage, and V_{out} will track V_{in}.

In figure 4.5b, M1-4 are transistors representing switches of S1-4. When en is low and enb is high, the circuit is in the pre-charge phase and the opposite occurs for the track phase. During the pre-charge phase, the path formed by M8 and M9 pulls the X node to ground to secure that the main switch M_S is open. Further, M1 and M2 are turned on and M5 is turned on to drive the M3 gate to VDD which turns off M3. M4, M6-7 and M_S are turned off. During the track phase, M6 is turned on, initially, then M3 is turned on because of presented voltage difference between gate and source stored on C_B. The X node becomes high enough to turn on M4 and M7. M4 passes a voltage to node N. M7 is larger than M6 to provide a low impedance path, which further boosts the above settling processing for floating node P, N, and X.

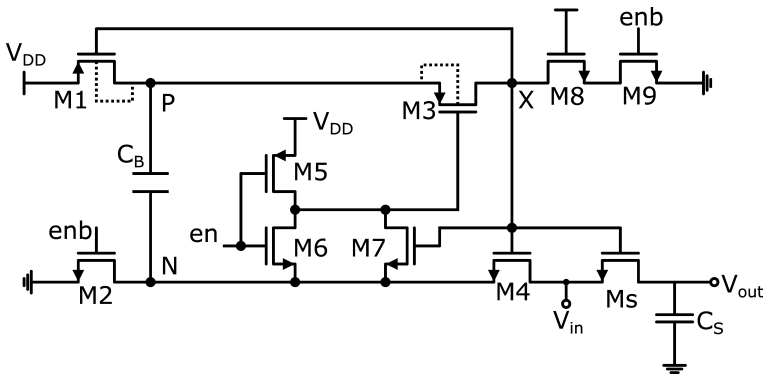
Some connections in the schematic may not be obvious to the reader. For example, the gate of M1 cannot be directly connected to enb. During the track phase, as node P is lifted to $V_{DD} + V_{in}$ it could turn on M1 because of the large source-to-gate voltage that would result from the gate of M1 being connected to enb. Further, the local body connection of M1 and M3 is to node P, since it is the highest voltage in the track phase. Otherwise, with M1 bulk connected to V_{DD}, the source-to-bulk diode junction could be forward-biased. Finally, node X is higher than V_{DD} in the track phase, so M8 is inserted to avoid excess drain-to-source voltage for M9.

4.3 1.5bit ADC

The 1.5bit ADC schematic is shown in figure 4.6. The circuit consists of two main comparators and a code conversion block by AND and XOR gates.



(a) simple diagram of bootstrap switch



(b) circuit implementation

Figure 4.5: bootstrap switch

Both the signal input and reference input are differential. Comparator 1 generates the decision of logic test $if\ (vip - vin) > (vthp - vthn)$ and Comparator 0 tests $if\ (vip - vin) > (vthn - vthp)$. The comparator decision and corresponding output code are shown in table 4.2 with respect to the input signal level.

From the basic 1.5bit ADC theory[13], we know that the decision threshold voltage is a quarter of the whole ADC reference range. In this case, the whole reference range is [-200mV, 200mV], so that the threshold voltage is [-50mV, 50mV], which is $\frac{1}{4}$ reference voltage.

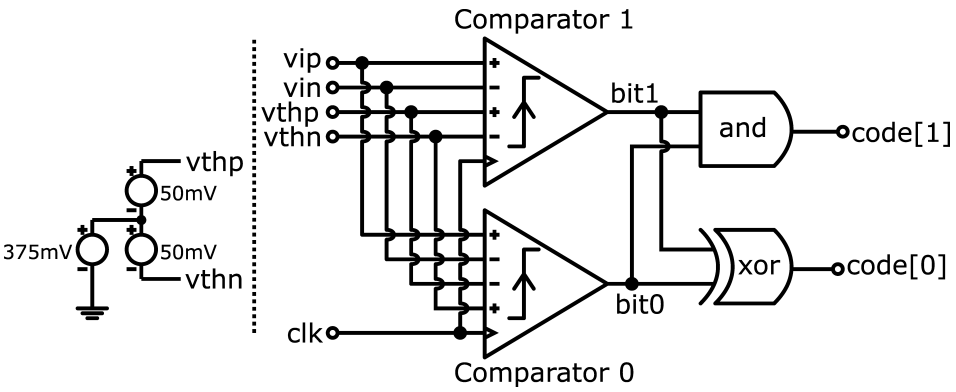


Figure 4.6: 1.5 bit ADC circuit

input differential voltage	bit1 & bit0	code[1:0]
$<V_{refn}$	"00"	"00"
$>V_{refn}$ & $< V_{refp}$	"10"	"01"
$>V_{refp}$	"11"	"10"

Table 4.2: 1.5bit ADC input voltage to output code conversion

4.4 Comparator

The complete comparator circuit used in the 1.5bit ADC is shown in figure 4.7. It is a so called double tail dynamic comparator where dynamic refers to that circuit does not have any static power dissipation. The double tail refers to the two active stages, which benefits from better isolation between the input and large swing output, especially compared to the traditional strongArm comparator, leading to lower kick-back noise. The dual input differential pairs are for differential signal and differential comparison voltage at the same time. The outputs v3p,n are cascaded with an SR-latch to keep the previous decision during reset phase.

The circuit is falling-edge triggered. When clk is high and clkb is low, the circuit is in reset mode. The outputs of the first stage, v1n and v1p, are reset to ground. The second stage is suspended and nodes v2p, v2n, v3p, v3n are set

to supply voltage. When clk toggles from high to low, the clamping from M5-6, M15-18 is released and the amplification and decisioning starts.

