2020

Cost-efficient solutions for analog and mixed-signal test and calibration challenges

Shravan Kumar Chaganti
Iowa State University

Cost-efficient solutions for analog and mixed-signal test and calibration challenges

by

Shravan Kumar Chaganti

A dissertation submitted to the graduate faculty

in partial fulfillment of the requirements for the degree of

DOCTOR OF PHILOSOPHY

Major: Electrical Engineering (Very Large Scale Integration)

Program of Study Committee:
Degang Chen, Major Professor
Randall Geiger
Nathan Neihart
Cheng Huang
Chinmay Hegde

The student author, whose presentation of the scholarship herein was approved by the program of study committee, is solely responsible for the content of this dissertation. The Graduate College will ensure this dissertation is globally accessible and will not permit alterations after a degree is conferred.

Iowa State University

Ames, Iowa

2020

DEDICATION

To my family.

TABLE OF CONTENTS

Page

LIST OF FIGURES

Page

LIST OF TABLES

Page

NOMENCLATURE

ADC	Analog to Digital Converter
AMS	Analog and Mixed-Signal
ATE	Automated Test Equipment
DAC	Digital to Analog Converter
DNL	Differential Non-Linearity
IC	Integrated Circuit
INL	Integral Non-Linearity
IoT	Internet of Things
SoC	System on Chip

ACKNOWLEDGMENTS

First of all, I would like to express my deepest gratitude to my major professor Dr. Degang Chen for his invaluable guidance, support and feedback throughout the course of my Ph.D.

I would also like to thank my committee members: Dr. Randall Geiger, Dr. Nathan Neihart, Dr. Cheng Huang and Dr. Chinmay Hegde for their time and valuable feedback on this work.

I would like to thank Dr. Amit Majumdar from Xilinx, Abalhassan Sheikh and Dr. Srivaths Ravi from Texas Instruments, and Xiankun Jin and Dr. Doug Garrity from NXP for their collaboration, support and inspiration on various research projects.

I would like to thank Semiconductor Research Corporation, Xilinx, NXP Semiconductors, Texas Instruments, and Iowa State University for the financial and technical support of my research work.

I am grateful towards my colleagues Kushagra Bhatheja, Tao Chen, Yuming Zhuang, Li Xu, Nanqi Liu, Zhiqiang Liu, Subhanwit Roy, Leandro Fuentes, Hao Meng, Sindhusha Dhulipala, and many more wonderful people at Iowa State University whose friendship and help during my time here has made my Ph.D a wonderful experience. I would like to convey my gratitude towards all my dear friends, for the memorable times.

Finally, I am deeply thankful to my family, and to Sir, for their love, support and sacrifice that has gotten me through the toughest of times. Words cannot express my love and gratitude for my parents, and my dearest sister Sakshi, for being there for me.

ABSTRACT

Integrated Circuits (ICs) are used in a myriad of applications and impact our lives every single day. Some of these applications are mission critical, like automotive, medical equipment, aircrafts etc., and thus have stringent quality and reliability requirements. Extensive testing is needed to guarantee adherence to these requirements. Test time and cost are typically very high for Analog and mixed-signal circuits. Analog-to-Digital Converters (ADCs) and Digital-to-Analog-Converters (DACs) are critical components of many of these ICs, and the cost associated with their testing often dominates the overall test cost of the ICs. There is thus an urgent need to develop methods that reduce test time and cost of data converters, and also ensure their reliability post deployment.

In this dissertation, we will provide several solutions to address these issues, with a focus on DACs. First, a segmented linearity testing method called uSMILE will be presented for reducing test time and thus cost of DAC linearity test by reducing the number of measurements required to estimate the nonlinearity of the DAC. Next, the uSMILE-ROME algorithm for DAC testing will be described. This not only reduces test time but significantly reduces test cost by eliminating the need for high precision measurement devices for linearity testing of high resolution DACs. A cheap on-board/on-chip digitizer with comparable resolution and worse linearity than the DAC under test can be used to get an accurate estimation of the DAC INL. The algorithm will further be adapted so that it can be run on-chip, enabling Built-in Self-test and Self-calibration of DACs, which ensures their long-term reliability. The uSMILE algorithm will then be modified to estimate and calibrate dynamic errors in the DAC, in addition to static errors. This enables low-cost high purity sine wave generation using a non-linear DAC. Finally, a Concurrent Sampling (CS) method will be introduced for measuring a multitude of analog DC voltages on-chip

concurrently using local comparators and a calibrated DAC. The previously mentioned test and calibration schemes can be used for DAC calibration.

uSMILE and its variants are currently used for production testing of multiple products at Texas Instruments and other semiconductor companies to reduce DAC linearity test time. A uSMILE-ROME based DAC built-in self-test and self-calibration scheme is being developed on a chip at NXP semiconductors. These BIST techniques, combined with Concurrent Sampling, enable real-time measurement of analog voltages, and will have a significant impact on the semiconductor industry because it addresses a growing need for automotive test and reliability.

CHAPTER 1. GENERAL INTRODUCTION

Semiconductor chips, or Integrated Circuits (ICs) pervade every single aspect of our lives today. We can find ICs today in phones, automobiles, homes, computers, watches, refrigerators, roads, aircrafts, medical equipment, … and even basketballs and shoes! The age of Internet-of-Things and 5G has mandated that people interact all day every day with appliances with some sort of ICs in them. Many of these applications, like automotive, medical, aircrafts etc., impact the lives of human beings directly. The failure of the electronics which are integral to the functioning of these systems can lead to catastrophic results. Hence, these devices need to meet extremely stringent quality and reliability requirements, both at the system level and at the chip level.

One way to guarantee that these devices meet these stringent safety and reliability standards, especially for mission critical systems like automotive, is through extensive testing. Moreover, aging effects and environmental changes cause performance degradation of IPs over time. The International Organization for Standardization (ISO) 26262 standard has therefore been developed to address the functional safety of automotive electronic systems [1]. Solutions on the digital front are fairly robust and mature. However, robust and reliable hardware solutions for analog, required to meet the standard, are lacking.

Ensuring that these devices meet these stringent quality and safety standards via extensive testing can be very expensive. The cost of test and calibration for Analog and mixed-signal ICs has been steadily rising to the point where it is a significant contributor to the overall cost of build of the IC. Increasing design complexity, new process nodes and defect models, etc. are pushing cost of test to the forefront of chip development cost. These are propelled by the increasing levels of integration to drive the system BOM (bill of materials) low, and the increasing quality needs from customers especially in automotive and other low dppm (defective parts per million) markets.

Data converters (ADCs and DACs) in particular are critical components of integrated circuits used in control/actuation and sensing applications. 5G communication, Internet of Things also drive a growing demand for chips containing data converters.

1.1 Background

Data convertors - Digital-to-Analog-Converters (DACs) and Analog-to-Digital-Convertors (ADCs) - are key interfaces between the physical analog world and the digital world and are widely used in mixed-signal integrated circuits today. With demand for high performing data convertors with increasing resolutions, the time required for testing them increases exponentially and hence, so does the test cost, which leads to an increase in the cost of the device.

Many parametric specifications of the DAC may need to be tested before shipping a part out to the customer. One category is measuring the integral nonlinearity (INL), the differential nonlinearity (DNL), offset and gain error. The other category is measuring the spectral performance of the DAC, including the Signal-to-Noise Ratio (SNR), Total Harmonic Distortion (THD), Spurious Free Dynamic Range (SFDR), etc. [2], [3]. There are also transient characteristics like settling time, glitch impulse area, etc.

Static linearity testing of DACs frequently dominates the overall test time of SoCs, and this directly translates to high test costs. Accurate testing of DACs in a time and cost-efficient way is a very challenging task. Conventional DAC static linearity testing is done by sweeping the input DAC code from 0 to the maximum code, and capturing the corresponding analog output voltages with a digitizer. Also, since the noise in measurement reduces only as the square root of the number of measurements per code, depending on the expected level of noise due to various sources, multiple hits per code are usually required to average out the noise to acceptably low levels [3]. The testing time is long and the test equipment is expensive. As the resolution of the DAC

increases, the number of codes to test increases exponentially. Typically, the DAC code update rate also reduces with increasing resolution. Hence, the test times, and thus test cost, also increase exponentially. For an n-bit DAC, if H samples per code are needed for noise averaging, then a total of $H \times 2^n$ output voltages will have to be measured. For a 16-bit DAC, with H=256, over 16 million samples would be required. If the DAC has a sampling rate of 500KSPS, the data acquisition time alone would be around 40 seconds. Multi-site testing will reduce this time, but this still corresponds to a very significant test cost.

In addition to long test times, expensive measurement devices are required as the performance of the DAC increases. Traditional testing of DAC static linearity is done using a digital voltmeter (DVM) or a digital waveform recorder [3]. IEEE standards dictate that the equipment is required to have significantly better accuracy and resolution than the specifications of the DAC itself. Especially in production test environments, the purchase of Automated Test Equipment (ATE) with high performance resources for precision voltage measurement can tremendously increase total test cost. The number of channels are also limited, which can be a bottleneck for massive multisite testing.

In the past, many researchers have proposed methods to reduce the cost of DAC linearity testing. The proposed method in [4] used stimulus error identification and removal (SEIR) [5] to obtain the ADC linearity first and estimate DAC INL/DNL with the ADC. However, the DAC INL/DNL estimation accuracy is limited and the test time is long. In [6], the authors developed a circuit with deterministic dynamic element matching (DDEM) ADC and a dithering DAC to test the DAC. It is capable of testing a 14-bit DAC with ADC at 6-bit linearity. But it has to use the proposed ADC circuit and it also has the long test time problem. In [7], Huang, et al. presented a static loopback testing technique for an ADC/DAC pair. The effective DAC resolution was raised

by scaling down the output during ADC testing using local histogram test. Then the effective ADC resolution was raised by scaling up the DAC output during DAC testing. This method is applicable for a segmented current steering DAC. It is architecture dependent and it takes a long time. In [8], Ting, et al, tested the current-steering DAC by measuring the major transition current difference with a current-controlled oscillator and counter. This method is fast and low-cost but it is highly architecture-dependent. Although many other BIST based methods have been proposed for DAC linearity testing to reduce cost [8]–[12], not many methods have been investigated to reduce the test time. Methods that reduce test cost are usually highly architecture dependent or require specialized ADC circuits.

The counterpart to the DAC, the Analog-to-Digital Converter (ADC) faces many of the same challenges for linearity testing, and thus, it is instructive to investigate and possibly, port over some of the techniques which have been introduced to reduce the time and cost of static linearity tests for ADCs to DACs. The ultrafast Segmented Model Identification of Linearity Errors (uSMILE) algorithm was first introduced for ADCs in [13]. The method significantly reduced the number of hits per code required to test an ADC by using a segmented non-parametric model for the INL. The USER-SMILE method [14], [15] then combined this method with SEIR to additionally relax the linearity requirement on the input ramp for ADC testing.

There is thus an urgent need to develop test time and cost reduction strategies for DACs. Methods will have to be developed for reducing production test and calibration cost for DACs so that the device cost does not increase. Secondly, viable techniques will need to be developed which can help ensure the reliability and safety of the device over its lifetime. BIST, self-calibration and self-diagnosis is the most promising solution to many of the challenges described above. More significantly, self-calibration based on BIST can lead to performance improvement and combined

with self-diagnosis, can ensure consistent reliability over time. The major focus of this dissertation is on optimizing production test time and cost of DACs, BIST of DACs, and the opportunities for in-field monitoring of analog circuits that this enables.

1.2 Dissertation organization

This dissertation is organized as follows. Chapter 2 will focus on reducing DAC linearity test time by reducing the number of samples to be measured. This will be done by reviewing and re-visiting the segmented model introduced in the uSMILE algorithm for ADCs. An alternative albeit equivalent Extrapolated Reconstruction (ER) method, which is also based on the segmented model but is more time and memory efficient, will also be developed [16]. The applicability of uSMILE and ER will be discussed for DACs with different architectures. Additionally, it will be shown that the segmented model is not suitable for architectures which involve interpolation. Hence, a new interpolated segmented model will be developed for accurate linearity testing of these DACs. Based on this new model, two new methods, namely, ultrafast Interpolated Segmented Model Identification of Linearity Errors (uISMILE) and Extrapolated Reconstruction with Interpolation (ER+I), will be developed for interpolated DACs.

In chapter 3, rigorous theoretical foundations will be laid down for the segmented model for the DAC INL along with the boundary conditions required to solve for unknowns using the least squares method. A fast and memory efficient implementation of uSMILE will also be discussed, which avoids the least square solution altogether. Additionally, rigorous noise analysis and derivation of the time savings factor compared to the conventional method will be performed for both uSMILE and uISMILE.

Chapter 4 will introduce the uSMILE-ROME algorithm for accurate linearity testing of DACs with dramatically reduced test time and cost [17]. Firstly, the segmented model from uSMILE will be used to reduce the number of measurements required. This reduction in test time

directly translates to reduction in test cost. Secondly, the method will use an on-board/on-chip digitizer for measurement of the DAC output, instead of the traditionally used high accuracy digital voltmeter. Shifted and non-shifted versions of the DAC output will be processed off-chip to remove the nonlinearity of the measuring device. It will hence be shown that the algorithm relaxes the stringent linearity requirement on the measurement device by removing the errors introduced due to the nonlinearity of the device. This will result in further cost savings. Extensive analysis will be done to evaluate the effect of noise on the algorithm as well as the shift non-constancy.

In chapter 5, a complete on-chip DAC BIST and self-calibration solution will be presented based on uSMILE-ROME, with a sub-radix 2 DAC structure used as an example. The theoretical basis for the modifications to the algorithm computation will be developed, and a complete on-chip processing algorithm, optimized for time and memory, will be presented. The solution can be implemented either on hardware or in software if a processor is available on-chip.

In chapter 6, a method for low-cost high purity sine wave generation will be presented. A dynamic version of the uSMILE algorithm will be developed, which will enable use of a non-linear DAC to generate a pure sine wave, by calibration of both static and dynamic errors.

