Master's Thesis

Digital correction of DAC mismatch in continuous time ΔΣ AD converters

Siyu Tan
Yun Miao

Department of Electrical and Information Technology,
Faculty of Engineering, LTH, Lund University, October 2014.

Digital correction of DAC mismatch in continuous time $\Delta\Sigma$ AD converters

Siyu Tan & Yun Miao

Supervised by:
Joachim Rodrigues
Mattias Andersson
Pietro Andreani

LUND INSTITUTE OF TECHNOLOGY
Lund University

Master's Thesis
Lund, 2014

Abstract

The popularity of the continuous time $\Delta\Sigma$ A/D converter has increased dramatically in recent years. A major limitation in resolution in such a structure is unit element component mismatches during chip fabrication in the first feedback DAC. Such mismatches cause the DAC to become non-linear, which degrade the signal to noise ratio of the whole $\Delta\Sigma$ modulator. In this work, a 4-bit, third order, single-loop, continuous-time $\Delta\Sigma$ converter with digital background correction of component mismatch in the first DAC has been designed, simulated and fabricated in a 65nm CMOS process. The modulator is clocked at 144 MHz with an oversampling ratio of only 8. The mismatches found in the first feedback DAC is digitally estimated based on cross-correlation and correction can be done using derived correction factors thereafter. In the analog modulator, low-power loop filter with excess loop delay compensation is implemented. The feedback loop is formed by resistive current mode DAC. The SNDR is 67.5dB within 9MHz bandwidth. The digital correction runs in background.

Acknowledgments

We would like to express our sincere gratitude to our supervisors, Joachim Rodrigues, Mattias Andersson and Pietro Andreani, who gave us a great opportunity to design and fabricate a chip in data converter field. With their support, we gain a deeper understanding toward IC design.

Half of this work is digital, and we got a lot of help from Joachim. He is always very kind to answer our questions. In the analog part, Pietro and Mattias contributed many advices in designing circuits and also in the simulation.

Finally, we are very thankful to our friends and our families, who support us during our study in Lund University. It's a wonderful memory that will be remembered in our life.

Siyu Tan & Yun Miao
Lund, Sweden, 2014

Contents

Abstract iii

Acknowledgments v

Contents vii

List of Acronyms ix

List of Symbols xi

Introduction 1

1 Introduction 1
1.1 Motivation 1
1.2 Advantages of CT structure 1
1.3 Mismatches in the system 3
1.4 Organization of this work 3

2 Continuous-time $\Delta\Sigma$ modulator design 5
2.1 Introduction 5
2.2 Discrete-Time(DT) modulator design 7
2.3 DT-CT conversion 8
2.4 Excess Loop Delay (ELD) in CT modulator 10
2.5 Other nonidealities and optimization 11

3 Digital calibration unit 13
3.1 Introduction 13
3.2 Crosscorrelator and mismatch correction factor generation 14
3.3 Division 15
3.4 Correction of the mismatch 16

4 System level simulation in MATLAB 17
4.1 Simulink module construction 17
4.1.1 Loop filter design 17
4.1.2 Quantizer 17
4.1.3 Feed back DAC 19
4.1.4 Mux 20
4.2 Analog part simulation 22
4.2.1 Without non-idealities 22
4.2.2 With non-idealities 23
4.3 System level simulation 25

5 Circuit level considerations (Analog) 29
5.1 Analog part overview 29
5.2 OP-Amplifier design 29

5.2.1 Schematic 32
5.2.2 Layout 32
5.2.3 Simulation 32
5.3 Quantizer 35
5.3.1 Comparator in flash ADC 35
5.3.2 Resistor ladder 35
5.4 Resistive DAC 35
5.4.1 Unit element 38
5.4.2 DAC 38
5.5 MUX 38
5.5.1 ROM 41
5.5.2 Switching logic 41
5.6 Bias circuits 42
6 Circuit level considerations (Digital) 45
6.1 VHDL implementation 45
6.1.1 Design overview 45
6.1.2 Cross correlation function Finite State Machine . . 47
6.1.3 Division Finite State Machine 49
6.1.4 Mismatch coefficient generation Finite State Machine 50
6.1.5 Mismatch coefficient utilization Finite State Machine 52
6.1.6 Behavioral simulation results. Matlab and VHDL simulation match. 53
6.2 Optimization and synthesis 55
6.2.1 Optimization in synthesis flow 55
6.2.2 Post synthesis simulation results 57
6.3 Place and route flow 57
6.3.1 Place and route 57
6.3.2 Post layout simulation results 61
7 Post layout simulation and conclusion 63
7.1 Layout view of the chip 63
7.2 Post layout simulation 63
7.2.1 Analog modulator simulation in Spectre simulator . 65
7.2.2 Circuit level simulation in AMS simulator 66
7.3 Conclusion 66
References 71

List of Acronyms

A/D	Analog-to-digital
ADC	Analog-to-digital converter
BL	Bit line
BW	Bandwidth
CIFB	Cascaded integrators with distributed feedback
CMFB	Common-mode feedback
CMOS	Complementary metal oxide semiconductor
CT	Continuous time
D/A	Digital-to-analog
DAC	Digital-to-analog converter
DC	Direct current
DFF	Data flip-flop
DR	Dynamic range
DSM	Delta-sigma modulator
DT	Discrete time
ETF	Error transfer function
FOM	Figure of merit
FFT	Fast fourier transform
FSM	Finite state machine
GBW	Gain band-width
GE	Gain error
HD2	Second harmonic distortion
IB	In-band (inside the desired bandwidth)
LTE	3GPP long-term evolution
LF	Loop filter in the modulator
MOS	Metal Oxide Semiconductor
NRZ	Non-return-to-zero
NTF	Noise transfer function

OSR	Oversampling ratio
PCB	Printed circuit board
PI	Proportional integrating
PM	Phase margin
PSD	Power spectral density
RZ	Return-to-zero
SFDR	Spurious-free dynamic range
S/H	Sample-and-hold
SNDR	Signal-to-noise-and-distortion ratio
SNR	Signal-to-noise ratio
STF	Signal transfer function
SW	Switch signal uesd to control the feedback path
TF	Transfer function
TTF	Test transfer function
WL	Word line

List of Symbols

α	Time instant where feedback DAC pulse starts
β	Time instant where feedback DAC pulse ends
HD_n	n^{th} order harmonic distortion
f_s	Clock/sampling frequency [Hz]
k	Boltzmann's constant, $\approx 1.381 \times 10^{-23}$ [J/K]
T_s	Sampling period [s]
V_{th}	MOS transistor threshold voltage

Chapter 1

Introduction

1.1 Motivation

$\Delta\Sigma$ AD converters have increased in popularity in recent years. To gain higher SNR in the overall system, designers are willing to use a higher order structure with a multi-bit quantizer. Thus multi-bit feedback DACs are required which causes a problem. The multi-leveled DACs are not perfectly linear, and may result in a severe performance degradation. The nonlinearity in the outer most DAC placed at the input impairs the SNR the most because any mismatch in it will be directly feeded into the system in parallel with the input signal. The more errors that occur, the lower the SNR becomes for the system.

In this work, we implemented a digital correction circuit [1] to compensate for the mismatch found in the first DAC to regain the SNR. The system's output value is the subtraction of the converted digital output signal and the small correction value. Therefore, the ideal SNR can be mostly recovered. Figure 1 is the System block diagram. The analog part is the classical CIFB continuous time Delta-Sigma converter and the digital part is the digital correction.

This ADC targets to the cellular standard LTE which has a 9MHz bandwidth. So that the input bandwidth is also 9MHz in our system. We chose an OSR value equals 8, which comparing with the state of art is quite small. Thus, the sample frequency of our system is 144MHz. Designed ideal SNDR is 70dB.

1.2 Advantages of CT structure

In the analog part, continuous time CIFB (Cascade of Integrators with distributed FeedBack) loop filter topology is chosen. The advantages of the CT $\Delta\Sigma$ ADC are all based on the displacement of the sampler inside the modulator loop [2], thus:

- Allowing the filters to be implemented as continuous-time circuits, consequently reducing the speed requirements of the filters

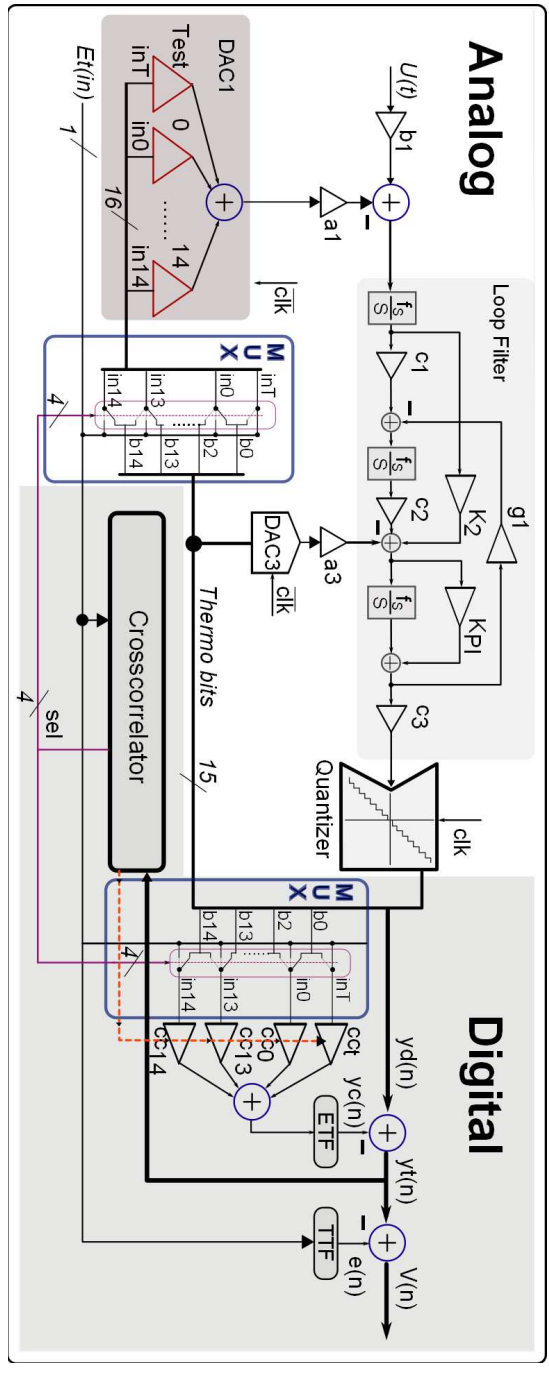


Figure 1: System block diagram

- Providing an implicit antialiasing filter
- Not requiring a precise input sample and hold circuit

More over, continuous-time(CT) loop filters have speed advantages over their discrete-time(DT) counterparts, enabling a higher clock rate or a lower power consumption. [3] Thus it has been a good choice in our project design.

The starting point of designing a CT loop filter is by looking at an equivalent DT CIFB modulator. 'The Delta-Sigma Toolbox' for Matlab is oulined in the book *Understanding Delta-Sigma Data Converters* [4] is a traditional way to implement a DT structure. After that, we conduct the DT-CT conversion using the impulse invariant transform to generate the equivalent CT structure. Finally, the CT modulator is found by doing some mathematical conversions and compensation for excess loop delay(ELD). Details on this part will be covered in chapter 2.

1.3 Mismatches in the system

When designing a chip, there is a spread in component value due to the non-ideal fabrication process, misalignment of the mask, gradients in the wafer and temperature variations introduce mismatches between components. There are ways to compensate these mismatches so that the fabrication phase does not degrade the performance of our design. For example, common-centroid arrangement in the layout can multigate process gradients. However, even with help of modern technology, small mismatches from random variations still exist and would affect the performance. These mismatches are what we mainly focus on in this work and should be corrected by other means than layout.

In the structure shown in figure 1, mismatch inside the outer-most DAC *DAC1* could be regarded as an additional input source to the system. If this mismatch component is large enough, it impairs SNDR a lot. Therefore, we introduced a digital correction block placed at the output of the analog modulator to solve this problem. It calculates the mismatch in each unit element in *DAC1*, represented by the correction coefficients *cc*.The corrected output *yt(n)* is the subtraction of original digital output *yd(n)* and the calculated mismatch value *yc(n)*. If the correction factors could successfully represent all the mismatches, there will be no mismatch component at the final output *V(n)* after subtraction of the injected test signal *e(t)*.

1.4 Organization of this work

Following this introduction, chapter 2 discusses the analog part, especially loop filter design. Some mathematical derivation of the coefficients and simulation result are included. Next chapter, chapter 3 deals with the digital correction mechanism, which mainly covers cross-correlation based mismatch coefficients

generation. Chapter 4 presents the system level simulation in Matlab. It demonstrates how efficient our system could eliminate mismatch produced in outer-most DAC. Chapter 5 and chapter 6 present circuit level implementation of both analog and digital part. Finally, chapter 7 contains a conclusion of this work.

Chapter 2

Continuous-time $\Delta\Sigma$ modulator design

This chapter illustrates the design of the analog part in the system–the $\Delta\Sigma$ modulator. It contains three major parts:

- Loop filter (LF) design
- 4-bit quantizer design
- Feedback DAC design

2.1 Introduction

An ADC is a device that converts a continuous physical quantity (usually voltage) to a digital number that represents the quantity's amplitude [5]. The ADC acts as a 'bridge' that connects the sensor's analog output to the digital processor. For example, everything we record in a microphone, or the temperature sensed by a temperature sensor can be processed by the processor. A simple ADC consists of two major parts: a sample-hold(S/H) block (Sampler) used for uniform sampling in time, and a quantizer to quantize the signal in amplitude. According to *Nyquist theorem*, the original baseband spectrum can be reconstructed when:

$$f_S \geq 2f_B = f_N \tag{1}$$

where f_N is the *Nyquist frequency*, f_S is the *Sampling frequency* and f_B is the *Bandwidth*. As already mentioned in [6], an antialiasing filter is a must in the Nyquist-Rate ADC design, so that higher frequencies beyond the bandwidth does not alias with the in-band signal after periodical sampling. Because this filter is placed at the input stage of the system, it's quality influences the ADC performance. Thus a high quality S/H block is necessary. The operation principle of an ADC is illustrated in figure 2.

Nyquist rate ADCs do not have any feedback loop. Thus, there is no memory of previous samples, and there exists an one-to-one correspondence between the input and output samples. Each input sample is seperately processed, regardless of the earlier input samples [4]. The complete quantization

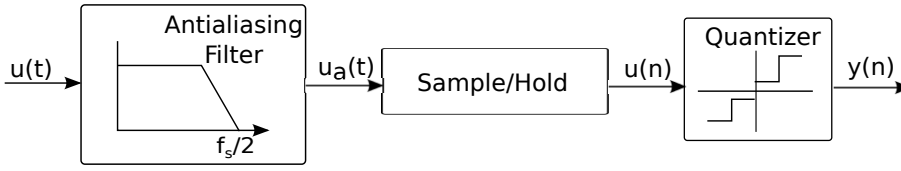


Figure 2: Operation principle of an ADC

noise uniformly lies in the frequency band of interest. When high resolution is demanded, the matching of internal analog components become very important. Therefore to achieve high ENOB, it is wise to choose oversampled Noise-Shaping converters–ΔΣ ADCs. Such structure can 'push' the quantization noise from in-band to out-band, which is usually called *Noise Shaping*. It is capable to gain high SNR with only a small number of output bits at reasonably high conversion speed. Figure 3 is the FFT of a Nyquist rate ADC's output and figure 4 is the result from a 3$^{\text{rd}}$ order 4-bit DT ΔΣ ADC output. In each figure, red line indicates the upper edge of the bandwidth. The noise shaping effect can be clearly seen for the ΔΣ ADC.

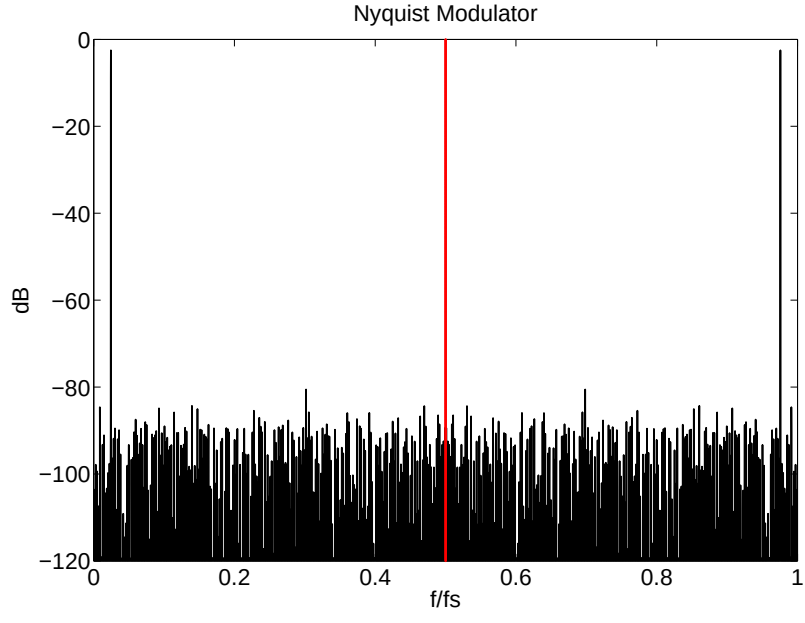


Figure 3: A Nyquist ADC: BW:9MHz, f_S:18MHz,12bit quantizer

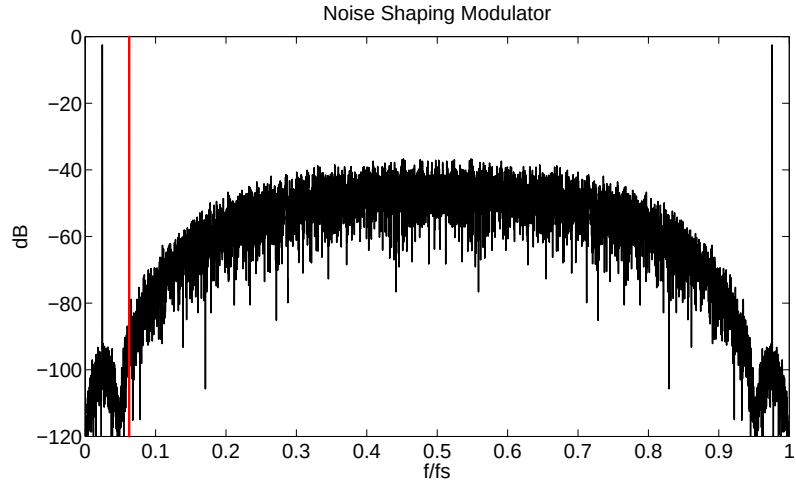


Figure 4: An oversampled ADC: BW:9MHz, f_S:144MHz, 4bit quantizer, order:3

2.2 Discrete-Time(DT) modulator design

To begin with, a 3$^{\text{rd}}$ order CIFB DT modulator is chosen as reference. That is because large numbers of existing topologies are based on Discrete-Time structure, and DT simulation is faster. Figure 5 is the corresponding block diagram of the structure. It contains a cascade of three delaying integrators, with the feedback signal along with the input signal feeding into the input terminals of each integrator by multiplying corresponding weight factors $a_{1,2,3}$ and b_1. The feedback path $-g_1$ together with two latter integrators form a resonator used to move the NTF zero from DC to somewhere in-band to improve SNR. Besides, the Delta-Sigma toolbox in [4] provides a variety of useful functions, which could ease the design phase.

Firstly we use the function 'synthesizeNTF' in the toolbox to synthesize a noise transfer function (NTF) for our delta-sigma modulator with out-of-band gain H_{inf}=6:

$$NTF = \frac{(z-1)(z^2-1.908z+1)}{(z-0.1808)(z^2-0.02173z+0.08153)} \tag{2}$$

Computed CIFB coefficients are listed in table 1. Toolbox function 'simulateDSM' then simulates the generated DT modulator by providing NTF and the input signal. Figure 4 shows the result.

To evaluate the maximum achievable SNR and its corresponding input signal amplitude, tool 'simulateSNR' is used. As shown in figure 6, maximum

SNR of this DT modulator is 72.2dB at -2dB input amplitude, which meets our specification of 70 dB. For a full scale input, the modulator will be unstable and the SNR drops dramatically.