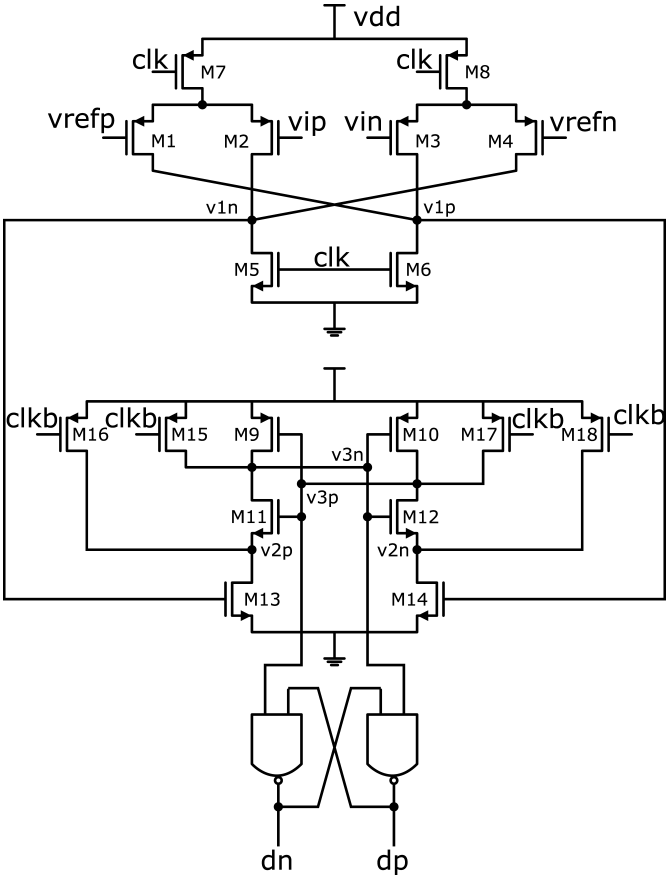


Figure 4.7: comparator circuit

4.5 1.5bit Stage

The stage has the timing shown in figure 4.8. What is special in the proposed timing schedule is the "early decision". During the P1 phase, the signal will gradually settle on the sampling capacitor. At the end of the P1 phase, which is indicated by t2, it is the best timing for the comparator to trigger the decision since the signal is substantially settled. Once the decision is taken and a code is ready, the DAC still has a delay to fully respond to the code, which occupies a part of the P2 period. To fix this problem, we can trigger the decision in advance, moved from t2 to t1.

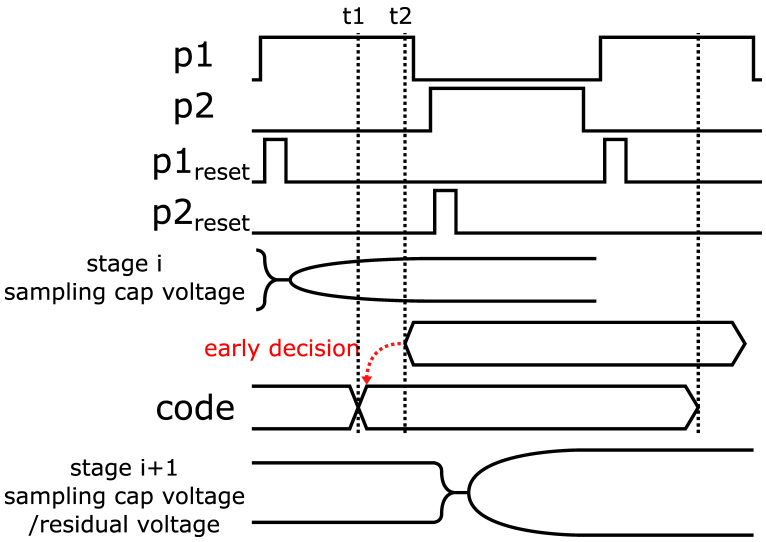


Figure 4.8: timing of the stage

4.6 Top Level

Figure 4.9 illustrates the top-level schematic of the pipeline ADC and the connection between different stages. All stages share the same reference voltages. The p1 and p2 phases are swapped for any two adjacent stages. If stage i is in the phase of the residual voltage generation, the next stage (i+1) is in the sampling phase. To be more power and area efficient, the stages for the less significant bits are scaled down. The scaling down is mainly for sampling capacitor as shown in figure 4.9. To boost the speed, the fisrt stage use a double size amplifier. Because the noise and nonlinearity of the current stage will be attenuated by the previous stages of their gain, the requirement of the current one is relaxed.

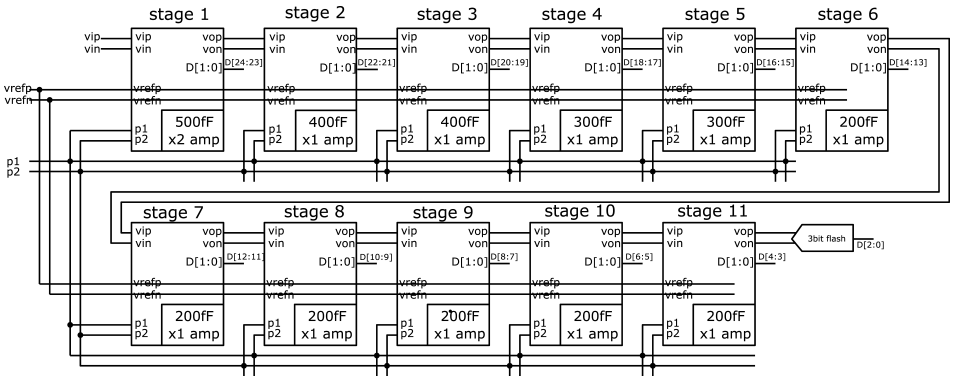


Figure 4.9: ADC core connection with stages

Chapter 5
Simulation Results

This chapter presents various simulation results of the pipeline ADC with frequency and amplitude sweeps, across corners, and the use of different levels of post-correction. It is worth noting that the use of post-correction, as an alternative to analog calibration, is an important prerequisite for the particular implementation chosen in this work.