Chapter 7 will introduce a low-cost method for separation of ADC noise, aperture jitter, and clock jitter, and accurate ADC spectral testing under non-coherent sampling conditions. This method will relax the need for a high purity clock source for separation and accurate estimation of the intrinsic aperture jitter of the ADC by using a dual channel test setup. This jitter separation and estimation method will therefore enable accurate estimation of the spectral performance for high speed ADCs.

Finally, in chapter 8, an innovative Concurrent Sampling method will be introduced, which will provide an alternative to the analog test bus for measurement of on-chip DC voltages for test

and debug. A local comparator and calibrated DAC will be used to digitize the voltages locally, which will then be streamed out using IJTAG. The concept of fault propagation graphs (FPG) which has been developed previously, can help identify a set of analog nets, which when monitored using Concurrent Sampling, will provide near complete analog fault coverage. Concurrent sampling, enabled by a calibrated DAC using the methods from previous chapters, combined with the FPG, will not only help improve fault coverage but also enable real time fault monitoring for AMS circuits in the field.

1.3 References

[1] 14:00-17:00, "ISO 26262-1:2018," *ISO*. [Online]. Available: http://www.iso.org/cms/render/live/en/sites/isoorg/contents/data/standard/06/83/68383.html. [Accessed: 11-Nov-2019].

[2] M. Burns, G. W. Roberts, and F. Taenzler, *An introduction to mixed-signal IC test and measurement*, vol. 2001. Oxford University Press New York, 2001.

[3] "IEEE Standard for Terminology and Test Methods of Digital-to-Analog Converter Devices," *IEEE Std 1658-2011*, pp. 1–126, Feb. 2012.

[4] Y. Zhuang, B. Magstadt, T. Chen, and D. Chen, "High-Purity Sine Wave Generation Using Nonlinear DAC With Predistortion Based on Low-Cost Accurate DAC-ADC Co-Testing," *IEEE Transactions on Instrumentation and Measurement*, vol. PP, no. 99, pp. 1–9, 2017.

[5] L. Jin, K. Parthasarathy, T. Kuyel, D. Chen, and R. L. Geiger, "Accurate testing of analog-to-digital converters using low linearity signals with stimulus error identification and removal," *IEEE Transactions on Instrumentation and Measurement*, vol. 54, no. 3, pp. 1188–1199, Jun. 2005.

[6] H. Xing, D. Chen, and R. Geiger, "On-chip at-speed linearity testing of high-resolution high-speed DACs using DDEM ADCs with dithering," in *2008 IEEE International Conference on Electro/Information Technology*, 2008, pp. 117–122.

[7] X. L. Huang and J. L. Huang, "ADC/DAC Loopback Linearity Testing by DAC Output Offsetting and Scaling," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 19, no. 10, pp. 1765–1774, Oct. 2011.

[8] H.-W. Ting, S.-J. Chang, and S.-L. Huang, "A Design of Linearity Built-in Self-Test for Current-Steering DAC," *J Electron Test*, vol. 27, no. 1, pp. 85–94, Feb. 2011.

[9] K. Arabi, B. Kaminska, and M. Sawan, "On chip testing data converters using static parameters," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 6, no. 3, pp. 409–419, Sep. 1998.

[10] I. H. S. Hassan, K. Arabi, and B. Kaminska, "Testing digital to analog converters based on oscillation-test strategy using sigma-delta modulation," in *Proceedings International Conference on Computer Design. VLSI in Computers and Processors (Cat. No.98CB36273)*, 1998, pp. 40–46.

[11] K. P. S. Rafeeque and V. Vasudevan, "A built-in-self-test scheme for digital to analog converters," in *17th International Conference on VLSI Design. Proceedings.*, 2004, pp. 1027–1032.

[12] J.-L. Huang, C.-K. Ong, and K.-T. Cheng, "A BIST scheme for on-chip ADC and DAC testing," in *Proceedings Design, Automation and Test in Europe Conference and Exhibition 2000 (Cat. No. PR00537)*, 2000, pp. 216–220.

[13] Z. Yu and D. Chen, "Algorithm for dramatically improved efficiency in ADC linearity test," in *2012 IEEE International Test Conference*, 2012, pp. 1–10.

[14] T. Chen and D. Chen, "Ultrafast stimulus error removal algorithm for ADC linearity test," in *2015 IEEE 33rd VLSI Test Symposium (VTS)*, 2015, pp. 1–5.

[15] X. Jin *et al.*, "An on-chip ADC BIST solution and the BIST enabled calibration scheme," in *2017 IEEE International Test Conference (ITC)*, 2017, pp. 1–10.

[16] S. K. Chaganti, A. Sheikh, S. Dubey, F. Ankapong, N. Agarwal, and D. Chen, "Fast and accurate linearity test for DACs with various architectures using segmented models," in *2018 IEEE International Test Conference (ITC)*, 2018, pp. 1–10.

[17] S. K. Chaganti, T. Chen, Y. Zhuang, and D. Chen, "Low-cost and accurate DAC linearity test with ultrafast segmented model identification of linearity errors and removal of measurement errors (uSMILE-ROME)," in *2018 IEEE International Instrumentation and Measurement Technology Conference (I2MTC)*, 2018, pp. 1–6.

CHAPTER 2. FAST AND ACCURATE LINEARITY TEST FOR DACS WITH VARIOUS ARCHITECTURES USING SEGMENTED MODELS

Shravan K. Chaganti*, Abalhassan Sheikh†, Sumit Dubey†, Frank Ankapong†, Nitin Agarwal†, and Degang Chen*

*Iowa State University, Ames, USA

†Texas Instruments Inc.

Modified from a manuscript published in the proceedings of the IEEE International Test Conference 2018

Abstract

Production test of parametric specifications is a significant contributor to the overall cost of build for analog and mixed-signal products. Data converters (ADCs and DACs) in particular are critical components of integrated circuits used in control/actuation and sensing applications. If left un-optimized, their production test time often dominates the overall system-on-chip (SoC) test time. In this chapter, we specifically focus on static linearity test of DACs and propose architecture-aware test methods that are combined with best-in-class fast linearity test concepts in the literature to minimize test time without compromising test quality.

The proposed methods exploit the hypothesis that the number of device errors which contribute to linearity errors can be captured by a significantly fewer number of variables than the number of codes at which linearity needs to be tested. We introduce a new time and memory efficient method called Extrapolated Reconstruction (ER) to calculate DAC INL and DNL, based on the segmented model introduced in uSMILE. We also demonstrate that since the segmented model techniques do not account for interpolation, they are not suitable for interpolated DACs.

We thus develop an interpolated segmented model and enhance both uSMILE and ER to obtain two new methods that provide correct estimations for interpolated DACs. A linearity test time reduction of 15x-20x was seen in actual silicon measurement results for multiple 12-bit DACs and >100x was seen in simulation case studies for many 16-bit DACs

2.1. **Introduction**

Increasing design complexity, new process nodes and defect models, etc. are pushing cost of test to the forefront of chip development cost. These are propelled by the increasing levels of integration to drive the system BOM (bill of materials) low, and the increasing quality needs from customers especially in automotive and other low dppm (defective parts per million) markets.

Data convertors - Digital-to-Analog-Converters (DACs) and Analog-to-Digital-Convertors (ADCs) - are key interfaces between the physical analog world and the digital world and are widely used in mixed-signal integrated circuits today. With demand for high performing data convertors with increasing resolutions, the time required for testing them increases exponentially and hence, the test cost.

Many parametric specifications of the DAC may need to be tested before shipping a part out to the customer. One category is measuring the integral nonlinearity (INL), the differential nonlinearity (DNL), offset and gain error. The other category is measuring the spectral performance of the DAC, including the Signal-to-Noise Ratio (SNR), Total Harmonic Distortion (THD), Spurious free dynamic range (SFDR), etc. [1], [2]. There are also transient characteristics like settling time, glitch impulse area, etc. In this chapter, we specifically focus on optimizing DAC linearity test time.

Static linearity testing of DACs frequently dominates the overall test time of SoCs, and this directly translates to high test costs. Accurate testing of DACs in a time and cost-efficient way is a very challenging task. Conventional DAC static linearity testing is done by sweeping the input

DAC code from 0 to the maximum code, and capturing the corresponding analog output voltages with a digitizer. Depending on the expected level of noise due to various sources, multiple hits per code are usually required to average out the noise to acceptably low levels [2]. The testing time is long and the test equipment is expensive. These problems are only worsened as the resolution of the DAC increases.

In the past, many researchers have proposed methods to reduce the cost of DAC linearity testing. The proposed method in [3] used stimulus error identification and removal (SEIR) [4] to obtain the ADC linearity first and estimate DAC INL/DNL with the ADC. However, the DAC INL/DNL estimation accuracy is limited and the test time is long. In [5], a circuit with a deterministic dynamic element matching (DDEM) ADC and a dithering DAC was developed to test the DAC. A 14-bit DAC can be tested by an ADC with just 6-bit linearity using this method, but the proposed specialized ADC circuit has to be used. In [6], Huang, et al. presented a static loopback testing technique for an ADC/DAC pair. The effective test resolution was raised by scaling and offsetting the DAC output. The effective DAC resolution was raised by scaling down the output during ADC testing. Conversely, the effective ADC resolution was raised by scaling up the DAC output during DAC testing. This method is applicable for a segmented current steering DAC. In [7], Ting, et al, tested the current-steering DAC by measuring the major transition current difference with a current-controlled oscillator and counter. This method is fast and low-cost but it is highly architecture-dependent. Although many other BIST based methods have been proposed for DAC linearity testing to reduce cost [7]–[11], not many methods have been investigated to reduce the test time and cost by reducing the number of samples that need to be measured.

The counterpart to the DAC, the Analog-to-Digital Converter (ADC) faces many of the same challenges for linearity testing, and thus, it is instructive to investigate and possibly, port

over some of the techniques which have been introduced to reduce the time and cost of static linearity tests for ADCs to DACs. The ultrafast Segmented Model Identification of Linearity Errors (uSMILE) algorithm was first introduced for ADCs in [12]. The method significantly reduced the number of hits per code required to test an ADC by using a segmented non-parametric model for the INL. The USER-SMILE method [13], [14] then combined this method with SEIR to additionally relax the linearity requirement on the input ramp for ADC testing. This algorithm was used as the inspiration to develop the uSMILE-ROME method for DAC testing [15], which solved the dual challenges of reducing the number of samples to be taken as well as reducing the test cost by enabling use of low-linearity on-board/on-chip digitizers as opposed to high accuracy digital voltmeters. ROME or "Removal of Measurement Error" method relaxes the linearity requirement of the digitizer which is used to measure the output voltages of the DAC. However, it requires the ability to add a constant voltage shift between the DAC and the ADC. Although implementing the voltage shift is not very hard, this facility is sometimes unavailable and a high accuracy digitizer is used to measure the output. For such cases, the segmented model can still be applied independently to reduce the number of measurements that need to be taken.

This chapter will focus on reducing test time by reducing the number of samples to be measured. This will be done by reviewing and re-visiting the segmented model introduced in the uSMILE algorithm. An alternative albeit equivalent Extrapolated Reconstruction (ER) method, which is also based on the segmented model but is more time and memory efficient, is developed. The applicability of uSMILE and ER is discussed for DACs with different architectures. Additionally, it is shown that the segmented model is not suitable for architectures which involve interpolation. Hence, a new interpolated segmented model is developed for accurate linearity testing of these DACs. Based on this new model, two new methods, namely, ultrafast Interpolated

Segmented Model Identification of Linearity Errors (uISMILE) and Extrapolated Reconstruction with Interpolation (ER+I), are developed for interpolated DACs. Extensive simulations have been performed to demonstrate the different methods, and their applicability to DACs with various architectures. Measurement results of multiple 12-bit DACs also show the effectiveness of the different methods in drastically reducing the static linearity test time for DACs over the conventional method.

The remainder of the chapter is organized as follows. Section II reviews the conventional method and uSMILE, and develops the extrapolated reconstruction method for DACs. Section III presents some simulation results for the uSMILE and ER methods. Section IV illustrates the limitations of the previous methods for DACs which have interpolation and develops new algorithms which are suitable for testing such DACs. Section V presents simulation results for these new algorithms. Section VI demonstrates the effectiveness of the proposed methods with measurement results. Section VII discusses advantages and disadvantages of the different proposed methods as well as on-chip implementation. Section VIII finally concludes the chapter.

2.2. Segmented Models for DAC Linearity

2.2.1 Conventional method and its drawbacks

The conventional method for testing the linearity of a DAC involves sweeping the input code from 0 to the maximum code, and then measuring the output voltages using a digitizer. The digitizer can be in the form of a Digital Voltmeter or a higher resolution ADC with significantly better specifications than the DAC under test. Whichever the case, multiple measurements per code are required to average out the noise. Once the output voltages have been measured, the output voltages are subtracted from either an end-point fit line or a best fit line, and divided by the average voltage difference between 2 consecutive codes to get the INLs at each code in units of

LSBs. As the resolution grows, the number of input codes grows exponentially, and so does the number of measurements that need to be taken. There are 2^n input codes for an n-bit DAC. Let's say that h number of measurements need to be taken per code to average out noise. The total number of measurements would then be $h \times 2^n$. For a 16-bit DAC, with h=64, over 4 million measurements would need to be taken. At a sampling rate of, say 500KSPS, the data acquisition alone will take around 9 seconds! Even with multi-site testing, this will result in significant test time and cost per chip.

2.2.2 uSMILE : ultrafast Segmented Model Identification of Linearity Errors

The conventional method essentially treats the INL/DNL error at each code as unrelated to each other, and so, the number of variables to be estimated is equal to the number of DAC codes. This is highly inefficient. In reality, especially for high resolution DACs, the number of truly independent error sources due to non-idealities of analog components is much smaller than the number of codes. For example, take a 16-bit R-2R DAC. The number of resistor mismatches is just $2 \times 16 - 1 = 31$ which is dramatically less than $2^{16} = 65,536$. Although there will be many more error sources, it is true that a limited number of independent error terms are sufficient to capture the errors in the input output transfer curve of the DAC. In other words, all the INL/DNL errors are highly correlated and are deterministic functions of a much smaller number of independent errors.