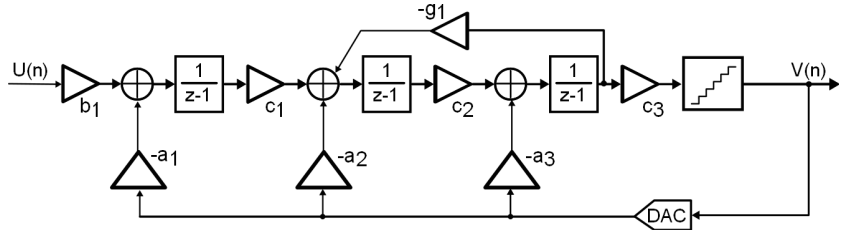


Figure 5: 3^{rd} order DT CIFB structure

Table 1: DT coefficients

Coefficient	Value
a_1	0.8682
a_2	2.5907
a_3	2.7975
b_1	0.8682
c_1, c_2, c_3	1
g_1	0.0897

2.3 DT-CT conversion

The next step in modulator design is doing a DT to CT conversion. A lot of methods can be found to achieve this conversion, such as *The modified Z-transformation* and *The impulse-invariant transformation.* [3] We chose the latter one and the derivation process as follow (The local feedback g_1 is temporarily ignored):

- According to the basic theory of Z-Transform, the impulse response of 1^{st}, 2^{nd} and 3^{rd} order DT integration is illustrated in table 2:

Table 2: The impulse response of 1^{st}, 2^{nd} and 3^{rd} order DT integration

Z-domain	Impulse response
$\frac{a_{3,DT}}{z-1}$	$a_{3,DT}$
$\frac{a_{2,DT}}{z-1^2}$	$a_{2,DT}(n-1)$
$\frac{a_{1,DT}}{z-1^3}$	$a_{1,DT}(\frac{n^2}{2}-\frac{3}{2}n+1)$

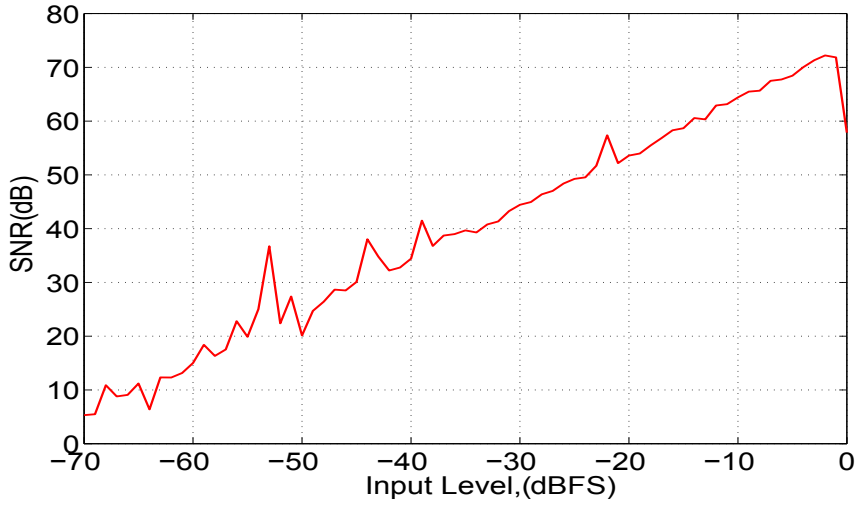


Figure 6: Input amplitude vs. output SNR curve

- Sampled impulse response for 1st, 2nd and 3rd order CT integration is shown below in table 3, where α and β are the starting and the ending point of the feedback DAC pulse (see figure 7):

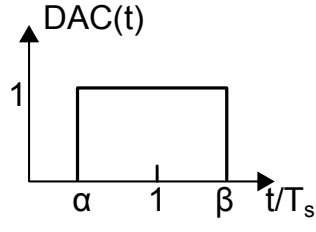


Figure 7: The starting and the ending point of a feedback DAC pulse

Table 3: The impulse response of 1st, 2nd and 3rd order CT integration

S-domain	Impulse Response
$[\frac{1}{s}(e^{-s\alpha_3 T} - e^{-s\beta_3 T})]\cdot \frac{a_{3,CT}}{s}$	$a_{3,CT}(\beta_3 - \alpha_3)$
$[\frac{1}{s}(e^{-s\alpha_2 T} - e^{-s\beta_2 T})]\cdot \frac{a_{2,CT}}{s^2}$	$a_{2,CT}[(\beta_2 - \alpha_2)n + \frac{\alpha_2^2 - \beta_2^2}{2}]$
$[\frac{1}{s}(e^{-s\alpha_1 T} - e^{-s\beta_1 T})]\cdot \frac{a_{1,CT}}{s^3}$	$\frac{a_{1,CT}}{2}[(\beta_1 - \alpha_1)n^2 + (\alpha_1^2 - \beta_1^2)n + \frac{\beta_1^3 - \alpha_1^3}{3}$

- Sampled CT impulse response should match DT impulse response, as equation 3 shows. Thus, the conversion demonstrated in table 4 is conducted. When the feedback DAC waveform(RZ or NRZ) and it's starting point α and ending point β are given, CT coefficients can be calculated from their DT equivalent. A new parameter is introduced in the equation: $\gamma_i = min(\beta_i, 1)$. [3]

$$\begin{aligned} a_{3,DT}+a_{2,DT}(n-1)+a_{1,DT}(\frac{n^2}{2}-\frac{3}{2}n+1) = \\ a_{3,CT}(\beta_3-\alpha_3)+a_{2,CT}[(\beta_2-\alpha_2)n+\frac{\alpha_2^2-\beta_2^2}{2}]+ \\ \frac{a_{1,CT}}{2}[(\beta_1-\alpha_1)n^2+(\alpha_1^2-\beta_1^2)n+\frac{\beta_1^3-\alpha_1^3}{3}] \end{aligned} \tag{3}$$

Table 4: CT coefficients vs. DT coefficients

CT	DT
$a_{1,CT}$	$\frac{a_{1,DT}}{\beta_1-\alpha_1}$
$a_{2,CT}$	$\frac{a_{2,DT}+\frac{a_{1,DT}}{2}(\alpha_1+\beta_1-3)}{\beta_2-\alpha_2}$
$a_{3,CT}$	$\frac{a_{3,DT}+a_{2,DT}[\frac{1}{2}(\alpha_2+\beta_2)-1]+a_{1,DT}[1-\frac{1}{6}(\alpha_1^2+\alpha_1\beta_1+\beta_1^2)+\frac{1}{4}(\alpha_2+\beta_2)(\alpha_1+\beta_1-3)]}{\beta_3-\alpha_3}$
$a_{4,CT}$	$\frac{2a_{3,DT}-2a_{3,CT}(\gamma_3-\alpha_3)-a_{2,CT}[2(\gamma_2-\alpha_2)+\alpha_2^2-\gamma_2^2]-a_{1,CT}[(\gamma_1-\alpha_1)+(\alpha_1^2-\gamma_1^2)+\frac{\gamma_1^3-\alpha_1^3}{3}]}{2}$

2.4 Excess Loop Delay (ELD) in CT modulator

Due to the finite respond time of the DACs and the required decision time in the quantizer, feedback pulse from DACs could not start exactly at the quantization time. This delay is called Excess Loop Delay(ELD) and can lead to instability and changes in the NTF. In this design, a fixed 50% delay between feedback DACs and quantizer is added so that quantizer has sufficient decision time. Such a delay is implemented by injecting inverting clock signals. Without ELD compensation, the system will not work as intended. The basic idea is to realize a fast feedback loop around the quantizer (-a_{PI} path in figure 8) [7–9]. However in this way, one more DAC around the quantizer is required to satisfy such implementation which consumes extra chip area. In this work, a feed-forward proportional path through the third integrator which is called PI-integrator is implemented (Gain block K_{PI} and the last integrator in figure 1). It was first proposed in [10] and used in [11]. Only an additional resistor is required instead of a DAC in the schematic. The K_{PI} coefficient is calculated by comparing this structure with the direct feedback path structure shown in figure 8.

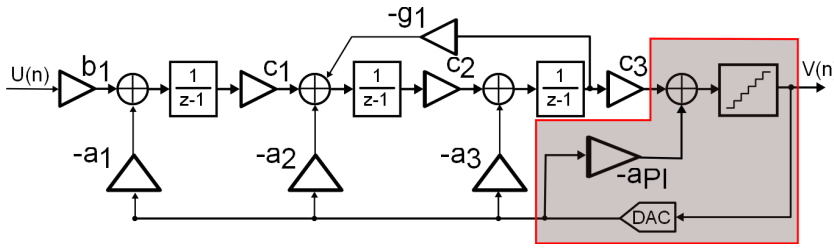


Figure 8: Direct feedback path ELD compensation

2.5 Other nonidealities and optimization

Table 5: CT coefficients

Coefficient	Value
$a_C T1$	0.8682
$a_C T3$	0.7292
b_1	0.8682
c_1	0.2500
c_2	1.6667
c_3	2.4000
g_1	0.0538
k_2	0.7143
$k_P I$	0.9497

In the loop filter implementation, other non-idealities still exist. For example, OP-amplifiers in the continuous-time integrators are not ideal. The finite Gain-Bandwidth(GBW) and the additional gain error(GE) influence the performance of the loop-filter. As mentioned in paper [1], the real integrator can be modeled by an ideal integrator with additional delay and gain error (figure 9). In circuit level simulation, we broke the feedback loop and check the step-respond of each DAC. The phase shift $\tau_{A1,2}$ and τ_{A3} to corresponding integrator's output is measured. Only the delay in the third integrator dominates in this work. To ease calculation, delays in the first and the second integrator have been ignored. Thus the total additional delay is $\tau_D = (0.18 + 0.5)Ts = 0.68Ts$, where Ts is the sample period. Putting $\alpha = 0.68$ and $\beta = 1.68$ into the equation in table 4, new CT coefficients could be derived. Further more, the full scale in circuit level quantizer (533mV) is much lower than the input signal's full scale(1.2V), thus scaling factors c_1,c_2 *and* c_3 should be modified to meet the dynamic range of the quantizer. We set $c_3 = 2.4$, so that the maximum input voltage to the quantizer will not above the full scale. Then c_1, c_2, a_{CT3}, k_2 *and* g_1 should be modified accordingly. Finally, to further save chip area and decrease power consumption, second feedback DAC is replaced by a feed-forward gain

path (K_2) from the first integrator's output to the third integrator's input. This path has the same effect comparing with using a DAC feedback loop. Final loop-filter structure is the *Loop Filter* part in figure 1 and derived CT coefficients are illustrated in table 5.

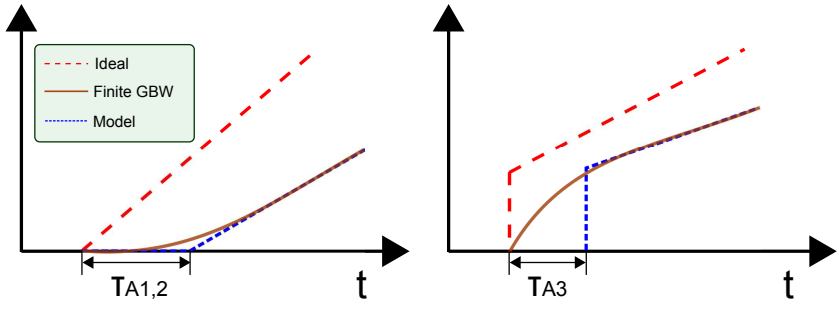


Figure 9: Modeling of finite GBW induced delay and gain-error

Chapter 3

Digital calibration unit

This chapter demonstrates the implementation of the digital part in our system. It contains three major parts:

- Division
- Crosscorrelator
- Mismatch coefficient generation

3.1 Introduction

Error shaping and error correction technique are the two major techniques for linearization $\Delta\Sigma$ modulator architectures. Error shaping techniques also known as dynamic element matching work very reliable, but they do not support low OSRs. Error correction techniques can be implemented in either analog or in digital domain. When we compensate errors in the digital domain, the mismatch error to be corrected should be determined in advance.

The method of determining unit element gain errors used in our implementation is employing the correction techniques as presented in [1]. The function is outlined as follow.

As shown in figure 1, DAC on the feedback loop of the $\Delta\Sigma$ modulator is extended by one additional unit element t and a multiplexer. This extension allows the insertion of test signal E_t into the modulator through a selected unit element in the DAC. In the digital domain, the same multiplexer is used for background correction. The correlation based error estimation is realized by the remaining part of the digital domain in figure 1.

First, the gain mismatches of the unit elements in the DAC need to be estimated. Therefore, a digital test signal E_t is inserted through a selected unit element in the DAC, which appears again at the output of the $\Delta\Sigma$ modulator. A characteristic value of individual gain k_{1_bi}, where i is the number of the unit element under test, can be calculated through the crosscorrelation between test signal E_t and output signal from the modulator. With the multiplexer, the test signal E_t is switched rotationally into any unit element i as shown in figure 15.

A 4-bit select signal named *sel* does the selection of the unit element under test. The inserted test DAC unit element becomes a part of the system and be used on the fly for background calibration.

There are three requirements on the inserted test signal E_t. First the test signal should be single-bit, because it will be switched into the unit element of the DAC which is single-bit. Secondly, the test signal is feedback into the $\Delta\Sigma$ modulator along with the actual input signal U(t). Thus, the test signal must be uncorrelated with the input signal so that the crosscorrelation is only dependent on the test signal and the gain of the selected unit element under test. Finally, there is also frequency requirement on the test signal. The error estimation is using the $\Delta\Sigma$ modulator to convert analog signal from the DAC with test signal inserted to digital output signal, which is used to compute the characteristic value for the selected test unit element by the crosscorrelation. However, the reliable frequency range for A/D conversion of a non-ideal modulator is at low frequencies (in band) of the modulator, where the STF is almost ideally flat, frequency independent and independent of parameter variations in the analog filter. Thus, the test signal E_t is limited to the inband of the modulator by inserting it at a fraction of the sampling frequency.

According to the requirements listed above, the test signal we choose is a pseudo random signal at 1/8192 of the sampling frequency, which leads to 32 changes of the test signal during one sample length. Because the sample length is L = 2^{18}, the logic of test signal changes every 8192 samples. And the actual test signal changes according to sequence S, as equation 4:

$$\begin{aligned} S = [&1 \quad -1 \quad -1 \quad 1 \quad 1 \quad -1 \quad 1 \quad -1 \quad 1 \quad -1 \quad 1 \\ &1 \quad 1 \quad 1 \quad -1 \quad -1 \quad 1 \quad 1 \quad -1 \quad 1 \quad -1 \quad -1 \\ &1 \quad -1 \quad -1 \quad -1 \quad 1 \quad 1 \quad -1 \quad -1 \quad 1 \quad -1] \end{aligned} \tag{4}$$

This sequence has an average of 0. The test signal is input to the system from FPGA in the hardware implementation. With this test signal, the following crosscorrelation can be computed.

3.2 Crosscorrelator and mismatch correction factor generation

With the test signal and output from the modulator Y_t, the crosscorrelation CCF is:

$$CCF_i = \sum_{n=1}^{L} e_t(n) y_t(n) \approx k_{1bi} s_{e_t}^2 \tag{5}$$

Where CCF_i is the crosscorrelation for unit element i, L is the number of samples for computing the crosscorrelation, k_{1bi} is the individual gain of the unit element under test,and $s_{e_t}^2$ is the variance of the inserted test signal. The

crosscorrelation values CCF_i contains the mismatches of each unit element of the DAC. Thus, relating CCF_i to a reference CCF_{ref} as equation 6 reveal the correction factor c_i:

$$c_i = k_{1b}\big(\frac{CCF_i}{CCF_{ref}} - 1\big) \tag{6}$$

where c_i is the correction factor for unit element i in DAC, k_{1b} is the ideal gain factor of DAC, and CCF_{ref} is chosen as the CCF value of the extra test unit element, which in the following part of this thesis called as CCF_t. Deduction of this equation is presented as [12].

In the implemented system, the sample length L is 2^{18}. The accuracy of c_i is set to be 16 bits for high accuracy. The difference between real and ideal correction factor will be illustrated later in figure 18.

In the calculation of c_i, division is a very important part, the realization of an efficient divisor is introduced in next section.

3.3 Division

During the calculation of c_i division is needed for $\frac{CCF_i}{CCF_{ref}}$. There are different methods to calculate division. The way we used here is Newton-Raphson method [13] [14].

Newton-Raphson division is a division method using functional iteration.

Division can be written as the product of the dividend and the reciprocal of the divisor. In this algorithm, a priming function is chosen, which has a root at the reciprocal. By efficiently computing the reciprocal of the divisor, the quotient can be easily computed.

$$Q = \frac{N}{D} = N \times \frac{1}{D} \tag{7}$$

The widely used target root is the divisor reciprocal $\frac{1}{D}$, whose priming function is:

$$f(x) = \frac{1}{X} - D \tag{8}$$

Where the root of X is $\frac{1}{D}$. The well-known Newton-Raphson equation which converges quadratically is given by:

$$X_{i+1} = X_i - \frac{f(X_i)}{f'(X_i)} \tag{9}$$

Apply equation 9 to equation 8, this iteration can be used to find an approximation to the reciprocal:

$$X_{i+1} = X_i - \frac{f(X_i)}{f'(X_i)} = X_i + \frac{\frac{1}{X_i} - D}{\frac{1}{X_i^2}} = X_i \times (2 - D \times X_i) \tag{10}$$

The error in the reciprocal decreases quadratically after each iteration.

3.4 Correction of the mismatch

With the computed correction factors, the correction value Y_c can be computed. Each correction factor c_i is multiplied with the single bit of Y_d that is feed into the corresponding i^{th} unit element in the DAC. Then by summing up the multiplication result, the correction value Y_c is computed, also as shown in equation 11.

$$Y_c = \sum_{i=0}^{15} Y_d(i) \times c_i \tag{11}$$

In this implementation, Y_c is a 17 bits binary signal. The first 5 bits represent the integer part with first bit as sign, while the following 12 bits contain the fraction part. In the following part of this article, the format of the value will be present in $[a, b]$ form, where a indicates the integer part, and b indicates the fraction part. i.e. Y_c is in $[5, 12]$ format.

Before continue the calculation, Y_c should pass through an Error Transfer Function (ETF) which in our case can be simplified to its low frequency equivalent test signal transfer function $TTF = -1$ as the frequency of E_t lies in well below the bandwidth. The minus sign in the TTF or ETF is because the signal coming from *DAC1* will be reversed when subtracted by the input before entering the system.

The correction then could be done by subtracting the reformed output signal from the modulator Y_{d_bin}, which is also in $[5, 12]$ format, by the correction value Y_c:

$$Y_t = Y_{d\_bin} - Y_c \tag{12}$$

where Y_t is the corrected signal which is used for the following crosscorrelation. For the first 16 sets of samples, which is in total 16×2^{18} samples, Y_t is equal to Y_d as Y_c is 0. As the correction is in background mode, the corrected signal is continuously closing to the ideal value every time it is calculated. However, because the accuracy of Y_c can only be 12 bits in fraction part, this iterated correction will never reach the ideal value.

As signal Y_t contains the inserted test signal, the final output signal Y_{out} can be easily computed:

$$Y_{out} = Y_t - E_t \times TTF \tag{13}$$

where E_t also passes a TTF which is simplified to a low frequency equivalent, $TTF = -1$. In a real implementation, both ETF and TTF become a delay in the circuit.

Chapter 4

System level simulation in MATLAB

Before implementing the system in circuit level, we start by running simulations in Matlab. First, a Simulink Model is constructed. Then mismatches in the feedback DAC are introduced to test the digital dorrection technique. Finally, system level simulation will be performed.

4.1 Simulink module construction

Simulink is a block diagram environment for multi-domain simulation and Model-Based design. Compared with using MATLAB script-based simulation,Simulink further eases the design process, since it provides many useful construction blocks, which present the identical functionality of real circuit blocks. Further more, it's graphical view makes the structure much more clear to understand. Figure 10 plots system level model of this work. In the figure, the white part is the analog modulator, the red part is the digital correction block and the blue part is pseudo random test signal injection path.

4.1.1 Loop filter design

Based on previous discussion in chapter 2, we have derived the CT loop filter coefficients. Now corresponding blocks need to be placed in Simulink Graphical Editor and properties be properly set. The loop filter is demonstrated in figure 11. $\frac{fs}{S}$ represents a continuous time integrator, where *fs* stands for sampling frequency. $t_{(1-3)}$ are the transport delay and $GE_{(1-3)}$ are gain blocks. They are used to model the finite GBW in the real integrators, whose gain is finite and have limited bandwidth, as previously mentioned. When $GE_{(1-3)}$=1 and $t_{(1-3)}$=0, all three integrators in LF are ideal, which means they all have infinite GBW.

4.1.2 Quantizer

The 4-bit internal quantizer circuit is realized by a mid-rise flash ADC and it's Simulink block diagram is shown in figure 12. The terminology *mid-rise* stands

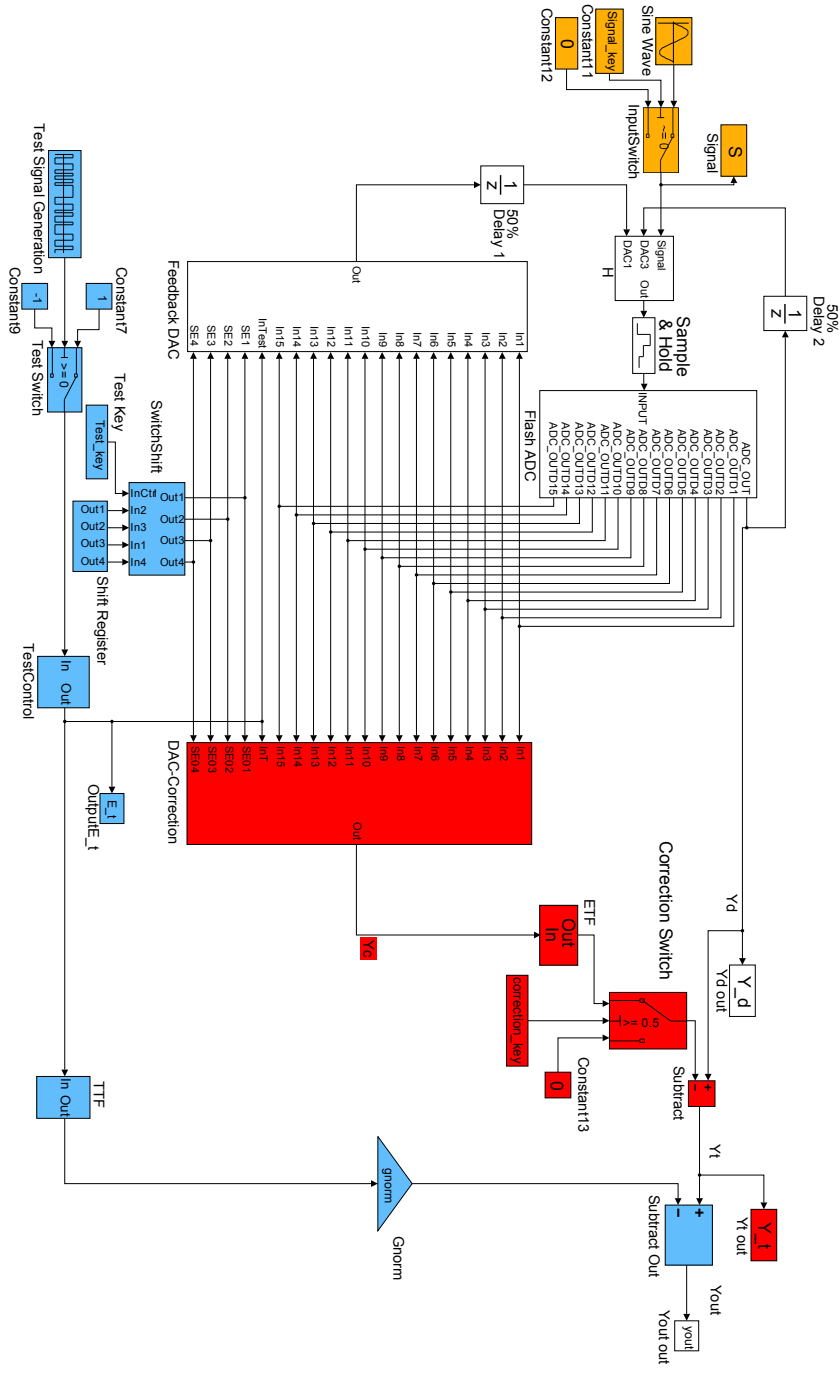


Figure 10: Simulink model

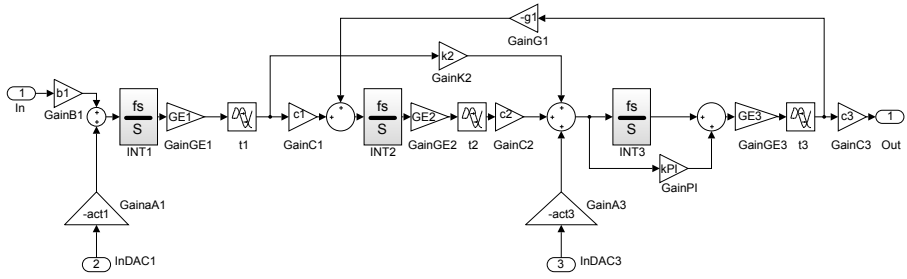


Figure 11: LF model in Simulink

for that when input is at zero, the output is half the step size. Full scale (-1 1) is divided by 15 comparators into 16 levels. So that the thresholds are:

$$Threshold = -1 + \frac{1}{2} \times \Delta \ + \Delta \ \times i, i \ is \ [1, 2, ..., 15], \Delta \ = \frac{2}{16} \qquad (14)$$

The digital output is 15-bit thermometer code. When input analog level is above any of the thresholds, all comparators below it including that threshold comparator output '1', while the others output '-1'. All '-1' stands for minimum value and all '1' is the maximum. By counting the number of '1's, the quantized input value is derived. In the circuit, however, a '0' is represented by '-1' and '1' remains the same. This conversion, of course, has no influence to the following digital operation and output result.

