AN ABSTRACT OF THE THESIS OF

Brandilyn Coker for the degree of Master of Science in Electrical & Computer Engineering presented on February 27, 2015.

Title: Limitations and Optimization of a Blind Calibration Algorithm for Nonlinearity in Analog to Digital Converters

Abstract approved: _____

Un-Ku Moon

Analog to digital converters (ADCs) are a critical part of communication between the physical world and the increasingly digital systems humans use every day. ADCs have inherent non-idealities that degrade performance. Nonlinearity is one of the most prevalent non-idealities that designers face. While calibration methods for nonlinearity exist in the analog domain, digital calibration is preferred since it typically takes less resources (chip area, power consumption) and can be implemented off chip if need be. A blind digital calibration algorithm for nonlinearity correction in ADCs was developed at Oregon State University that continuously corrects harmonic distortion using the concepts of downsampling and orthogonality of sinusoidal signals. It can calibrate for multiple harmonics simultaneously with no need for an external test signal.

This work explores the blind calibration algorithm in order to determine some of the limitations inherent to both the theoretical design and with respect to a practical implementation in hardware. Based upon these limitations, various methods of algo-

rithm optimization were characterized through discussion of design trade-offs and ways to improve performance.

Limitations and Optimization of a Blind Calibration Algorithm for Nonlinearity in Analog to Digital Converters

by

Brandilyn Coker

A THESIS

submitted to

Oregon State University

in partial fulfillment of
the requirements for the
degree of

Master of Science

Presented February 27, 2015
Commencement June 2015

Master of Science thesis of Brandilyn Coker presented on February 27, 2015.

APPROVED:

Major Professor, representing Electrical & Computer Engineering

---

Head of the School of Electrical Engineering and Computer Science

---

Dean of the Graduate School

I understand that my thesis will become part of the permanent collection of Oregon State University libraries. My signature below authorizes release of my thesis to any reader upon request.

Brandilyn Coker, Author

ACKNOWLEDGEMENTS

First and foremost, I'd like to thank Dr. Moon for giving me the opportunity to complete my graduate work under him. You took a chance on me and I will be forever grateful for your guidance and understanding over the years, and I appreciate how honest and caring you are with your grad students. You always have our backs.

To my family, thank you for your support and patience through the many years of schooling. You've always provided a happy place to go home to, and a willing ear along with good advice for when I needed a little perspective on life or a good pep talk.

To my research group, thank you for always being willing to help and for making this graduate school journey an entertaining one. From lunches to beers to football games, the camaraderie I experienced was something I'll never forget.

To my friends Jerry and Cal, you were the best partners I could have had during undergrad; all of those late nights working on projects would have been a lot rougher without your optimism and the laughter we shared.

To the TekBots crew, thank you for making the lab a fun place to hang out when I needed a break from the seriousness of the grad offices, and for reminding me what it means to be a good TA and mentor when it is far too easy to become jaded.

To Don and Tuan, without you guys I don't know how I would have made it through. Your friendship and support means everything to me, and I look forward to staying close with you as I start this next phase of my life. You've been there through everything, keeping it real even when everything seemed like it was falling apart.

To my cat, thank you for keeping me warm on those cold rainy days, and for always greeting me at the door when I get home from a long day at work.

TABLE OF CONTENTS

TABLE OF CONTENTS (Continued)

LIST OF FIGURES

LIST OF FIGURES (Continued)

LIST OF TABLES

Chapter 1: Introduction

1.1 Introduction

While the amount of digital information available each year is growing rapidly, the world is still inherently analog. In order to communicate between our digital systems and the analog world, data converters are needed. Data converters are divided into two types: analog to digital converters (ADCs) and digital to analog converters (DACs). These data converters are required to record and play back audio, to monitor and control the temperature of a room automatically, and to monitor your heart rate. These are just some of the many situations and applications in which ADCs are needed. The design of ADCs is a complicated matter of balancing the trade-offs of power, size, and speed. Due to the effects of Moore's Law over the past 50 years, transistors have been decreasing in size at a rapid rate [1]. This improves digital designs by decreasing power consumption and increasing speed. However, for analog designs the smaller feature sizes cause decreased intrinsic gains and require lower voltage supplies. In order to utilize smaller semiconductor process technologies, researchers have increasingly directed their attention towards ADC architectures that can tolerate and even leverage these limitations while still achieving the desired amount of performance.

1.2 Motivation

In order to improve ADC performance, analog designers must take into account non-idealities intrinsic to their chosen design. Common non-idealities include noise, matching, jitter, and nonlinearity. Nonlinearity and other non-idealities can be reduced by good design, but to get peak performance ADC calibration is a consistent method to improve results. Traditional calibration methods require external input and various test signals in order to work. Recently a novel blind calibration algorithm for nonlinearity was developed that does not require additional inputs or test signals, utilizing only the digital output from the ADC. [2] Further increasing our understanding of the blind calibration algorithm and determining its limitations with the intent to implement in silicon as future work is of considerable value.

1.3 Thesis Organization

This thesis is organized as follows. Chapter 2 will cover the background of ADCs, nonlinearity in ADCs, and a sampling of previous digital calibration techniques utilized to combat this non-ideality. Chapter 3 discusses the blind calibration algorithm for nonlinearity in ADCs developed at Oregon State University. Chapter 4 explores the limitations of the algorithm, both those inherent in the theoretical design and those related to eventual implementation in silicon. Chapter 5 presents the optimizations that can be made to the algorithm based upon the limitations introduced in the previous chapter. Finally, the thesis ends with the conclusion in Chapter 6.

Chapter 2: Background

2.1 Introduction

Every analog design has its pros and cons. Power consumption, speed, and size are all factors that play into the decision to pick a specific architecture. The negatives for a design are often based upon the kinds of non-idealities inherent to the architecture, such as noise. One of the most common and troubling non-idealities found in analog designs is nonlinearity. ADCs are no exception to these issues.

2.2 Analog to Digital Converters

An analog to digital converter converts a continuous time physical input signal to a discrete time digital output signal through quantization. ADCs are typically defined by their bandwidth (or sampling rate) and resolution, which is found by calculating the effective number of bits (ENOB) from the signal to noise distortion ratio (SNDR) determined by the frequency response of the ADC. There are a wide variety of ADC architectures, including flash, successive approximation (SAR), pipeline, and $\Delta\Sigma$ [3]. ADCs are everywhere, from satellite communications, to medical imaging, and smartphones. They are more important than ever in an increasingly digital world as they provide the interface to quantify and interact with the physical world around us.

Flash ADCs are the fastest ADC architecture, as they use a voltage ladder made up of resistors or capacitors and then feed the results at each point simultaneously into a

Figure 2.1: Symbol for an Analog to Digital Converter

series of comparators. However, to get high resolutions they require a large area and power consumption and thus are usually limited to applications where extremely large bandwidths are needed. SAR ADCs are often used for medium to high resolution applications at relatively slow sample rates, and are known for their low power consumption and small area requirements, leading to widespread use in the many energy constrained systems. SAR ADCs successively generate a digital signal by comparing the analog input to a specified voltage reference range that narrows after each quantization until the final bit is found. Pipeline ADCs are a versatile architecture that can attain a wide range of sampling rates and resolutions, and they consist of a number of stages containing a low resolution flash comparator and DAC. A coarse quantization is done, and then leftover residue is amplified and fed into the next stage until the end of the pipeline and the residue left is the smallest bit (LSB). $\Delta\Sigma$ ADCs (also known as $\Sigma\Delta$) are mostly used for low speed-high resolution applications. The high resolution is attained via the use of oversampling the input, followed by a filter band to restrict the output to the desired bandwidth. [4]

2.2.1 Non-Idealities

There are a number of non-idealities that are common to ADC designs, including noise, component mismatch, jitter, and nonlinearity. Noise is inherently produced inside the