5.1 Post Processing and Performance Impact

Figure 5.1 shows the principle of how data is corrected and the correction weights are estimated with a known signal. The underlying assumption is that there is a so called reference ADC available with significantly better linearity but that may operate at a lower speed and lower SNR. The correction is basically a problem of linear fitting. For each stage output, it uses 3 parameters for the fitting, with w_1 being the weight for stage output "1", while w_2 is the weight for stage output "-1" while w_3 is the weight for a single tap memory compensation. All the weights are determined by the least square error algorithm.

Figure 5.2-5.6 show the results using different levels of correction to improve the performance when clocked at 4 GHz and with input tone close to Nyquist, using tt corner, 0.75V supply voltage at 80°C. The output spectrum is based on an analog representation of the digital signal and a fullscale tone represents -14dBFS. Figure 5.2 is the result using binary scaled weights for all stages demonstrating very poor performance both with regards to noise and distortion. In figure 5.3, the results are based on using w_1 to correct for inter-stage gain errors, clearly leading to a substantial increase in performance but with a few residual higher order harmonics remaining. SFDR is improved from 24 to 63 dB. Adding the memory tap coefficient w_3 the SFDR improves to more than 77 dB, see Figure 5.4. The method shown in figure 5.5 uses w_1 and w_2 parameters to estimate weights for positive and negative code separately, denoted correction for PN mismatch. In this case, since no mismatch is applied to circuit components and reference voltages are ideal, the estimated w_1 and w_2 are almost identical, and the same performance is achieved as shown in figure 5.3. Figure 5.6 uses all the parameters listed above. Here, all distortion components are below the noise floor of the simulation. The SNDR and SFDR approaches 58.9dB and 77.8dB, respectively. The simulation includes transient noise.

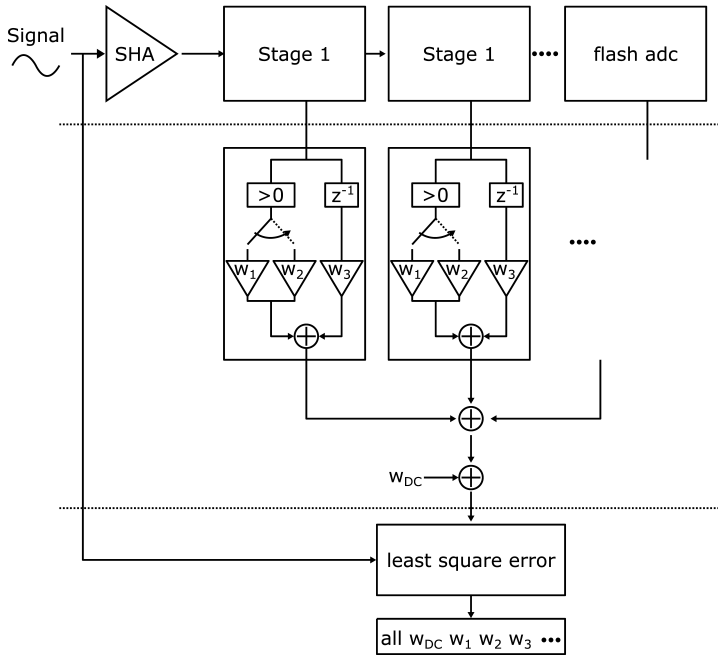


Figure 5.1: post-processing architecture

The same simulation without transient noise shows an SNDR of 69 dB and SFDR 80 dB. Here, SNDR is basically equal to SQNR. The total gain of the pipeline stages is approximately 345, found by examining the weights of the pipeline stages. From (2.1), the SQNR is calculated as 70dB, which matches the simulated SNDR of 69 dB.

5.2 Simulation Sweeps

5.2.1 Performance across Corners

Figure 5.7 and 5.8 show SNDR and SFDR performance, respectively, across PVT using a 4GHz clock, and a 0 dBFS input tone near Nyquist. The result is after applying all the calibration. From the sweep, we find that for different corners, temperature shows largest influence on circuit performance. At the tt corner, the circuit has a good performance down to 0°C. The combination of low supply voltage, low temperature and ss corner leads to very poor performance at this high clock speed.

In the typical corner, the current consumption is 20 mA with a 0.75 V supply, which yields a Schreier FoM of 167.3 dB.