It is also worth to mention that because the quantizer lies inside the feedback loop, small non-idealities are suppressed by the loop and thus make no difference to the resulting SNDR [6]. Therefore, this flash ADC structure is ideally constructed in Simulink and mismatch and delay are not taken into account.

4.1.3 Feed back DAC

The most important part in the $\Delta\Sigma$ AD converter design is the feedback DAC, especially the outer-most one. As it's output directly feeds into the system in parallel with input signal, any internal DAC unit element mismatch could be regarded as an additional input source. This mismatch component is highly relevant to it's digital input pattern that should be treated seriously. In Simulink, the DAC design is a summation of all the digital bits with a multiplication of a fixed gain g_{norm}. Thus, the analog output swings from -1 to 1 as expected. When dealing with the first DAC, however, additional mismatches can be added, which simulates real circuit situation. Those mismatches are what

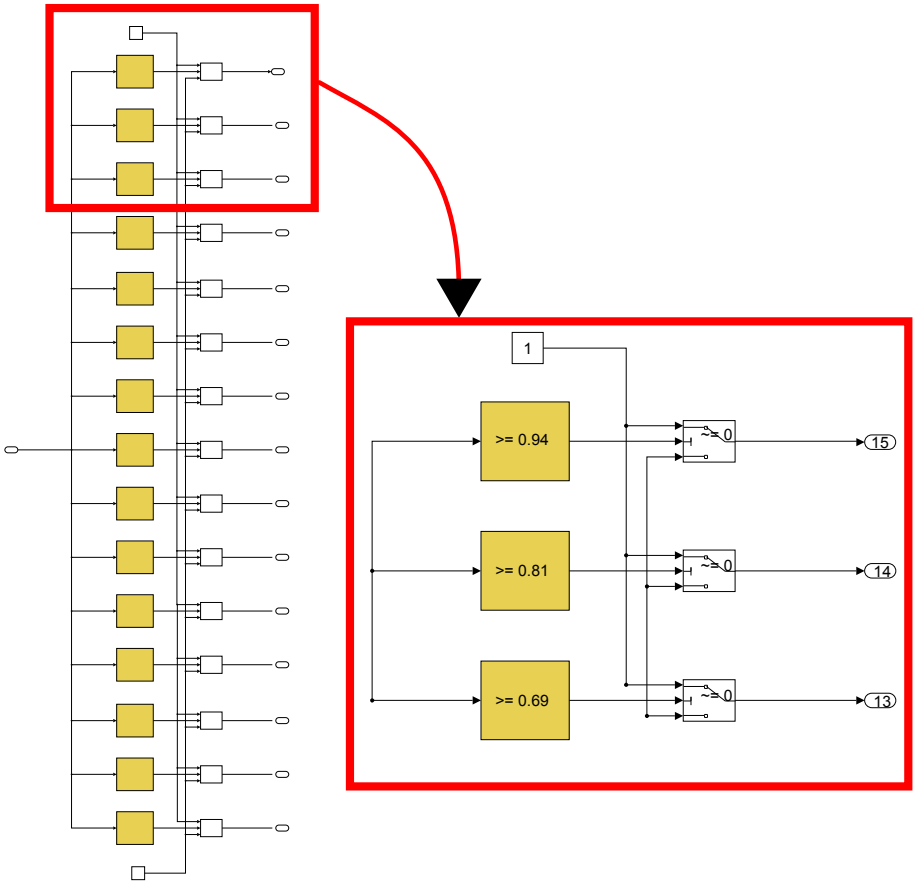


Figure 12: Quantizer model in Simulink

we would like to correct. One more unit element is included which acts as an additional signal path for test signal routing. Thus, in total 15+1 unit elements are used. Figure 13 plots the Simulink model of first DAC. Each individual gain (gain(i)) is set in the MATLAB script. When the mismatch components are added, small offsets are added to the original DAC levels, which will be directly seen at the output, due to flat signal transfer function (STF) of the modulator.

4.1.4 Mux

Figure 15 is a Simulink model of the MUX block. To do digital correlation, a single bit test signal should be injected through the first DAC to the system.

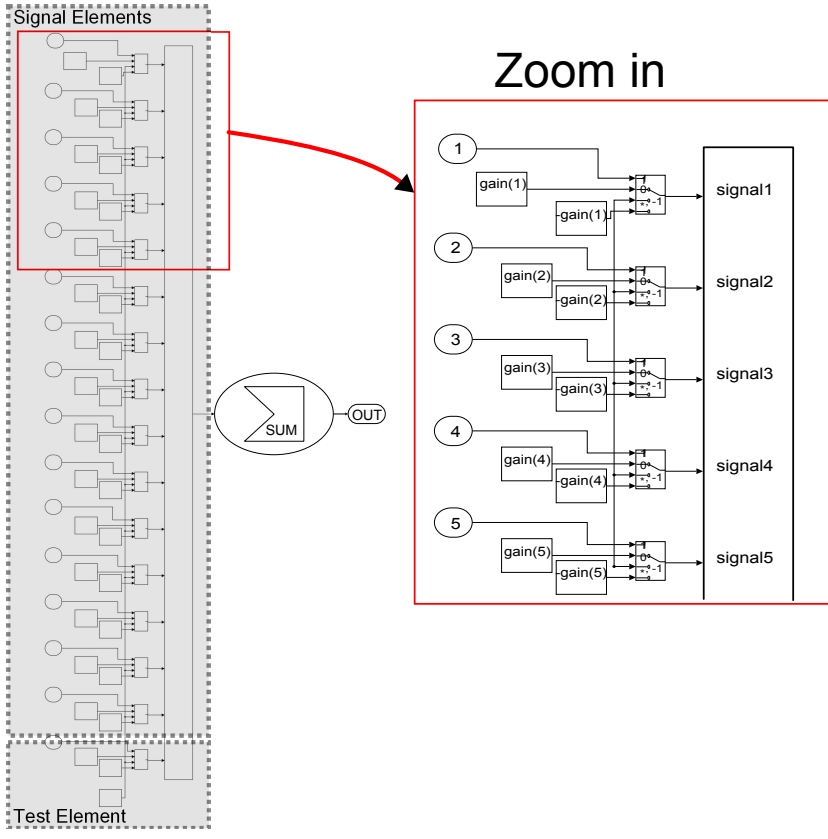


Figure 13: 1^{st} DAC model in Simulink

Thus a MUX component is designed to switch digital signal injection pattern and to control test signal path. It is located between the quantizer output and the first DAC input. A conceptual functionality figure is illustrated in figure 14. Four-bit selection signal *sel* is used to control the signal routing. When *sel=0000*, signal routing doesn't change. Otherwise test input will inject to corresponding DAC unit element whose mismatch is under measurement. Normal AD conversion is not interrupted because the digital feedback path still exists using the additional *Test Element* in figure 13. After 16 iterations, all unit elements have been tested and a mismatch estimation can be done. In simulink model, the path is controlled according to a look-up table.

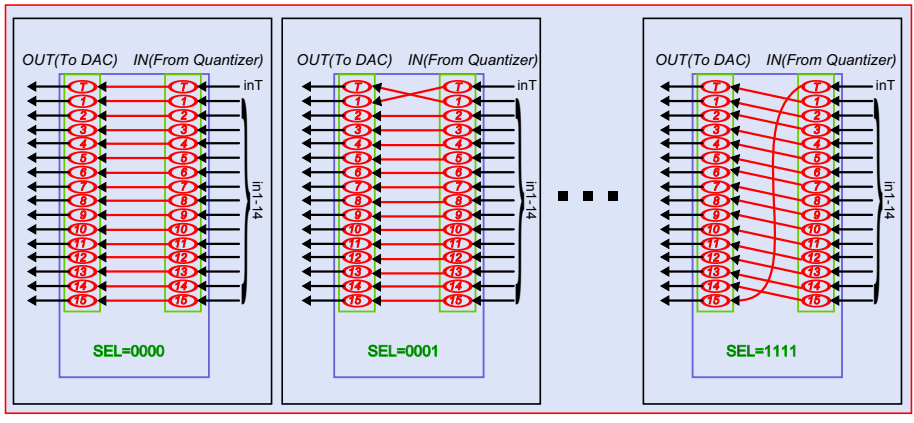


Figure 14: Conceptual functionality of MUX block

4.2 Analog part simulation

This section discusses simulation in MATLAB. We start from explaining ideal condition. Key parameters are illustrated in table 6. *Test input frequency* stands for the pseudo random test signal's frequency, which in our case is well below input frequency so that it runs in background without interrupting normal operation. The detail of this part has been illustrated in section 3.1. The input frequency is low to make sure higher order harmonic distortion lies inside of the bandwidth.

Table 6: Simulation parameters

Parameter	Value
Bandwidth	9MHz
Sample frequency(F_s)	144MHz
Input frequency(F_{in})	1.4766MHz
Amplitude in dB	-2.5dB
Test input frequency	17.578KHz

4.2.1 Without non-idealities

To estimate the resulting model performance, we first run an ideal simulation. Figure 16a is the ideal case simulation between CT Simulink model and the original DT model with same specifications. Generally speaking, two plots don't show large differences. The small mismatch components in CT model's result is because of the accuracy of Simulink. Any conversion or transition mismatch in the model will result in SNDR deterioration. By doing simulation at different input amplitude, we found out that -2.5dB input amplitude could

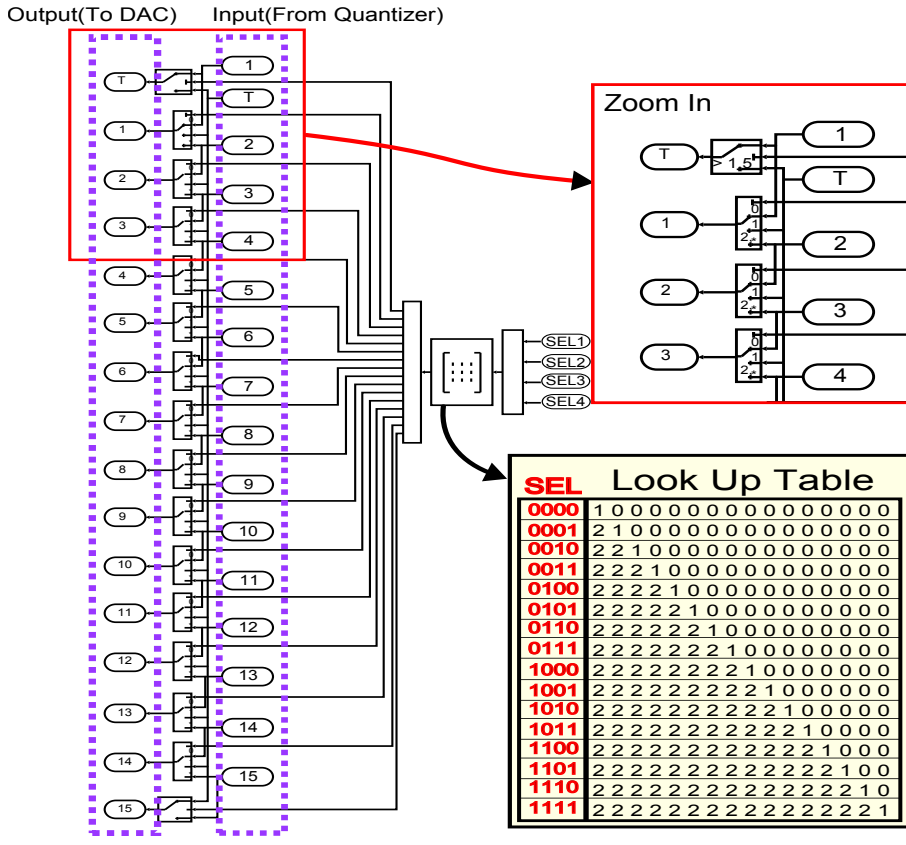


Figure 15: Simulink model: MUX block

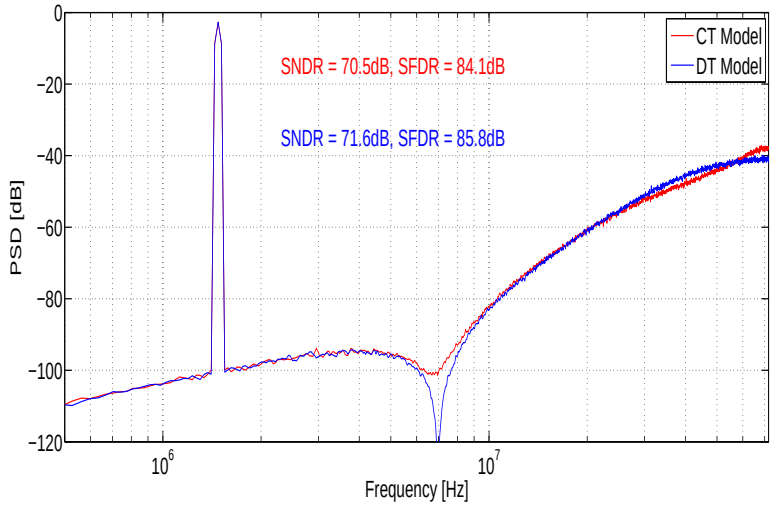
result in highest SNDR 70.4dB. This is a starting point to test the whole system.

4.2.2 With non-idealities

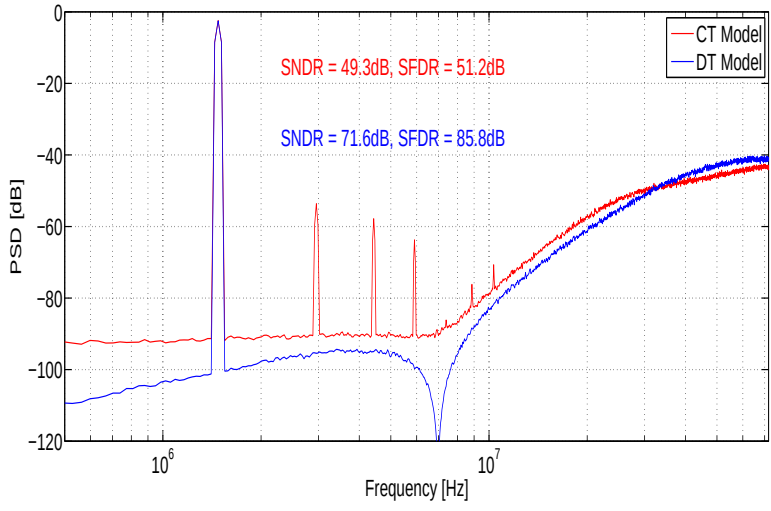
In the real circuit, mismatch in the first DAC influence modulator's performance the most. So that we added some arbitrary small gain errors *ge(i)* inside each gain component *gain(i)* in first DAC, as equation shown below:

$$gain(i) = gnorm \times (1 + ge(i)) \tag{15}$$

Where *gnorm* is a gain $\frac{1}{15}$ used to normalize the DAC output full scale range. These gain errors *gain(i)* could be seen from DAC's output because in Simulink model (figure 13), DAC analog output is a summation of input word times



(a) Ideal case simulation: CT Vs. DT



(b) Non-ideal case simulation: CT(non-ideal) Vs. DT(ideal)

Figure 16: Ideal vs. non-ideal case simulation

corresponding gain factors. It follows signal route and will be directly seen in digital output Y_d in figure 10. Figure 16b proves that such mismatch influence can be clearly seen. SNDR degrades a lot compared with the ideal case result in figure 16a. This mismatch component is what we mainly focus on in this work, which can be efficiently corrected by a digital block.

4.3 System level simulation

System level simulation will be discussed in this section. In the following, we have combined the previously mentioned blocks. We ran a $L \times 16$ samples simulation, where 'L' stands for the sample length of one cross-correlation length. A relatively large 'L' of 2^{18} is chosen, so that the correction coefficients calculation becomes more accurate. Test signal length is the same as 'L' and dynamic range is influenced by one LSB due to test signal injection. Thus input amplitude should be reduced by the same amount to meet the dynamic range in the system. After running 16 cross-correlations, the mismatch can be estimated, as explained in chapter 3. To do the SNR calculation, the FFT length is set to a smaller number of 2048, and averaging is used to provide a smoother FFT result.

The complete simulation result is plotted in figure 17. There are 6 subplots in the figure. The first one is the output without correction. Up to 5^{th} order harmonic distortion could be clearly seen and the SNDR is heavily degraded. The second to the fourth plots show results from three iterations of digital calibration process. The correction coefficients *cc(i)* is derived from each cross-correlation between digital output Y_t and the pseudo random test signal E_t. Figure 18 plots real vs. ideal correction coefficients in these three iterations. To enable VHDL digital implementation, we limited the resulting *cc(i)* resolution to 16-bit, so the maximum absolute difference between real and ideal coefficients is around 1.025×10^{-4}.

After three iterations, the fifth plot is the result of simulation using correction factors calculated previously, and the test input is disabled. Compared with the correction result using ideal correction factors shown in the last plot, designed ideal SNDR could be almost recovered. In ideal case, correction factors *cc(i)* have full resolution. However, the reason for the remaining mismatch existence is the non-ideal ETF setting. In our estimation, STF should be flat in the signal band. So that mismatches in first DAC are directly shown in the result which stands for ETF='-1'. However in real case, STF is not exactly 1 due to e.g. DT-CT conversion accuracy, ETF='-1' is not a accurate representation of the error transfer path. We rerun the simulation, changed ETF from '-1' to '-0.9', and the distortion increased while ideal case SNDR dropped from 70.2dB to 68.8dB. Thus we can conclude that the ETF will influence the correction accuracy and will make the DAC internal mismatches been hardly full canceled. This is the limitation of our correction system.

Finally, in the simulation result, there is 13.4dB SNDR and 21.4dB SFDR improvement. It is based on the 2% mismatch estimation. In the real circuit, there could be only small mismatches in the resistive feedback DAC, so that 2% mismatch may be a little bit exaggeration. However in the simulation, even this large mismatch can be efficiently corrected and a satisfied result is achieved. We can conclude that in real case, mismatch correction system in this work should work fine.