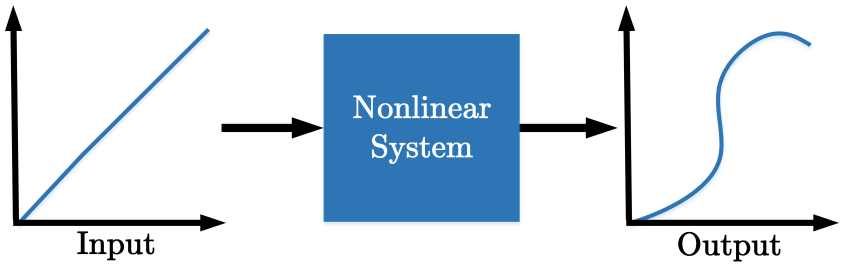


Figure 2.2: Example of a nonlinear system response

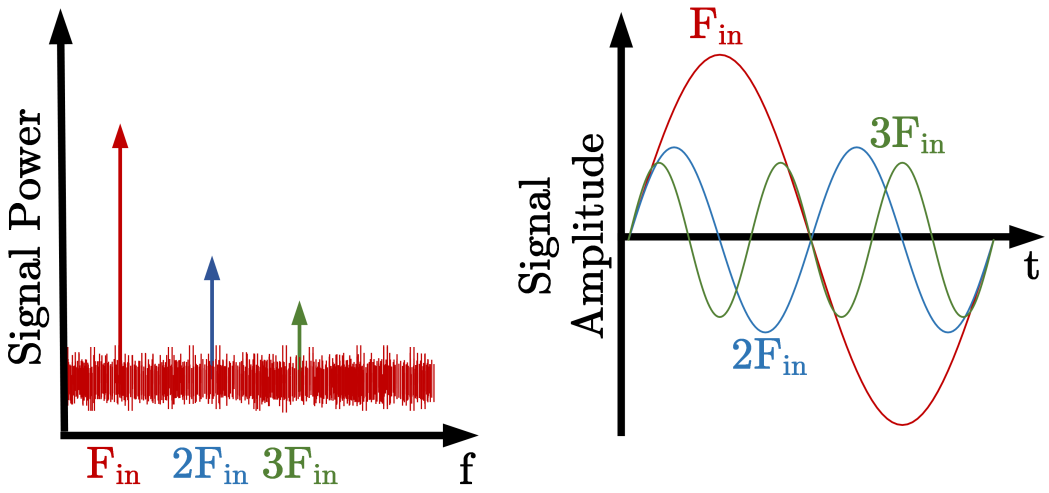
ADC circuits by resistors and capacitors ($\frac{kT}{C}$ noise), which appears in what is known as the input-referred noise. Quantization error is noise inherent to ADCs as well, and is dependent upon the ADC resolution. Its effects can be mitigated by oversampling. Jitter is the deviation of a clock edge from its ideal location, causing uncertainty in what time a data sample has actually been quantized. This uncertainty and thus error appears as more noise when examining the ADC output.

2.3 Nonlinearity

Nonlinearity is present when the output and input of a system are not directly proportional to each other, breaking the rule of superposition shown in Equation 2.1. Nonlinear systems are a well known phenomenon in the fields of physics and mathematics. Due to this, much effort has been put forth to find efficient methods for modelling nonlinear systems.

$$f(x+y) = f(x) + f(y) \tag{2.1}$$

When looking at the frequency domain for a system's outputs, nonlinearity manifests in the form of frequencies other than the original input. These additional signals created



(a) Harmonics in the frequency domain (b) Harmonics in the time domain

Figure 2.3: Harmonic distortion

by nonlinearity are known as harmonic distortion. For a signal with multiple tones, nonlinearity causes both harmonic distortion and undesired intermodulation. A signal from a system with second and third order harmonic distortion could be represented by Equation 2.2, where y_{out} is the system output, x_{in} is the input signal, and α_2 and α_3 are the distortion coefficients.

$$y_{out} = \alpha_1(x_{in}) + \alpha_2(x_{in})^2 + \alpha_3(x_{in})^3 \tag{2.2}$$

These new frequencies are generated at multiples of the input. In the case of audio signals, harmonic distortion in significant amounts is easily heard. In radio frequency (RF) systems, harmonic distortion and intermodulation can cause transmitted data to be corrupted and therefore lost.

2.3.1 Nonlinearity in ADCs

Nonlinearity is inherent to analog systems due to the nonlinear components used. As seen in the previous section, it presents itself in the form of harmonics that can be observed in both the time and frequency domains. However, the frequency domain is where most ADC performance analysis occurs. The additional tones decrease the performance of the ADC in the form of lower SNDR, a defining metric for ADC performance. The digital output of an ADC suffering from nonlinearity can commonly be written as Equation 2.3, where α_1 and α_3 are the distortion coefficients correlating to the fundamental frequency and third harmonics respectively.

$$D_{out} = \alpha_1 D(V_{in}) + \alpha_3 D(V_{in})^3 \tag{2.3}$$

There is a wide variety of possible sources for nonlinearity in ADCs. Common sources include amplifier nonlinearity and capacitor mismatches. Both analog domain and digital domain solutions exist for these problems. In the analog domain, using larger values for passive components such as capacitors and resistors can improve matching, in addition to larger loop gains and wider bandwidths. However, these methods come with the compromise of higher power consumption and area. They are also becoming harder to design due to process scaling.

Due to this, digital calibration is becoming the preferred method to improve linearity. There are a number of techniques that have been proven to reduce or remove nonlinearity in ADCs, each with their own limitations. These methods include bootstrapped calibration, split ADC based calibration, and multiple pseudo-random number based calibration.

2.4 Digital Nonlinearity Calibration Techniques

A number of digital calibration techniques have been developed to solve the widespread problem of nonlinearity in ADCs. One method involves the use of a bootstrapped calibration algorithm, in which the calibration digital to analog converter (DAC) is calibrated by the ADC and vice versa [5]. This technique runs the ADC at a higher clock than the input is sampled at, allowing the ADC/DAC pair to process calibration signals and update in the background. Two known codes are given to the DAC and then passed to the ADC. The difference in the ADC output from ideal, coupled with varying the input DAC codes across the entire ADC range, allows the nonlinearity to be estimated and thus its impact on the performance reduced.

Another technique is known as split ADC based calibration. As the name would suggest, the ADC design is split into two identical halves. Then a known offset delta (Δ) is given to each input; one gets a positive delta and the other gets a negative delta. After the conversion is complete the difference between the digital outputs is found, and for an ideal ADC the result should equal 2Δ. Any error found from that value is then fed into into a least mean squares (LMS) engine to calibrate for odd order harmonics (3rd, 5th, etc.) [6].

The multiple pseudo random number based calibration presented in [7] passes a sum of multiple independent pseudo random number sequences to the input of the ADC, and then multiplies the output with the product of the same pseudo random number sequences. This allows for estimation of the nonlinearity coefficient because independent pseudo random number sequences are orthogonal to each other.

While all of the discussed methods have presented good results, they are either limited to a specific ADC architecture or require known test signals in order to complete the

calibration. The blind calibration algorithm for nonlinearity is not limited in either of these ways, and will be discussed in the following chapter.

Chapter 3: Blind Calibration Algorithm for Nonlinearity in ADCs

3.1 Introduction

In this chapter a previously developed blind calibration algorithm for nonlinearity in analog to digital converters is discussed. The benefits of this algorithm include not requiring any external calibration signal, being entirely digital, and the ability to remove both even and odd harmonics simultaneously in the background [2]. Performance was demonstrated using a MATLAB simulation on data from a ring oscillator based ADC with nonlinearity limited performance. To reduce the nonlinearity generated by the ADC in th digital output, the calibration algorithm relies on the properties of downsampling and the orthogonality of sinusoids.

3.2 Downsampling

A downsampled signal in a discrete time system is one in which every Nth sample of the original signal $x[n]$ is kept to create a new signal $x_{ds}[n]$, as shown in Equation 3.1. For example, with a downsampling factor (N_{ds}) of 5 the new signal would contain every 5^{th} sample and the total number of samples would be one fifth of the original [8].

$$x_{ds}[n] = x[nN_{ds}] \tag{3.1}$$

Downsampling in the time domain is observed in the dropping of data samples as seen in Figure 3.1. In the frequency domain, downsampling is observed via the input frequency

spectrum being spread out by the downsampling factor, including any harmonics that might be present (Figure 3.2). This effect is one of the important concepts for the blind calibration algorithm as it allows for the creation of new signals at known frequencies with respect to the input.

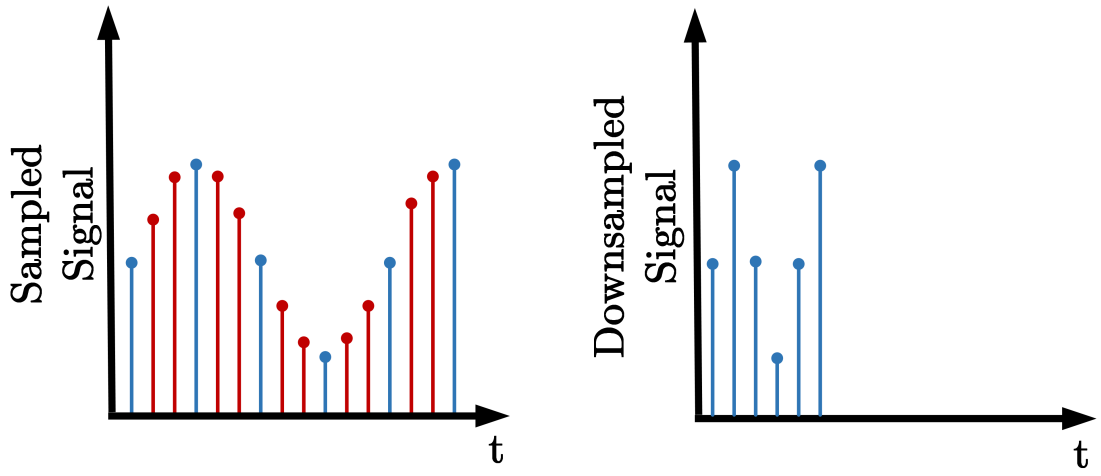


Figure 3.1: Downsampling in the time domain, $N_{ds} = 3$

Due to the harmonics generated by nonlinearity being at integer multiples of the input frequency f_{in}, downsampling allows the signal to stretch and have the original tone be at the same frequency as the nonlinearity introduced by the non-ideal ADC. This is accomplished by choosing the downsampling factor (N_{ds}) to be the same as the harmonic being calibrated (e.g. $N_{ds} = 3$ for third order correction, $N_{ds} = 2$ for second order correction, etc.). If the original signal is at $0.25f_s$, and it is downsampled by 3, the new signal would be at $0.75f_s$.