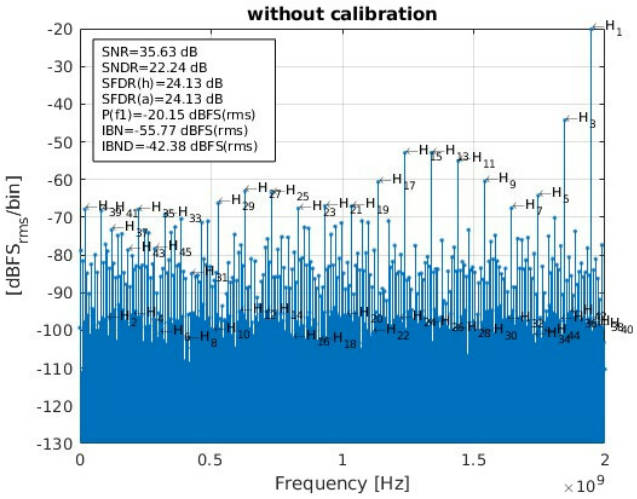


Figure 5.2: ADC output spectrum without any correction

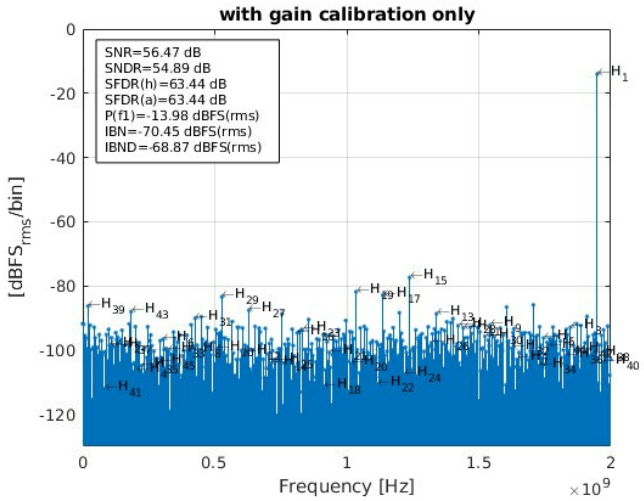


Figure 5.3: ADC output spectrum with gain correction

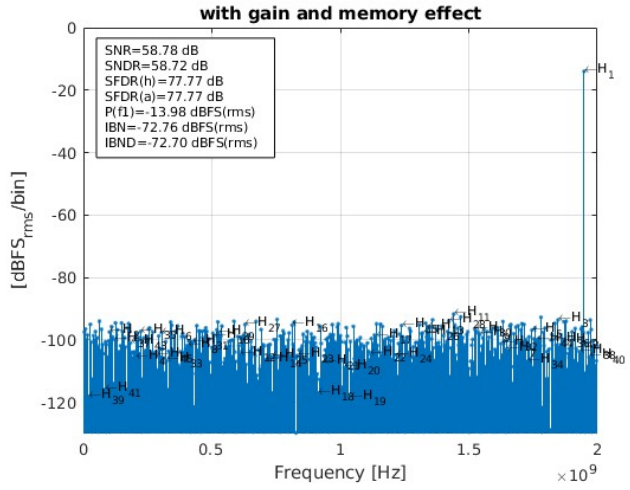


Figure 5.4: ADC output spectrum with gain and memory corrections

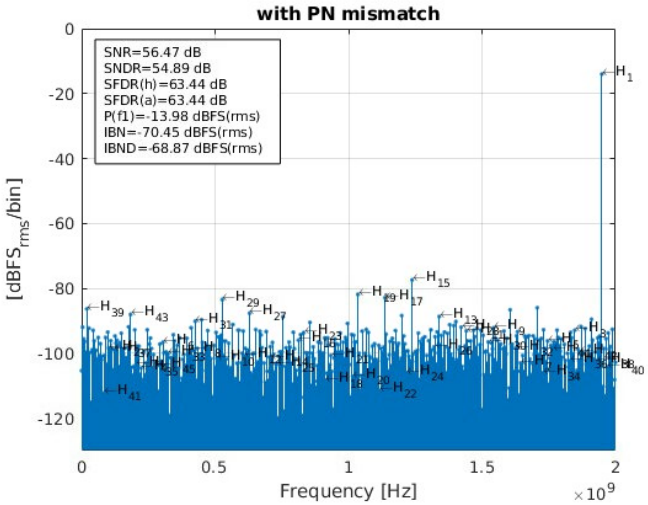


Figure 5.5: ADC output spectrum with PN mismatch corrections

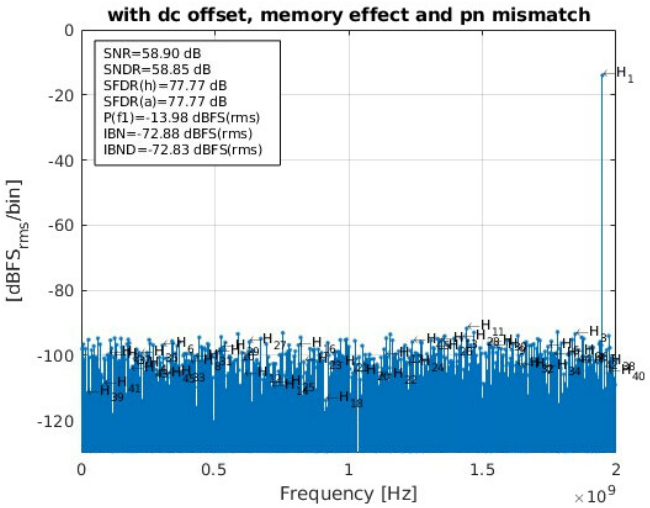


Figure 5.6: ADC output spectrum with all corrections

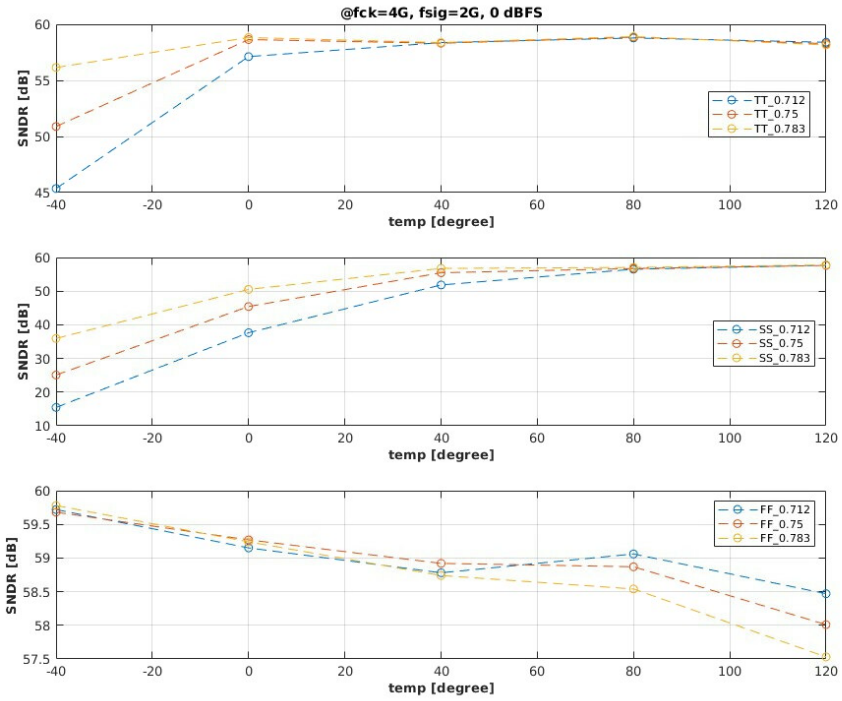


Figure 5.7: SNDR over PVT

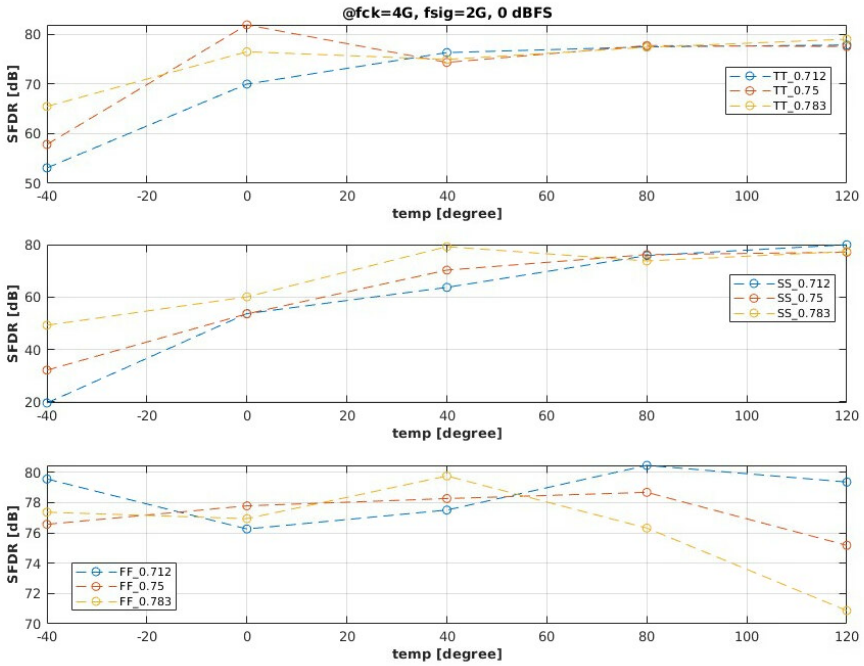


Figure 5.8: SFDR over PVT

5.2.2 Performance vs Clock Frequency

Figure 5.9 and 5.10 shows performance change with respect of clock frequency. We choose 2 corners, which are tt and ss, of their typical and worst case.

5.2.3 Performance vs Signal Level

Figure 5.11 and 5.12 show performance with respect to signal power. The purpose is to test the robustness of post-processing method and see if the algorithm may have issues in finding the correction coefficients with small input signals. However, the plots show a robust and gradual dB-for-dB decrease in SFDR and SNDR versus input signal level.

But there's problem when we check the corresponding weights for -15dBFS and -20dBFS. For -20dBFS case, the first 3 stages have purely "01" code output, and weights for these 3 stages are 0. -15dBFS case has two problematic stage weights. The reason is when signal level is low, for instance limited to one sub-range of the 1.5bit stage transfer curve, then the stage digital output will only show one code. From the LMS algorithm, this behavior will be regarded as "no correlation" with signal, which leads to 0 weight.