This correlated nature of the INL/DNL DAC errors makes a strong case for a model based approach to DAC linearity testing. The uSMILE method models the DAC's INL curve with a segmented non-parametric model. The INL curve of the DAC is broken into many MSB segments according to the MSB (Most Significant Bits) value of the DAC input code. Take a 16-bit DAC for example. If 6 bits are used as the MSB, then the INL curve is divided into 64 different segments.

Each of these segments has an error term associated with it, say $e_M(C_{MSB})$, where C_{MSB} ranges from 0 to 63. Each of these segments in turn can be further divided into smaller segments. Say the next 5 bits are used as ISB (Intermediate Significant Bits), then each MSB segment gets divided into 32 ISB segments, each of which has an error term associated with it, say $e_I(C_{ISB})$. If we stop the segmentation here, the variations within each ISB segment away from the ISB average values are captured by the 32 LSB errors (5 LSB bits). The error term associated with each LSB segment is denoted as $e_L(C_{LSB})$. The final INL value for code C will be

$$INL(C) = e_M(C_{MSB}) + e_I(C_{ISB}) + e_L(C_{LSB}) \tag{2.1}$$

For higher resolution DACs, segmented architectures are inherently used to avoid the exponential growth of components. This segmented non-parametric model can thus be applied to binary weighted, R-2R, current steering, mDAC, hybrid DACs etc. The segmented model of the DAC's INL enables us to estimate the INLs with significantly fewer output voltage measurements because it drastically reduces the number of variables to be estimated. Let's say that we have an n-bit DAC, and we sweep the DAC input code from 0 to max code, while taking 1 measurement per code. We calculate the preliminary INLs at each code, which will obviously have a significant amount of noise. Let's say we do an nMSB-nISB-nLSB segmentation of the DAC. We can write equation (2.1) at every code, giving us $M = 2^n$ equations. The number of unknowns $K = (2^{nMSB} + 2^{nISB} + 2^{nLSB})$. Since this is an overdetermined system with more number of equations than unknowns, the method of least squares can be used to compute the unknowns. Let's define e_M as a column vector:

$$e_M = \begin{bmatrix} e_M(0) \\ e_M(1) \\ . \\ . \\ e_M(2^{nMSB}-1) \end{bmatrix} \tag{2.2}$$

e_I and e_L are defined in a similar way. We can then write $e_M(C_{MSB})$ as:

$$e_M(C_{MSB}) = [0 \quad 0 \quad \ldots \quad 1 \quad \ldots \quad 0]e_M \tag{2.3}$$

with the 1 being placed at the $C_{MSB}+1$ location. The INL at code C can thus be written as:

$$INL(C) = [0\ldots1\ldots0\ldots1\ldots0\ldots1\ldots0]\begin{bmatrix} e_M \\ e_I \\ e_L \end{bmatrix} \tag{2.4}$$

with locations of the 1s depending on C_{MSB} , C_{ISB} and C_{LSB} . We can then combine all the equations at every code in matrix form as:

$$P = H \times e + noise \tag{2.5}$$

where P is the preliminary full code INL column matrix of length M , H is an $M \times K$ coefficient matrix which has three +1s in each row, the placement of which depend on the DAC code, and $e = [e_M \quad e_I \quad e_L]^T$ is a column matrix of length K. The unknown vector e can be estimated using least squares as:

$$\hat{e} = H_{inv}P \tag{2.6}$$

where $H_{inv} = (H^T H)^{-1} H^T$.

Once the unknowns are estimated, the full code noise-free INL vector F can be reconstructed as

$$F = H \times \hat{e} \tag{2.7}$$

The method of least squares naturally averages out the noise. The average number of measurements per unknown $= M / K$. We will call this the time saving factor (tsf). This also gives us the equivalent number of hits per code. For a 16-bit DAC with an 8-4-4 segmentation,

$M = 2^{16} = 65535$ and $K = 2^8 + 2^4 + 2^4 = 288$. Hence, the equivalent hits per code $= 65535 / 288 \approx 228$ i.e. the estimated INLs using uSMILE with just 1 measurement per code should be as accurate as if we had taken 228 measurements per code and calculated INL using the conventional method. The uSMILE algorithm essentially takes a noisy INL estimation (P) as an input and gives a noise-free estimation of the INL (F) as an output. In this sense, it acts like a noise filter, based on the segmented model. Thus, uSMILE enables us to estimate the INL/DNL of the DAC at each code with a much-reduced number of samples. The time and space complexity of uSMILE is discussed in Section VII.

Note that this segmented model is not valid for string or thermometer-coded type architectures. For example, if you have a segmented 15-bit DAC implemented as a 7-bit thermometer coded resistor DAC and an 8-bit R-2R DAC, then the segmentation of the INL curve must be carefully chosen such that the MSB bits are greater than or equal to 7, since the thermometer coded part does not have a segmented architecture. For example, a 7-4-4 segmentation of the INL curve is valid for this DAC, and so is an 8-3-4 segmentation, but a 6-5-4 or 5-5-5 segmentation is not valid. As will be seen in the next section, the segmented model also cannot be applied directly to DACs which have interpolation.

2.2.3 Extrapolated Reconstruction (ER) method

The basic idea behind uSMILE is that a limited number of non-idealities determine the deviation from the ideal output voltages of a DAC. If we can determine these accurately, then the INL at each code can be determined. When the architecture of the DAC is segmented by MSBs, ISBs and LSBs, we say that the INL at code C can be written as the error due to the MSB DAC plus the error due to the ISB DAC plus the error due to the LSB DAC, as shown in equation (2.1) . This is derived from the fact that the output voltage at code C is equal to the output voltage of the

MSB DAC plus the output voltage of the ISB DAC plus the output voltage of the LSB DAC, or in other words,

$$V(C) = V_M(C_{MSB}) + V_I(C_{ISB}) + V_L(C_{LSB}) \tag{2.8}$$

This means that if we can estimate $V_M(C_{MSB})$, $V_I(C_{ISB})$ and $V_L(C_{LSB})$ accurately, then we can extrapolate and re-construct the output voltages for all codes! For both ADCs and DACs, the non-linearities are defined per code. Unlike in ADCs, where we do not know the output code that the ADC will produce, DAC linearity testing offers a unique opportunity – we can actually obtain the measurements for specific output codes. As we will see, we can measure the output of the DAC accurately at specific limited number of codes, and derive $V_M(C_{MSB})$, $V_I(C_{ISB})$ and $V_L(C_{LSB})$ from these measurements and thus reconstruct the output for all codes.

Let's say we do an nMSB-nISB-nLSB segmentation of the DAC. We define the output voltage at code k as

$$v(k_{MSB}, k_{ISB}, k_{LSB}) = V_M(k_{MSB}) + V_I(k_{ISB}) + V_L(k_{LSB}) = V(k) \tag{2.9}$$

where $k_{MSB}, k_{ISB},$ and k_{LSB} are the MSB, ISB and LSB codes of code k.

First, we measure the outputs at all the "MSB points", which are basically all codes k for which the ISB and LSB codes are 0. Let's call these measurements $v_M(j) = v(j,0,0)$ where j varies from 0 to $2^{nMSB} - 1$. Next, we choose any one MSB segment (all codes for which $k_{MSB} = m$, say), and then measure the outputs at all the "ISB points" in this segment (all codes for which $k_{MSB} = m$ and $k_{LSB} = 0$). Let's call these measurements $v_I(j) = v(m, j, 0)$ where j varies from 0 to $2^{nISB} - 1$. Finally, we choose any ISB segment in this MSB segment (all codes for which $k_{MSB} = m$ and $k_{ISB} = i$, say), and measure outputs at all the "LSB points". Let's call these measurements

$v_L(j) = v(m,i,j)$ where j varies from 0 to $2^{nLSB} - 1$. We can now write the voltage at any code C as:

$$\begin{aligned}\hat{V}(C) &= V_M(C_{MSB}) + V_I(C_{ISB}) + V_L(C_{LSB}) \\ &= [V_M(C_{MSB}) + V_I(0) + V_L(0)] \\ &\quad + [V_M(m) + V_I(C_{ISB}) + V_L(0)] - [V_M(m) + V_I(0) + V_L(0)] \\ &\quad + [V_M(m) + V_I(i) + V_L(C_{LSB})] - [V_M(m) + V_I(i) + V_L(0)] \\ &= v(C_{MSB}, 0, 0) \\ &\quad + v(m, C_{ISB}, 0) - v(m, 0, 0) \\ &\quad + v(m, i, C_{LSB}) - v(m, i, 0) \\ &= v_M(C_{MSB}) + v_I(C_{ISB}) - v_I(0) + v_L(C_{LSB}) - v_L(0)\end{aligned} \tag{2.10}$$

Thus, we can "reconstruct" the output voltage at all codes by measuring the output voltages at just a few codes. We can then calculate the end-point fit/best fit INLs and DNLs for all the codes just like we would have done in the conventional method.

Let's take a 12-bit DAC as an example, with a two level 7-5 segmentation to make things easier. Every 32 codes form one MSB segment. First, we measure the outputs at all the MSB points i.e. at codes 0, 32, 64, … , 2048, 2080, … , and 4064 with multiple hits per code to average out the noise. Then we measure the outputs for all the LSB points in one MSB segment. If we choose this at mid-code, that means we measure at codes 2048, 2049, … , 2080. From just these measurements, we can extrapolate and reconstruct the outputs for all the DAC codes.

For an n-bit DAC, if the conventional method requires measuring the output voltages at $M = 2^n$ number of codes with, say h hits per code to average out the noise, then the proposed Extrapolated Reconstruction (ER) method would require measurement at $K = (2^{nMSB} + 2^{nISB} + 2^{nLSB})$ number of codes with the same h hits per code. Hence, the time saving factor $tsf = M / K$. For a 16 bit DAC with 8-4-4 segmentation, this value is nearly equal to 228. Notice that the tsf for ER is exactly the same as the tsf for uSMILE. In uSMILE, we measure the output voltage at all the codes, with 1 measurement per code. In ER, we measure the output at significantly fewer codes, but with multiple measurements per code. To give equal noise averaging

capabilities, the total number of measurements for both methods will be the same. Although both methods are equivalent with respect to the accuracy of estimation, the computation and memory requirements are less for ER. Details of the advantages and disadvantages of each method will be discussed in Section VII.

2.3. Simulation Results for uSMILE and ER

We will now show the effectiveness of uSMILE and ER via simulations. First, a 16-bit R-2R DAC was modeled in Matlab with resistor mismatches. 0.5LSB of additive noise was added to the output. For simplicity, the DAC was segmented as 8-8 for uSMILE and ER. For ER, the output voltages were measured at $2^8 + 2^8 = 512$ codes, with 256 measurements per code. For uSMILE, measurements were taken at all 2^{16} codes, with 2 measurements per code. This makes the equivalent number of hits per code equal to 256, because $tsf = 2^{16} / (2^8 + 2^8) = 128$.

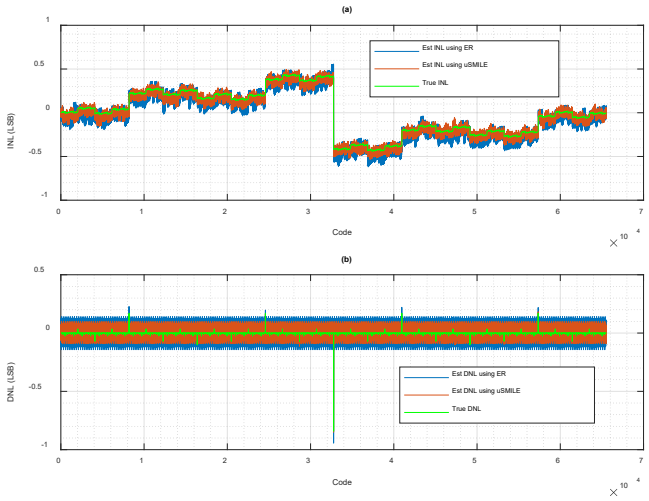


Figure 2.1. Simulation results of a 16b R-2R DAC
(a) INL estimations using uSMILE and ER
(b) DNL estimations

The true and estimated INL curves are shown in Figure 2.1(a), along with the INL estimation errors. Similarly, the DNL curves are shown in Figure 2.1(b). We can clearly see that there is very good correlation between the true and estimated INLs and DNLs.

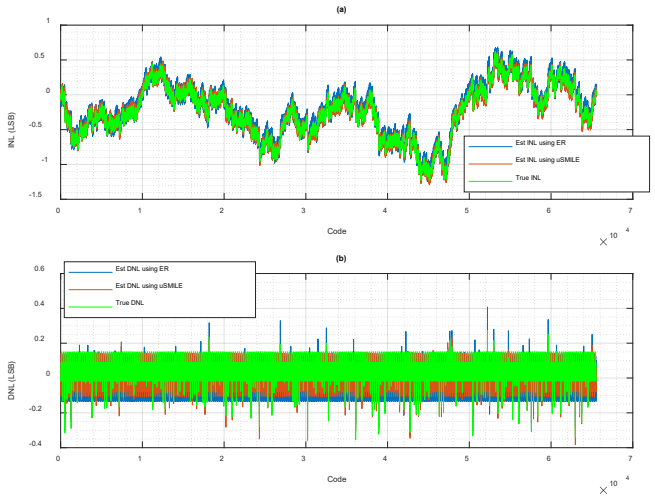


Figure 2.2. Simulation results of a 16b Segmented Hybrid DAC
(a) INL estimations using uSMILE and ER
(b) DNL estimations

Next, a 16-bit segmented hybrid DAC implemented as an 8-bit thermometer coded resistor DAC and an 8-bit R-2R DAC is modeled with resistor mismatches, with the additive noise at 0.5 LSB level. With the same parameters as for the R-2R DAC, the INLs are estimated with the 2 methods. The true and estimated INL curves are shown in Figure 2.2, along with the DNL estimation errors. Once again, the correlation is very good between the true and estimated INLs and DNLs. These simulation results validate that uSMILE and ER can accurately estimate the INL/DNL with a significantly reduced number of measurements.