One possible side effect of downsampling is signal aliasing. Aliasing, also known as frequency folding, is an effect that causes different signals to become indistinguishable from one another when sampled at the same rate. This occurs when the Nyquist theorem

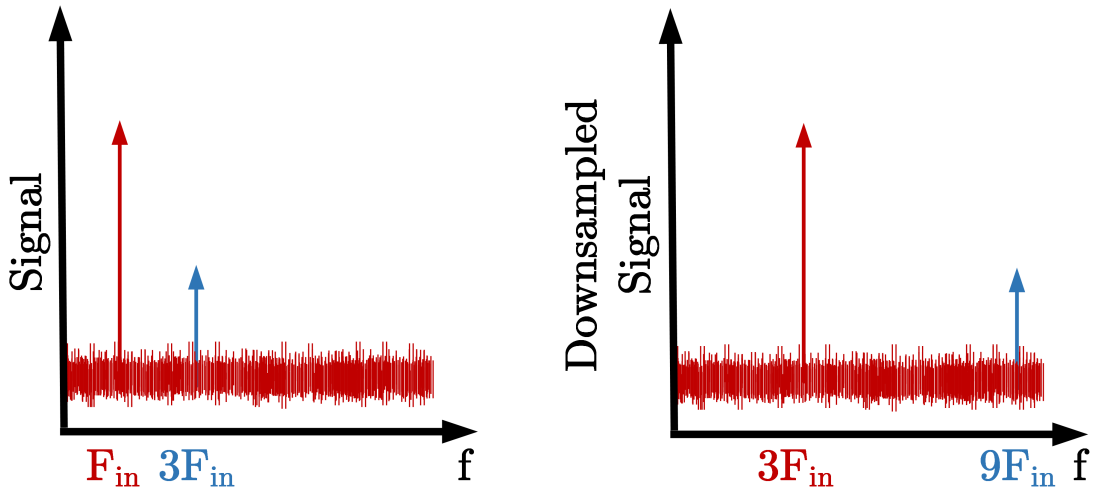


Figure 3.2: Downsampling in the frequency domain, $N_{ds} = 3$

is not met by both frequencies of interest, i.e. $f < 0.50 f_s$, and thus one of the frequencies "folds" around f_s and appears to be a different frequency. In general, a signal with frequency $f_{sampled}$ after sampling could have been obtained by the sampling of signals of frequency f_{in}

$$f_{sampled} = |f_{in} - k f_s| \tag{3.2}$$

where k is any integer. Thus for $k = 1$, with a normalized f_{in} of $0.75 f_s$, $f_{sampled}$ would appear to have a frequency of $0.25 f_s$. As can be seen in Figure 3.3, while the original signals are two different frequencies, they appear to be the same frequency after sampling. To reduce or remove any aliasing after downsampling, an anti-aliasing filter is often used to restrict the bandwidth of the input signals to $0.5 f_s$ and insure no frequency folding occurs.

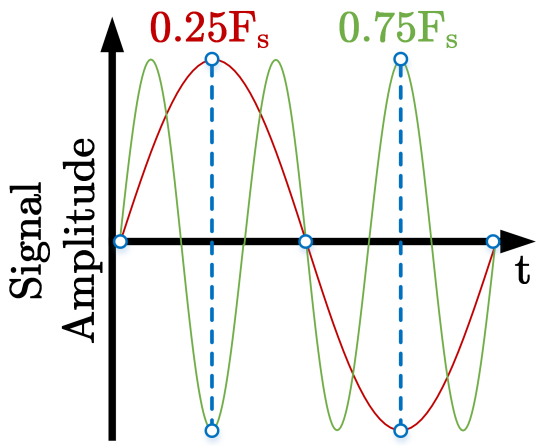


Figure 3.3: Example of aliasing that can occur when an input frequency is higher than the Nyquist bandwidth of $0.5F_s$

3.3 Orthogonality

A pair of signals are orthogonal if the sum of their inner product is zero, as shown in Equation 3.3.

$$x \perp y \Leftrightarrow \int x(t)y(t)dt = 0 \tag{3.3}$$

A key property of sinusoidal signals is that they are orthogonal at different frequencies [9]. In continuous time, this statement holds regardless of amplitude and phase for sinusoids of infinite duration (e.g. summed from $-\infty$ to ∞ as seen in Equation 3.4).

$$\int_{-\infty}^{\infty} sin(\omega_1 t)sin(\omega_2 t) = 0 \Rightarrow \omega_1 \neq \omega_2 \tag{3.4}$$

However, digital signal processing typically takes place in discrete time with sampled signals, such as those produced from ADCs and used for the discrete Fourier transform

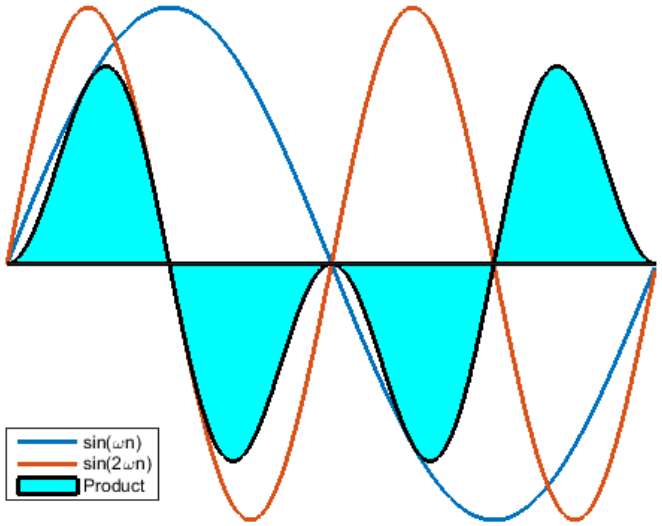


Figure 3.4: Graph illustrating the orthogonality of two sine waves

(DFT) [10]. While math can be done for infinite duration signals, in the real world there is a set of sampled data of length N to work with. Thus two discrete time signals are orthogonal if over N data points their inner products sums to 0 (Equation 3.5).

$$\sum_{n=1}^{N-1} sin(\omega_1 n) sin(\omega_2 n) = 0 \Rightarrow \omega_1 \neq \omega_2 \tag{3.5}$$

For discrete time sinusoidal signals of length N, the exact rule of orthogonality holds only for the harmonics of the sampling rate f_s divided by N. This means that the product of two sinusoids will only sum to zero if their frequencies are integer divisions of the total number of sampled data points with respect to the sampling rate. The list of orthogonal frequencies for a given set of data samples is described by Equation 3.6.

$$f = k\frac{f_s}{N}, \text{ for } k = 0, 1, 2, ...N-1 \tag{3.6}$$

These frequencies are the only frequencies that have a whole number of periods in N samples, and thus fulfill the expectation that sinusoidal signals are orthogonal. If frequencies besides these are used, the signals will not be perfectly orthogonal, and thus will not sum perfectly to zero.

3.4 Algorithm

By combining the properties of downsampling and orthogonality of sinusoidal signals, the distortion coefficients for a non-ideal ADC can be estimated. [2] This is done by incorporating a second and identical ADC with a scaled input, thus allowing the removal of the original signal and leaving only the harmonics available for processing. The coefficient is estimated by summing the product of the downsampled signal D_{ds3} along with the signal-free signal D_{nosig}, thus calculating the error (err) present in the current coefficient estimate. This error is then used to update the coefficient value, and the cycle repeats until the error is minimized, thus providing the final distortion coefficient which is then used to calibrate incoming data. This feedback for minimizing the error is part of what is known as a least mean squares (LMS) engine, a concept often used in calibration. The blind calibration algorithm can calibrate for multiple orders of distortion simultaneously. However, for the sake of simplicity and understanding the following equations and Figure 3.6 explaining the algorithm only show calibration for a third order harmonic.