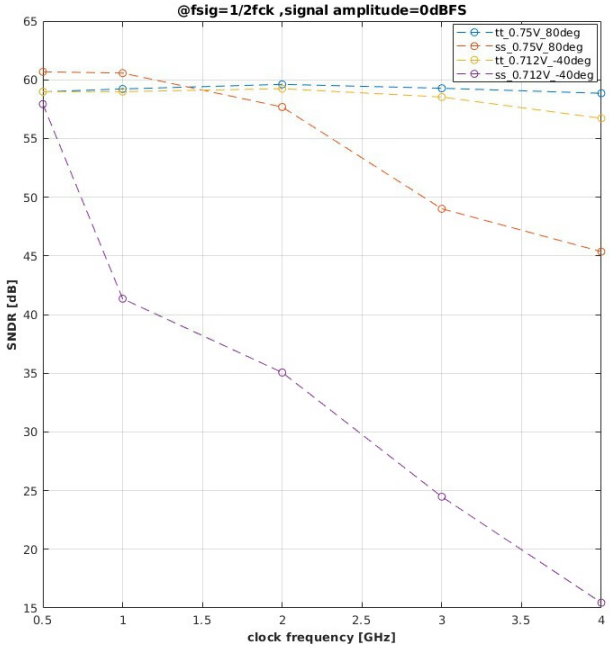


Figure 5.9: SNDR as a function of clock frequency

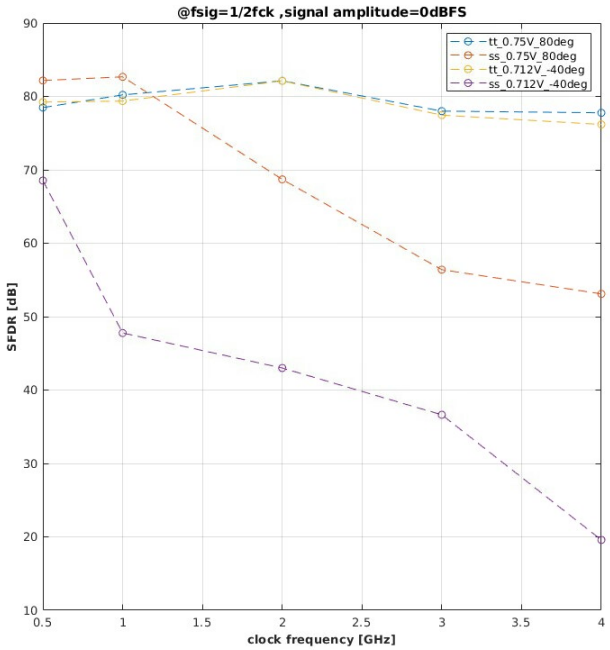


Figure 5.10: SFDR as a function of clock frequency

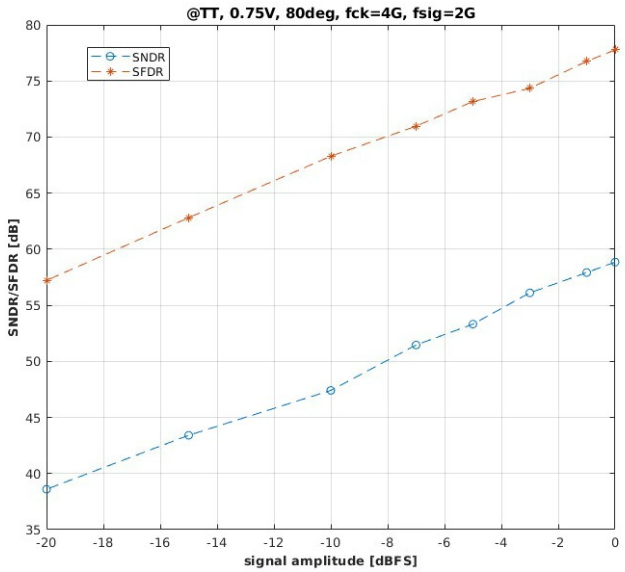


Figure 5.11: SNDR and SFDR as a function of input signal level

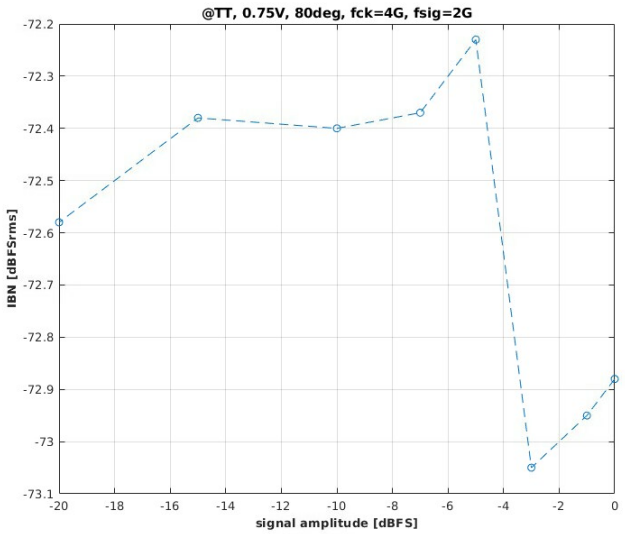


Figure 5.12: IBN as a function of input signal level

5.2.4 Performance vs Signal Frequency

Figure 5.13 shows performance change with respect of signal frequency. The clock is fixed to be 4GHz. The plot shows great uniformity with change of signal frequency.

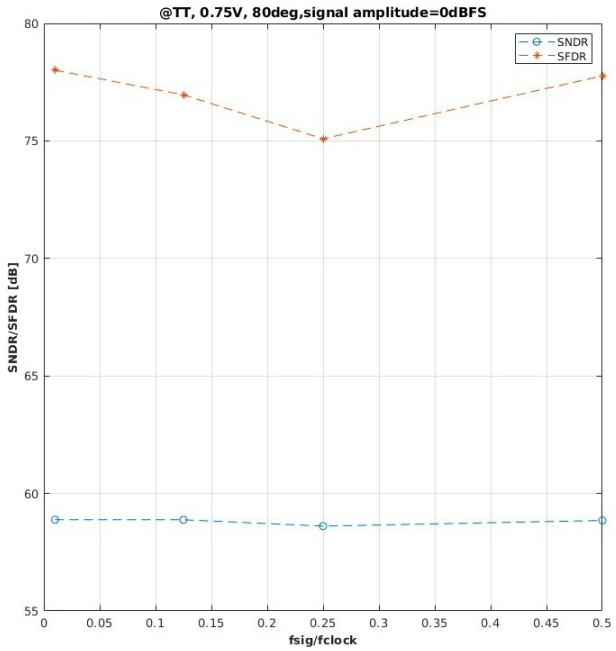


Figure 5.13: SNDR and SFDR as a function of input signal frequency

Chapter 6
Future Work and Conclusion

6.1 Future Work

The design is promising for its good linearity performance. One of the difficulties for real circuit design is the reference voltage generation, which is much more complex than the typical design, since we need double amount of biasing voltages to support the proposed push pull stage. This problem grows further if one would implement multi-bit stages. As a result, it suggested that future work should consider the implementation also of the reference generation. It is also advised to revisit the biasing scheme of the push pull source follower to see if a different method can improve robustness over corners.

In addition, all the post-processing has been done in Matlab. To further assess the efficiency of the proposed ADC, the additional circuit complexity and power consumption associated with the processing should be estimated.

As we mentioned before, weights derived from post-processing method could vary a lot for small and large signal amplitude. In order to get rid of this signal dependency, a dither based correction could be investigated since its estimation is fully based on injected dither signal assuming no correction between dither and input signal.