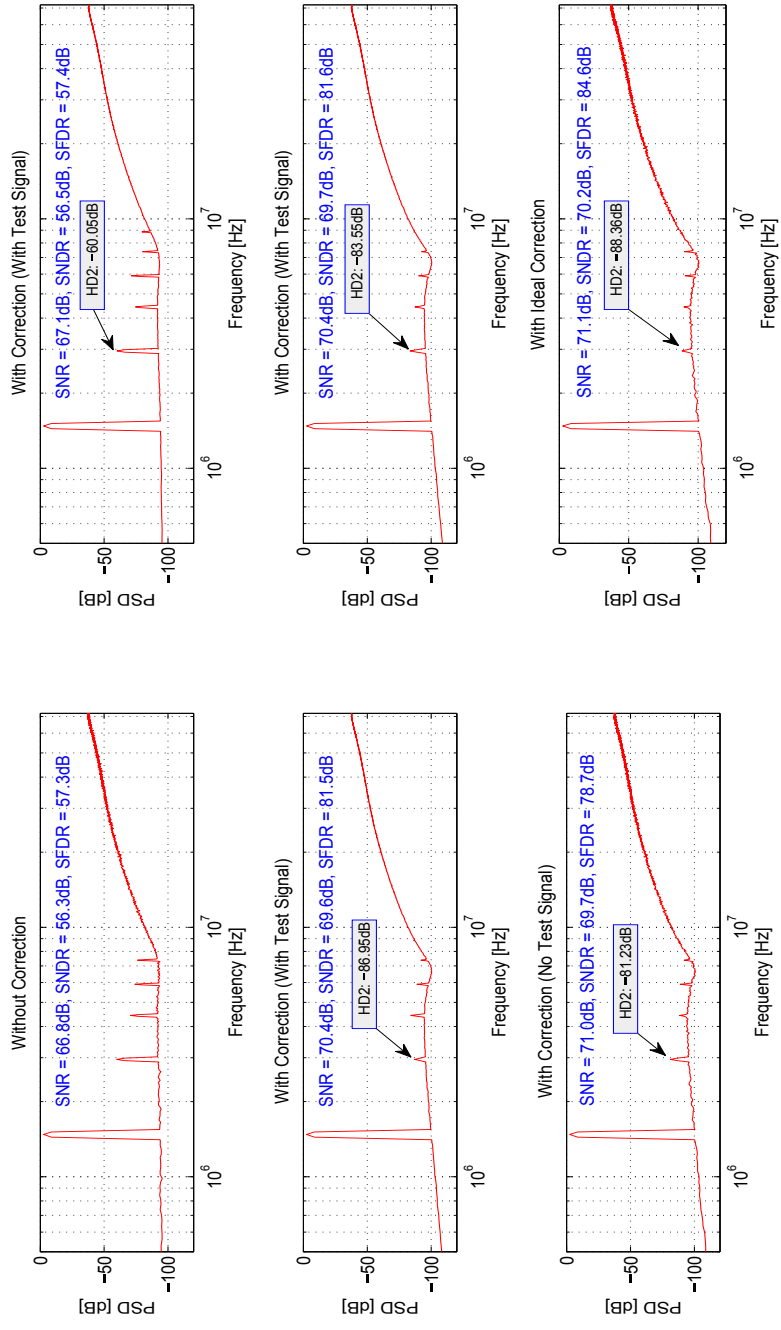
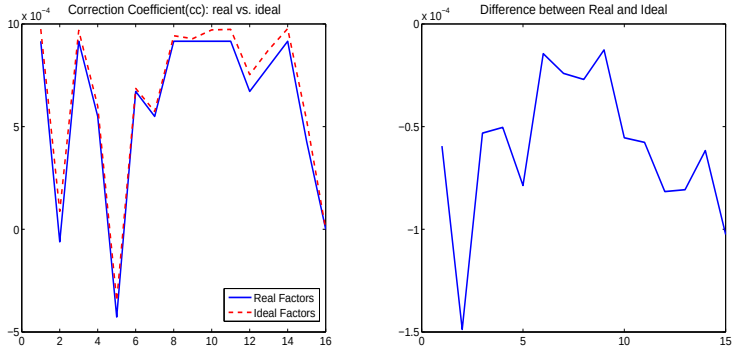
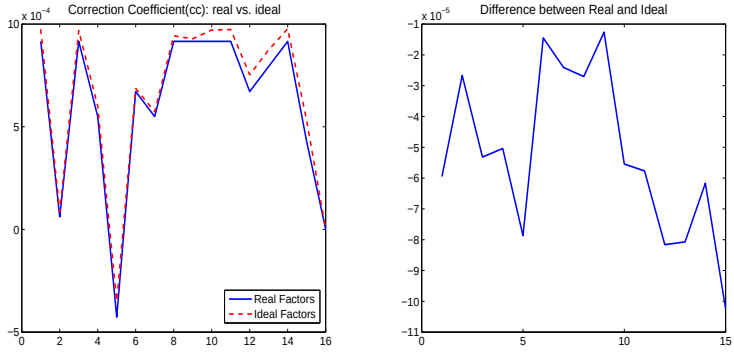


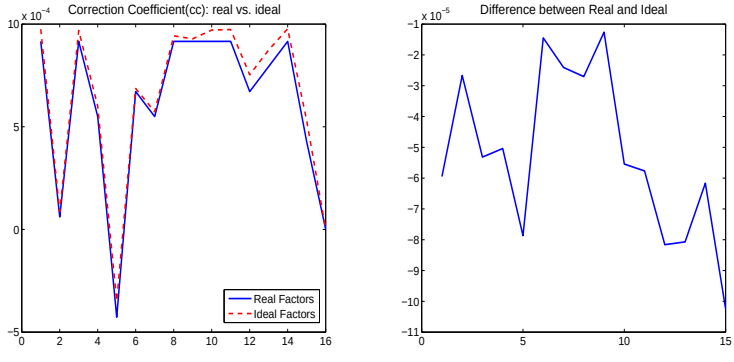
Figure 17: System level simulation with 2% maximum mismatch between unit elements



(a) 1^{st} Iteration



(b) 2^{nd} Iteration



(c) 3^{rd} Iteration

Figure 18: Three iterations in a single run

Chapter 5

Circuit level considerations (Analog)

This chapter presents circuit level implementation of the analog part in Cadence. This design is based on ST-65nm technology. Analog part and digital part are designed separately and manually connected in the top level.

5.1 Analog part overview

To begin with, we first have a look at the top level schematic of the analog modulator. As seen in figure 19, the arrangement is quite similar to the Simulink model mentioned in chapter 4. Two DACs, *dac1* and *dac3*, are placed in the feedback loop and the MUX block is implemented in front of the outer most DAC, *dac1*. There are two additional blocks, compared with the Simulink model: A thermometer to binary converter *therm2bin* to directly output uncorrected 4-bit binary signal and one *DeltaSigma_InSW* block that controls a 4-bit selection signal *sel* input either from on-chip digital block or from outside world. The reason of introducing such a precaution is that even if there are bugs in the digital block which cause malfunction of the circuit later on, we still have the ability to do the digital calibration using an external FPGA.

5.2 OP-Amplifier design

In the loop filter schematic(figure 20), high performance continuous time integrators are very important. The OP-AMP, the most critical component in a integrator, should be carefully designed. In the third order loop filter, the first and the third amplifiers both need to drive high capacitance load, the limitations for the first amplifier are due to its input referred linearity requirements, while the third amplifier is limited due to loop stability reasons because of the chosen PI-based ELD compensation. [1] Thus higher performance amplifiers are designed in them. In contrast, the second one doesn't have speed requirement, and any nonlinearity it introduces is suppressed by the loop. So high GBW is not that necessary. There are many specifications used to judge one amplifier's performance. In this design, the following items are of our interest:

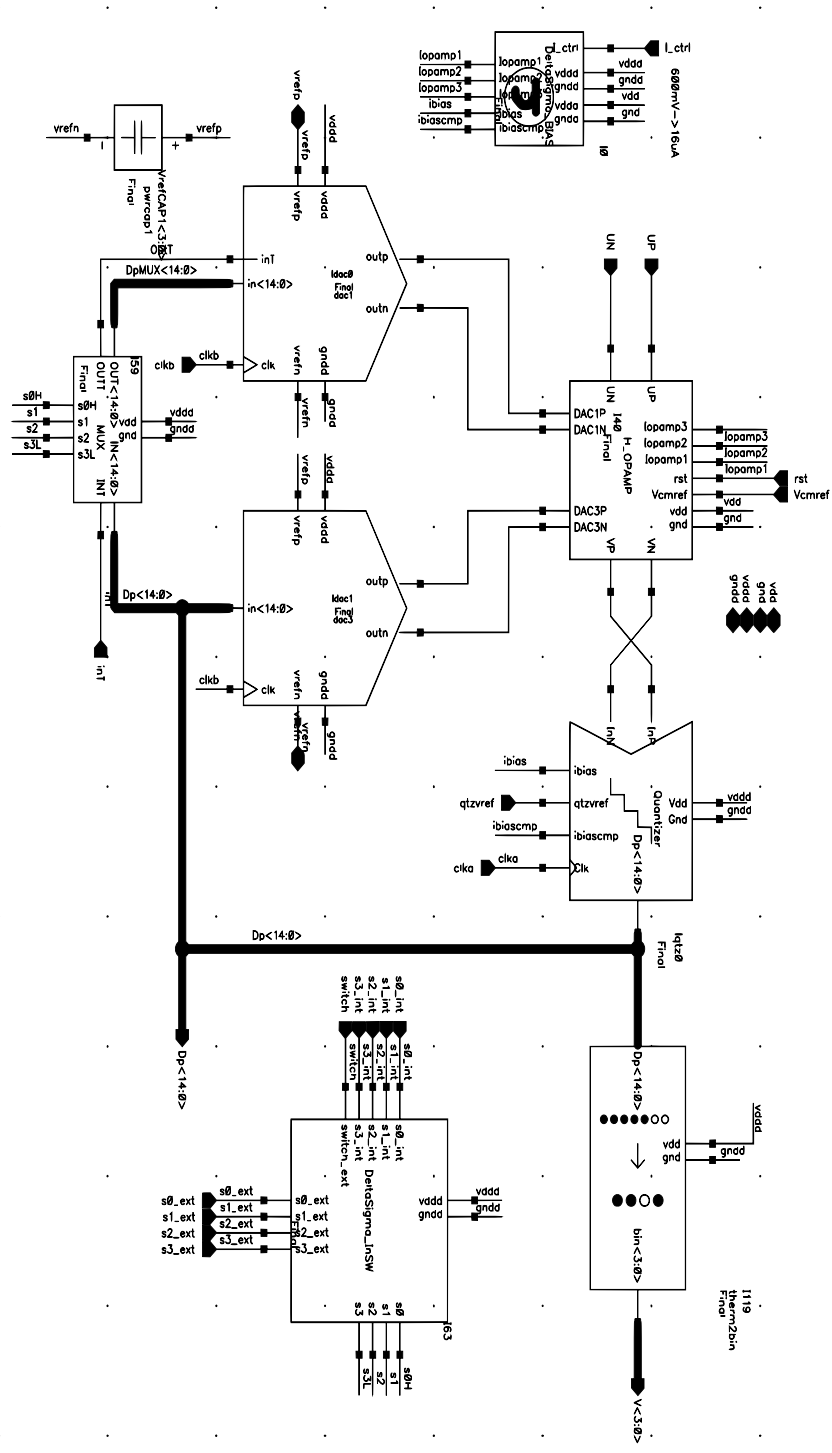


Figure 19: Top model schematic in Cadence

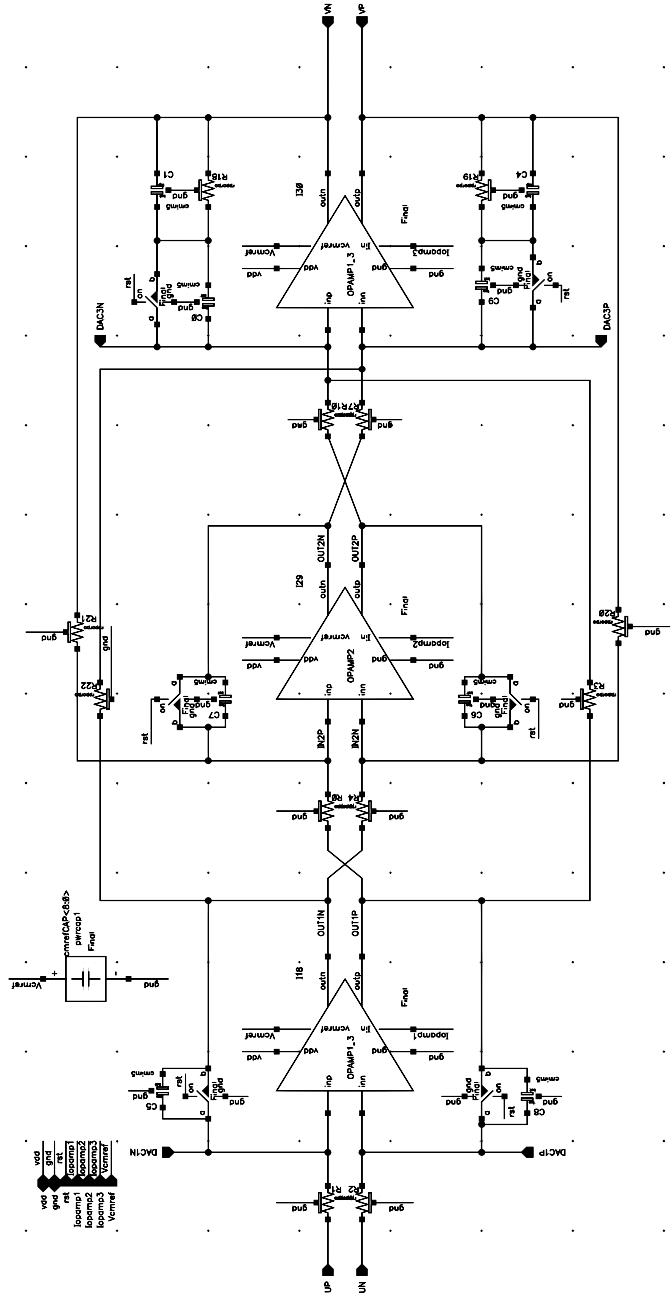


Figure 20: loop filter schematic in Cadence

- DC gain and gain bandwidth(GBW)
- Phase margin(PM) and stability
- Step response

5.2.1 Schematic

Figure 21 is the schematic of the OPAMPs. All three amplifiers have the same structure but various bias currents. The first and the third amplifiers, OPAMP1 and OPAMP3, are identical, while the second amplifier OPAMP2 is running at low power, which result in a relatively weaker performance.

The amplifier's structure is a two-stage class AB output fully differential OP amplifier. [3] An additional Common Mode Feed Back(CMFB) circuit is added to provide stable common mode bias point. It only controls part of the active load to limit the CMFB loop gain to prevent stability problem. Further more, in parallel with the first input stage differential transistors, two additional transistors are added to solve a start-up problem. For example, if the input and output are too low, CMFB circuit could not pull input node back to a proper bias voltage thus leads to start-up failure. This can happen when the amplifier is connected as an integrator. With the help of these start-up transistors, however, input bias voltage will always be stay in a proper range. Taking advantage of using low V_{th} transistors in 65nm CMOS technology, tail current source in the design are cascode configured for better current mirror matching without decreasing V_{gs} of the input stage transistor.

5.2.2 Layout

Based on schematic, a layout view is drawn as shown in figure 22. It occupies approximately 80um $\times$ 90um chip size. To reduce mismatch, we used common-centroid technique, which means matched transistors are in '$ABBA$' arrangement, where A is one transistor and B is another. As discussed in [15], using this technique could suppress the fabrication gradient problem. In the layout, input stage is placed on the left and the class AB output stage on the right. The middle part is tail current source, a large area is used due to cascode configuration.

5.2.3 Simulation

To test the performance of the amplifiers, a testbench is done by connecting each amplifier to proper load. A stability test is then ran to measure GBW and phase margin for amplifiers in all three integrators. Additional 200fF load capacitor is connected to all the output nodes. From the result in table 7, a large GBW difference can be seen between the first and the third integrator.

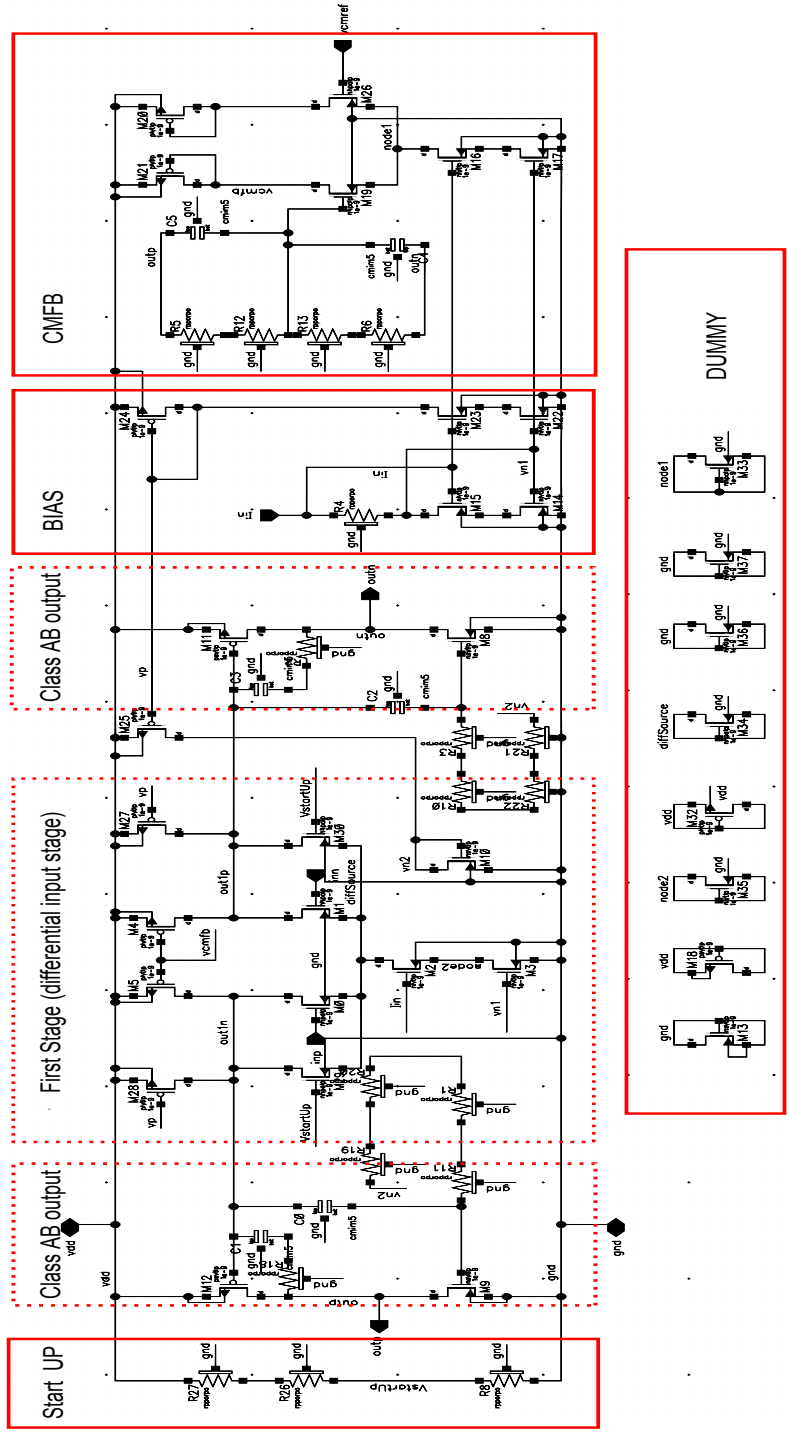


Figure 21: OP amplifier schematic

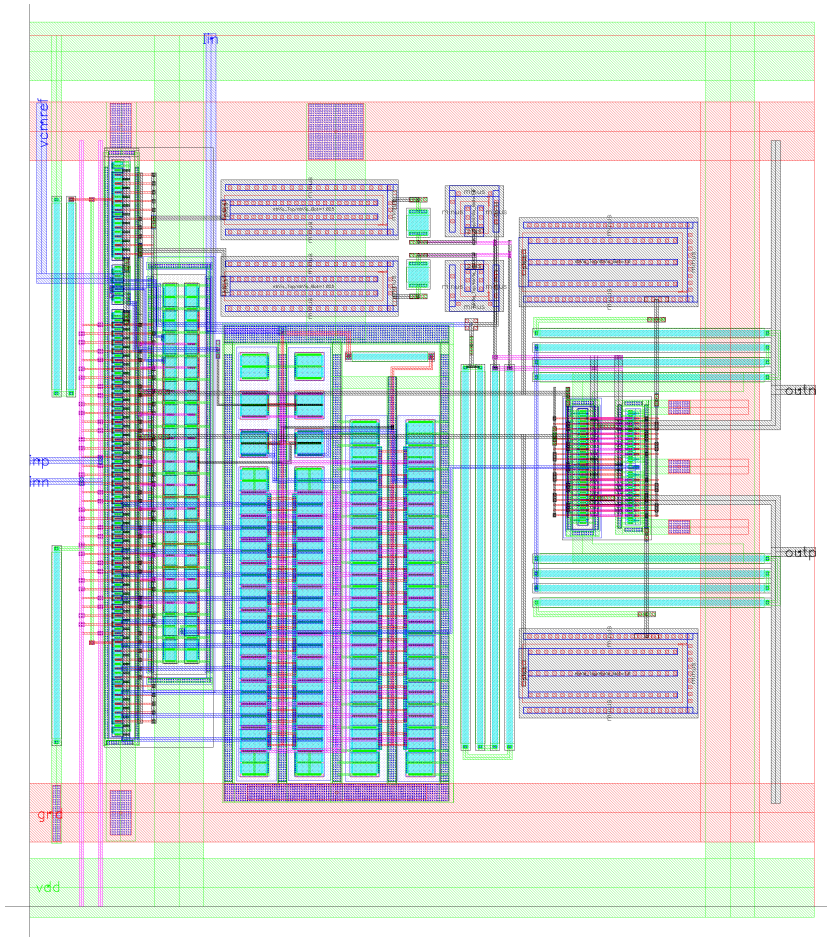


Figure 22: OPAMP layout view

Although the internal amplifiers are the same, different integrator R and C values could lead to performance variation.

Table 7: OPAMP performance

Integrator	GBW	PM	Slew Rate	Power
1^{st}	981.75MHz	82°	676V/us	1192.78μW
2^{nd}	493.02MHz	65°	300V/us	616.40μW
3^{rd}	383.89MHz	53°	676V/us	1192.78μW

5.3 Quantizer

Quantizer design will be discussed in this section. In $\Delta\Sigma$ system, a simple flash ADC is good enough. It is fast, accurate, and doesn't require large chip area. Furthermore, it finishes the conversion within one clock cycle, contrary to e.g. SAR or pipelined converters.