The periodic input signal V_{in} is fed into both ADC_1 and ADC_2, though before

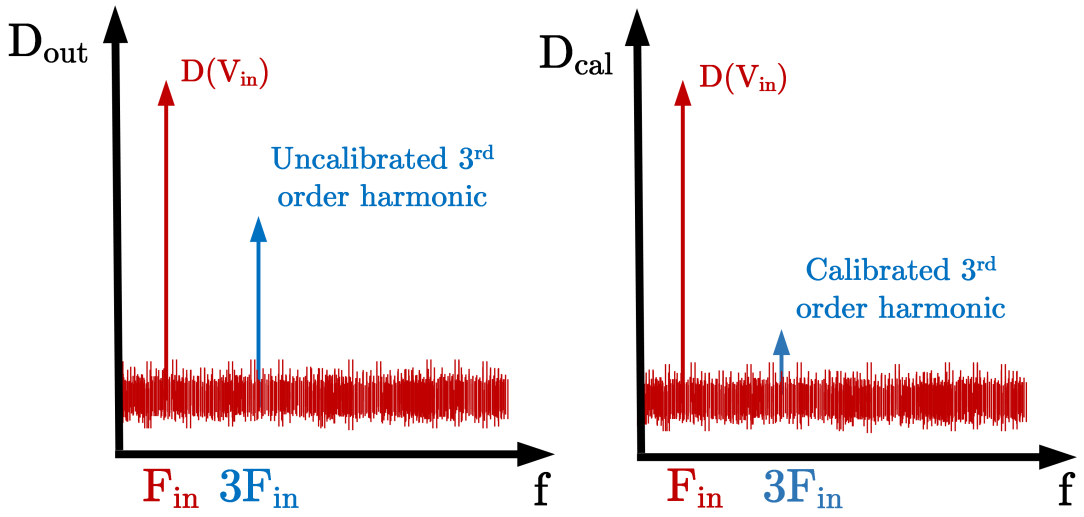


Figure 3.5: Ideal calibration of third order distortion

entering ADC_2 it is scaled down by a factor of 2 (Eq. 3.7c).

$$V_{in} = Asin(\omega t) \tag{3.7a}$$

$$D_{out} = D(V_{in}) + \alpha_3 D(V_{in}^3) \tag{3.7b}$$

$$D_{out2} = D(0.5V_{in}) + \alpha_3 D(0.5V_{in}^3) \tag{3.7c}$$

The digital outputs from the ADCs are then raised to the power of the coefficient being calculated (in this example, 3), and then multiplied by the current α_3 coefficient estimate. This new value is then subtracted from the original D_{out} to find the calibrated digital output D_{cal}. When the α_3 estimate has converged to the correct value, the harmonic distortion is greatly reduced.

$$D_{cal}[n] = D_{out}[n] - \alpha_3[n-1]D_{out}[n]^3 \tag{3.8}$$

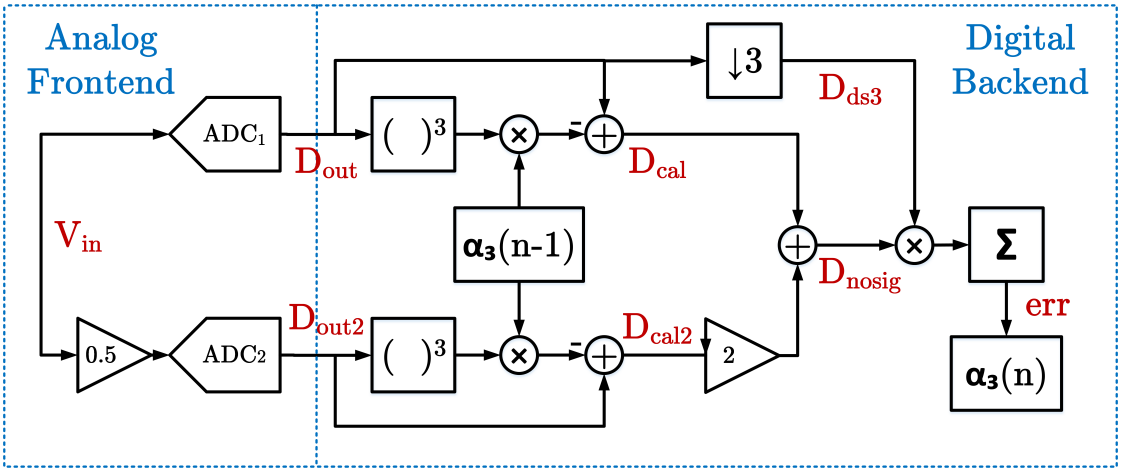


Figure 3.6: Simplified block diagram of blind calibration algorithm calibrating for $3rd$ harmonic

In order to converge to the correct α_3 using the LMS engine as shown in Equation 3.9, the error must be found. Note that μ is just a variable for adjusting the step size of the error term, and thus the rate of convergence for finding the stable and final value of the distortion coefficient.

$$\alpha_3[n] = \alpha_3[n-1] - \mu \times err \tag{3.9}$$

The error term is found by summing the product of the downsampled input D_{ds3} and the harmonics left in D_{nosig}. D_{ds3} and D_{nosig} are both centered around $3f_{in}$, which would dictate that their sum is nonzero based upon the rules of orthogonality. As the harmonics present in D_{nosig} are minimized, the err term decreases to zero until there are no harmonics left in D_{nosig} and thus none left in the calibrated output D_{cal}.

$$err = \sum D_{nosig} \times D_{ds3} \tag{3.10}$$

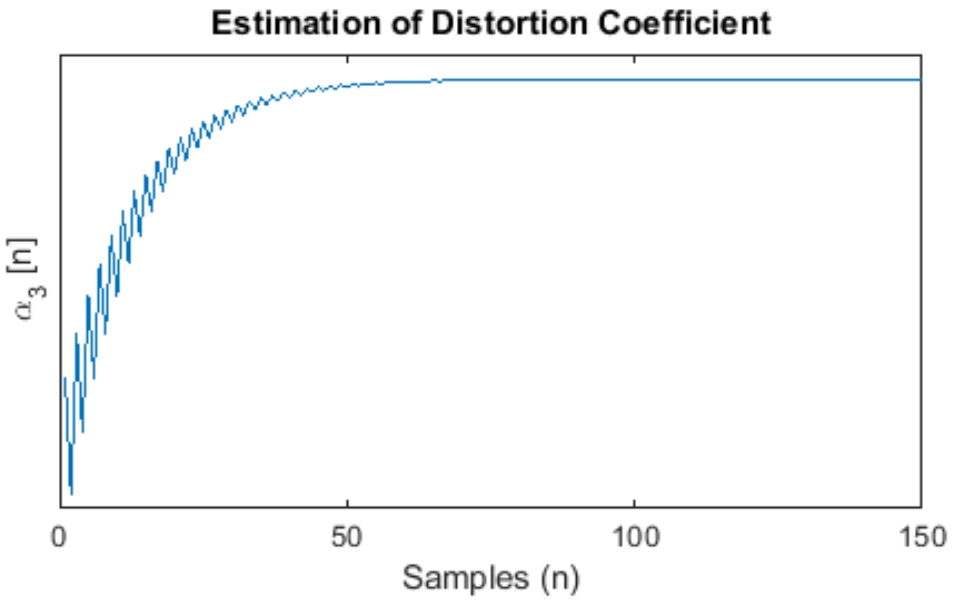


Figure 3.7: Convergence of distortion coefficient estimation

To isolate the nonlinearity from the original input signal for use in the error calculation, the second ADC is used to linearly scale the output down and up by 2. When the scaled version of D_{cal2} is subtracted from D_{cal}, the resulting signal free output is D_{nosig}. This works because the original signal will have scaled linearly, while the harmonics added by ADC nonlinearity will not have. A simplified graphical version of this using D_{out} and D_{out2} can be seen in Figure 3.8. D_{ds3} is created by simply taking every third signal from the original ADC output D_{out} as seen in Equation 3.11b.

$$D_{nosig} = D_{cal} - 2D_{cal2} \tag{3.11a}$$

$$D_{ds3} = D_{out}[3n] \tag{3.11b}$$

To calibrate for multiple harmonics, an additional downsampled stream is needed

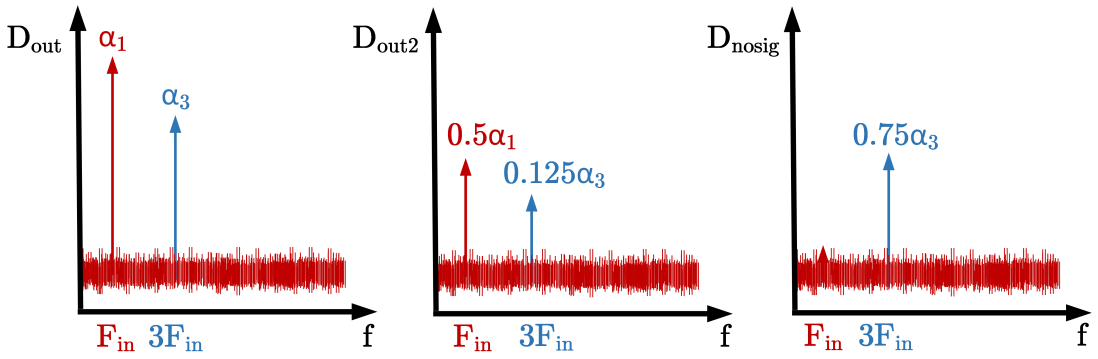


Figure 3.8: Scaling used to separate nonlinearity terms from original signal for LMS engine

for each additional harmonic, and some additional computation blocks in parallel to the third order distortion calibration path to find the calibrated output D_{cal}. A block diagram for the calibration of both the third and fifth harmonics can be seen in Figure 3.9.