6.2 Conclusion

In this thesis, a pipeline ADC core has been designed, analyzed and simulated, capable of operating at a speed up to 4 GSps with SFDR in excess of 70 dB with 15 mW of power consumption and a Schreier FoM of 167.3 dB. The ADC uses a source-follower based residual amplifier. The amplification of two comes from processing the signal in two opposite phases and combining them differentially at the output. The equivalent gain factor will deviate from two due to intrinsic gain loss of source follower and charge sharing happening in the DAC path. To overcome this loss, the deviation in weights are corrected using post-processing. It is found that the memory inside each stage has a large impact on linearity. The settling behavior will be impacted by the residue of the previous sample in each stage since it sets the initial start voltage for the settling. To overcome this, we introduced a corresponding circuit implementation and correction method. A

capacitor reset phase is added to the circuit, while the correction uses both the current data and 1-tap delayed data to fit with reference signal. The memory effect correction improves SFDR by around 11 dB at 4 GSps in the typical case.

Bibliography

[1] Franco Maloberti. *Data Converters.* Springer US, 2007.

[2] Chun C. Lee and Michael P. Flynn. "A SAR-Assisted Two-Stage Pipeline ADC". In: *IEEE Journal of Solid-State Circuits* 46.4 (2011), pp. 859–869. DOI: 10.1109/JSSC.2011.2108133.

[3] L. Jie et al. "An Overview of Noise-Shaping SAR ADC: From Fundamentals to the Frontier." In: *IEEE Open Journal of the Solid-State Circuits Society* 1 (2021), pp. 149 –161. ISSN: 2644-1349.

[4] Boris Murmann. *ADC Performance Survey 1997-2023.* [Online]. Available: https://github.com/bmurmann/ADC-survey.

[5] Athanasios. Ramkaj et al. "3.3 A 5GS/s 158.6mW 12b Passive-Sampling 8Œ-Interleaved Hybrid ADC with 9.4 ENOB and 160.5dB FoMS in 28nm CMOS". In: *2019 IEEE International Solid-State Circuits Conference - (ISSCC).* 2019, pp. 62–64. DOI: 10.1109/ISSCC.2019.8662490.

[6] Benjamin. Hershberg et al. "3.1 A 3.2GS/s 10 ENOB 61mW Ringamp ADC in 16nm with Background Monitoring of Distortion". In: *2019 IEEE International Solid-State Circuits Conference - (ISSCC).* 2019, pp. 58–60. DOI: 10.1109/ISSCC.2019.8662290.

[7] Bruno. Vaz et al. "16.1 A 13b 4GS/s digitally assisted dynamic 3-stage asynchronous pipelined-SAR ADC". In: *2017 IEEE International Solid-State Circuits Conference (ISSCC).* 2017, pp. 276–277. DOI: 10.1109/ISSCC.2017.7870368.

[8] Jiangfeng. Wu et al. "27.6 A 4GS/s 13b pipelined ADC with capacitor and amplifier sharing in 16nm CMOS". In: *2016 IEEE International Solid-State Circuits Conference (ISSCC).* 2016, pp. 466–467. DOI: 10.1109/ISSCC.2016.7418109.

[9] Bruno. Vaz et al. "A 13Bit 5GS/S ADC with Time-Interleaved Chopping Calibration in 16NM FinFET". In: *2018 IEEE Symposium on VLSI Circuits*. 2018, pp. 99–100. DOI: 10.1109/VLSIC.2018.8502306.

[10] B. Murmann and B.E. Boser. "A 12-bit 75-MS/s pipelined ADC using open-loop residue amplification". In: *IEEE Journal of Solid-State Circuits* 38.12 (2003), pp. 2040–2050. DOI: 10.1109/JSSC.2003.819167.

[11] Yun Chiu. "Digitally-assisted design of data converters". In: *Digitally-Assisted Analog and Analog-Assisted Digital IC Design*. Ed. by XichengEditor Jiang. Cambridge University Press, 2015, pp. 135173.

[12] Simon S. Haykin. *Adaptive filter theory*. Pearson, 2014. ISBN: 027376408X.

[13] Marcel J.M. Pelgrom. *Analog-to-Digital Conversion*. Springer International Publishing, 2022. ISBN: 9783030908072.

Appendix A

Calibration for DAC gain error

A.1 Linear Model

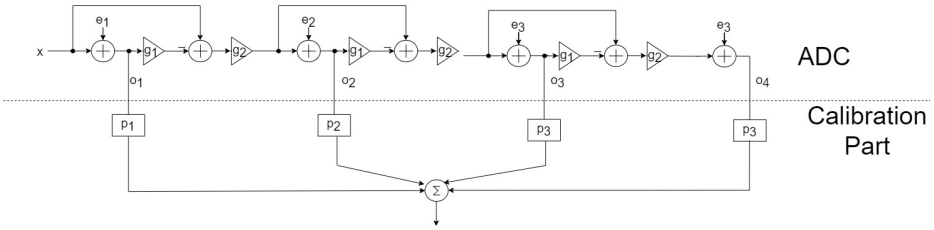


Figure A.1: 4 Stages system model with dac gain error

Figure A.1 is a 4 stage pipeline ADC linear model including residual amplifier and DAC gain error. Meanings of different symbols are:

- x is the input signal
- $e_1, \ldots, e_4$ stand for quantization error source
- $o_1, \ldots, o_4$ is the output of each stage
- $p_1, \ldots, p_4$ is the post-processing weights of each o_n
- g_1 is the DAC gain
- g_2 is the residual amplifier gain

Final output is the linear combination of o_n with its corresponding weight p_n

$$Output = \sum_{n=1}^{4} p_n o_n \tag{A.1}$$

If we donate $\mathbf{P}$ to be $[p_1\ p_2\ p_3\ p_4]$ and $\mathbf{O}$ to be $[o_1\ o_2\ o_3\ o_4]^{\mathbf{T}}$, above equation could be rewritten as

$$Output = \begin{bmatrix} p_1 & p_2 & p_3 & p_4 \end{bmatrix} \begin{bmatrix} o_1 \\ o_2 \\ o_3 \\ o_4 \end{bmatrix} = \mathbf{PO} \tag{A.2}$$