5.3.1 Comparator in flash ADC

The basic building block in an ADC unit is a comparator, whose schematic [3] is plotted in figure 23. The left part is a fully differential input stage, which pre-amplifies the input signal. In the middle it is a clocked core circuit. As mentioned earlier in chapter 1.2, a CT modulator has internal sample and hold functionality. The clock core together with a SR latch keep track of input signal during *clkb* high and hold the value in another half cycle. This is the idea of how sample and hold works.

5.3.2 Resistor ladder

To provide stable and accurate reference levels to the comparators , a carefully designed resistor ladder is made. From the schematic in figure 24, instead of only using a series of resistors to divide power supply, an additional voltage buffer is used. [3] Any disturbance in power or reference voltage should not affect output reference voltages. In our design, 15 reference levels are divided by 14 uniform steps from 366mV to 834mV.

5.4 Resistive DAC

The feedback DACs are critical components, since their linearity dominates the $\Delta\Sigma$ modulator's performance. In recent years, many DAC structures have been developed, such as current steering DAC, R-2R ladder DAC, Successive-Approximation DAC, etc. The working principle is that it senses all the input line's value using internal unit elements, and outputs a corresponding analog value, either in voltage or in current. In this work, a resistive current mode

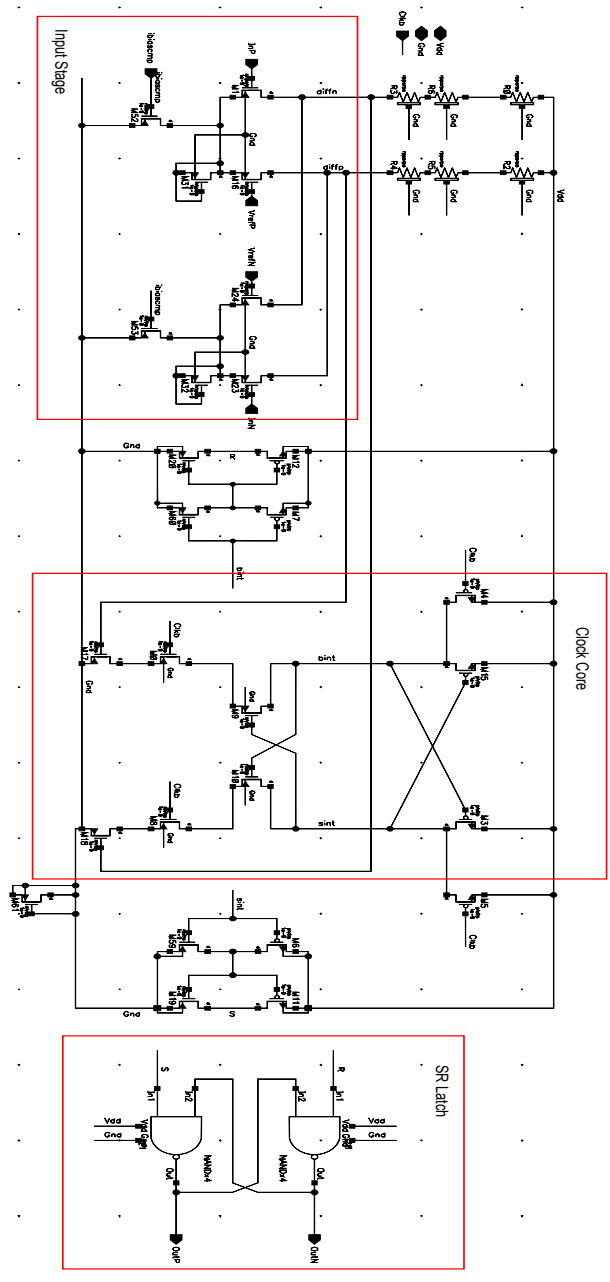


Figure 23: Comparator in flash ADC

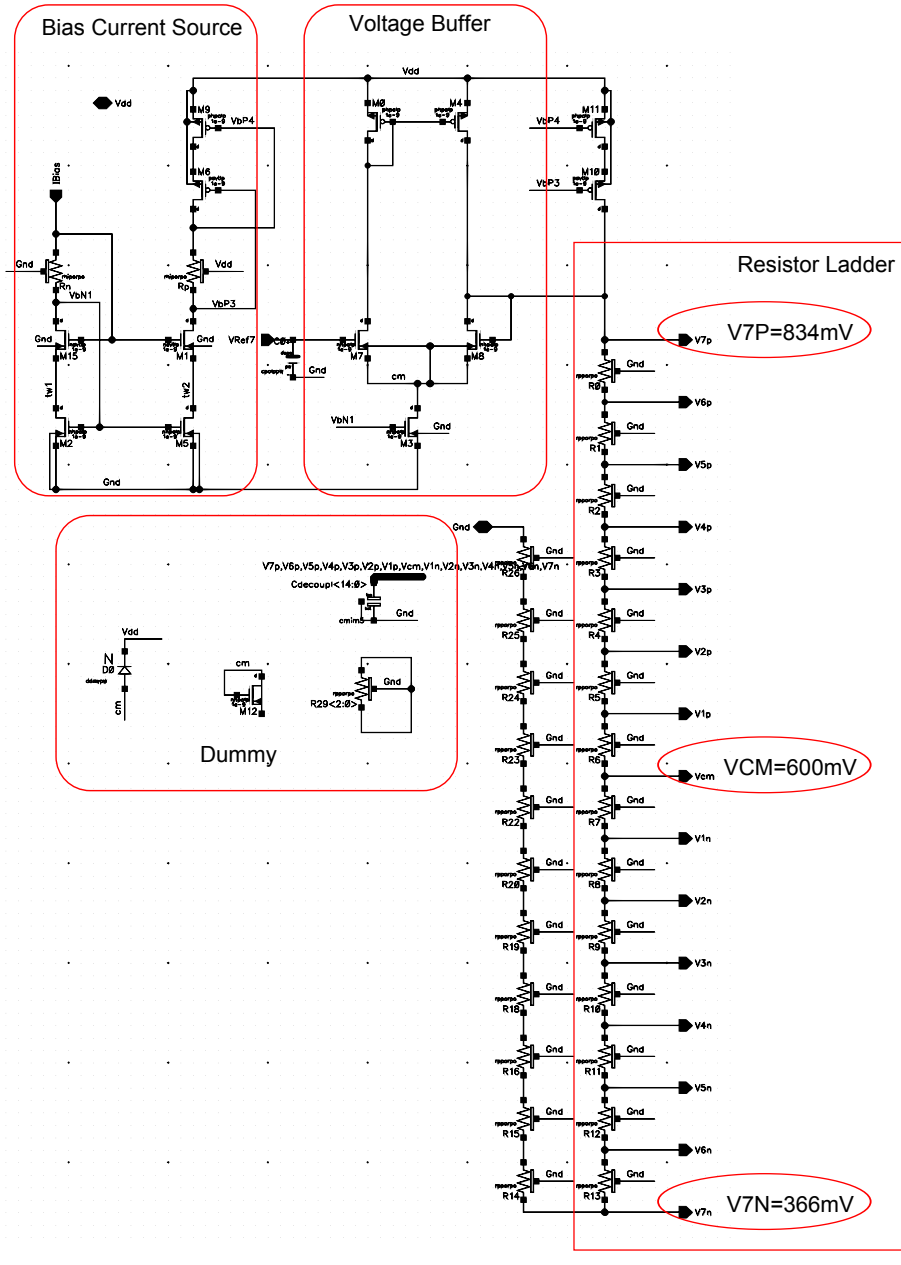


Figure 24: Resistor ladder

DAC structure is chosen. It represents that resistors are used in the unit elements to generate a proper current value. It can also be done using current sources however. The reason we chose resistors is that during chip fabrication, process mismatch in implementing a resistor is much smaller than implementing a current source. More importantly, the resistive DAC has less thermal noise than the current steering DAC. [3]

5.4.1 Unit element

Figure 25a illustrates the schematic of a single unit element in a feedback DAC. It consists of two major parts: A D-Flip-Flop that receives one sample at rising clock edge and a resistor to transform reference voltage into plus or minus unit current out. The D-Flip-Flop block, shown in schematic view below in figure 25b, is a traditional DFF design. During clock low, gate in blue (Dashed) box is opened and signal flows following blue (Dashed) arrow line. When the clock is high, input path is blocked and transmission gate in red (Solid) box starts to work. Previous value at point A flows through transmission gate and when the clock is low, this value is stored in the right most loop. In short, input can be captured and saved only at clock's rising edge. During any other time, input value would not disturb output result.

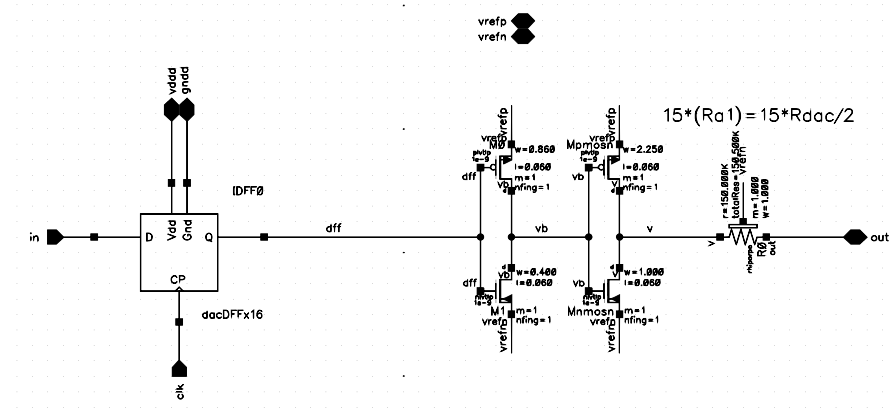
5.4.2 DAC

By putting all DAC unit elements in parallel and connecting their outputs together, a DAC block is constructed as demonstrated in concept figure 26a. The purple (Dashed) part is one additional unit in the first DAC used to inject test bit. Output current from all unit elements will be summed together, and the magnitude of the output current is the analog representation of the input digital word. Figure 26b shows the corresponding layout view. As can be clearly seen, unit elements are placed next to each other to reduce the mismatch. Top parts are the resistors and the DFFs are placed below. Analog part and digital block are well separated, so that digital switching noise influence is further reduced.

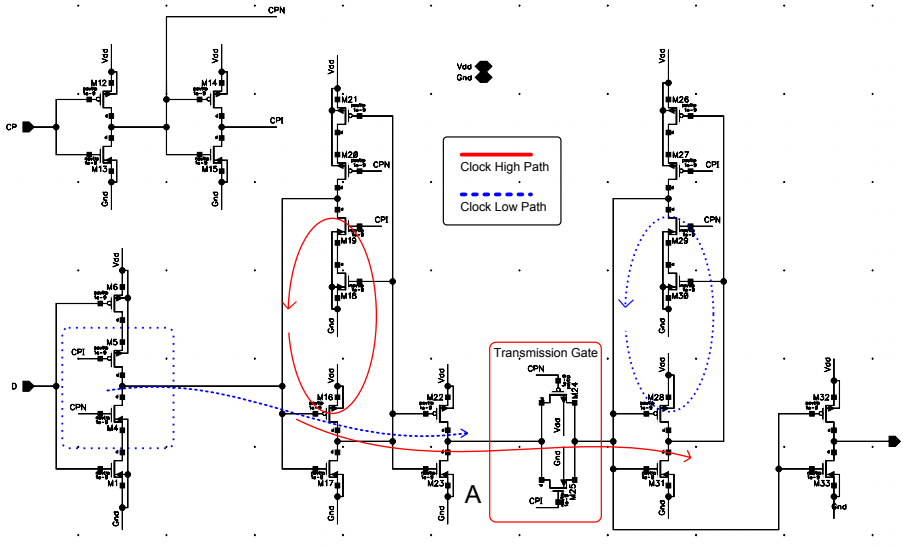
5.5 MUX

In this section, the MUX circuit placed before the first DAC will be discussed. It is a purely digital block which means that pre-designed standard cells can be used. Standard cells have equal height, and IOs are mostly placed at cell's boundary. Thus comparing with analog block design, the place and route effort can be reduced and the layout is less error prone. In principle, to match the Simulink model, a MUX block should contain three blocks:

- A ROM, generates a LUT to control each switch

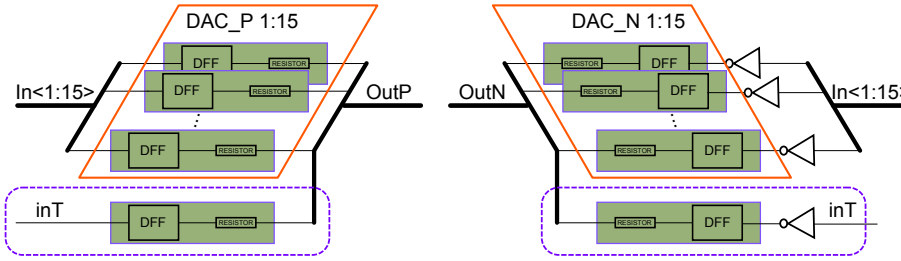


(a) Schematic

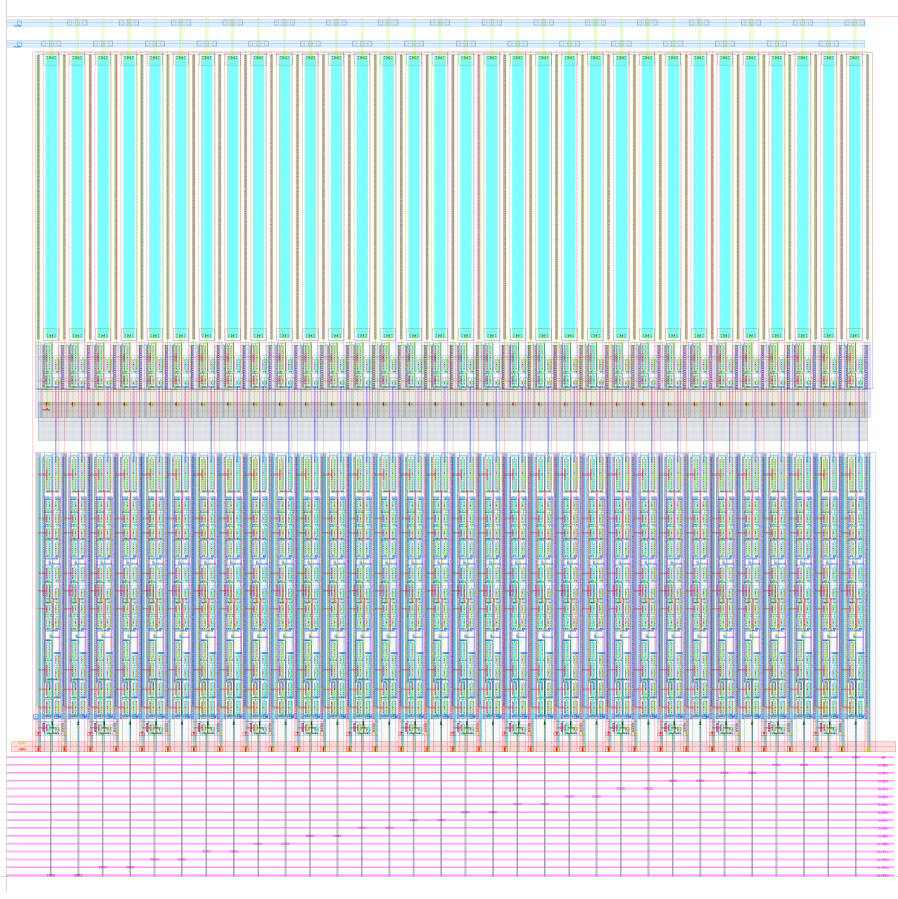


(b) DAC D-Flip-Flop

Figure 25: DAC unit element



(a) DAC concept diagram



(b) DAC layout view

Figure 26: DAC

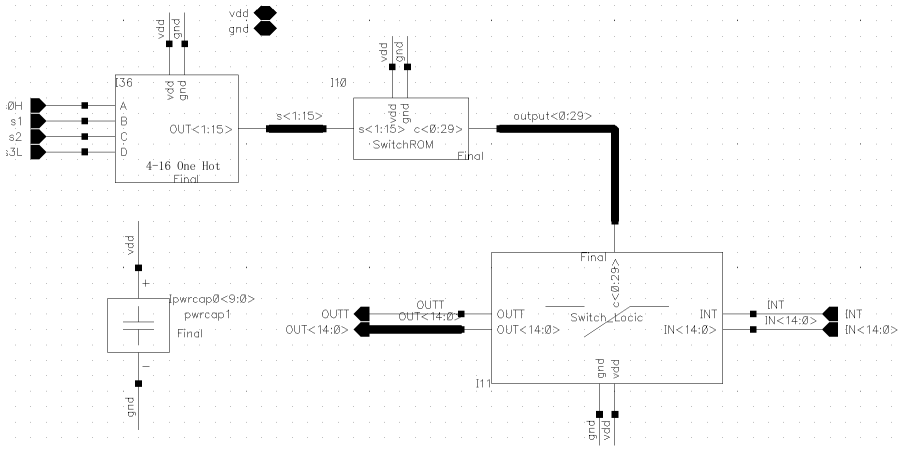


Figure 27: MUX

- Switch logic, the main part of MUX
- 4-bit binary input to 16-bit One-Hot code

Figure 27 is our MUX schematic. The 4-bit binary selection signal coming externally or from digital part is first converted to 16-bit one hot code. Then corresponding control word stored in the ROM is selected which controls signal routing in switching logic block. Detail discussion is performed in the following sub-sections.

5.5.1 ROM

Figure 28 shows the ROM schematic. There is nothing special in the structure, it only contains a MOS transistor matrix. The transistor existing on the junction or not determines corresponding word line(WL) value. For example when the bottom bit line(BL15) is '1', all transistors controlled by it will be on. Thus the corresponding WL will be pulled down and represents a '0', while the remaining is still '1'. So the result should be "01010101010101010101010101010100".

5.5.2 Switching logic

The main composition of the MUX is a switch logic. By changing signal routing according to control signals from ROM, test input signal would be injected into each unit element, as have already been illustrated in section 4.1.4. Figure 29 is schematic view. The internal 3-1 selector selects input signal injection either to A, B or C output depending on the value of control signal c(i+1),c(i).

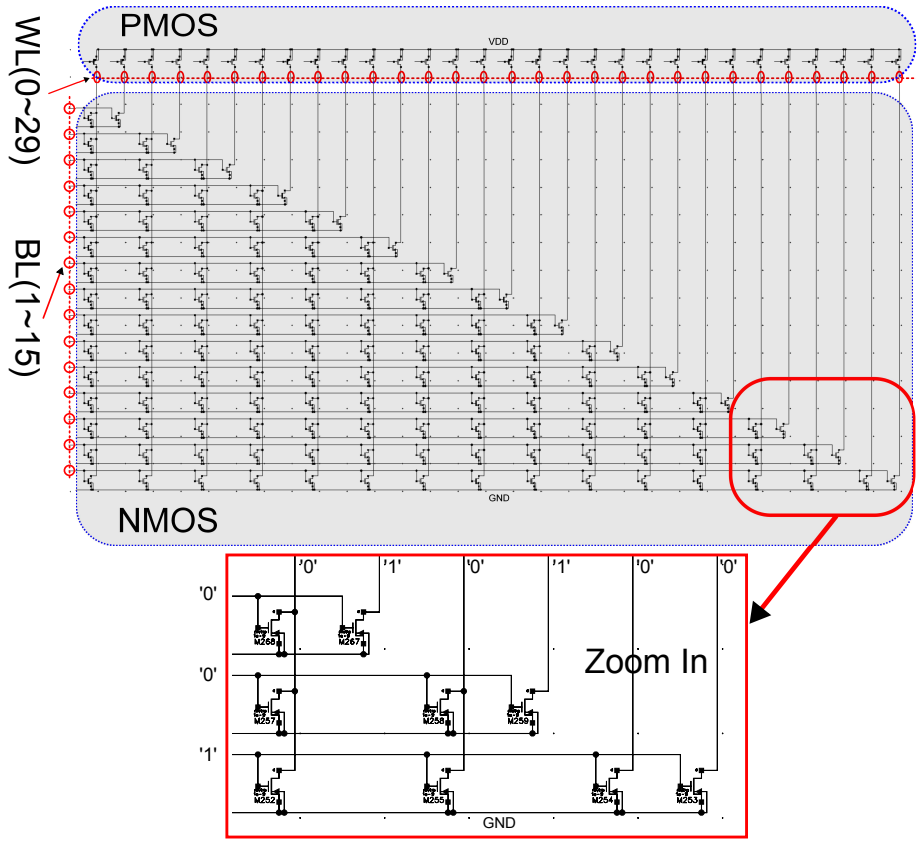


Figure 28: ROM schematic

5.6 Bias circuits

The final part in this chapter is about the bias circuit that supplying stable current used in all the on-chip circuits. In the OP-AMPs, $I_{in} = 16\mu A$ bias current is required to provide proper current for the other branches in the circuit. In the quantizer circuit, all the comparators require $I_{in} = 15\mu A$ bias current mirrored from an external current mirror, which also should be provided by this bias circuit. In the schematic shown in figure 30, a voltage follower is added to stabilize V_{GS} of the current mirror. It makes it so the NMOS transistor's gate voltage is precisely equal to the input reference voltage, regardless of the mirror's load. The current mirror generates a constant 10μA reference current and by mirroring this reference current using PMOS current mirrors, all the required current can be derived.