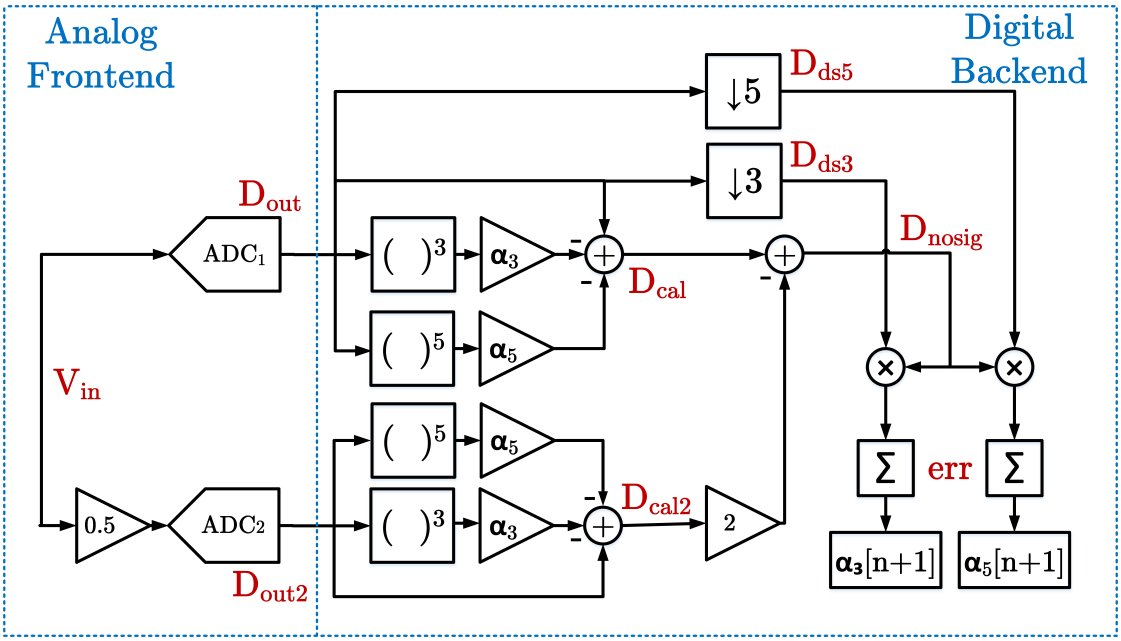


Figure 3.9: Simple block diagram of blind calibration algorithm for 3rd and 5th order distortion

Chapter 4: Limitations

4.1 Introduction

The implementation of mathematical theories in electrical hardware is a challenging process. Many times the original theory needs to be modified or simplified in a way to allow for it's reproduction in hardware. For the blind calibration algorithm discussed in the previous chapter, several aspects of the theory are difficult to achieve in hardware. This section discusses some of the limitations intrinsic to the algorithm and some of the considerations for creating a hardware implementation.

4.2 Generation of Higher Order Harmonics

In the effort to remove certain nonlinearities, higher order nonlinearities are introduced. This is mathematically shown in Equation 4.2 and simulation results are shown in Figure 4.1 These generated harmonics are proportional in size to the lower order nonlinearities, and can usually be neglected with respect to the higher order nonlinearities in terms of defining ADC performance via SNDR. If even better performance is desired, calibration of the generated harmonics can be added to the design.

$$D_{out} = D(V_{in}) + \alpha_2 D(V_{in}^2) \tag{4.1}$$

$$D_{cal} = D_{out} - \alpha_2 D_{out}^2 \tag{4.2a}$$

$$= D(V_{in}) - \alpha_2^2 D(V_{in}^3) - \alpha_2^3 D(V_{in}^4) \tag{4.2b}$$

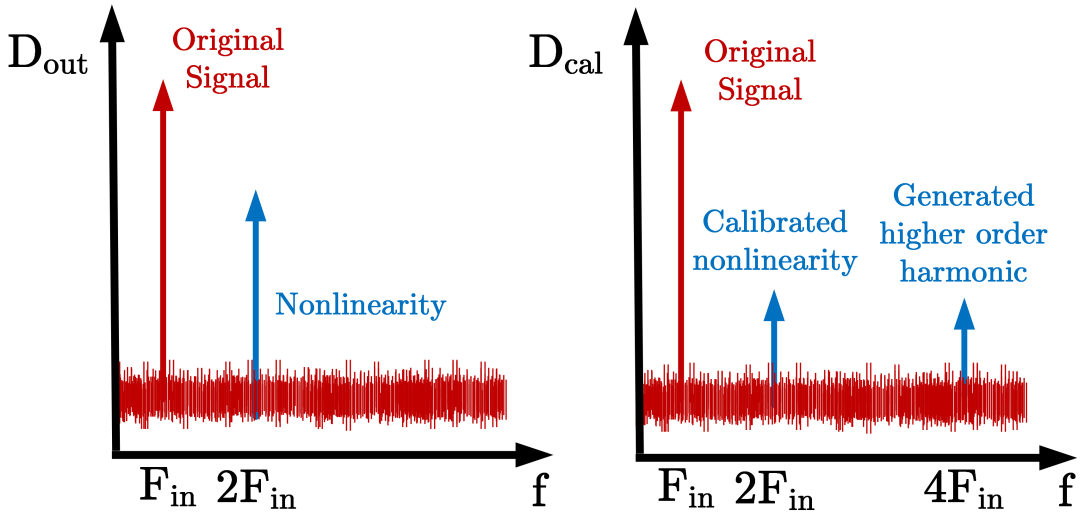


Figure 4.1: Generation of higher order harmonics by calibration algorithm

4.3 Background Mode

For the algorithm to run in the background, two identical ADCs are needed. This comes at the expense of additional power consumption and size in both the analog and digital domains, as most of the processing needs to be done twice. This is necessary in order to run in the background with any possible input. If only one ADC output was used for calibration, any digital output signal with multiple tones would get destroyed by the

algorithm, as shown in Figure 4.2. The red signal is the original data, which needs to stay the same after calibration. The blue signal is the distortion added by the nonlinear ADC during conversion. Since parts of the red and blue signal overlap at the harmonic being calibrated, there is undesired calibration. This is because with only one ADC, the LMS engine uses the product of the downsampled signal and the calibrated signal to find the error, and thus cannot distinguish between a tone that is part of a multi-tone input and a tone that is due to nonlinearity.

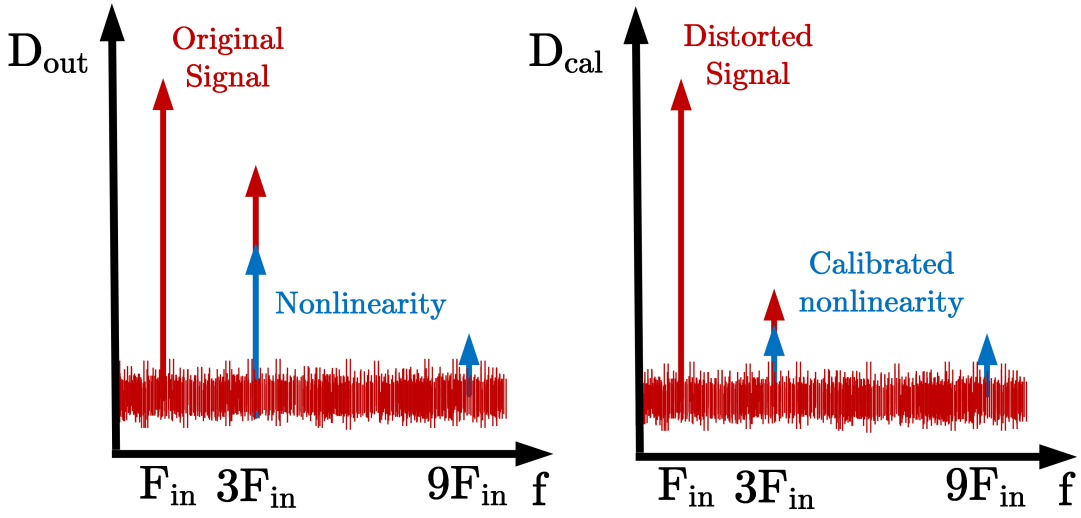


Figure 4.2: Spectrum result of calibration algorithm using a single ADC

4.4 Offset

Any offset introduced by the ADC to the signal must be removed before entering the calibration algorithm [11]. If offset is not removed from D_{out}, then D_{nosig} will also have an offset. This causes the algorithm error term to include the added offset in addition to the nonlinearity information. The algorithm would not converge to the

correct distortion coefficient. Removing any offset requires additional signal processing before the calibration algorithm begins. Since it is difficult to deduce whether or not the offset was present in the original signal or added to the digital output by the ADC it is best to carefully design the ADCs such that they do not add significant offset during data conversion.