P is the post-processing which conducts in the digital domain and is the part we focus on in this chapter. Output **O** is the superposition of signal response and noise response. In the model, signal works in a signal-input-multiple-output form,and noise works in a multiple-input-multiple-output form. Output can be described as a matrix function

$$\mathbf{O} = \mathbf{N} \cdot \mathbf{E} + \mathbf{S}x \tag{A.3}$$

$$\begin{bmatrix} o_1 \\ o_2 \\ o_3 \\ o_4 \end{bmatrix} = \begin{bmatrix} n_{11} & \dots & n_{14} \\ \vdots & \ddots & \vdots \\ n_{41} & \dots & n_{44} \end{bmatrix} \begin{bmatrix} e_1 \\ e_2 \\ e_3 \\ e_4 \end{bmatrix} + \begin{bmatrix} s_1 \\ s_2 \\ s_3 \\ s_4 \end{bmatrix} x \tag{A.4}$$

in which **N** is the noise transfer matrix, **S** is the signal transfer vector, n_{xy} is the forward gain factor of noise from error source e_y to port o_x, s_x is the gain factor of forwarded signal from source to port o_x. For example, n_{21} stand for path from e_1 to o_2, which is $-g_1g_2$, and s_2 stand for path from signal source to o_2, which is $(1-g_1)g_2$.

Above all is the description of the system, then the calibration method is how to derive a proper **P** while following some constraints. There are two constraints in the system: (1) push the total output noise to be minimum; (2) signal should be fully recovered.

A.1.1 Noise Transfer

Regardless of signal, the output is

$$\begin{bmatrix} o_1 \\ o_2 \\ o_3 \\ o_4 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ -g_1g_2 & 1 & 0 & 0 \\ -g_1g_2(1-g_1)g_2 & -g_1g_2 & 1 & 0 \\ -g_1g_2[(1-g_1)g_2]^2 & -g_1g_2(1-g_1)g_2 & -g_1g_2 & 1 \end{bmatrix} \begin{bmatrix} e_1 \\ e_2 \\ e_3 \\ e_4 \end{bmatrix} \tag{A.5}$$

As an intuitive result, one way to get minimum output noise is to null all noise source expect for e_4, which means after post processing, only a fractional part of e_4 is left.

$$\begin{bmatrix} p_1 & p_2 & p_3 & p_4 \end{bmatrix} \begin{bmatrix} o_1 \\ o_2 \\ o_3 \\ o_4 \end{bmatrix} = Ae_4 = \begin{bmatrix} 0 & 0 & 0 & A \end{bmatrix} \begin{bmatrix} e_1 \\ e_2 \\ e_3 \\ e_4 \end{bmatrix} \tag{A.6}$$

Currently, A is an unknown variable, which would be solved in next section. Then substituting **O** in (A.6) with (A.5)

$$\mathbf{P} \cdot \mathbf{N} \cdot \mathbf{E} = A \begin{bmatrix} 0 & 0 & 0 & 1 \end{bmatrix} \cdot \mathbf{E} \tag{A.7}$$

where we find equality $\mathbf{P} \cdot \mathbf{N} = A \begin{bmatrix} 0 & 0 & 0 & 1 \end{bmatrix}$, then $\mathbf{P}$ is solved by

$$\mathbf{P} = A \begin{bmatrix} 0 & 0 & 0 & 1 \end{bmatrix} \mathbf{N}^{-1} \tag{A.8}$$

A.1.2 Signal Transfer

In this model, the signal transfer vector $\mathbf{S}$ is

$$\mathbf{S} = \begin{bmatrix} 1 \\ (1-g_1)g_2 \\ [(1-g_1)g_2]^2 \\ [(1-g_1)g_2]^3 \end{bmatrix} \tag{A.9}$$

Regardless of noise, the final output is

$$Output = \mathbf{P} \cdot \mathbf{O} = \mathbf{P} \cdot \mathbf{S}x \tag{A.10}$$

Since signal should be fully recovered, and it requires

$$\mathbf{P} \cdot \mathbf{S} = 1 \tag{A.11}$$

Then factor A in (A.8) is determined

$$A = \frac{1}{\begin{bmatrix} 0 & 0 & 0 & 1 \end{bmatrix} \mathbf{N}^{-1}\mathbf{S}} \tag{A.12}$$

Combining (A.8) and (A.12), the calibration vector $\mathbf{P}$ is derived.

A.2 Case Test

A.2.1 case 1: without DAC error

For a traditional pipeline adc without DAC gain error, g_1 is 1, then signal and noise transfer matrices simplify to

$$\mathbf{S} = \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \end{bmatrix} \tag{A.13}$$

$$\mathbf{N} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ -g_2 & 1 & 0 & 0 \\ 0 & -g_2 & 1 & 0 \\ 0 & 0 & -g_2 & 1 \end{bmatrix} \tag{A.14}$$

Then using symbolic calculation in MATLAB, $\mathbf{P}$ is determined following (A.8) and (A.12)

$$\mathbf{P} = \begin{bmatrix} 1 & \frac{1}{g_2} & \frac{1}{{g_2}^2} & \frac{1}{{g_2}^3} \end{bmatrix}^{\mathbf{T}} \tag{A.15}$$

which follows the conclusion of traditional methods.

A.2.2 Case 2 : with DAC error

Using symbilic calculation in MATLAB with g_1 and g_2 factors, we derive

$$\mathbf{P} = \begin{bmatrix} g_1 & \frac{g_1}{g_2} & \frac{g_1}{{g_2}^2} & \frac{1}{{g_2}^3} \end{bmatrix}^{\mathbf{T}} \tag{A.16}$$

equation (A.16) is approximately the product of g_1 and equation (A.15).