To check their effectiveness for another DAC architecture, a 16b interpolated DAC is generated, with the 8-bit LSB R-2R DAC interpolating between the output voltages of the 8-bit MSB string DAC. The INL and DNL curves are shown in Figure 2.3. In this case, it is clearly visible that the DNL estimations are inaccurate. We see that there are spikes in the DNL

estimations at the MSB points. The reason for this will be explained in the next section. These figures indicate that the segmented model is not suitable for interpolated DACs.

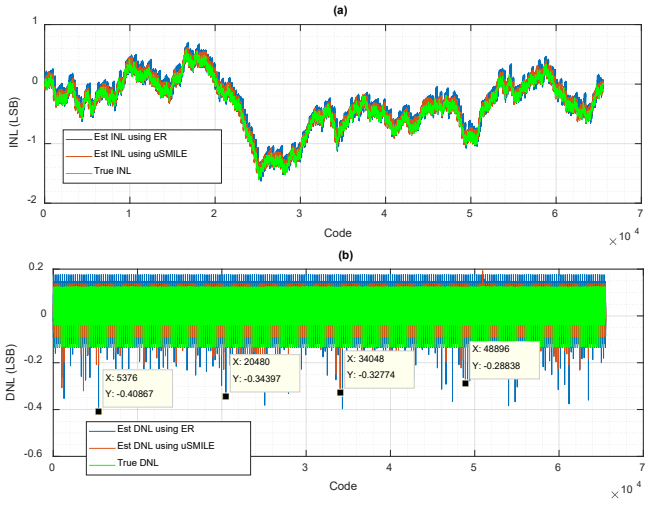


Figure 2.3. Simulation results of a 16b interpolated DAC
(a) INL estimations using uSMILE and ER
(b) DNL estimations

2.4. Interpolated Segmented Models for DAC Linearity

The simulation results in the previous section show that the uSMILE and ER methods give accurate INL and DNL estimations for R-2R and hybrid DACs, but are not suitable for DACs which have interpolation. This is to be expected because the segmented model does not account for interpolation. We see especially bad estimations at the MSB points, with large spikes in the DNL estimations at these codes. The reason for this is as follows. In the segmented model, we

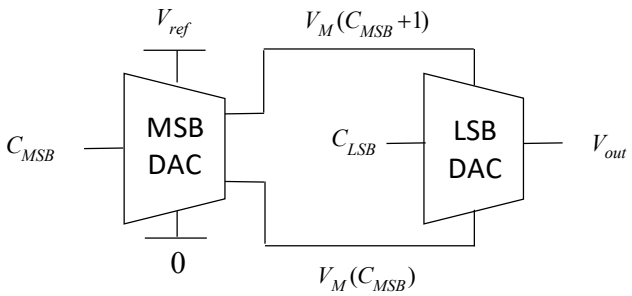


Figure 2.4. Interpolated DAC architecture

assume that the output voltage due to the LSB DAC gets added to the output voltage of the MSB DAC, assuming a 2 level segmentation for simplicity. But since the architecture of the interpolated DAC, shown in Figure 2.4, is such that $V(C_{MSB})$ and $V(C_{MSB}+1)$ effectively become the low and high references for the subsequent LSB DAC, all the voltages get interpolated or scaled between these two voltages.

Figure 2.5 gives a better visual representation of what is happening, for example, during extrapolated reconstruction. When we extrapolate and add the LSB DAC voltages from the m'th

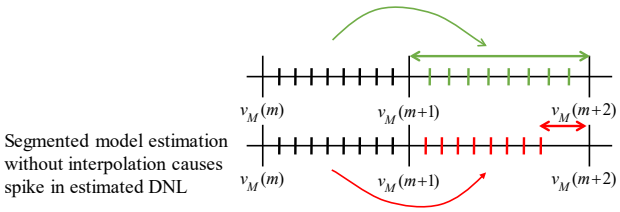


Figure 2.5. Actual output voltages vs segmented model estimation for a segmented DAC with interpolation

MSB segment to the (m+1)'th MSB segment, without interpolation, there might be a sudden jump up/down in voltage between the last LSB code and the next MSB code if $v_M(m+2)-v_M(m+1)$ significantly differs from $v_M(m+1)-v_M(m)$, whereas the actual DAC itself is interpolating and stretching the LSB DAC voltages between $v_M(m+1)$ and $v_M(m+2)$, which prevents any sudden jumps. Similar logic can be used for uSMILE, the difference being that uSMILE will try to estimate the LSB DAC voltages such that these jumps are minimized across all MSB segments in the least squares sense. Hence, in both the segmented model estimations, there will be sudden jumps up/down in voltage when we go from one MSB segment to the other, which manifest as large DNL errors. Given this deficiency of the segmented model, we need to come up with a modified version which accounts for interpolation.

2.4.1 uISMILE: ultrafast Interpolated Segmented Model Identification of Linearity Errors

We will now develop a model for the INL curve which is segmented and accounts for interpolation. In order to simplify the derivation of the equations, we will consider a 2 level nMSB-nLSB segmentation, assuming that the LSB DAC interpolates between the MSB DAC output voltages, as shown in Figure 2.4. This LSB DAC can always be further segmented as required. Figure 2.6 shows how the output voltage V_{out} is being generated for some code C. Let's say that

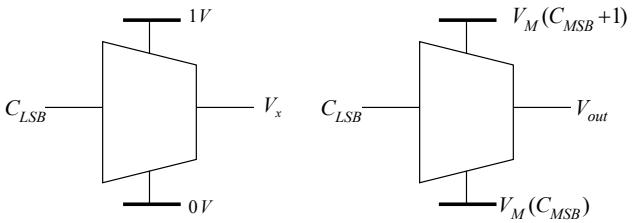


Figure 2.6. LSB DAC Output voltage generation

V_x is the output voltage of the LSB DAC for code C_{LSB} if the reference voltages were 0 and 1V. This voltage can be written as:

$$V_x(C_{LSB}) = \frac{C_{LSB}}{2^{nLSB}} + e_L(C_{LSB}) \tag{2.11}$$

where $e_L(C_{LSB})$ is the deviation from the ideal output voltage, or in other words, it is the INL in volts if the reference voltage of the LSB DAC were 1V. Similarly, the output voltage of the MSB DAC with reference voltage V_{ref} can be written as:

$$V_M(C_{MSB}) = C_{MSB} \times \left(\frac{V_{ref}}{2^{nMSB}} \right) + e_M(C_{MSB}) \tag{2.12}$$

Then, with the voltages $V_M(C_{MSB})$ and $V_M(C_{MSB}+1)$ from the MSB DAC being used as references for the LSB DAC, the output voltage V_{out} can be written as:

$$V_{out} = V_M(C_{MSB}) + [V_M(C_{MSB}+1) - V_M(C_{MSB})] \times V_x \tag{2.13}$$

V_{out} can also be written in terms of the INL of the whole segmented DAC as:

$$V_{out} = C_{MSB}\left(\frac{V_{ref}}{2^{nMSB}}\right) + C_{LSB}\left(\frac{V_{ref}}{2^{n}}\right) + INL(C) \tag{2.14}$$

Substituting equations (2.11), (2.12), and (2.14) in equation (2.13), we get:

$$\begin{aligned} & C_{MSB}\left(\frac{V_{ref}}{2^{nMSB}}\right) + C_{LSB}\left(\frac{V_{ref}}{2^{n}}\right) + INL(C) = \\ & \quad C_{MSB} \times \left(\frac{V_{ref}}{2^{nMSB}}\right) + e_M(C_{MSB}) \\ & + \left[\begin{array}{l} (C_{MSB}+1) \times \left(\frac{V_{ref}}{2^{nMSB}}\right) + e_M(C_{MSB}+1) \\ -C_{MSB} \times \left(\frac{V_{ref}}{2^{nMSB}}\right) - e_M(C_{MSB}) \end{array}\right] \times \left(\frac{C_{LSB}}{2^{nLSB}} + e_L(C_{LSB})\right) \end{aligned} \tag{2.15}$$

After rearranging and cancellation of terms, we finally get:

$$\begin{aligned} INL(C) = {} & e_M(C_{MSB}) + \left(\frac{V_{ref}}{2^{nMSB}}\right) e_L(C_{LSB}) \\ & + [e_M(C_{MSB}+1) - e_M(C_{MSB})] \times \frac{C_{LSB}}{2^{nLSB}} \\ & + [e_M(C_{MSB}+1) - e_M(C_{MSB})] \times e_L(C_{LSB}) \end{aligned} \tag{2.16}$$

Since this INL is in units of volts, we can divide the whole equation by $V_{ref} / 2^{n}$ where $n = nMSB + nLSB$, to get the INL in units of LSBs. Next, if we define $e_M'(C_{MSB}) = e_M(C_{MSB}) / (V_{ref} / 2^{n})$ and $e_L'(C_{LSB}) = e_L(C_{LSB}) \times 2^{nLSB}$, then we can finally write the model for INL as:

$$\begin{aligned} INL(C) \text{ in LSBs} = {} & e_M'(C_{MSB}) + e_L'(C_{LSB}) \\ & + [e_M'(C_{MSB}+1) - e_M'(C_{MSB})] \times \frac{C_{LSB}}{2^{nLSB}} \\ & + err \end{aligned} \tag{2.17}$$

where $e_M'(C_{MSB})$ and $e_L'(C_{LSB})$ are representative of the nonlinearities due to the MSB DAC and LSB DAC respectively, in units of DAC LSBs. The error term $err = [e_M'(C_{MSB}+1) - e_M'(C_{MSB})] \times e_L'(C_{LSB}) / 2^{nLSB}$ can be ignored since it is a multiplication of two

nonlinearities and will be small. The coefficient of $e_M'(C_{MSB})$ is $\left(1-C_{LSB}/2^{nLSB}\right)$ whereas the coefficient of $e_M'(C_{MSB}+1)$ is $\left(C_{LSB}/2^{nLSB}\right)$. This means that unlike in normal segmented DACs, where the INLs for codes in an MSB segment are dependent on one MSB code nonlinearity, the INLs for codes in an MSB segment for interpolated DACs are dependent on the weighted average of the 2 adjacent MSB code nonlinearities.

Now that we finally have a segmented non-parametric model of the INL for interpolated DACs, we can follow exactly the same procedure as in uSMILE – we can calculate the preliminary INLs P and then write equation (2.17) for each DAC code, get the coefficient matrix H, put the equations in matrix form, and estimate the vector $e' = [e_M' \quad e_L']^T$ using least squares. The final estimated INLs F can be calculated as $H_{inv} \times e'$. It should be noted that although we used the ideal value of $1LSB = (V_{ref}/2^n)$ for going from equation (2.16) to equation (2.17), the units of the estimated INLs will be whatever units the preliminary INLs are in (actual LSBs). The scaling factor will just be absorbed into the unknowns vector e'.

2.4.2 Extrapolated Reconstruction with Interpolation (ER+I)

Even for segmented DACs with interpolation, it is possible to extrapolate and reconstruct the output voltages for all codes by measuring the voltages at the same specific codes mentioned in Section II C. We will just have to tweak the equation that we use for reconstructing the output voltages. We will again consider a two level nMSB-nLSB segmentation for simplicity. We first measure the output voltages at the MSB points, and call them $v_M(j) = v(j,0)$ where j varies from 0 to $2^{nMSB}-1$. We then measure all the voltages in one MSB segment: $v_L(j) = v(m,j)$ where j varies from 0 to $2^{nLSB}-1$. Now, to reconstruct the output at any code, we can simply re-use

equation (2.13). But for that, we require $V_x(j)$ which are the output voltages of the LSB DAC when the reference voltages are 0 and 1V. Following logic similar to what is described in Section II C, but additionally accounting for interpolation, we can get $V_x(j)$ by subtracting $v_M(m)$ from $v_L(j)$ and dividing by $v_M(m+1)-v_M(m)$. Hence, all the output voltages can be reconstructed using the following equation:

$$\hat{V}(C) = v_M(C_{MSB}) + \left(v_M(C_{MSB}+1)-v_M(C_{MSB})\right)\times\left(\frac{v_L(C_{LSB})-v_M(m)}{v_M(m+1)-v_M(m)}\right) \tag{2.18}$$

To calculate the INLs, these can be subtracted from the end-pint/best fit line. One interesting property of interpolated DACs is that if the MSB point voltage deviations from the ideal output voltage are not extremely high, because of the interpolation, the DNLs from one LSB segment can be approximated to repeat every 2^{nLSB} codes, and so, we only need to calculate the DNLs for one LSB segment. In fact, since the interpolation prevents any sudden jumps when going from one MSB segment to the next, the consequent low DNL values are one of the motivations to pursue an interpolated DAC topology. It should be noted that all the discussions about the time saving factor for ER are valid for ER+I too.

2.5. Simulation Results for uISMILE and ER+I

We will verify that the uISMILE and ER+I methods give accurate INL and DNL estimations for interpolated DACs. uISMILE and ER+I were used to estimate the INLs of the same interpolated DAC that was generated in Section III, with the additive noise set to 0.5LSBs. As before, for ER and ER+I, 256 measurements were taken per code, whereas for uSMILE and uISMILE, 2 measurements per code were taken. Figure 2.7 shows the estimated INL plots using all 4 methods, compared with the true INL.