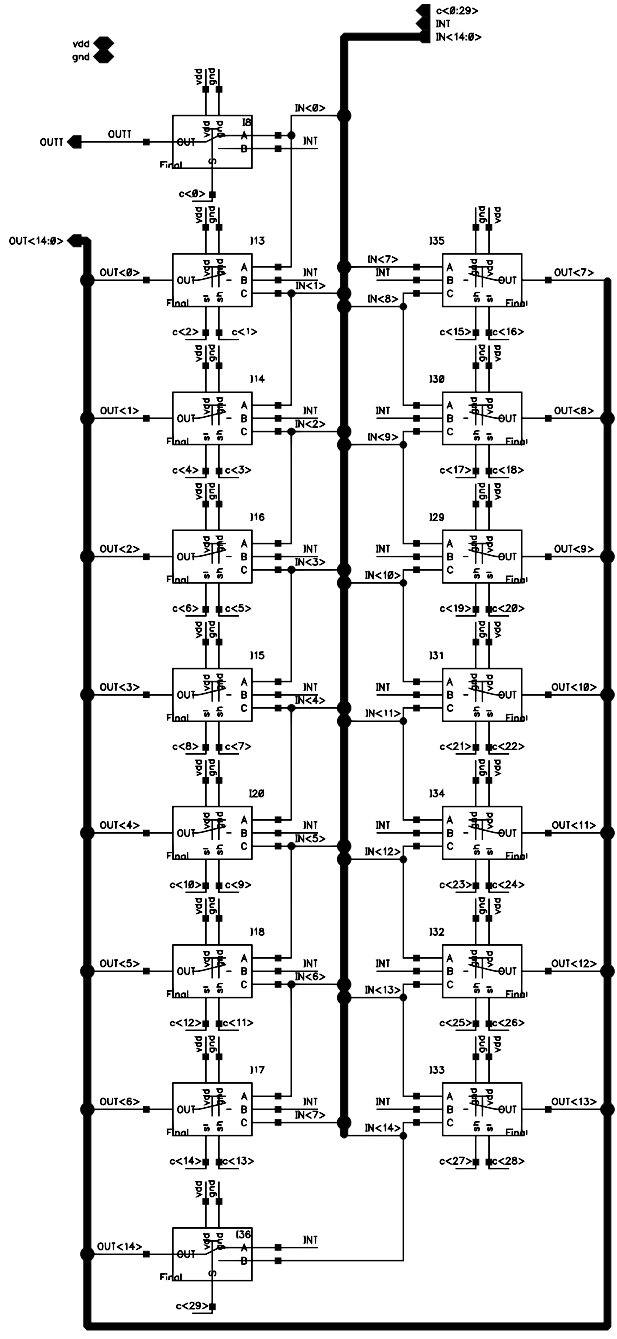


Figure 29: Switch logic schematic

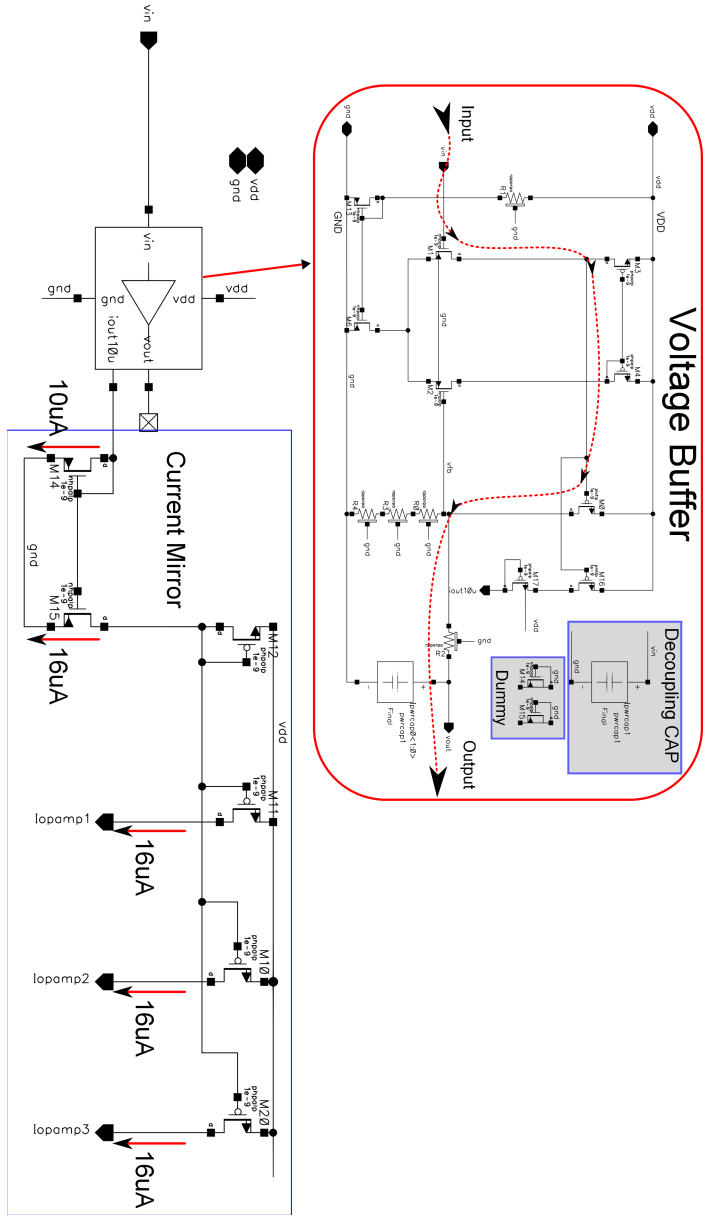


Figure 30: Bias circuit

Chapter 6

Circuit level considerations (Digital)

This chapter describes the implementation of the background digital calibration module from front end VHDL implementation to back end place and route flow.

6.1 VHDL implementation

The behavioral model is implemented in VHDL and simulated with Questa Sim from Mentor Graphics. Matlab modeling simplifies VHDL modeling and offers a trustable reference for the behavior as well. The behavioral simulation results are shown in this section.

6.1.1 Design overview

As introduced in chapter 3, the digital calibration system focuses on three modules: division, cross correlation function and mismatch coefficient generation. In addition to those three modules for calculating mismatch coefficients, there is also a module using the mismatch coefficients to calibrate the system. Furthermore, some surrounding logics, such as look up table for data format transformation, also exist in the system.

As previously shown in figure 1, the gray block on the right side illustrates the detailed block diagram of the digital calibration system. A more detailed diagram is shown in figure 31. At the beginning of the datapath, input signal Y_d is generated from the quantizer. It goes through a look up table in order to transfer a 15-bit thermometer code to a 5-bit signed number, which represent values from -15 to +15 with a step of 2. At the same time, the input test signal E_t is mapped from '0' or '1' to '-1' or '1' by adding a multiplexer when it is used. The reason for this transformation is that:

- Y_d is an unsigned signal of 16 possibilities representing integer value from 0 to 15 while E_t is 0 or 1. As the input is not symmetric around 0, calculation with these values would introduce an offset in the result of cross correlation function and lead to malfunction of the following calculation.

- On the other hand, calculation in Matlab model is done with signed value which is symmetric around zero. In order to match with Matlab model which can offer a reliable reference, Y_d is mapped from $[0, 15]$ to $[-15, 15]$ with an interval of 2 instead of 1, while E_t is mapped from '0' or '1' to '-1' or '1'.

After encoding the input signal Y_d to Y_{d_bin}, it is subtracted by the correction signal Y_c, generated by the feedback loop, results in Y_t. Y_t is the superposition of the calibrated signal and the test signal.

On the feedback path, Y_t is feed to the cross correlation function module named $CrosCorrFunc$ to calculate the cross correlation with E_{t_d3} which is E_t delayed for 3 clock cycles. Details about this calculation process will be discussed in the cross correlation function Finite State Machine section.

In the following, a set of data refers to 2^{18} samples.

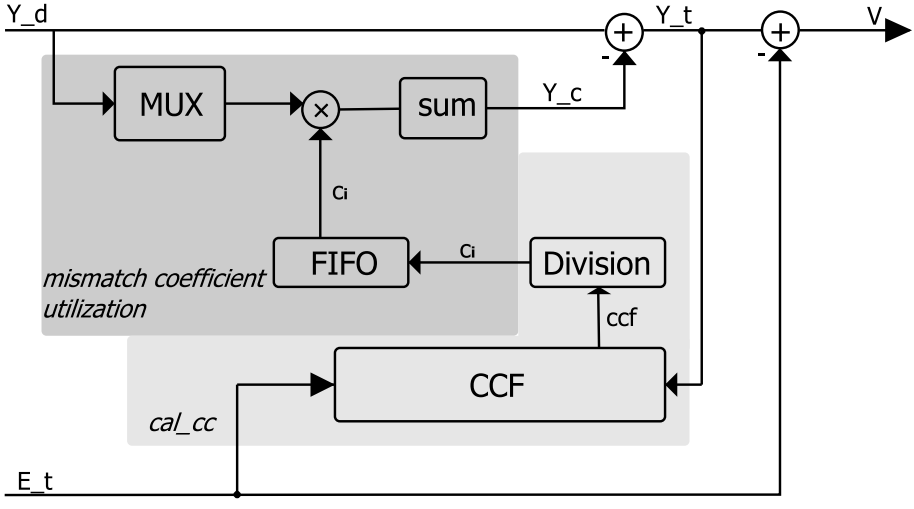


Figure 31: Digital system block diagram

Cross correlation function module outputs the value CCF_t and CCF_i sequentially. The first set of data of Y_t is used for calculation of CCF_t, and the following sets of data are for CCF_i.

The next module is *cal_cc* module, which calculates the correction factor c_i. There is another four bits output signal *sw_cnt* from this module indicating in which DAC element the test signal should be inserted. This signal will be feedback to the DAC as control signal on test signal insertion. Details about correction factor calculation will be discussed in section 6.1.4.

After correction factors are outputted from *cal_cc*, they are taken into *output_Y_c* module which is the utilization of the correction factor. In this

Table 8: Config signal truth table

config(1 downto 0)	input signal	output signal
00	Y_d	V
01	Y_{bin}	V
11	Y_{bin}	c
10	Y_d	c

module, the correction signal for Y_d which is named Y_c is calculated. In section 6.1.5, there are more discussions on this module. Then Y_d can be simply corrected by subtracting Y_c from Y_d, as equation 13.

From now on, Y_t is a corrected signal but combined with the injected test signal. Subtract the test signal E_t from Y_t is the last step for getting the final output signal V. In order to have more observation on the implemented system, the mismatch correction factors c_i can be outputted by setting a config signal properly. Setting for config signal is show in the follow table 8. Y_d is the signal output from the ADC modulator, Y_{bin} is 4-bit signal from external source, c is the calculated correction factor. At the same time, the correction system can be bypassed by setting *corr_en* signal to '0', and vice versa.

6.1.2 Cross correlation function Finite State Machine

The implementation of cross correlation datapath is demonstrated as figure 32.

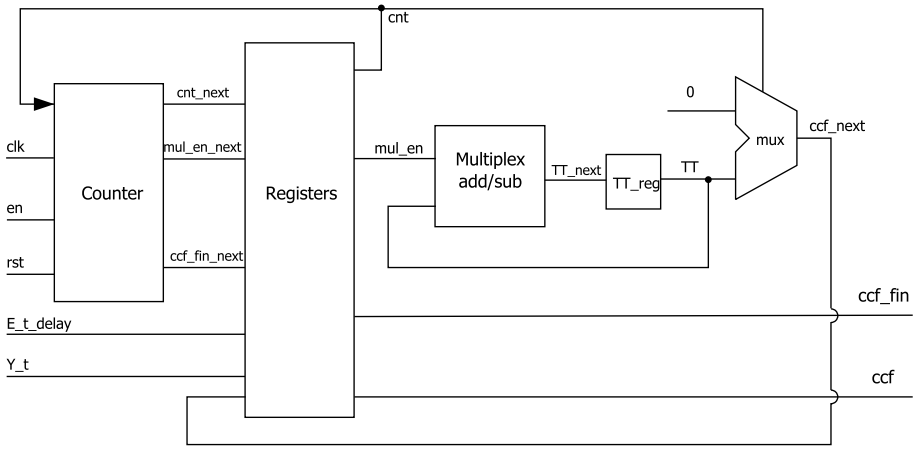


Figure 32: Cross correlation datapath

Figure 33 illustrates the FSM of cross correlation function module.

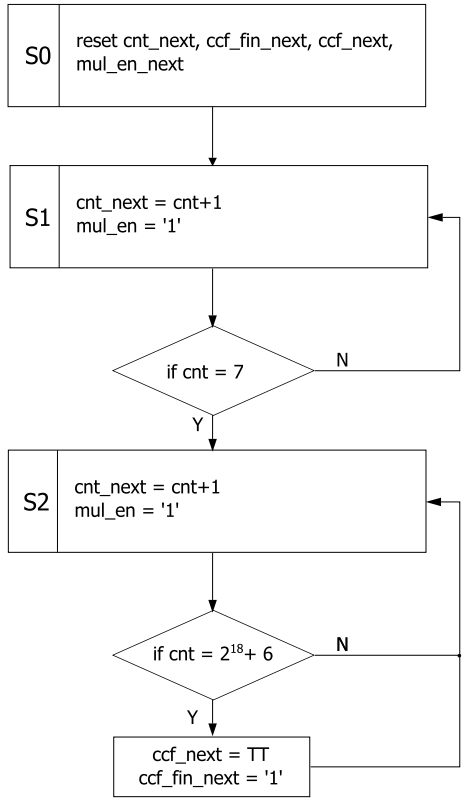


Figure 33: Cross correlation FSM

Because the $\Delta\Sigma$ modulator is using a 3-stage architecture which will introduce 3 clock cycles of delay on Y_t, test signal E_t also need to be delayed by 3 clock cycles to fit Y_d. E_{t_d} is the delayed test signal.

The implementation of equation 5 is done in two steps: the multiplication and then the accumulation. In the multiplication, the multiplicand E_{t_d} is either '0' or '1' which is mapped to (-1,1). Thus, the signal Y_t will only change sign accordingly. Therefore the accumulation and the multiplication in equation 5 can be performed with equation 16. And it is implemented with a multiplexer with E_{t_d} as select signal.

$$TT_{next} = \begin{cases} TT + Y_t & E_{t\_d} = `1' \\ TT - Y_t & E_{t\_d} = `0' \end{cases} \tag{16}$$

In equation 5, TT is the result of accumulation, TT_{next} is the registered value of TT. After accumulating for a set of data Y_t, the result TT then becomes

the CCF value of that specific set of data. And this CCF value is outputed for futher computation.

6.1.3 Division Finite State Machine

In this section, the implementation of divisor is introduced.

The division module is built according to Newton-Raphson algorithm as introduced in section 3.3. It estimates the reciprocal of CCF_t which is the crosscorrelation function used as CCF_{ref}. CCF_t is calculated with the first set of data, and will remain constant for the remaining 15 sets of data to compute $\frac{CCF_i}{CCF_t}$ in equation6 . Therefore, division only need to be calculated once every 16 sets of data.

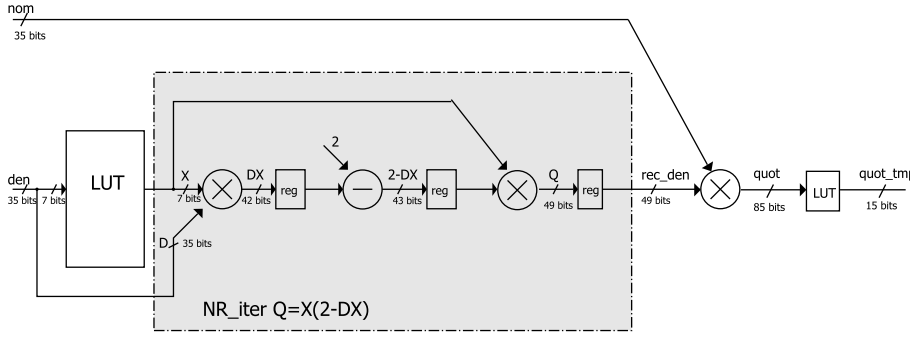


Figure 34: Division datapath

Before starting the first estimation of $\frac{1}{CCF_t}$, CCF_t is shifted to format [1,34], and den_{shift_cnt} records the number of shifts. CCF_i is shown as signal *den* and signal D in the datapath of division figure 34. The look up table (LUT) contains the precalculated mapping from 7 bits divisor to 7 bits reciprocal of the divisor. The first estimation is done by that look up table with $CCF_t(33\ downto\ 27)$ as input. The output of the LUT is signal X in figure 34. Finally, the iteration is used for a second estimation. In order to simplify the implementation, only one iteration is implemented. That's because iteration may significantly increase the output word length due to division. The FSM of this iteration is illustrated in figure 35. In this implementation, the output word length from the division after one time iteration becomes 49 bits wide. By multiply with the 35 bits wide nominator, the result for $\frac{CCF_i}{CCF_t}$ is in total 85 bits wide. As different shifts are done and recorded by den_{shift_cnt}, this 85 bits result need to be restored with those shifts. On the other hand, 85 bits signal is too wide for the following computation. Therefore, another look up table is used to select the validate bits to be outputted. In this look up table, the mapping from different shifts to the selected 15 out of 85 bits division result is conducted. Finally, 15 bits of

the 85 bits signal in format [1,14] is outputted as a result of the division.

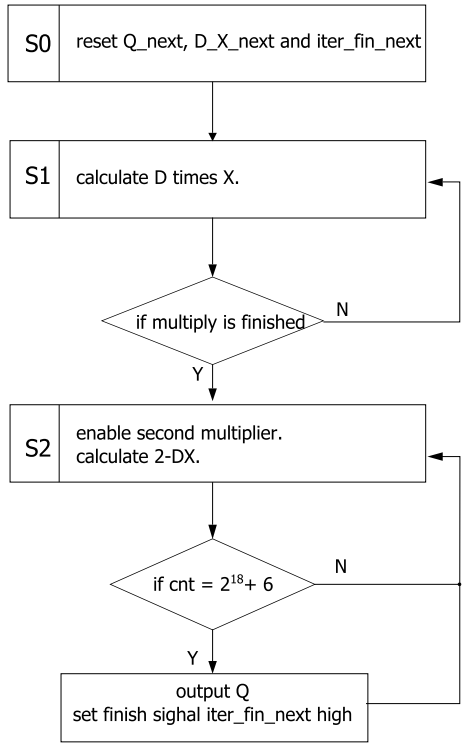


Figure 35: Newton Raphson iteration FSM

6.1.4 Mismatch coefficient generation Finite State Machine

In this section, the generation of the correction factor is introduced.

According to equation 6, CCF_i and CCF_t should be determinated place in order to compute the correction factor c_i. There is a counter *cnt* in module *cal_cc* to demonstrate the sequence of all the $CCFs$. The first CCF computed by $CrossCorrFunc$ is taken as CCF_t, and this CCF_t is then streamed to the division module for computing $\frac{1}{CCF_t}$. The following $CCFs$ are regarded as CCF_i. Then $\frac{CCF_i}{CCF_t}$ in equation 6 can be easily computed by multiply CCF_i and $\frac{1}{CCF_t}$. The subtraction in equation 6 requires format transformation of the subtrahend "1". This subtrahend should be in format [2,14], thus it is transformed into "0100000000000000". Then, the multiplication with the ideal gain factor k_{1b} of the DAC, is straight forward. Therefore, correction factor c_i is calculated, and it in format [2,14].

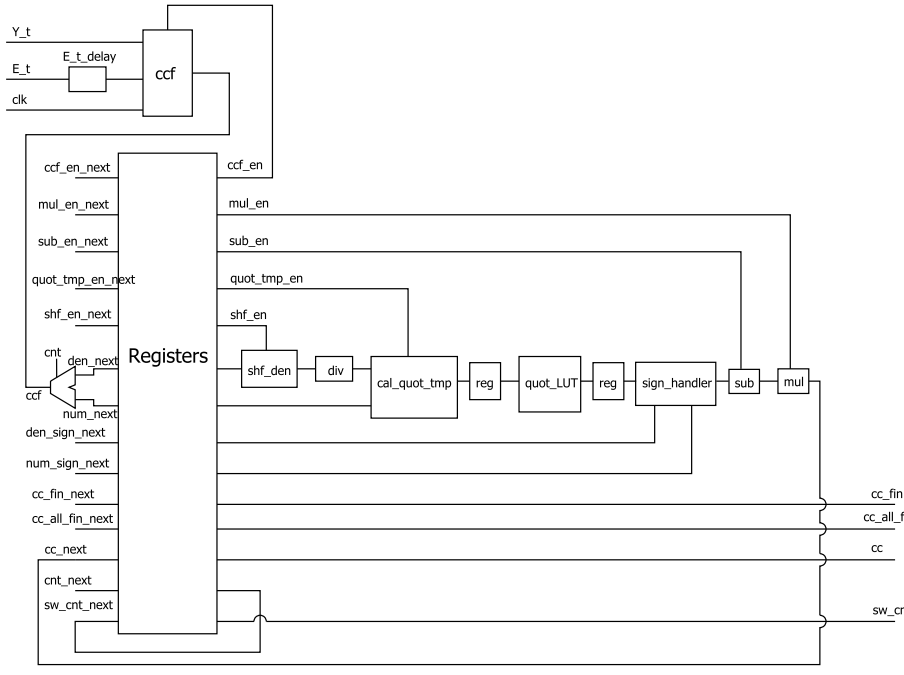


Figure 36: Mismatch coefficient generation datapath

The counter sw_{cnt} counts which unit element in the DAC is the correction factor c_i is corresponding to. This sw_{cnt} is then outputed to control the sequence of test signal insertion in the DAC. The datapath of this module is illustrated in figure 36, and the FSM is illustrated in figure 37.