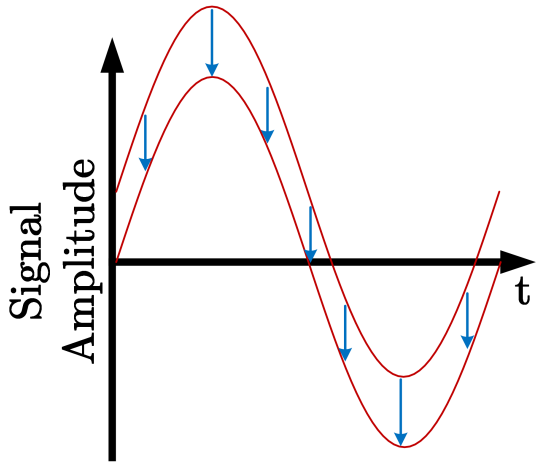


Figure 4.3: Removal of offset by centering periodic signal around 0

4.5 Input Buffers

In order to perfectly match each sample over time with its correlating downsampled value for the LMS engine, infinite length shift registers are needed to buffer and hold all incoming data. This implies a data set that is continually increasing in length as more samples are collected, as shown by Equation 4.3b where N_{buff} is the required buffer length, w is the length of the data being summed by the LMS engine, and n is the index of the coefficient error update. As this is impossible when it comes to a hardware

Table 4.1: Theoretical Input Buffer

n	Growing Buffer Length	D_{ds3}
0	[1 2 3 4 5 6 7 8 9 10]	[1 4 7 10]
1	[2 3 4 5 6 7 8 9 10 11 12 13]	[4 7 10 13]
2	[3 4 5 6 7 8 9 10 11 12 13 14 15 16]	[7 10 13 16]

Table 4.2: Practical Input Buffer

n	Finite Buffer Length	D_{ds3}
0	[1 2 3 4 5 6 7 8 9 10]	[1 4 7 10]
1	[2 3 4 5 6 7 8 9 10 11]	[2 5 8 11]
2	[3 4 5 6 7 8 9 10 11 12]	[3 6 9 12]

implementation, an alternative solution was developed. For third order calibration, a buffer of finite length $N_{buff,finite}$ is chosen such that there are w samples in D_{ds3} and D_{nosig} being fed into the LMS engine for each coefficient estimate error update.

$$err[n] = \sum_{k=1}^{w} D_{nosig}[k] D_{ds3}[k] \tag{4.3a}$$

$$N_{buff} = 2n + (3w - 2) \text{ for } w > 0, n = 0, 1, 2, ...\infty \tag{4.3b}$$

$$N_{buff,finite} = 3w - 2 \text{ for } w > 0 \tag{4.3c}$$

In order to achieve a buffer of finite length, the downsampled data used by the LMS engine is collected in a slightly different manner. Instead of continuously downsampling D_{out} from the data in the growing input buffer to form D_{ds3}, downsampling occurs only on the data in the finite input buffer. A visual comparison of the original theoretical

implementation to the practical implementation is shown in Tables 4.1 and 4.2. For this example, assume a w of 4. The values in the brackets are the indices of the input buffer and the downsampled signal, respectively. The data is continually shifting through the buffer and generating updated err terms for the distortion coefficient α_3 estimation.

All further discussion moving forward utilizes the practical input buffer method such that any limitations due to this change in the way downsampled data is collected from the input stream are discovered.

4.6 Input Frequency Resolution

During testing, the Equation 4.4 is used to define a periodic signal at frequency f_{in} that contains an integer number k of complete cycles within the given number of samples N to satisfy the concept of coherent sampling. This equation is used to create a test signal capable of generating a frequency spectrum using the Fast Fourier Transform (FFT). The FFT uses sample sets of length 2^n, which is required for the fast computation time. The generated input frequency is such that it is an integer ratio of the sampling frequency f_s. In order to find the frequency response of a system, you are limited in the resolution of the input frequencies passed in to the ADC by the resolution of the FFT you want to run. This limitation typically only applies during the testing phase of ADC design, as in real world usage the input frequency only needs to be within the bandwidth.

$$f_{in} = k\frac{f_s}{N}, \text{ for } k = 0, 1, 2, ...N-1 \tag{4.4}$$

Notice that Equation 4.4 is exactly the same as Equation 3.6, which is a limitation on the orthogonality of sampled sinusoids. While the limitation on f_{in} for the FFT is

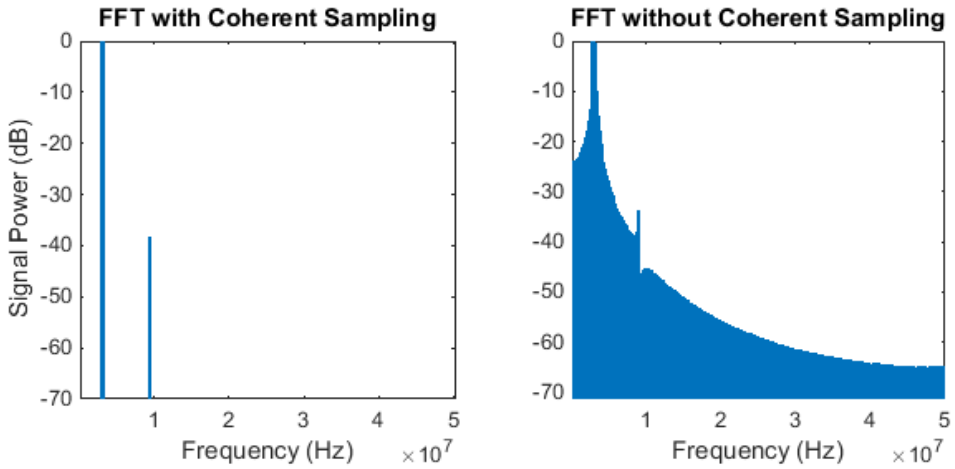


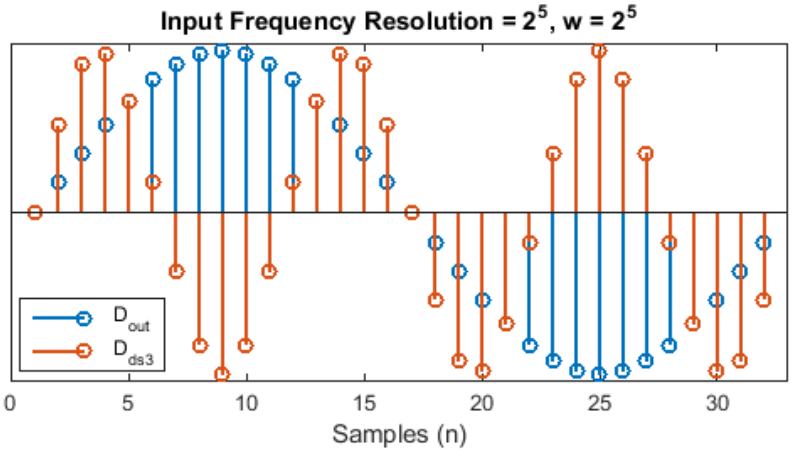
Figure 4.4: Example FFT output with coherent sampling and without coherent sampling

only applicable to specific testing cases, since the blind calibration algorithm needs to continuously run in the background this limitation on the input frequency required for exact orthogonality in sampled sinusoids is notable.

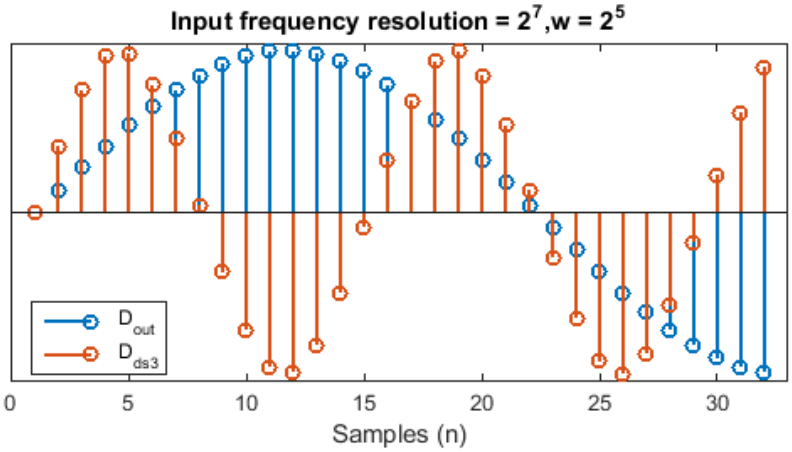
$$f_{in} = k\frac{f_s}{w}, \text{ for } k = 0, 1, 2, ...w - 1 \tag{4.5}$$

Since the error term of the coefficient estimate is calculated using the concept of orthogonality (see Eq. 4.3a), the length w of D_{nosig} and D_{ds3} must be carefully chosen. In the case of the algorithm, the input frequencies are thus limited to those seen in Equation 4.5, where the input frequency must have an integer number of complete cycles in w in order for exact orthogonality rules to apply. When there is not an integer number of periodic cycles in the sampled window of length w, the algorithm can no longer accurately estimate the distortion coefficient error as the error term will never be completely minimized as signals of different frequencies will not sum perfectly to zero.