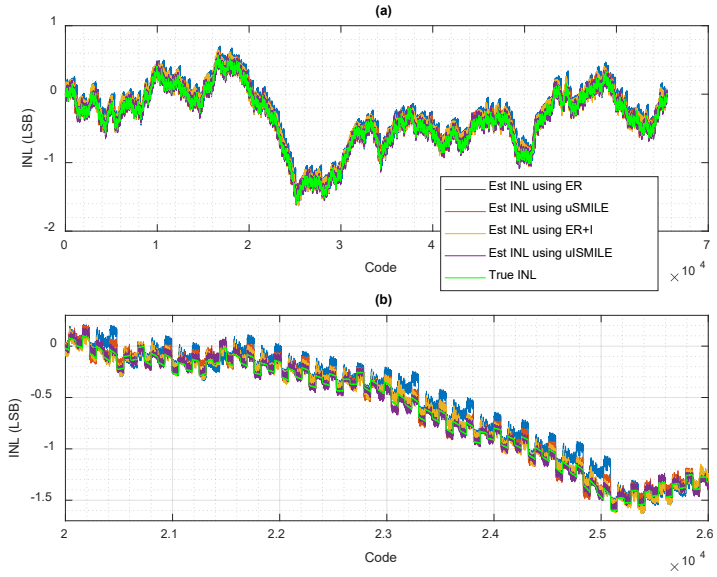


Figure 2.7. INL simulation results of a 16b interpolated DAC
(a) INL estimations using the different methods
(b) Zoomed in view of the INL plots

Figure 2.8 shows the DNL plots.

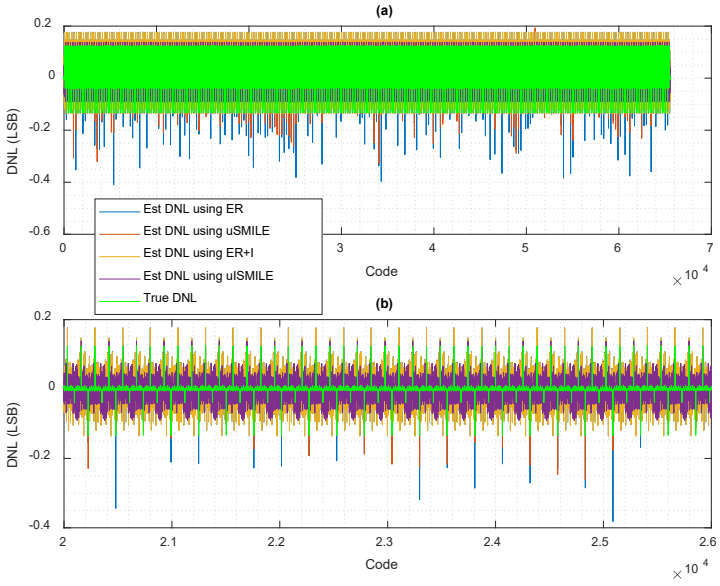


Figure 2.8. DNL simulation results of a 16b interpolated DAC
(a) DNL estimations of the 16b DAC using the different methods
(b) Zoomed in view of the DNL plots

The estimation errors are summarized in Table 2.1. We can clearly see the jumps at the MSB codes in the DNLs estimated by the algorithms which do not account for interpolation i.e. uSMILE and ER, whereas uISMILE and ER+I give good estimations for both the INL and DNL.

Table 2.1 Simulation results of a 16b interpolated DAC

(a) INL estimation errors in LSBs

	Max(abs(INL_est-INL_true))	(INLmax_est-INLmax_true)	(INLmin_est-INLmin_true)
ER	0.36	0.131	-0.027
ER+I	0.19	0.078	-0.026
uSMILE	0.25	0.039	0.030
uISMILE	0.12	0.034	-0.006

(b) DNL estimation errors in LSBs

	Max(abs(DNL_est-DNL_true))	(DNLmax_est-DNLmax_true)	(DNLmin_est-DNLmin_true)
ER	0.29	0.051	-0.27
ER+I	0.14	0.052	-0.002
uSMILE	0.33	0.068	-0.19
uISMILE	0.09	0.014	0.038

To verify the robustness of the ER+I and uISMILE methods, 100 16b DACs were randomly generated. Figure 2.9 shows a scatter plot with the true absolute maximum INLs/DNLs on the X-axis and the estimated absolute INLs/DNLs on the Y-axis. There is excellent correlation between the true and estimated INLs and DNLs for both the methods.

2.6. Silicon Measurement Results

The uSMILE algorithm has been shown to work in simulations in the previous sections, and in practice for ADCs in previous literature. The focus here will be to show the effectiveness of the uISMILE algorithm and the Extrapolated Reconstruction with Interpolation (ER+I) method with measurement results on a production mixed-signal IC. This IC has a 12b DAC and a higher resolution ADC available on-chip. The 12b DAC has an m-n segmentation, with interpolation

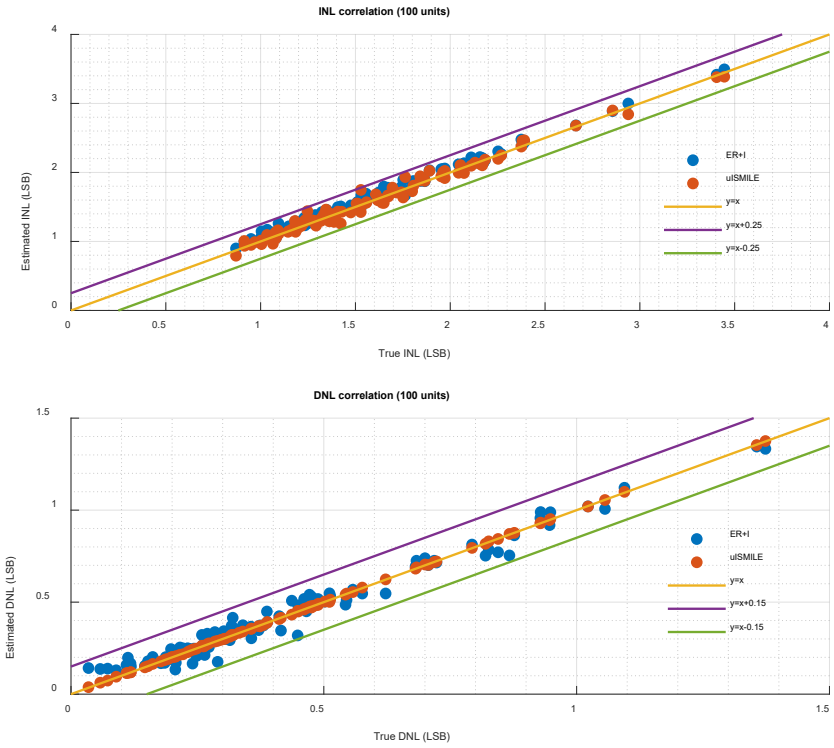


Figure 2.9. Simulation results – Robustness test for 16b interpolated DACs
(a) INL correlation using uISMILE and ER+I
(b) DNL correlation using uISMILE and ER+I

between the output voltages of an m-bit MSB DAC. The ADC was used to measure the output of the DAC.

Since the linearity of this DAC is specified for a sub-range of codes, the end-point fit INLs/DNLs were calculated and compared in this range. For uSMILE/uISMILE, the DAC input code was swept from min code to max code. From a simple calculation like the one presented in previous sections, it is found that the time saving factor $tsf \approx 20$ i.e. if one were to take one measurement per code, the equivalent hits per code for uSMILE/uISMILE would be ~20. Hence, 12 measurements were taken per code to make the effective number of hits per code 240. For the ER/ER+I methods, the outputs for only the MSB codes, and one LSB segment were measured, with 240 hits per code to make it comparable to uISMILE. The LSB segment was chosen at mid code (2048). For the conventional INL, the DAC input code was swept from min code to max code

with 240 measurements taken per code. The data acquisition time for all the fast linearity methods is 20x less than the conventional method. To ensure that the impact of noise is minimal, the conventionally measured INL was averaged across 10 runs to get "reference" INL with an effective hits per code of 2400.

The INL and DNL per code were estimated using the four different proposed methods (ER, ER+I, uSMILE, UISMILE) on three different units of the 12-bit DAC. The complete reference and estimated DAC INLs per code for DAC-1 are shown in Figure 2.10 (a).

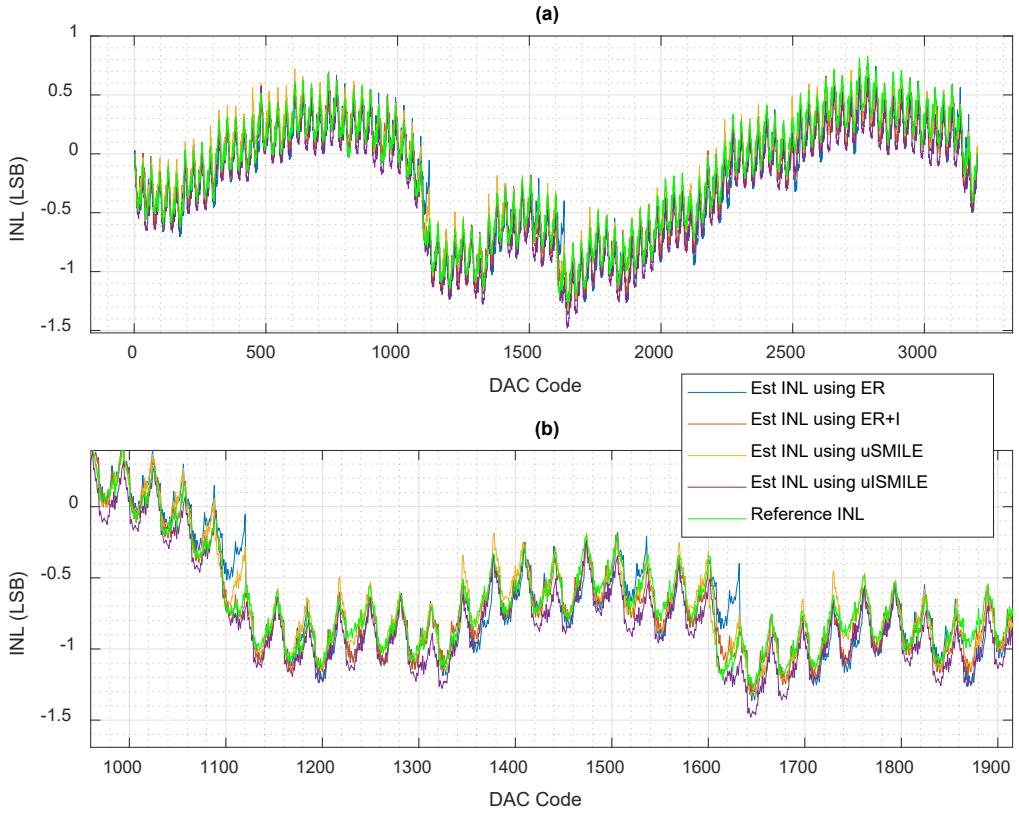


Figure 2.10. INL Measurement results for DAC-1
(a) INL estimations using the different methods
(b) Zoomed in view of the INL plots

Visually, all the estimation methods seem to track the INL well. Figure 2.10 (b) shows a zoomed in view of the plots over specific codes, where the INL estimation using ER (without interpolation) deviates from the reference INL in one LSB segment. The maximum absolute

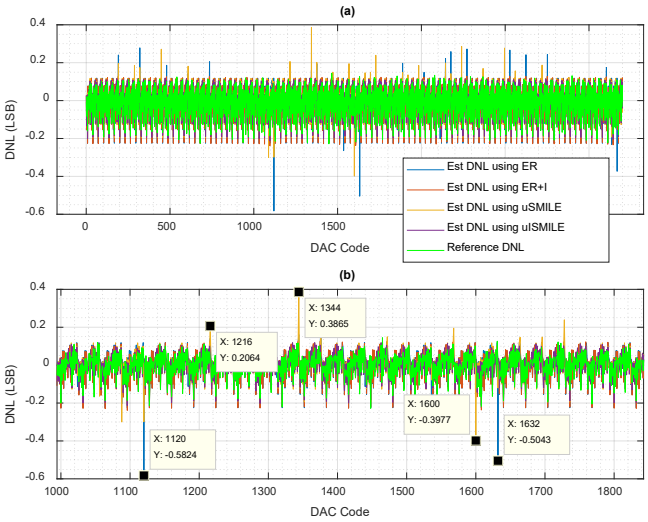


Figure 2.11. DNL Measurement results for DAC-1
(a) DNL estimations using the different methods
(b) Zoomed in view of the DNL plots

estimation error across codes, the estimation error for INLmax, and the estimation error for INLmin have been summarized in Table 2.2 (a). The INL estimation accuracies look slightly better for ER+I and uISMILE. There is a more significant difference in the DNL estimation curves shown in Figure 2.11.

Similar to what was noticed in the simulation results, for the algorithms which do not account for interpolation (ER and uSMILE), high DNL estimation errors can be seen at the MSB points. The summary for DNL estimation in Table 2.2 (b) also clearly shows that ER+I and uISMILE give significantly more accurate results for DNL.

Table 2.2 Measurement results of DAC-1

(a) INL estimation errors in LSBs

	Max(abs(INL_est-INL_true))	(INLmax_est-INLmax_true)	(INLmin_est-INLmin_true)
ER	0.52	-0.056	-0.092
ER+I	0.3	-0.06	-0.047
uSMILE	0.37	-0.068	-0.065
uISMILE	0.34	-0.172	-0.109

(b) DNL estimation errors in LSBs

	Max(abs(DNL_est-DNL_true))	(DNLmax_est-DNLmax_true)	(DNLmin_est-DNLmin_true)
ER	0.6	0.133	-0.354
ER+I	0.16	-0.016	-0.01
uSMILE	0.42	0.242	-0.17
uISMILE	0.11	-0.025	0.016

Figure 2.12 and Figure 2.13, which show the INL and DNL estimations for DAC-2 and DAC-3 respectively, further validate this point.

Since the specifications for DNL (commonly +/-1 LSB) are far more stringent than INL specifications, ER+I and uISMILE should be used for linearity testing of interpolated DACs.