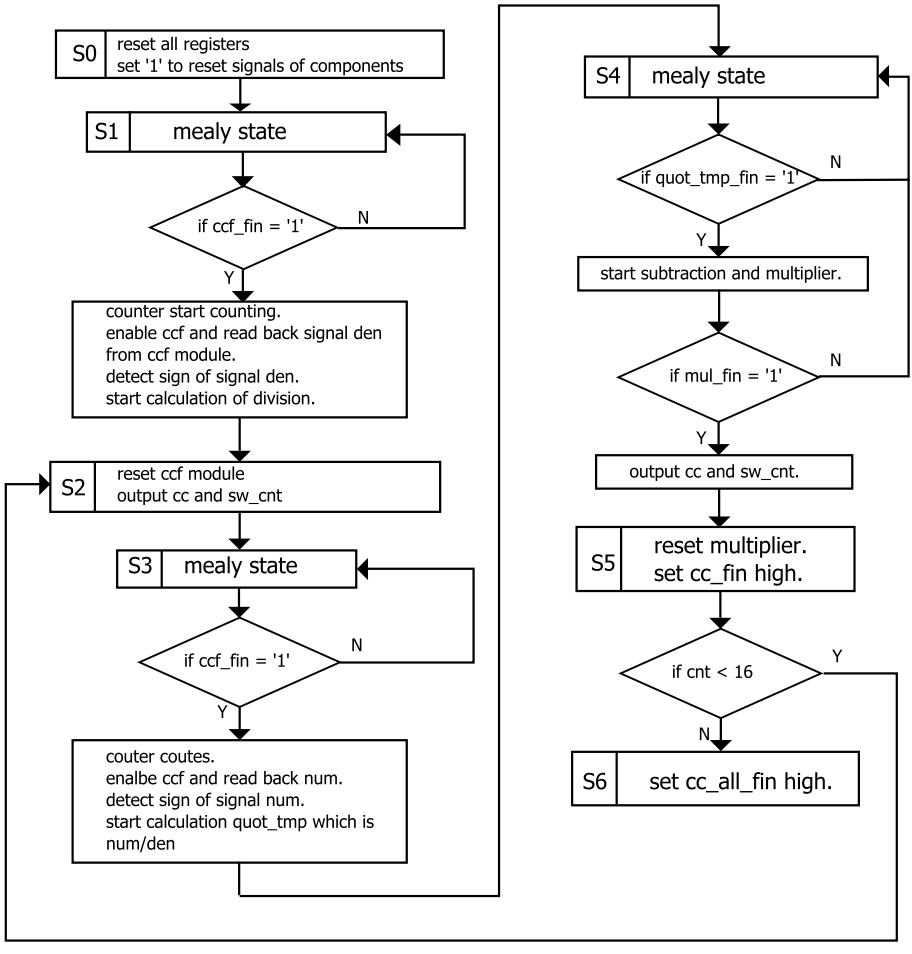


Figure 37: Mismatch coefficient generation FSM

6.1.5 Mismatch coefficient utilization Finite State Machine

As shown in figure 1. There are in total 16 unit elements in the DAC, 15 orginal DAC1 elements and 1 test DAC1 element. c_i is the correction factor for the i^{th} unit element in the DAC correspondingly. The next step is to implement

equation 6, it seems straight forward. However, the test signal inserted in DAC1 propagats throught the modulator, carries the mismatch error of the unit element it is feed into, and affects signal Y_d. The mismatch error generated with the test signal also need to be corrected. Therefore, the test signal should also engage the calculation of the correction value Y_c. For example, when test signal is inserted in to the i^{th} element in DAC1, it should be corrected with correction factor c_i during computation of Y_c. This procedure is implemented with the MUX mentioned in section 5.5 for switching test signal into the DAC1. Signal sw_{cnt} determines which unit element is inserted with test signal. Therefore, Y_d becomes Y_{d_tmp} by inserting test signal into it before entering the correction unit.

Hardware implementation of this module is more complex than implementing equation 6 itself. The datapath of it is shown in figure 38. Firstly, the correction factors are calculated on the fly, one by one while we expect to used it in the equation in parallel. Thus, a FIFO-like memory is implemented for temporarily store the correction factors. The memory is structured with two stages of registers. First stage stores the correction facter c_i into dedicated registers, the finish signal cc_{fin} is used as write enable signal in this stage. Second stage outputs all 16 c_i in parallel when they are calculated, signal cc_{all_fin} controls the output of this stage. Actually, from cal_{cc} module, cc_{fin} signal is triggered only for 15 times. Because the first two set of data outputs one c_i while each of the following set of data leads to one c_i. The extra c_i not calculated is the correction factor of the reference unit element also known as test unit element. This correction factor c_t equals 0, as it is considering relating CCF_{ref} with itself. Or in another word, it is taking CCF_{ref} as CCF_i in equation 6.

Then, the MUX, or switch in the figure, mensioned above is implemented. Afterwards, signal Y_{d_tmp} is used in the multiplication bit by bit, thus we simplify the multiplication to a multiplexer with each bit of Y_{d_tmp} as select signal. As shown in equation 17.

$$c_{Y\_r} = \begin{cases} c_i & Y_{d\_tmp} = \text{`1}' \\ -c_i & Y_{d\_tmp} = \text{`0}' \end{cases} \tag{17}$$

where c_{Y_r} is the internal signal represent the multiplication result of $Y_{d\_tmp}(i)$ and c_i.

Finally, the remaining part of equation 11 is the summation of 16 products.

6.1.6 Behavioral simulation results. Matlab and VHDL simulation match.

The testbench for behavioral simulation is build only for the digital calibration, the $\Delta\Sigma$ ADC is not modeled. The 15-bit thermocode input signal Y_d for

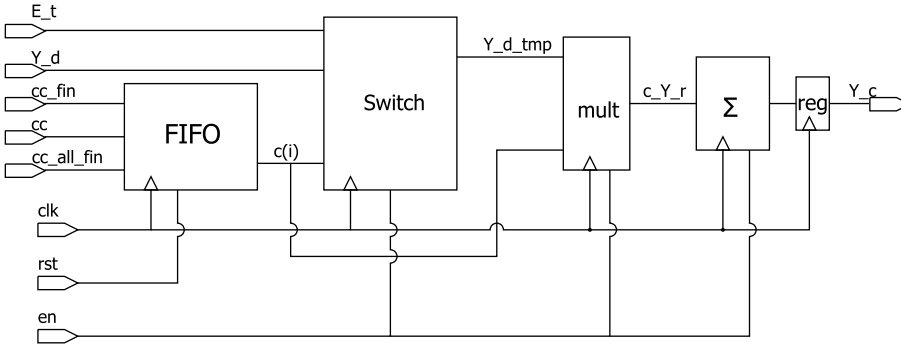


Figure 38: Mismatch coefficient utilization datapath

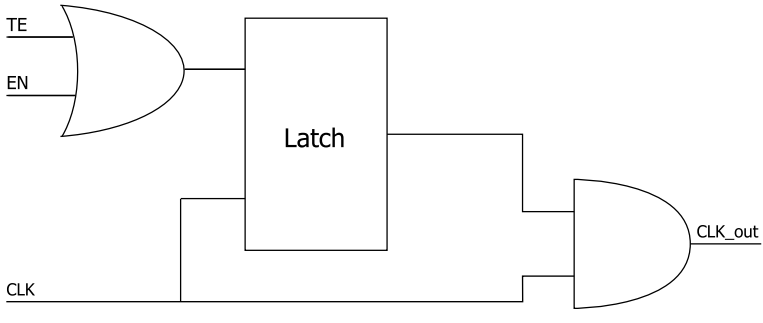


Figure 39: Clock gating structure

the simulation is generated Matlab simulink model. And the behavior is also compared with the result from Matlab simulink model.

The FFT of output signal V from both Matlab model and VHDL behavioral simulations are plotted in figure 40. When the first set of data is input in to the system, the mismatch correction factor is under calculation. So that, the output, as plotted in figure 40a is not corrected for mismatches in the DAC. Figure 40b illustrates the FFT after the first iteration, when correction is introduced. SNR, SNDR and SFDR rises compared with figure 40a. Figure 40c presents the result after the second iteration. The mismatch correction is becoming more accurate, but the performance is only marginally improved. It is clear that after two iterations, the background calibration can be bypassed as the accuracy imporvement is negligible in further iterations. With these simulation results, we can conclude that the VHDL behavioral model performs the same as the Matlab model.

6.2 Optimization and synthesis

6.2.1 Optimization in synthesis flow

After the behavioral check of the design, synthesis on the design is performed. In this phase of design, the main objective is to fit timing. The first trial synthesis with high threshold voltage (HVT) cells did not fit timing. Therefore, two methods are tried for fitting timing:

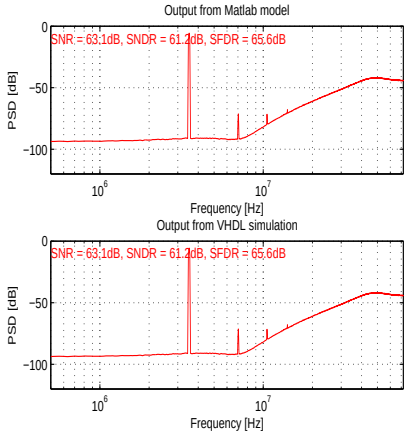
- Change the libraries used for synthesis.
- Optimize the design with pipelining.

As cells in library with HVT is slower than the ones in standard threshold voltage (SVT) or low threshold voltage (LVT) libraries, it is easier to fit timing with SVT library.

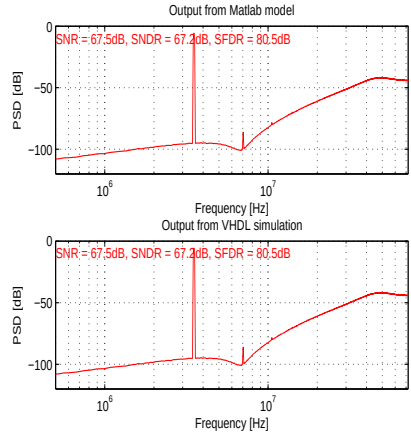
There are two way to implement a pipeline. One is automatical pipelining by the tool Design Compiler (DC) with the command *optimize_registers*. However, this kind of optimization does not work well with feedback loop. Feedback loop appears in our implementation: signal Y_t is calculated in relation with correction factor c_i while c_i is computed from Y_t and E_t.

The failure on automatic pipelining leads to the other path of pipelining. The top design module is tore down to small design modules and pipelined separately.

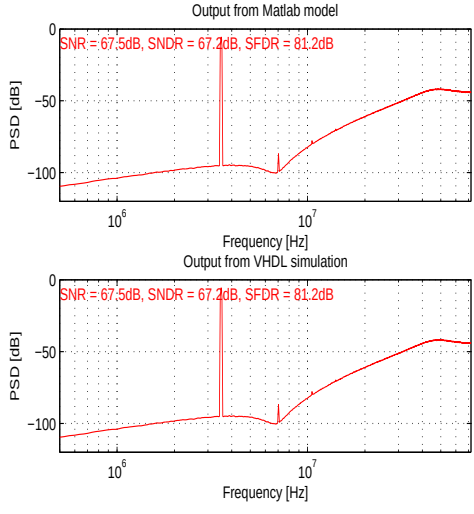
According to the synthesis timing report without pipelining, the critical path is along the summation for calculating Y_c. Then registers are inserted at the output of the summation. With those inserted registers, the compiler can cut down the critical path by moving them into the summation block. This is the mechanism of command *optimize_registers*. Other paths can be optimized



(a) 1^{st} set of data



(b) 2^{nd} set of data, 1^{st} iteration



(c) 3^{rd} set of data, 2^{nd} iteration

Figure 40: Behavioral simulation results

in the same way. As estimated by the endpoint slack check, most of the longer paths are related with *division* module. Then optimization is required on these modules and blocks which can be implemented as mentioned before. After the optimization, there are enough slack for timing of the design. To prevent the tool from modifying the optimized modules in the following synthesis steps, variable *dont_touch* are set on those optimized blocks.

On the other hand, there are 86% of the logics in the design are sequential logics, thus clock gating can be done to optimize the power consumption.

Clock gating is a technique generally used for saving dynamic power dissipation on flip-flops by gated-off the input clocks. It is implemented with a combination of AND gate and a latch to avoid glitches on the clocks. The structure is illustrated in figure 39, where TE is test enable input, EN is enable input, CLK is clock input, CLK_out is the gated clock output.

In this implementation, clock gating is done by inserting intergated clock gating cell into the module *cc_cal* to prevent the clock ticking of some blocks in the module. Division module is not always in use. As introduced in section 6.1.3, it is invoked once every 16 sets of data. Therefore, the clock for division module can be turned off when it is not in use. The multiplication for $\frac{CCF_i}{CCF_{ref}}$ is also not in use for most of the time. It is called once every set of data.

Besides the modules mentioned above, other registers are clock gated automatically by Design Compiler. The tool determines if the register is capable with clock gating.

All commands for compile and synthesis are saved as scripts for replication of the synthesis flow.

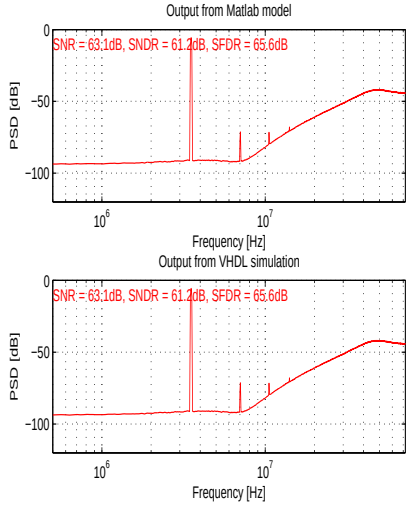
6.2.2 Post synthesis simulation results

After synthesis, post synthesis simulation is performed to check the functionality of the design. The simulation is done with the same testbench environment as the behavioral simulation in section 6.1.6. Test subject is the verilog netlist generated by Design Compiler. The result in figure 41 shows the same performance as the behavioral simulation. Thus, the netlist generated from synthesis flow can be used for place and route.

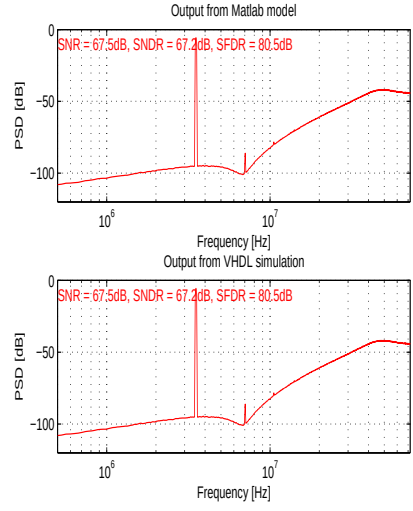
6.3 Place and route flow

6.3.1 Place and route

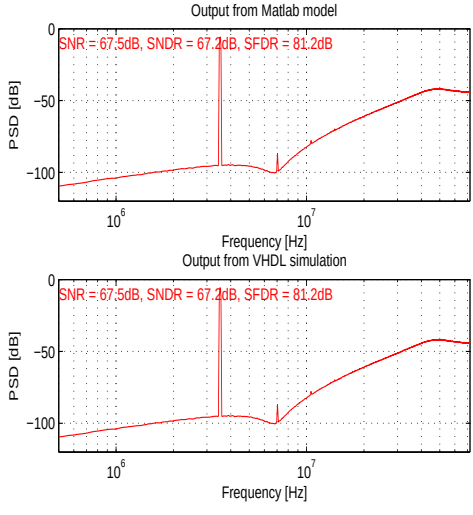
Digital place and route is implemented with Cadence Encounter in a 65 nm process from STMicro electronics. Standard libraries used are listed in table 10. The dimension of the die is 1000um $\times$ 1000 um. 53 pads are distributed along the edge of the die in this implementation. Details of the distribution of digital pads are listed in table 9.



(a) 1^{st} set of data



(b) 2^{nd} set of data, 1^{st} Iteration



(c) 3^{rd} set of data, 2^{nd} Iteration

Figure 41: Post synthesis simulation results

Table 9: Pads list for digital core

Pad name:	Placement	Description
Bottom row, left to right		
Pad: VDD_S	S	VDD south
Pad: GND_S	S	GND south
Pad: DATAIN_PAD2CORE_3	S	Data input(3)
Pad: DATAIN_PAD2CORE_2	S	Data input(2)
Pad: DATAIN_PAD2CORE_1	S	Data input(1)
Pad: DATAIN_PAD2CORE_0	S	Data input(0)
Right bottom to top		
Pad: DATAOUT_CORE2PAD_15	E	Data output(15)
Pad: GND_DIG_E	E	GND east
Pad: VDD_DIG_E	E	VDD east
Pad: DATAOUT_CORE2PAD_14	E	Data output(14)
Pad: DATAOUT_CORE2PAD_13	E	Data output(13)
Pad: DATAOUT_CORE2PAD_12	E	Data output(12)
Pad: DATAOUT_CORE2PAD_11	E	Data output(11)
Pad: DATAOUT_CORE2PAD_10	E	Data output(10)
Pad: DATAOUT_CORE2PAD_9	E	Data output(9)
Pad: DATAOUT_CORE2PAD_8	E	Data output(8)
Pad: DATAOUT_CORE2PAD_7	E	Data output(7)
Pad: DATAOUT_CORE2PAD_6	E	Data output(6)
Pad: DATAOUT_CORE2PAD_5	E	Data output(5)
Top row left to right		
Pad: RST_PAD2CORE	N	reset
Pad: CONFIG_PAD2CORE_1	N	config(0)
Pad: CONFIG_PAD2CORE_0	N	config(1)
Pad: EN_PAD2CORE	N	enable
Pad: HOLD_N_PAD2CORE	N	hold not
Pad: DATAOUT_CORE2PAD_0	N	Data output(0)
Pad: DATAOUT_CORE2PAD_1	N	Data output(1)
Pad: DATAOUT_CORE2PAD_2	N	Data output(2)
Pad: DATAOUT_CORE2PAD_3	N	Data output(3)
Pad: DATAOUT_CORE2PAD_4	N	Data output(4)

The digital core occupies 230.8 um × 504.4 um area, which is 116415.52 um^2, leads to 57% core utility. It is placed at the south-east corner of the die. There is a 26 um gap in the core along one power domain which decreases the core utility to 70%.

In order to have a better investigation of the power consumption, the design of the digital part occupies two power domains. One of the them supplies the core and the other is for the input buffers, output multiplexer, output buffers and the pads. The latter one is shared with the analog part as surrounding supply. CPF file is used to setup the power domains in this implementation.

When the power planning with special routing is finished, well-taps are added for reverse bias on nwell and pwell to avoid latchup. Well-tap cells are picked from $PRHS65\_7.0.a$ library listed in table 10.

Then, the phycial cells are placed. Pins for communication with the analog part are placed at the specific location. The design is optimized for fixing the DRC violations include max capacitance violations, max fan-out violations and hold violations. As the placement of the cells is a non-deterministic process, the design optimization may sometimes not be able to fix all the violations. If it happens, the optimization is repeated at this stage until all the violations are fixed.

With the DRC violation clear, the clock tree synthesis is performed. Clock buffers for CTS are picked from $PRHS65\_7.0.a$ library, which is listed in table 10.

Table 10: IP libraries used

Libraries:	
Core and clock related:	CLOCK65LPSVT CORE65LPSVT
Fillers and well-taps:	PRHS65_7.0.a
IOs and IO fillers:	Pads_Oct2012 (custom designed pads by Oskar Andersson)

As the digital core is placed at the south-east corner of the die, the north-east corner is left blank for placing decoupling capacitors. There are 10 digital pads on the north side, 6 pads on the south side and all 13 pads on the east side are used by digital core. The north-east part of the die should not be included for routing by the tool. Therefore, routing blockages are added.

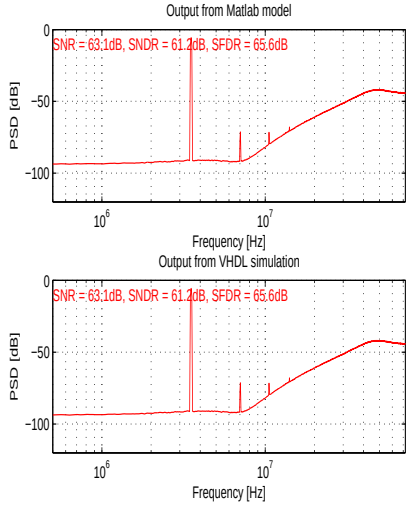
Afterward, a routing script is invoked. Runtime for routing is much longer than the previous procedures, as a better routing result is not achieved by route only once. In our case, the design is routed 5 times. First route involves basic settings with MultiCutVia disabled. While second route mainly focus on optimize via. Third route is a detailed route with MultiCutVia enabled. Final route optimize the route. For a better TPA (timing power and area) estimation

result, ExtractRCMode is set with qrc scripts. Then the design is optimized again to fix violations.

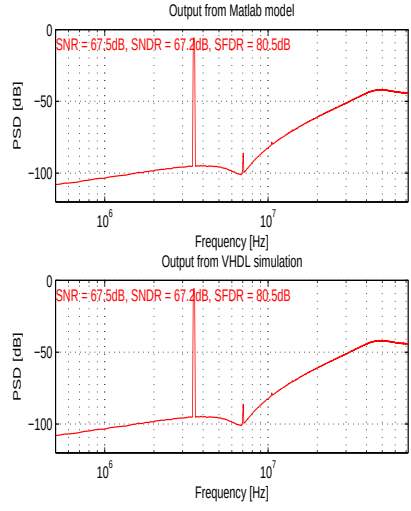
Adding fillers, generating the netlist and gds2 file are the last two steps in place and route. Fillers are picked from the same library as clock buffers, $PRHS65\_7.0.a$, listed in table 10. The generated netlist is used for post layout simulation. While the gds2 file is streamed in to Cadence Virtuoso for combination with the analog circuit. The final signoff is done with Cadence Virtuoso.