Figure 4.6 illustrates this effect for a w of 2^5. For a nonlinearity coefficient α_3 of



(a) Example of exact orthogonality for a sampled data set



(b) Example of non-orthogonal signals for a sampled data set

Figure 4.5: Discrete time orthogonality of sinusoids

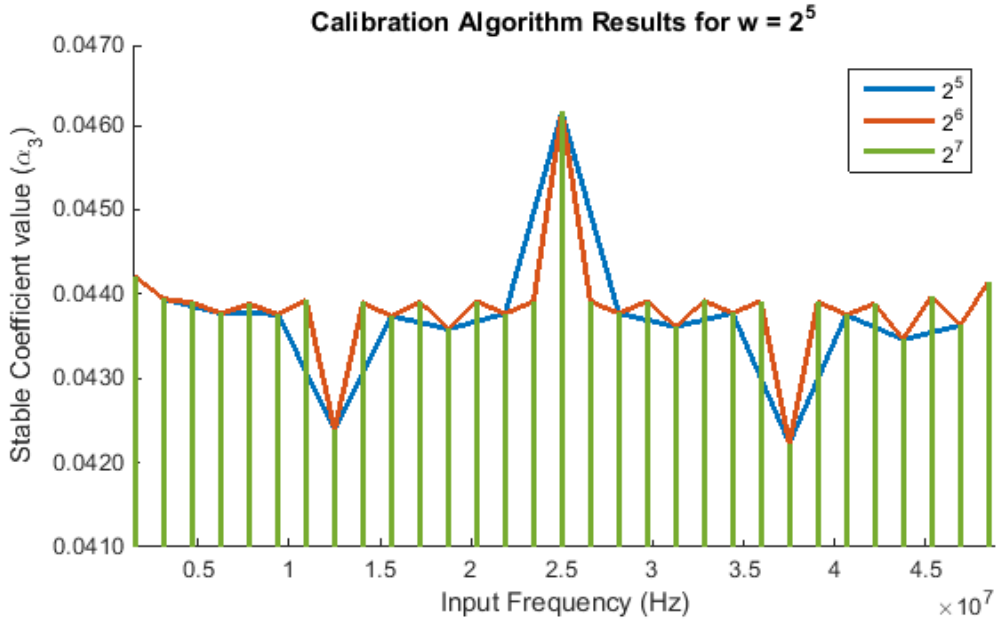


Figure 4.6: Deterioration of algorithm performance when input frequency is not at an integer division of the window size

0.05, the algorithm converges to approximately 0.044 during perfect operation (it does not reach 0.05 exactly due to the higher order harmonics generated by the algorithm beginning to dominate the performance). As the resolution of the input frequency (e.g., how many frequencies are tested across the bandwidth) increases past the size of the error summing window length w, the performance of the points in between integer divisions of the window size severely deteriorates. The resolution of the input frequencies was increased by a power of 2 each time based upon the assumption that any output would be examined using an FFT.

Thus, as the summing window of length w increases in size, the more input frequencies the calibration algorithm can accept and still produce consistent results. This is an unfortunate correlation, as increasing w quickly increases the amount of memory and

time needed for each error calculation.

Chapter 5: Optimizations

5.1 Introduction

Based on simulation results, a number of optimizations that can be used to improve the blind calibration algorithm with respect to a hardware implementation are discussed in the following sections.

5.2 μ

This coefficient determines the settling time and accuracy of the coefficient estimation by altering the step size of the error term in the blind calibration algorithm (see Equation 5.1). The specific choice of μ is not critical to the final coefficient value if it is given enough time to converge, as can be seen in Figure 5.1. As μ increases, the estimator converges more quickly. From Figure 5.2, it can be seen that while at a larger μ it converges faster, it also oscillates more. At large values of μ the estimator eventually becomes unstable and will not converge to a final value. If the number of cycles it takes to converge is not a limiting factor, it is then ideal to use a small μ to improve final accuracy.

$$\alpha_3[n] = \alpha_3[n-1] - \mu \times err \tag{5.1}$$

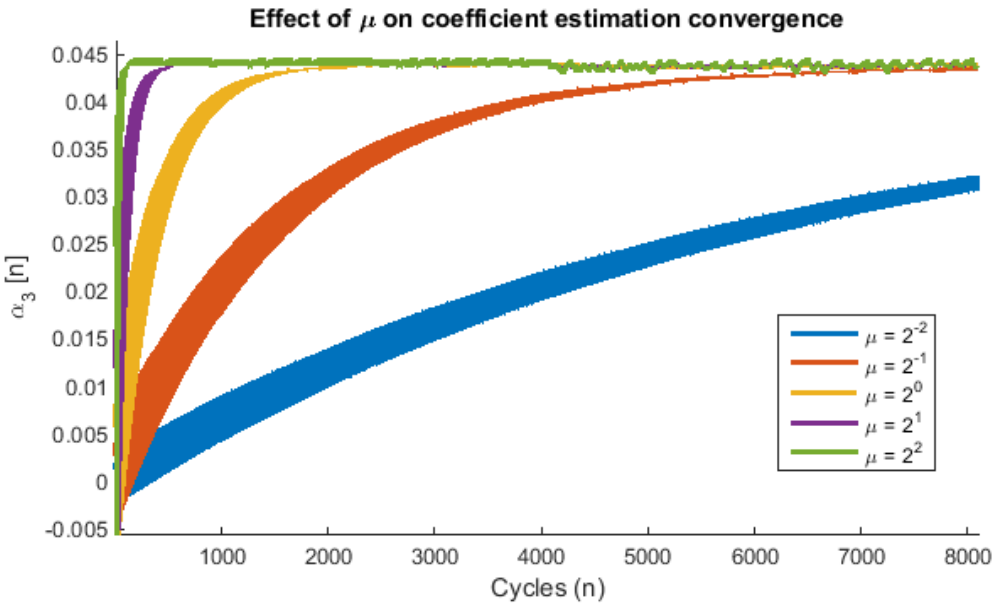


Figure 5.1: Impact of changing μ on estimator convergence time

5.3 Error Estimation Window

As discussed in Chapter 4, the theoretical algorithm relies on multiplying data by a downsampled version of the input. For a hardware implementation, this would require an infinitely growing buffer to store all past data for use with the current data. As this is not practical, steps were taken to determine a more efficient way to process the data while still maintaining algorithm performance. The concept of windowing, examining a small section of the original signal at a time, was found to be the most applicable. It is worthwhile to examine the trade-offs between window size and error and the rate at which the window moves through the data, among other things.

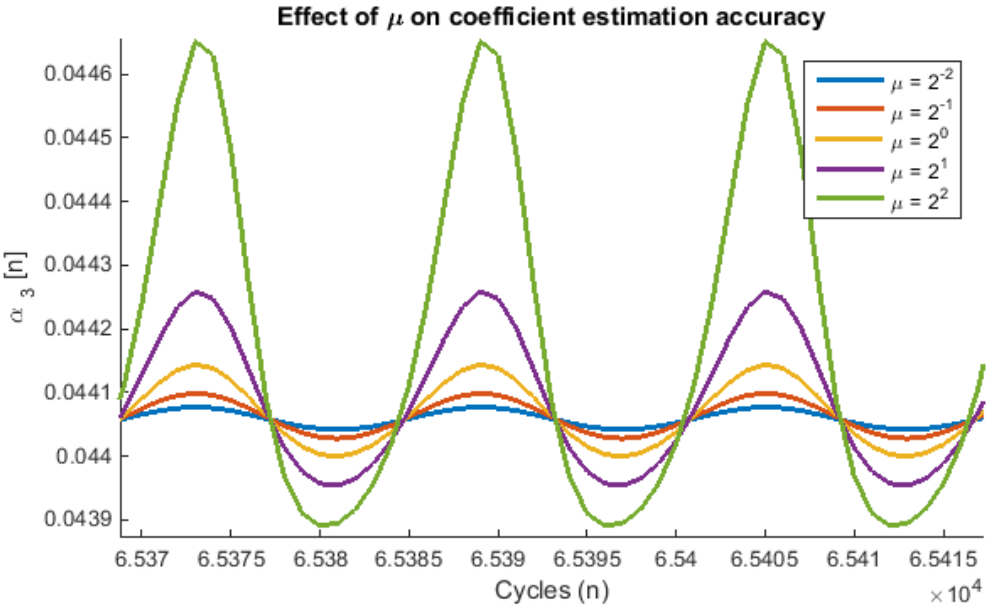


Figure 5.2: Impact of changing μ on estimator final accuracy

5.3.1 Shifting

Currently an updated α_3 is calculated with every new sample D_{out} that comes in. This means that the entire calibration algorithm calculation has to finish in one cycle of the ADC output. This is theoretically possible if the algorithm logic is clocked at a much higher rate than the ADC output speed. However, it would be ideal if it could run at or near the same speed as the ADC, and thus shifting the window by different amounts was simulated. This would allow the algorithm more time to complete before each distortion coefficient update. From the results in Figure 5.3, it can be seen that increasing the amount by which the window shifts for each α_3 estimation update does not impact performance; however, it does increase the convergence time, as the coefficient is not getting updated as often as before.