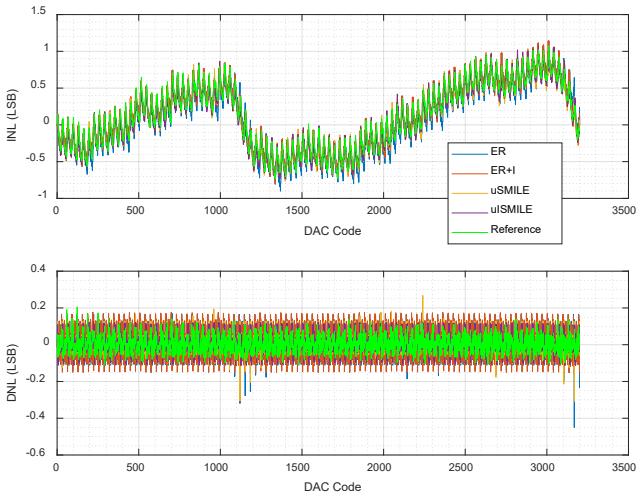


Figure 2.12. INL and DNL measurement results for DAC-2

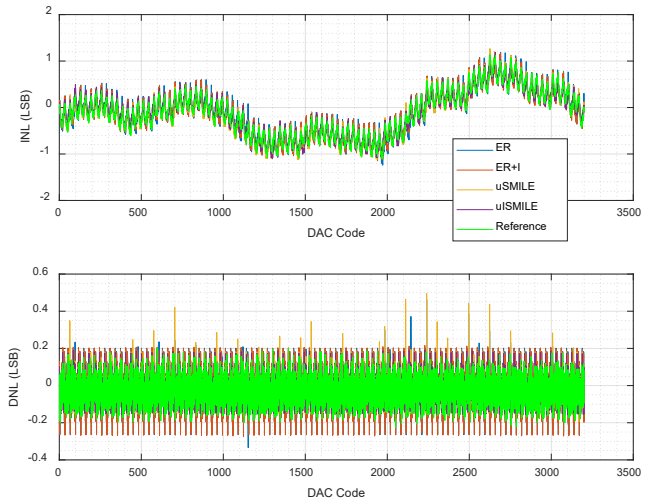


Figure 2.13. INL and DNL measurement results for DAC-3

Theoretically, the test time reduction as compared to the conventional method with 240 hits per code should be 20x. In practice, the reduction in total test time, which comprises data acquisition + on-chip calculation was ~19.8x for ER+I and ~16x for uISMILE. The reason for this is discussed in the next section. For higher resolution DACs, the test time reduction will be even more significant.

2.7. Discussion

The uSMILE, uISMILE, ER and ER+I methods altogether provide 4 different ways to significantly reduce the linearity test time for DACs, depending on the architecture. There are pros and cons for both classes of methods: uSMILE/uISMILE and ER/ER+I. In uSMILE/uISMILE, since we need to take measurements at all the codes, we have to wait for the output of the DAC to settle before capturing the data for every code. Conversely, since ER/ER+I involve taking measurements at fewer codes with multiple measurements per code, we do not have to wait for the output to settle when taking multiple measurements at the same code. It should be noted, though, that the settling time will be slightly lower for uSMILE/uISMILE because we are only stepping up one code at a time.

Another advantage that ER/ER+I has over uSMILE/uISMILE is that the data processing is relatively very simple, and so, will be faster, and more memory efficient. In fact, if one is interested in knowing just the maximum and minimum INL/DNL, then it is not even necessary to store INLs/DNLs at all codes. The voltage output measurements at only the mentioned specific few codes need to be stored, and the outputs at all the other codes can be reconstructed and the INL calculated, one code at a time, while keeping track of just the max and min INL/DNL values. If the uSMILE/uISMILE methods are implemented as presented in this chapter, they will take up a very long time and a lot of memory compared to ER/ER+I, because huge matrix inversions are involved. But, as mentioned earlier, there is a significant advantage that DAC uSMILE linearity testing offers over ADC uSMILE linearity testing, which we can utilize to reduce the time and memory requirements. This stems from the fact that, unlike for ADCs, we know exactly which code we are sending before we even capture the output data. Basically, the H matrix, which is dependent on the output codes for an ADC, is solely dependent on the input codes for a DAC, which are already known to us. Hence, the least squares solution matrix H_{inv} can be pre-computed and stored, which drastically reduces computational test time and memory requirements. In fact, if one observes the H_{inv} matrix carefully, one will find that there are patterns in it which repeat, and thus, an extremely low number of distinct pre-computed values need to be stored for H_{inv} in memory. Although the net test time of uSMILE will usually be higher than for ER, if the H_{inv} matrix is pre-computed and stored, then this increase in test time is not very significant. If we have an ADC on-chip to measure the DAC output, it paves the way for a complete BIST solution.

The uSMILE/uISMILE methods are not without their advantages over the ER/ER+I methods. One significant advantage is that since we are actually measuring the output for each and every code in uSMILE/uISMILE, they have the potential to catch sparkle INLs/DNLs if they

happen to occur. This can be done by performing a simple sanity check on the preliminary noisy INLs. In ER/ER+I, since we only measure the output at a few specific codes, and not all codes, they will not be able to catch these kinds of errors. Another significant advantage of uSMILE/uISMILE is that they can be combined with the ROME method [15] in order to facilitate measurement of DAC nonlinearities with low-linearity digitizers in addition to reducing the number of samples to be taken. Thirdly, the INL model used in uSMILE/uISMILE can easily be modified to account for nonlinearities from sources other than resistor/capacitor mismatches. Therefore, both methods have their advantages and disadvantages and can be applied depending on the requirements. Both ER/ER+I and uSMILE/uISMILE have the potential to be implemented on bench tests, for faster processing on the tester, or even on-chip during probe or final production test. In fact, due to the large time saving factor, we can take more measurements and increase the accuracy of our estimations.

2.8. Conclusion

Novel linearity testing methods that significantly reduce test time have been discussed for DACs with various architectures. The uSMILE algorithm and the Extrapolated Reconstruction (ER) method, which are based on a segmented model, have been shown to be very effective in reducing the linearity test time for different segmented DAC architectures. Shortcomings of the segmented model have been discussed for DACs with interpolation, and new methods that account for interpolation (uISMILE and ER+I) have been developed. Extensive simulation results of many 16-bit DACs and measurement results for multiple 12-bit DACs show that all the proposed methods are effective in significantly reducing time for linearity testing by reducing the number of samples that need to be measured by >100x and 20x respectively, while giving accurate estimations of the INL and DNL errors.

2.9. References

[1] M. Burns, G. W. Roberts, and F. Taenzler, *An introduction to mixed-signal IC test and measurement*, vol. 2001. Oxford University Press New York, 2001.

[2] "IEEE Standard for Terminology and Test Methods of Digital-to-Analog Converter Devices," *IEEE Std 1658-2011*, pp. 1–126, Feb. 2012, doi: 10.1109/IEEESTD.2012.6152113.

[3] Y. Zhuang, B. Magstadt, T. Chen, and D. Chen, "High-Purity Sine Wave Generation Using Nonlinear DAC With Predistortion Based on Low-Cost Accurate DAC-ADC Co-Testing," *IEEE Transactions on Instrumentation and Measurement*, vol. PP, no. 99, pp. 1–9, 2017, doi: 10.1109/TIM.2017.2769238.

[4] L. Jin, K. Parthasarathy, T. Kuyel, D. Chen, and R. L. Geiger, "Accurate testing of analog-to-digital converters using low linearity signals with stimulus error identification and removal," *IEEE Transactions on Instrumentation and Measurement*, vol. 54, no. 3, pp. 1188–1199, Jun. 2005, doi: 10.1109/TIM.2005.847240.

[5] H. Xing, D. Chen, and R. Geiger, "On-chip at-speed linearity testing of high-resolution high-speed DACs using DDEM ADCs with dithering," in *2008 IEEE International Conference on Electro/Information Technology*, May 2008, pp. 117–122.

[6] X. L. Huang and J. L. Huang, "ADC/DAC Loopback Linearity Testing by DAC Output Offsetting and Scaling," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 19, no. 10, pp. 1765–1774, Oct. 2011, doi: 10.1109/TVLSI.2010.2063443.

[7] H.-W. Ting, S.-J. Chang, and S.-L. Huang, "A Design of Linearity Built-in Self-Test for Current-Steering DAC," *J Electron Test*, vol. 27, no. 1, pp. 85–94, Feb. 2011, doi: 10.1007/s10836-010-5187-2.

[8] K. Arabi, B. Kaminska, and M. Sawan, "On chip testing data converters using static parameters," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 6, no. 3, pp. 409–419, Sep. 1998, doi: 10.1109/92.711312.

[9] I. H. S. Hassan, K. Arabi, and B. Kaminska, "Testing digital to analog converters based on oscillation-test strategy using sigma-delta modulation," in *Proceedings International Conference on Computer Design. VLSI in Computers and Processors (Cat. No.98CB36273)*, Oct. 1998, pp. 40–46, doi: 10.1109/ICCD.1998.727021.

[10] K. P. S. Rafeeque and V. Vasudevan, "A built-in-self-test scheme for digital to analog converters," in *17th International Conference on VLSI Design. Proceedings.*, 2004, pp. 1027–1032, doi: 10.1109/ICVD.2004.1261065.

[11] J.-L. Huang, C.-K. Ong, and K.-T. Cheng, "A BIST scheme for on-chip ADC and DAC testing," in *Proceedings Design, Automation and Test in Europe Conference and Exhibition 2000 (Cat. No. PR00537)*, 2000, pp. 216–220, doi: 10.1109/DATE.2000.840041.

[12] Z. Yu and D. Chen, "Algorithm for dramatically improved efficiency in ADC linearity test," in *2012 IEEE International Test Conference*, Nov. 2012, pp. 1–10, doi: 10.1109/TEST.2012.6401561.

[13] T. Chen and D. Chen, "Ultrafast stimulus error removal algorithm for ADC linearity test," in *2015 IEEE 33rd VLSI Test Symposium (VTS)*, Apr. 2015, pp. 1–5, doi: 10.1109/VTS.2015.7116249.

[14] X. Jin *et al.*, "An on-chip ADC BIST solution and the BIST enabled calibration scheme," in *2017 IEEE International Test Conference (ITC)*, Oct. 2017, pp. 1–10, doi: 10.1109/TEST.2017.8242032.

[15] S. K. Chaganti, T. Chen, Y. Zhuang, and D. Chen, "Low-cost and accurate DAC linearity test with ultrafast segmented model identification of linearity errors and removal of measurement errors (uSMILE-ROME)," in *2018 IEEE International Instrumentation and Measurement Technology Conference (I2MTC)*, May 2018, pp. 1–6, doi: 10.1109/I2MTC.2018.8409877.

CHAPTER 3. ANALYSIS OF SEGMENTED MODEL ALGORITHMS FOR DAC LINEARITY TEST TIME REDUCTION

Shravan K. Chaganti*, Kushagra Bhatheja*, Abalhassan Sheikh†, and Degang Chen*

*Iowa State University, Ames, USA

†Texas Instruments Inc.

Modified from a manuscript to be submitted to IEEE Transactions on Instrumentation and Measurement

Abstract

High resolution data converters pose a test challenge in terms of long test times and equipment costs. For modern SOCs with large number of data converters, test costs associated with them might begin to dominate the overall system costs. Reduction of data converter test cots thus becomes imperative. This chapter focuses on the time reduction of linearity testing of DACs. The test time reduction is attributed to the fact that the number of variables required to estimate the non-linearity at a given code is much less than the number of codes. Two algorithms, uSMILE and uISMILE have been discussed, which employ a segmented model for the DAC's nonlinearity.

Theoretical foundations have been laid down for the segmented model for the DAC INL along with the boundary conditions required to solve for unknowns using the least squares method. A fast and memory efficient implementation is also discussed, which avoids the least square solution altogether. Additionally, rigorous noise analysis has been performed for both uSMILE and uISMILE methods. Measurement results have been presented to validate the methods. A linearity test time reduction of 15x-20x was seen in actual silicon measurement results for production 12-bit DACs at TI and >300x was seen for a16-bit DAC.

3.1 Introduction

Digital to analog converters (DACs) form an indispensable part of most modern SoCs. With IoT becoming ubiquitous, the number of DACs embedded into a SOC are bound to see an exponential increase. Testing of high resolution DACs poses problems, including but not limited to observability of the DAC outputs and long test times.

DAC testing can broadly be categorized into linearity testing and spectral testing. While linearity testing involves estimation of INL/DNL of the DAC, spectral testing involves estimating characteristics such as THD, SFDR, etc [1]. This chapter focuses on the linearity testing of a DAC.

Conventional static testing for DACs employs an accurate voltmeter that records the voltage being produced by the DAC in response to the input code. In order to minimize the effect of noise, multiple readings are taken per code. The increasing resolution of the DACs implies that more codes need to be tested. This translates to long test times. Further, [2] states that the measuring device should be at least 2 bits more accurate than the device under test. The long test times in conjunction with requirements for a highly accurate measurement device results in increased test costs.

Test cost reduction for DACs can be visualized as a two-pronged approach. One aspect of the cost is the long test times. The other aspect is the requirement of utilizing a very accurate measurement device.

Several authors have attempted to come up with solutions addressing the concern of increasing test times for DACs. In [3], the author proposed a method of selecting a subset of test vectors such that a 100% test coverage is ensured, thereby bringing down the test time. [4] Modeled DACs based on wavelet theory. This was used in [5], which employed wavelet transform to bring down the test time. BIST solutions have also been proposed as a means to reduced test time. In [6] an on-chip DAC testing circuit was proposed that employed a comparator in order to compare

against the expected voltages. [7] improved upon the idea by generating an accurate linear ramp using a delta-sigma modulator and then comparing it with the dac output voltages. In [8] a circuit with deterministic dynamic element matching (DDEM) ADC and a dithering DAC was employed to test the DAC. In [9], a loopback system was employed in which the ADC was tested followed by testing of the DAC using the ADC. The required stimulus accuracy for ADC testing is obtained by scaling down the DAC output voltage and adding an offset (in order to ensure that the entire ADC range is covered). Conversely, while testing the DAC, the DAC output voltage is scaled up. [10] used the SEIR [11] algorithm to test the ADC and then used the ADC's estimated INL to estimate the DAC INL.