6.3.2 Post layout simulation results

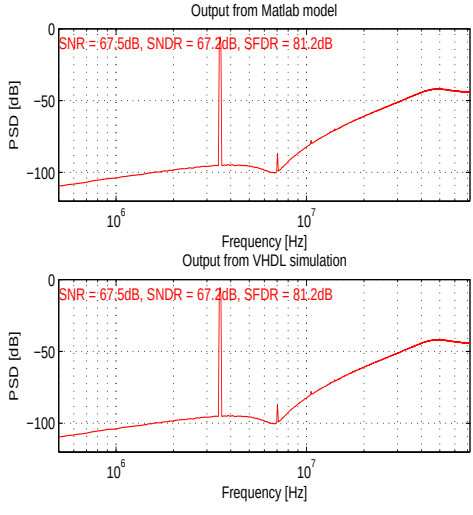
Finally, the post layout simulation on the netlist generated from place and route is performed with the same test bench environment. The result of it is illustrated in figure 42. The same performance is achieved as the behavioral simulation and the post synthesis simulation, which proves that the design is implemented as desired.



(a) 1^{st} set of data



(b) 2^{nd} set of data, 1^{st} Iteration



(c) 3^{rd} set of data, 2^{nd} Iteration

Figure 42: Post layout simulation results

Chapter 7

Post layout simulation and conclusion

This chapter concludes this work. First, a layout view of the chip is presented, followed by post-layout simulation. Finally, a brief introduction of Cadence AMS simulation is covered.

7.1 Layout view of the chip

To begin with, let's have a look at our chip layout in figure 43. The chip occupies 1mm × 1mm space, and in total 53 pads are placed surrounding the core. The circuits in the left red box is the analog modulator, while circuits in the right blue box is the digital calibration part. The digital block is generated by *SoC Encounter*, which is a program that place and route standard cells automatically. Obviously, digital part's layout density is much higher. The analog block, on the other hand, is drawn manually. There is no such 'standard cell' in the analog domain. Much more considerations are taken so that parasitic capacitance influence is minimized. For example, the metal wire should be in appropriate width and transistors are placed next to each other. As could be clearly seen, in the gaps, there are a lot of square capacitors to decouple the power supply.

7.2 Post layout simulation

In this subsection, post layout simulations in Cadence 'Virtuoso' is presented. First, the analog modulator's performance is shown. In circuit level, the non-ideal amplifiers in the integrators cause a SNR degradation. The additional amplifier-introduced delay could cause serious excess loop delay, which should be compensated by optimizing coefficients as already discussed in section 2.5. Comparing with the schematic level simulation, the layout has more parasitic components, causing the delay and GBW of the amplifiers to change a little bit. Thus, the coefficients we calculated based on non-idealities found in the schematic level circuit would not perfectly compensate the parasitics introduced in the layout phase. Therefore, running a post layout simulation are important. Secondly, the AMS simulator is needed to do a co-simulation of the

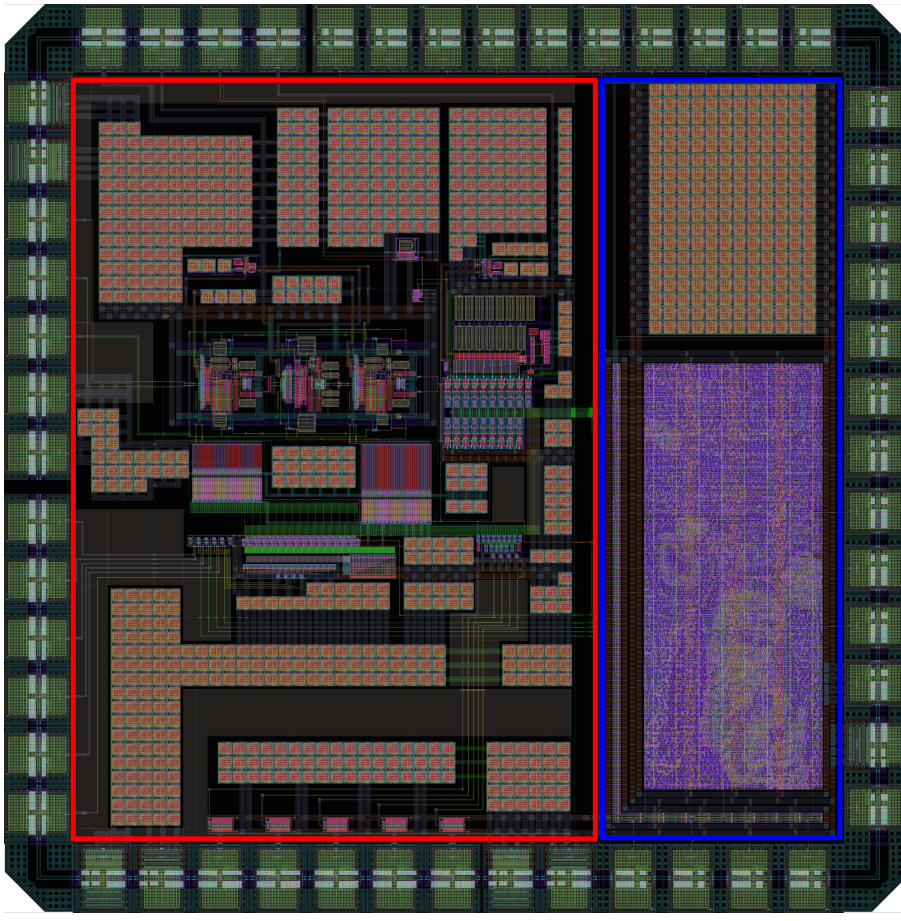


Figure 43: Final chip layout

combination of the analog and the digital part. It is essential to make sure the inter-connect wires between the two parts is correct. To do such a simulation, the VHDL behavioral model is imported to Virtuoso as a top block and is connected to analog part in the test bench. Due to the long sample length required for correction coefficients calculation, we shorten the simulation time and only monitor the switch signal coming from the digital part. If the switching pattern is as desired, we can conclude that the digital VHDL block is working. The correctness of the inter-connection between analog modulator and digital correction block is also checked.

7.2.1 Analog modulator simulation in Spectre simulator

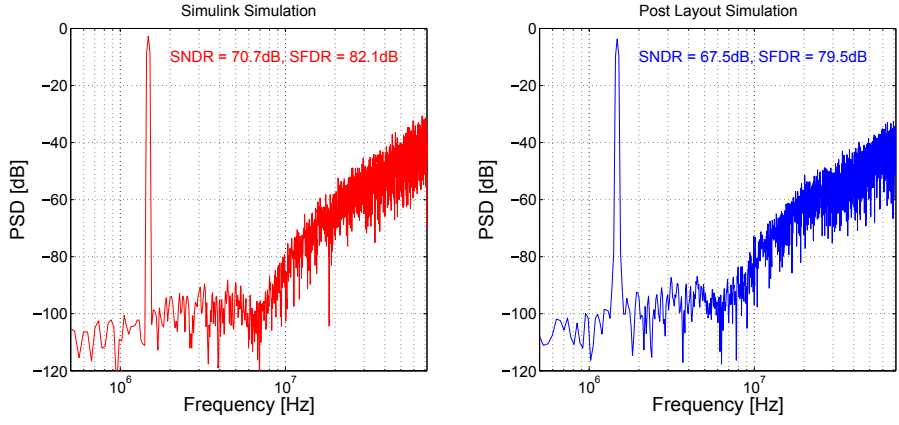


Figure 44: Matlab simulation vs. post layout Simulation

The more accurate way of predicting the modulator's performance is running a Spectre simulation in Cadence. Comparing with the simulation in Matlab, however, the real non-ideal circuit has worse performance. Thus a comparison between post layout result in Cadence and result from Matlab, is necessary. To speed up the simulation and make full utilization of all CPU cores, Cadence® Spectre® Accelerated Parallel Simulator (APS) is used. It is tightly integrated with the Virtuoso custom design platform and allows us to simulate the design faster. More than 10 times simulation time is saved with identical simulation accuracy compared with the basic mode. Figure 44 shows simulation result from both Simulink and Cadence model. From the figure, we see that when the structure is implemented in a real circuit, 3dB SNDR is lost due to all the non-ideal effects.

7.2.2 Circuit level simulation in AMS simulator

This part mainly focus on AMS simulation. Analog circuit simulation is performed in Cadence tool Virtuoso while digital VHDL model simulation is usually done in digital verification tools such as Modelsim. In our project, however, both analog and digital circuit are integrated in one single chip, but their layout are designed by different tools. Without a co-simulation, we could never feel confidence that the inter-connection is correct. Running the traditional circuit simulation in Spectre is not enough because it can not handle VHDL code. Cadence AMS simulator, which is an ideal tool for fulfilling such requirement, is used in our project design phase. It regards digital VHDL blocks as a black box which can be added in analog schematic. By connecting analog part and digital part properly, a co-simulation testbench will be constructed that makes circuit level system simulation possible. Figure 45 is the test bench.

To start AMS simulation, first, VHDL code should be imported to Virtuoso. The crosscorrelation length in the code is greatly shortened to speed up the simulation. A digital top level symbol appears in the library together with all the components used in the top-module. Then, in a new testbench, we add the existing analog top-module and already imported digital top-module. We finish connection between these two blocks afterwards. In figure 45, analog part is our $\Delta\Sigma$ modulator which provides the signal to the digital part on the right. An additional 4-bit DAC is used to convert the digital output signal back to the analog domain for monitoring the output in the *Wave Form* window. Finally, we launch ADE XL tool from schematic, choose simulator as AMS and set 'Files on irun command line' in AMS option to the proper VHDL package path. We start the simulation and when successfully finished, the output waveform is checked to see if it is as expected, see figure 46. We know that switching signal 'Internal SW (3:0)' shifts normally and the MSB, LSB definition in digital part is correct.

Further more, some large arbitrary selected component mismatches are added in the first DAC, and identically in the Simulink model. The same four-bit external selection SW signal pattern is applied in both simulations. The result shown that the signal definition of the analog SW input is correctly designed. It proves that the digital part can properly control the MUX (MUX has been discussed in chapter 5.5) and the interconnection is correct. This schematic view can be used as a guide when doing manual wire interconnection in top level layout later on.

7.3 Conclusion

This work has been implemented in 65nm process and a chip has been fabricated. The analog modulator performance is similar to the result from the ideal DT modulator Matlab simulation. ENOB of our system is 10.92, where

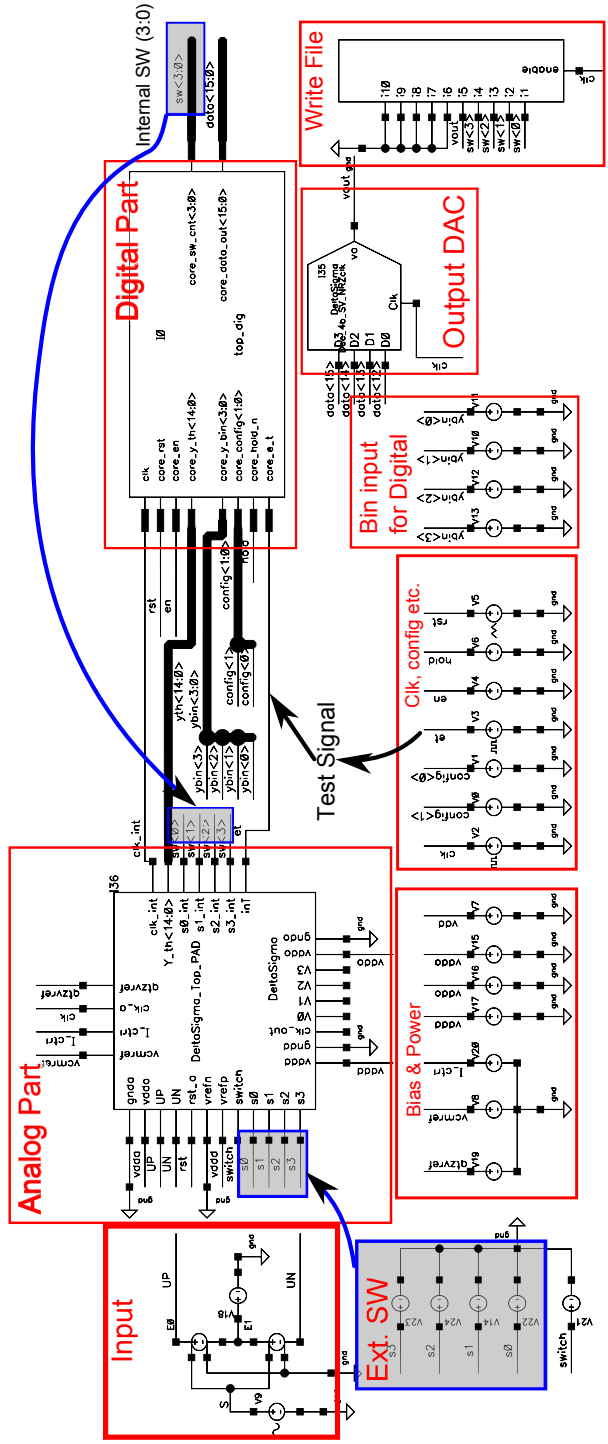


Figure 45: Co-Simulation test bench

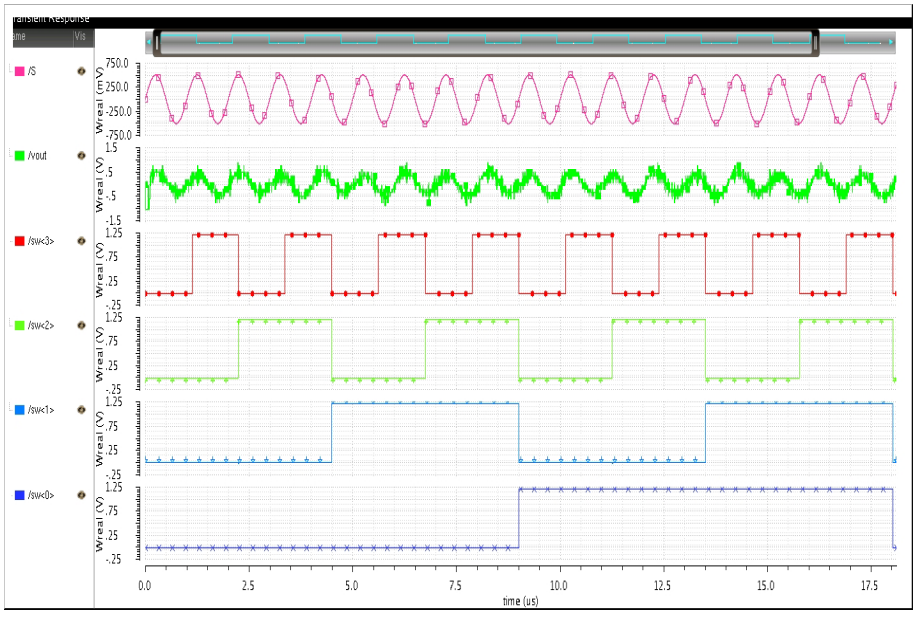


Figure 46: Co-simulation result

ENOB is given by

$$ENOB = \frac{SNDR - 1.76}{6.02} \tag{18}$$

and a *SNDR=67.5dB* from post-layout simulation. The power consumption is listed in table 11:

The total power consumption is 6.861mW. Thus, Figure of Merit(FOM) is:

$$FOM = \frac{P}{(2^{\mathrm{ENOB}} \times 2f_{\mathrm{B}})} = 196.73fJ/conversion \tag{19}$$

Table 11: Power consumption

Digital power(in quantizer and DACs)	1.914mW
Analog power(loop filter)	3.658mW
DAC reference power	0.139mW
Digital correction circuit	1.15mW

Although the FOM is not small enough comparing with the state of art, in table 12, the concept of digital correction method is innovative and there exists no published work that has implemented both the analog $\Delta\Sigma$ modulator and the pre-seated digital correction on a single chip. Therefore, this work provides

real data on silicon area and power consumption of the complete system. A potential drawback of this digital correction (realized late in the design process) is that the number of bits going to the digital decimation filters have increased from 4 to 16, This may require more complex filters. Further work can be done to decrease power consumption of the integration. We also found out that the error transferfunction must be exactly unity (ETF=-1). A non unity EFT limits the accuracy of correction factors *cc(i)* and the compensation will not be that effective.

Table 12: FOM comparison

Ref.	fs (MHz)	BW (MHz)	SNDR (dB)	P (mW)	Tech. (nm)	FOM (fJ/c.)
[16]	420	20	70	8.5	90	270
[17]	640	10	65	6.8	90	240
[18]	950	20	60	10.5	65	319
[19]	640	20	74	20	130	120
This work	144	9	67.5	6.861	65	196.73

During digital design, we started from using Matlab to do model building and simulation. Then the VHDL module is constructed guided by Simulink model. Within this procedure, the difference between high level model and hardware level implementation is witnessed. Design of division and crosscorrelation is a bit challenging. After front-end behavior simulation, back-end flow is also tricky. At the back-end phase, we realize that some changes are needed in the VHDL coding such as adding clock gating cells. This gives us a new vision on how to start our design at coding phase. Also we have learnt a lot during the backend flow as many different problems have been met and solved with the help from professors.

References

[1] J. G. Kauffman, P. Witte, J. Becker, and M. Ortmanns, "An 8.5 mW Continuous-Time $\Delta\Sigma$ Modulator With 25 MHz Bandwidth Using Digital Background DAC Linearization to Achieve 63.5 dB SNDR and 81 dB SFDR," *IEEE Journal of Solid-State Circuits*, 2011.

[2] M. Ortmanns and F. Gerfers, *Continuous-Time Sigma-Delta A/D Conversion - Fundamentals, Performance Limits and Robust Implementation.* Dordrecht, The Netherlands: Springer, 2005.

[3] M. Andersson, L. Sundström, M. Anderson, and P. Andreani, "Theory and design of a CT $\Delta\Sigma$ modulator with low sensitivity to loop-delay variations," *Analog Integr Circ Sig Process (2013) 76:353–366*, 2013.

[4] R. Schreier and G. Temes, *Understanding Delta-Sigma Data Converters.* IEEE, 2004.

[5] "$http://en.wikipedia.org/wiki/analog-to-digital\_converter$."

[6] F. Maloberti, *Data Converters.* Springer, 2007.

[7] M. Keller, A. Buhmann, J. Sauerbrey, M. Ortmanns, and Y. Manoli, "A Comparative Study on Excess-Loop-Delay Compensation Techniques for Continuous-Time Sigma–Delta Modulators," *IEEE Transactions on Circuits and Systems—I: regular papers, VOL. 55, NO. 11*, 2008.

[8] J. A. Cherry and W. M. Snelgrove, "Excess Loop Delay in Continuous-Time Delta–Sigma Modulators," *IEEE Transactions on Circuits and Systems——II: analog and digital signal processing, VOL. 46, NO. 4*, April 1999.

[9] S. Pavan, "Excess Loop Delay Compensation in Continuous-Time Delta-Sigma Modulators," *IEEE Transactions on Circuits and Systems——II: express briefs, VOL. 55, NO. 11*, November 2008.

[10] M. Vadipour, C. Chen, A. Yazdi, M. Nariman, T. Li, P. Kilcoyne, and H. Darabi, "A 2.1mW/3.2mW Delay-Compensated GSM/WCDMA $\Delta\Sigma$ Analog-Digital Converter," *Symposium on VLSI Circuits Digest of Technical Papers*, 2008.

[11] J. Sauerbrey, J. S. P. Garcia, G. Panov, T. Piorek, X. Shen, M. Schimper, R. Koch, M. Keller, Y. Manoli, and M. Ortmanns, "A Configurable Cascaded Continuous-Time $\Delta\Sigma$ Modulator with up to 15MHz Bandwidth," *Proc. European Solid-State Circuits Conf. (ESSCIRC)*, 2010.

[12] P. Witte and M. Ortmanns, "Background DAC Error Estimation Using a Pseudo Random Noise Based Correlation Technique for Sigma-Delta Analog-to-Digital Converters," *IEEE Transactions on Circuits and Systems—I: regular papers, VOL. 57, NO. 7*, July 2010.

[13] J. W.-g. WANG Liu-cheng, LIN Yong-cai, "Implementation of Fast and High-precision Division Algorithm on FPGA," *Computer Engineering VOL. 37, NO.10*, May 2011.

[14] S. F. Oberman and M. J. Flynn, "Division Algorithms and Implementations," *IEEE Transations on Computers, VOL. 46, NO. 8*, August 1997.

[15] R. J. Baker, *CMOS Circuit Design, Layout, and Simulation*, 3rd ed. IEEE, 2010.

[16] P. Malla, H. Lakdawala, K. Kornegay, and K. Soumyanath, "A 28mW Spectrum-Sensing Reconfigurable 20MHz 72dB-SNR 70dB-SNDR DT $\Delta\Sigma$ ADC for 802.11n/WiMAX Receivers," *IEEE Int. Solid-State Circuits Conference (ISSCC)*, 2008.

[17] P. Crombez, G. V. der Plas, M. Steyaert, and J. Craninckx, "A Single Bit 6.8mW lOMHz Power-Optimized Continuous-Time $\Delta\Sigma$ with 67dB DR in 90nm CMOS," *Proc. European Solid-State Circuits Conf. (ESSCIRC)*, 2009.

[18] V. Dhanasekaran, M. Gambhir, M. M. Elsayed, E. Sánchez-Sinencio, J. Silva-Martinez, C. Mishra, L. Chen, and E. Pankratz, "A 20MHz BW 68dB DR CT Delta-Sigma ADC Based on a Multi-Bit Time-Domain Quantizer and Feedback Element," *ISSCC Dig. Tech. Papers*, 2009.

[19] G. Mitteregger, C. Ebner, S. Mechnig, T. Blon, C. Holuigue, E. Romani, A. Melodia, and V. Melini, "A 14b 20mW 640MHz CMOS CT $\Delta\Sigma$ ADC with 20MHz Signal Bandwidth and 12b ENOB," *IEEE Int. Solid-State Circuits Conference (ISSCC)*, 2006.

LUND
UNIVERSITY

Series of Master's theses
Department of Electrical and Information Technology
LU/LTH-EIT 2014-411

http://www.eit.lth.se