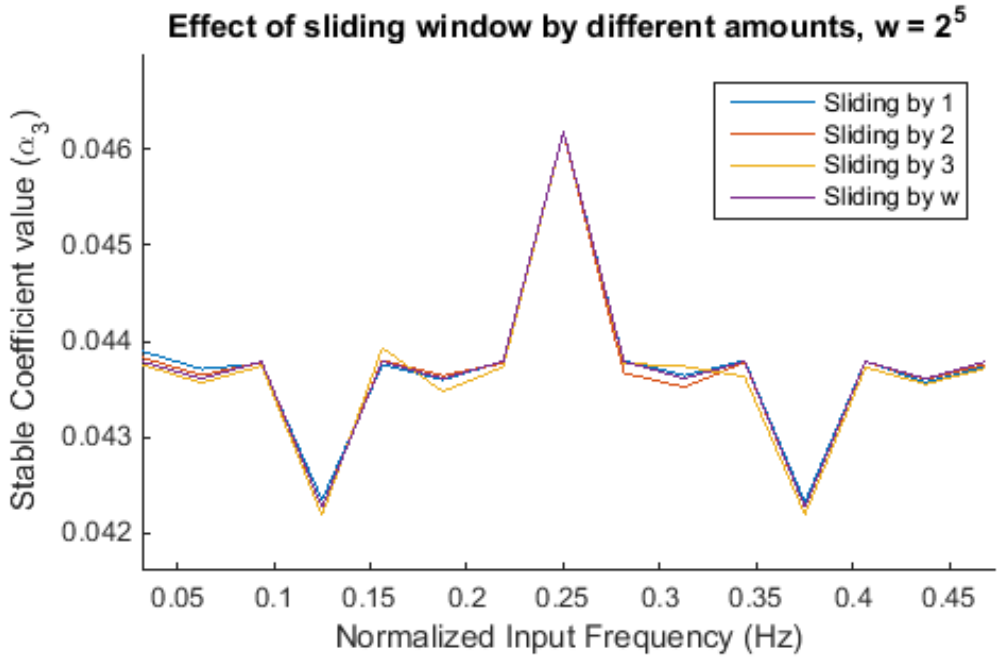


Figure 5.3: Effects of sliding algorithm window by varying amounts

5.3.2 Shape

The ultimate cause of the input frequency resolution limitation discussed in the previous chapter is that for a given window size, the error would not sum to 0 if there was not an integer number of periods in the data. In many cases there are only a few points missing that lead to this problem, as seen in Figure 4.5b. One way to get the results closer to those of exact orthogonality is to weight the data points at the center of the window as more important than those at the edges. This involves the use of classic window filters such as the Hanning, Hamming, and Blackman windows. The window responses are shown in Figure 5.4, and their impact on the input frequency resolution issue is shown in Figure 5.5. The windows are applied to D_{nosig} and D_{ds3} before the error calculation to get as close to exact orthogonality as possible.

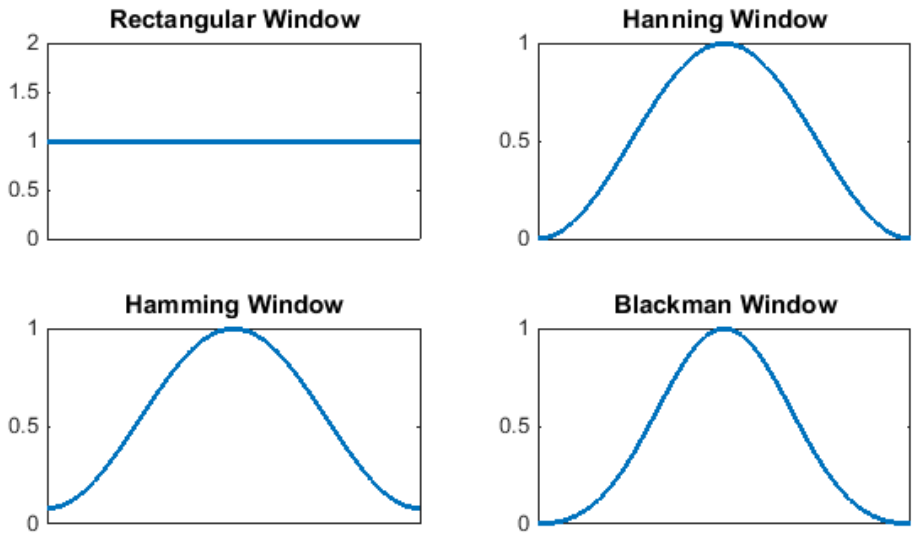


Figure 5.4: Classic window functions)

The three non-rectangular windows are all fairly similar in response, which can be seen in their behavior being nearly identical in Figure 5.5. Errors still exist around the band edges ($0F_s$ and $0.5F_s$) when using the window functions, though the middle of the band is consistently improved compared to the case without windowing (or rather, using a rectangular window). Around $0F_s$ the windows only contain a few (incomplete) cycles of the signal, such that any errors present are not averaged out. Around $0.5F_s$, or the maximum rate at which the data can be sampled accurately, there are plenty of cycles within the window, and thus a shaped window warps the data interfering with the algorithm accuracy.

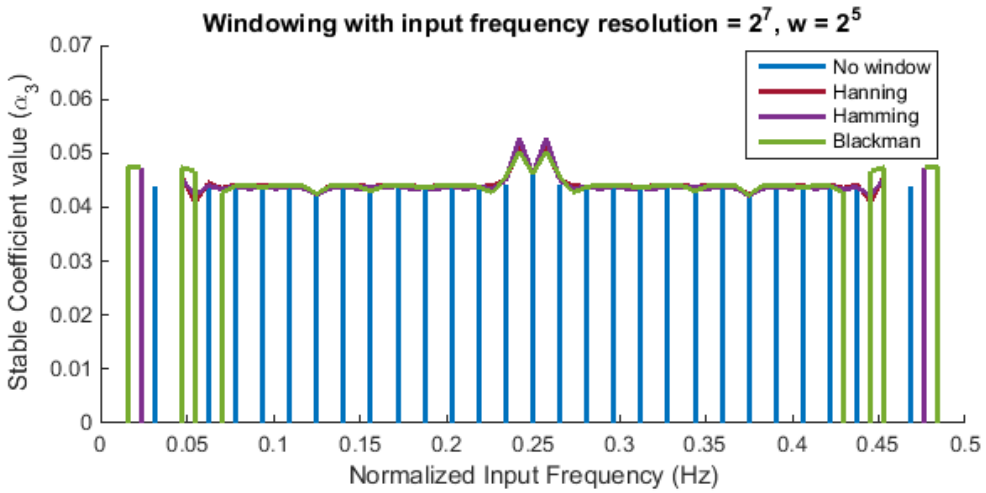


Figure 5.5: Improvement of algorithm performance after utilizing shaped windows (Hanning, Hamming, etc.)

5.3.3 Size

With the intention of minimizing hardware area and computation time, determining the smallest window size with an acceptable amount of error is a useful endeavor. However, the window needs to be large enough to capture at least one period of the signal for exact orthogonality. In order to account for a high resolution of input frequencies, an extremely large window must be used based upon the rules of orthogonality.

5.4 Fixed-Point

Floating point processors are typically more expensive and complicated than their fixed-point counterparts. For a synthesized version of the algorithm, fixed-point math could be used to determine the size of the internal data paths. The algorithm computations are somewhat prohibitive in the sense that for all of the required calculations, the data

paths grow in size very quickly. However, the algorithm can be designed based upon the resolution of the digital output from the ADCs. Another side effect of using fixed-point is the loss of precision due to overflow and rounding.

Chapter 6: Conclusion

While it is likely that nothing will ever completely solve the problem of nonlinearity, the blind calibration algorithm comes close. Looking at the limitations of the algorithm, along with optimizations that can be made, brings the design one step closer to actual practical implementation in silicon. Not all algorithms translate well to this final step, with some being near impossible, but with respect to the current results it appears that hardware implementation is still a viable option for future development. A further understanding of the novel blind calibration algorithm was presented such that future researchers have a better starting point for continuing the design.

Bibliography

[1] R. Schaller, "Moore's law: past, present and future," *Spectrum, IEEE*, vol. 34, pp. 52–59, Jun 1997.

[2] M. Gande, H.-Y. Lee, H. Venkatram, J. Guerber, and U.-K. Moon, "Blind background calibration of harmonic distortion based on selective sampling," in *Custom Integrated Circuits Conference (CICC), 2013 IEEE*, pp. 1–4, Sept 2013.

[3] T. Carusone, D. Johns, and K. Martin, *Analog Integrated Circuit Design*. Wiley, 2nd ed., 2011.

[4] J. LeClare, "A Simple ADC Comparison Matrix." Maxim Integrated, June 2003.

[5] C. Grace, P. Hurst, and S. Lewis, "A 12-bit 80-MSample/s pipelined ADC with bootstrapped digital calibration," *Solid-State Circuits, IEEE Journal of*, vol. 40, pp. 1038–1046, May 2005.

[6] J. McNeill, R. Majidi, and J. Gong, ""Split ADC" Background Linearization of VCO-Based ADCs," *Circuits and Systems I: Regular Papers, IEEE Transactions on*, vol. 62, pp. 49–58, Jan 2015.

[7] A. Panigada and I. Galton, "A 130 mw 100 ms/s pipelined adc with 69 db sndr enabled by digital harmonic distortion correction," *Solid-State Circuits, IEEE Journal of*, vol. 44, pp. 3314–3328, Dec 2009.

[8] A. V. Oppenheim and A. S. Willsky, *Signals & Systems*. Prentice Hall, 2nd ed., 1997.

[9] J. O. S. III, *Mathematics of the Discrete Fourier Transform (DFT): with Audio Applications*. W3K Publishing, 2nd ed., 2007.

[10] M. H. Richardson, "Fundamentals of the Discrete Fourier Transform," *Sound & Vibration Magazine*, March 1978.

[11] M. Gande, H. Venkatram, L. Ho-Young, J. Guerber, and U.-K. Moon, "Blind calibration algorithm for nonlinearity correction based on selective sampling," *Solid-State Circuits, IEEE Journal of*, vol. 49, pp. 1715–1724, Aug 2014.