This chapter addresses the problem of long test time by reducing the number of variables that need to be estimated in order to obtain the INL/DNL of the DAC. In contrast to conventional testing, the current method hypothesizes that the INL at each code can be expressed as a summation of a few variables and these variables are much less than the number of DAC codes. This concept was first introduced for ADCs in [12]. The concept was extended to DACs in [13] which is described in the previous chapter. In [13], the segmented model was also extended to account for interpolated DAC architectures using the uISMILE algorithm. Although the basics of the segmented model were discussed, certain parameters were not rigorously defined. Also, some unnecessary restrictions were placed on the data collection and least squares solution matrix formation. These are removed in this chapter. For the sake of completeness, there is a rigorous description of the segmented model, as well as boundary conditions which help define the terms more definitively.

The remainder of the chapter is organized as follows. Section 0 reviews the basic uSMILE method for DACs along with some rigorous definitions and boundary conditions required for least

squares estimation. Gaussian noise analysis is performed in section 0 to derive an accurate time savings factor compared to the conventional method. A time and memory efficient implementation of the algorithm is presented in section 0. Section 0 discusses the application of uSMILE under different conditions like an interpolated architecture (uISMILE), and presence of un-modeled errors. Section 0 demonstrates the effectiveness of uSMILE and uISMILE with measurement results including a 16-bit R2R DAC and volume production data of many 12-bit interpolated DACs.

3.2 Basic uSMILE for DACs

In this section, the basic concepts of the uSMILE method for DACs are reviewed and discussed. The modeling of DAC linearity errors with segmented errors is reviewed first. Then, the algorithm for estimating the segmented errors and thus the INLs is briefly discussed.

3.2.1 Modelling of DAC linearity errors

Traditional full code linearity testing of DACs involves taking multiple measurements of the output voltage at every DAC code in order to average out the noise. Since the linearity errors at every code are assumed independent of each other, many hits per code are required to accurately estimate the INLs/DNLs at every code. As has been discussed in [13], this results in very large test times, especially for high resolution DACs, owing to the number of codes increasing exponentially with the number of bits, as well as slower possible DAC code change rates at higher resolutions.

For high resolution DACs, to avoid exponential growth in the number of components, and thus die area, segmented architectures are often used. Mismatches and nonlinearities of a limited number of these components majorly determine the nonlinearity of the input-output transfer characteristics of the DAC. The segmented INL model of the uSMILE method discussed in [13] leverages this fact and utilizes the architecture of the DAC to drastically reduce the number of variables or unknowns to be estimated. Note that binary current steering, R2R DACs etc., are

inherently segmented bit-by-bit and thus the term "segmented DACs" is used as an umbrella term for these as well.

When we say that an n bit DAC is segmented as $nM - nI - nL$, it means that the first nM bits of the n bit DAC code form the MSB (Major Significant Bits) code k_M, the next nI bits form the ISB (Intermediate Significant Bits) code k_I and the last nL bits form the LSB (Intermediate Significant Bits) code. uSMILE models the INL at any DAC code k as the sum of three segment errors:

$$INL(k) = E_{\mathrm{M}}(k_{\mathrm{M}}) + E_I(k_I) + E_{\mathrm{L}}(k_{\mathrm{L}}) \tag{3.1}$$

where $E_M(k_M)$ is the MSB segment error at MSB code k_M, $E_I(k_I)$ is the ISB segment error at ISB code k_I, and $E_L(k_L)$ is the LSB segment error at LSB code k_L. The number of unknowns to be estimated has been reduced to $2^{nM} + 2^{nI} + 2^{nL}$ as opposed to 2^n for conventional full code INL testing.

Now, there are two options for setting "boundary conditions" in order to definitively peg all the values for the E_{M}'s, E_I's and E_L's. This will be better explained using an example. Figure 3.1 (a/d) shows a typical INL plot of a 12-bit binary current steering DAC. Let's say that we have segmented this as 4-4-4. The vertical lines divide the INL curve into 16 equally sized MSB segments. Every MSB segment is further broken into 16 ISB segments which will repeat for each MSB segment. Figure 3.1 (b/e) shows a zoomed in view of the 1st MSB segment which in turn is broken into 16 ISB segments. For every MSB segment k_M, there are 16 $E_I(k_I)$ errors for each 4-bit ISB code k_I. Similarly, each ISB segment is further divided into 16 LSB errors $E_L(k_L)$ for each 4-bit LSB code k_L, which are shown in Figure 3.1 (c/f).

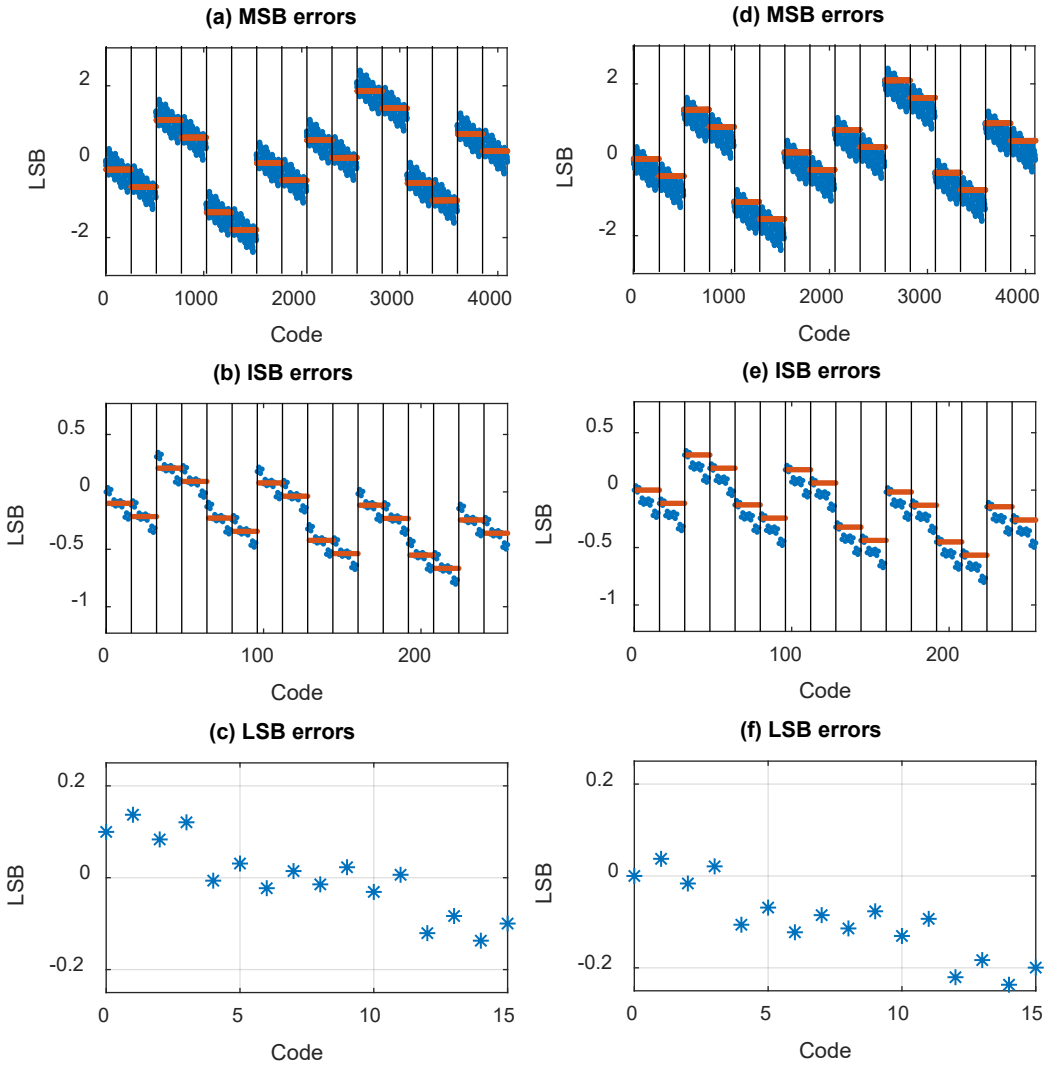


Figure 3.1. DAC INL segments. Option 1 (left) and Option2 (right)

The first option is to set the average of INL errors in the k_M'th MSB segment as $E_M(k_M)$. Then, after removal of the E_M 's, we set $E_I(k_I)$ as the average of the residual errors in the k_I'th ISB segment. These give us the following two boundary conditions for option 1:

$$avg(E_I) = \frac{\sum_{k_I=0}^{2^{nI}-1} E_I(k_I)}{2^{nI}} = 0$$
$$avg(E_L) = \frac{\sum_{k_L=0}^{2^{nL}-1} E_L(k_L)}{2^{nL}} = 0 \quad (3.2)$$

The MSB, ISB and LSB errors using option 1 are shown in the left side plots of Figure 3.1.

The second option is to set $E_M(k_M)$ as the first INL in the $k_M{}'th$ MSB segment. Then, after removal of the E_M 's, we set $E_I(k_I)$ as the first residual error within the $k_I{}'th$ ISB segment. These give us the following two boundary conditions for option 2:

$$E_I(0) = 0$$
$$E_L(0) = 0 \quad (3.3)$$

The MSB, ISB and LSB errors using option 2 are shown in the right side plots of Figure 3.1.

3.2.2 Least Squares matrix solution for uSMILE

Let's say that for an n bit DAC, we have "noisy" output voltage measurements for a set of M codes. The segmented model actually allows us to estimate the INLs at all codes even when we do not have measurements at all codes. Let's call this set of M codes as S_M . We assume that we have an accurate measurement device such that measurement errors, apart from noise, are negligible. Here, M could be equal to $N = 2^n$, which would mean that we have output voltage measurements for all the DAC codes, or it could be less than N , which means we have voltage measurements for a sub-set of all possible DAC codes. From these output voltages, we can calculate our preliminary noisy INLs at these M codes by subtracting these voltages from either an end-point fit or best fit line. Now, for each code $k \in S_M$, the following equation can be written:

$$P(i) = INL(k(i)) + m(i) \quad i = 0,1,2,...,M-1 \quad (3.4)$$

where $P(i)$ is the preliminary noisy INL estimation, $INL(k(i))$ is the accurate INL, and $m(i)$ is the measurement error at code $k(i)$. If we assume a segmented model for the INL, we can substitute (3.1) in (3.4) to get

$$P(i) = E_{\mathrm{M}}\left(k_{\mathrm{M}}(i)\right) + E_I(k_I(i)) + E_{\mathrm{L}}\left(k_{\mathrm{L}}(i)\right) + m(i) \tag{3.5}$$

We have M such equations, and $K = 2^{nM} + 2^{nI} + 2^{nL}$ variables or unknowns to be estimated. We write these equations in matrix form as:

$$P = H \times E + m = H \begin{bmatrix} E_M \\ E_I \\ E_L \end{bmatrix} + m \tag{3.6}$$

where P is column vector of length M with the $i'th$ element being $P(k(i))$, H is a matrix of size $M \times K$ and E is the column vector of segment errors that need to be estimated, of size $K \times 1$. Each row of H has all 0's except three 1's in three columns corresponding to the MSB, ISB and LSB code of the code k:

$$H(i, k_M(i)) = H(i, 2^{nM} + k_I(i)) = H(i, 2^{nM} + 2^{nI} + k_L(i)) = 1 \tag{3.7}$$

Typically, M>>K, so this is an over-determined system of equations. The least squares method can be used to estimate the values of the unknowns. The least squares solution is given by:

$$\hat{E} = (H^T H)^{-1} H^T \times P \tag{3.8}$$

The method of least squares will naturally average out gaussian noise. For now, we assume that Gaussian noise is the dominant source of measurement error. We will discuss other unmodelled errors and flicker noise effects in section 0. Also note that 2 other equations need to be included in H, which come from the boundary conditions, so either (3.2) or (3.3). If these equations are not included, the matrix will be rank deficient and might be unsolvable, because many combinations of segment errors can satisfy the measurements.

Once the segment errors have been estimated, we construct the H_1 coefficient matrix which is an $N \times K$ matrix with columns populated according to an all codes ramp from 0 to $2^n - 1$, similar to the H matrix, with:

$$\begin{aligned} 1 &= H(k, k_M) \\ &= H(k, 2^{nM} + k_I) \qquad k = 0,1,...,2^n - 1 \\ &= H(k, 2^{nM} + 2^{nI} + k_L) \end{aligned} \tag{3.9}$$

We then estimate the INL's at all codes by multiplying $\hat{E}$ by H_1 :

$$\hat{INL} = H_1 \times \hat{E} \tag{3.10}$$

In the conventional method, $N = 2^n$ number of unknowns need to be estimated, whereas for uSMILE, only $K(<< N)$ number of unknowns need to be estimated. Intuition dictates that the uSMILE method will require fewer number of total measurements by a factor of N / K. Let's call this value as the intuitive time saving factor:

$$tsf_i = \frac{N}{K} = \frac{2^n}{2^{nM} + 2^{nI} + 2^{nL}} \tag{3.11}$$

We will do a rigorous noise analysis in the next section to calculate an accurate value for the time savings factor and then compare with this intuitive value.

3.3 Noise Analysis and Time Savings

In this section, we will do a rigorous analysis of the effect of additive Gaussian noise on the uSMILE algorithm. We will start from (3.8) and substitute P from (3.6) into it. We will also assume that the measurement error's primary source is Gaussian noise, so we will use n instead of m to denote the additive noise vector. This gives us:

$$\begin{aligned} \hat{E} &= (H^T H)^{-1} H^T \times (H \times E + n) \\ &= (H^T H)^{-1} (H^T H) \times E + (H^T H)^{-1} H^T n \\ &= E + (H^T H)^{-1} H^T n \end{aligned} \tag{3.12}$$

Multiply both sides of this equation by H_1 in order to get the estimation of the INLs at all the codes.

$$\hat{INL} = INL + H_1(H^T H)^{-1} H^T n \tag{3.13}$$

The estimation error due to noise can then be written as

$$e = \hat{INL} - INL = H_1(H^T H)^{-1} H^T n \tag{3.14}$$

To analyze the relation between the INL estimation error and the noise variance, some matrix transformations need to be performed. Multiply both sides by their transpose:

$$ee^T = H_1(H^T H)^{-1} H^T nn^T H((H^T H)^{-1})^T H_1^T \tag{3.15}$$

Let's look at the LHS first. Once expanded, it can be observed that the diagonal terms are the square of the estimation error at each code.

$$ee^T = \begin{bmatrix} e(0)^2 & \cdots & e(0)\cdot e\left(2^n-1\right) \\ \vdots & \ddots & \vdots \\ e(0)\cdot e\left(2^n-1\right) & \cdots & e\left(2^n-1\right)^2 \end{bmatrix} \tag{3.16}$$

Next, the noise vector multiplied by its transpose becomes:

$$nn^T = \begin{bmatrix} n(1)^2 & \cdots & n(1)\cdot n(M) \\ \vdots & \ddots & \vdots \\ n(1)\cdot n(M) & \cdots & n(M)^2 \end{bmatrix} \tag{3.17}$$

Assuming that the noise terms are random with 0 mean and independent of each other, we can calculate the expected value of the matrix. If the variance of the thermal noise is σ_n^2 , then $n \sim N(0, \sigma_n^2 / h1)$ where $h1$ is the number of hits/measurements per DAC code.

$$E[nn^T] = \begin{bmatrix} E[n(1)^2] & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & E[n(M)^2] \end{bmatrix} = \frac{\sigma_n^2}{h1} I_M \tag{3.18}$$

Where I_M is an identity matrix of size $M \times M$. We calculate the expected value of both sides of (3.15) and substitute (3.18) in it to obtain:

$$E[ee^T] = (\sigma_n^{\,2} / h1) \times E[H_1 (H^T H)^{-1} H_1^{\,T}] \tag{3.19}$$

The H_1 matrix is a constant and H is a constant for a given set of input DAC codes. The expected values thus become:

$$E\left\{\begin{bmatrix} e^2(0) \\ e^2(1) \\ \vdots \end{bmatrix}\right\} = \frac{\sigma_n^2}{h1} \cdot \mathrm{diag}\left\{H_1 (H^T H)^{-1} H_1^{\,T}\right\} \tag{3.20}$$

The H matrix is dependent upon the set of input DAC codes, the segmentation of the DAC's INL, and the architecture of the DAC. For simplicity, we assume that $H = H_1$ i.e. that measurements are taken at all codes of the DAC. This is the typical case, and only the segmentation is sufficient to determine H. We should still add the 2 equations from either (3.2) or (3.3) to H in order to get an accurate estimate.

Let's call the value of the diagonal elements of $H_1 (H_1^{\,T} H_1)^{-1} H_1^{\,T}$ as $d(k)$. Then, the variance of the estimation error at code k is equal to:

$$\sigma 1_e^{\,2}(k) = \frac{d(k)}{h1} \sigma_n^{\,2} \tag{3.21}$$

Where $h1$ is the number of hits/measurements per DAC code for uSMILE. For the conventional method, the variance of the estimation error would be $\sigma 2_e^{\,2}(k) = \frac{1}{h2} \sigma_n^{\,2}$ where $h2$ is the number of hits per code for the conventional method. For equivalent noise averaging capability, $\sigma 1_e = \sigma 2_e$,which gives us the accurate time savings factor as:

$$tsf_a = \frac{h2}{h1} = \frac{1}{d(k)} \tag{3.22}$$

Let's take the example of a 14-bit DAC with 5-5-4 segmentation. The value of d turns out to be 4.76×10^{-3} for all codes. So, $tsf_a = 1/d \approx 210$. For the same DAC, using (3.11),

$tsf_i = \frac{2^{14}}{2^5+2^5+2^4} \approx 205$ which is quite close to tsf_a! Similarly, for a 12-bit DAC with 5-4-3 segmentation, $tsf_a \approx 76$ and $tsf_i \approx 73$. As can be seen, (3.11) is a good approximation and can be used for quick calculations of the time saving factor for basic uSMILE.

3.4 Time and memory efficient computation

Computing the least squares using the large H matrix can be quite inefficient. Let us investigate if there is a quicker and more memory efficient method for estimating the segment errors. For simplicity, let's assume that we have noisy measurements for all the DAC codes, i.e. $M = N$. Thus, we have N equations similar to (3.5), one for each code, with k varying from 0 to $2^n - 1$.

Let's first sum up all the P's for which the MSB code $k_M = C_M$, say. Since there is one equation per code, we can calculate that there will be 2^{nIL} such equations, where $nIL = nI + nL$. For each k_I, the LSB code k_L varies from 0 to $2^{nl} - 1$, so each $E_I(k_I)$ will be hit 2^{nL} number of times. Similarly, each $E_L(k_L)$ will be hit 2^{nI} number of times. This gives us the following equation:

$$\sum_{k\in[k_M=C_M]} P(k) = 2^{nIL} E_M(C_M) + 2^{nL}\left(\sum_{k_I=0}^{2^{nI}-1} E_I(k_I)\right) + 2^{nI}\left(\sum_{k_L=0}^{2^{nL}-1} E_L(k_L)\right) + \sum m(k) \tag{3.23}$$

From the equations in (3.2), we know that the 2nd and 3rd terms are equal to 0. Therefore, the E_M's can be estimated using the following equation:

$$\hat{E}_M(C_M) = \frac{\sum_{k\in[k_M=C_M]} P(k)}{2^{nIL}} \tag{3.24}$$

Similarly to (3.23), we can add up all the P's for which ISB code $k_I = C_I$:

$$\sum_{k\in[k_I=C_I]} P(k) = 2^{nML}E_I(C_I) + 2^{nL}\left(\sum_{k_M=0}^{2^{nM}-1} E_M(k_M)\right) + 2^{nI}\left(\sum_{k_L=0}^{2^{nL}-1} E_L(k_L)\right) + \sum m(k) \tag{3.25}$$

We know that the 3rd term is 0 from (3.2). We can then divide the whole equation by 2^{nML} to get:

$$\hat{E}_I(C_I) = \frac{\sum_{k\in[k_I=C_I]} P(k)}{2^{nML}} - avg(\hat{E}_M) \tag{3.26}$$

Following the same methodology, we can derive that:

$$\hat{E}_L(C_L) = \frac{\sum_{k\in[k_L=C_L]} P(k)}{2^{nMI}} - avg(\hat{E}_M) \tag{3.27}$$

Even when we do not have the noisy INLs for all the codes, (3.24), (3.26) and (3.27) provide a good approximately accurate estimation. These equations involve simple averaging and can even be updated on the fly if required. Once the segment errors are estimated, the INLs at all codes can be estimated easily by adding up the segment errors. Thus, an extremely time and memory efficient implementation of uSMILE has been presented which can be easily implemented either in software or hardware on-chip.

3.5 Application of the segmented model under various conditions

In this section, the application of uSMILE under different conditions is discussed. First, the modification of the basic uSMILE for interpolated DACs is discussed. Lastly, the presence of unmodeled errors on the segmented model has been discussed, along with how the segmented model compares to the widely used major carrier test for DAC testing.

3.5.1 Interpolated DAC architecture

The simple segmented model does not apply for DACs with interpolated architectures. In fact, if the simple segmented model is applied to an interpolated DAC, there will be high INL and DNL estimation errors, especially at the MSB code transitions [13]. Hence, a new interpolated segmented model was developed in [13]. Assuming that interpolation is between the MSB DAC and the ISB+LSB DAC, the following model for the INL of an interpolated DAC was derived:

$$\begin{aligned} INL(k) = {} & E_M(k_M) + E_I(k_I) + E_L(k_L) \\ & + [E_M(k_M + 1) - E_M(k_M)] \times \frac{k_{IL}}{2^{nL}} \end{aligned} \tag{3.28}$$

Where k_{IL} is the code of the ISB+LSB DAC. The extra terms, compared to (3.1) come in due to the interpolation effect. It should be noted that according to the derivation of this interpolated model, (3.3) must be used as boundary conditions. Additionally, one might wonder what the value of $E_M(C_M + 1)$ is for the last MSB segment (when $C_M = 2^{nM} - 1$). The model was derived assuming end-point fit INLs, hence, for the last MSB segment, $E_M(2^{nM}) = 1$.

Now, let us look at the effect of noise and the time savings factor for uISMILE. All the derivations leading to (3.20) and (3.21) and (3.22) still hold true, with the difference being that the values in H_1 and H matrices will be set according to the coefficients from (3.28). Let's take an example of an interpolated 14-bit DAC with 6-4-4 segmentation with interpolation between the voltages of the 6-bit MSB DAC. In this case, the value of d varies with code.

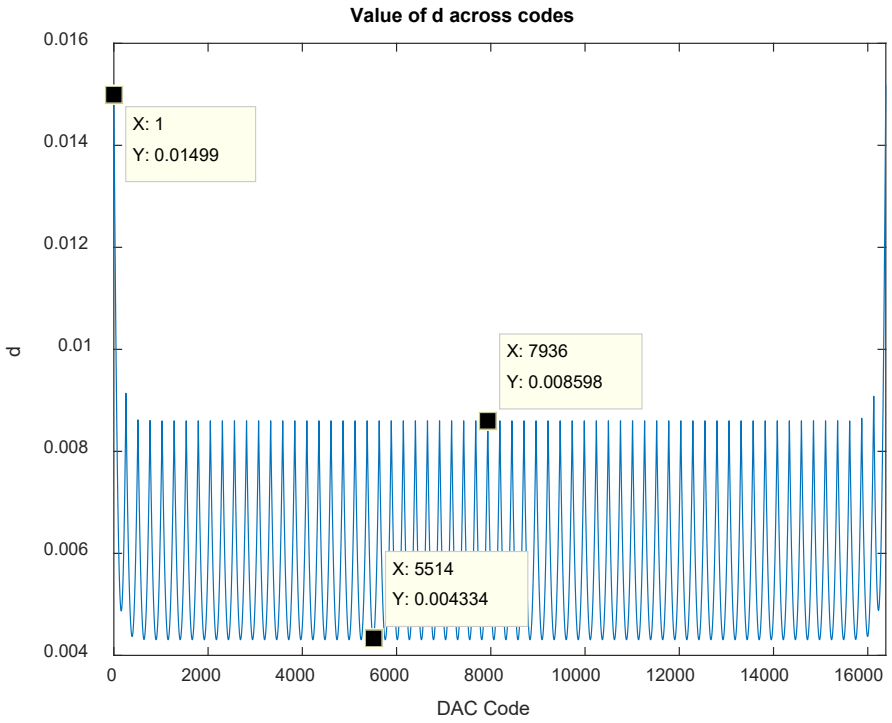


Figure 3.2. Value of d across codes for a 14-bit interpolated DAC with 6-4-4 segmentation

Figure 3.2 shows a plot of d across DAC codes. The mean value of $d = 5.8\times10^{-3}$. This gives us $tsf_a \approx 172$ which agrees with $tsf_i \approx 171$. If we instead take the worst case value of $d = 14.99\times10^{-3}$, then $tsf_a \approx 66$. If we ignore the first and last MSB segments, then the worst case value of $d = 8.6\times10^{-3}$ and $tsf_a \approx 116$. In any case, for the uISMILE algorithm, it is worth calculating tsf_a since it can be significantly lower than the quick calculation using tsf_i.

3.5.2 Effect of unmodeled errors

In the INL model, the segment errors for the lower bits is assumed to be identical in each MSB segment. However, this might not always hold true. The INL of the DAC itself can be broken

into two components, the linear component ($INL_{k,l}$) and the non-linear component ($INL_{k,nl}$). The INL at code k can be written as:

$$INL_k = INL_{k,l} + INL_{k,nl} \tag{3.29}$$

The linear component arises from resistor/capacitor/transistor mismatches and results in a segmented shape for the INL. The non-linear component arises typically due to voltage coefficients and DAC buffers. Often, the output of the DAC is buffered, and so, the buffer's nonlinearity, especially near supply rails, contributes to the DAC's overall INL curve. This nonlinearity typically has a smooth non-discontinuous shape. Though this is not explicitly modeled by the segmented model, the average value of this smooth nonlinearity within an MSB segment gets absorbed into the E_M's, if option 1 from Section 3.2.1 is followed. The deviation from the average nonlinearity is not captured and becomes an error term. Similar to the analysis performed in [14], the un-captured error is a function of the slope within an MSB segment. The maximum possible absolute unmodeled error can be derived as $\max\left\{\text{abs}\left[f_{\text{nl}}'(x)\right]\right\} \times V_{MSB}/2$ for option 1 and $\max\left\{\text{abs}\left[f_{\text{nl}}'(x)\right]\right\} \times V_{MSB}$ for option 2, where $f_{nl}(x)$ is the nonlinearity modeled as a function of the output voltage x and the unit is DAC LSB, and V_{MSB} is the voltage range of 1 MSB segment.

For a 16-bit DAC with 7-5-4 segmentation, there are 128 MSB segments and each segment corresponds to a voltage range of 1/128 (output range normalized to 1). Suppose the nonlinear function $f_{nl}(x) = 40x(x-1)^2$, then the maximum $INL_{k,nl}$ is around 6LSBs. The maximum slope is 40, hence the maximum error in uSMILE is $40 \times 1/128 \approx 0.31 LSB$ for option 1 and $0.63 LSB$ for option 2. This is an extreme case for the amount of smooth nonlinearity, but it is useful to demonstrate the error in estimation due to uSMILE compared to the very commonly used major carrier test (MCT) for DAC linearity [15], in which measurements are taken at only n major code transitions for an

n-bit DAC. Matlab simulations were run for a 16-bit R2R DAC with resistor mismatches to illustrate the effect of smooth nonlinearity. The best-fit true INL has been compared to the uSMILE estimated INL and MCT estimated INL in Figure 3.3 (a), and the respective estimation errors are plotted in Figure 3.3 (b) and in Figure 3.3 (c).

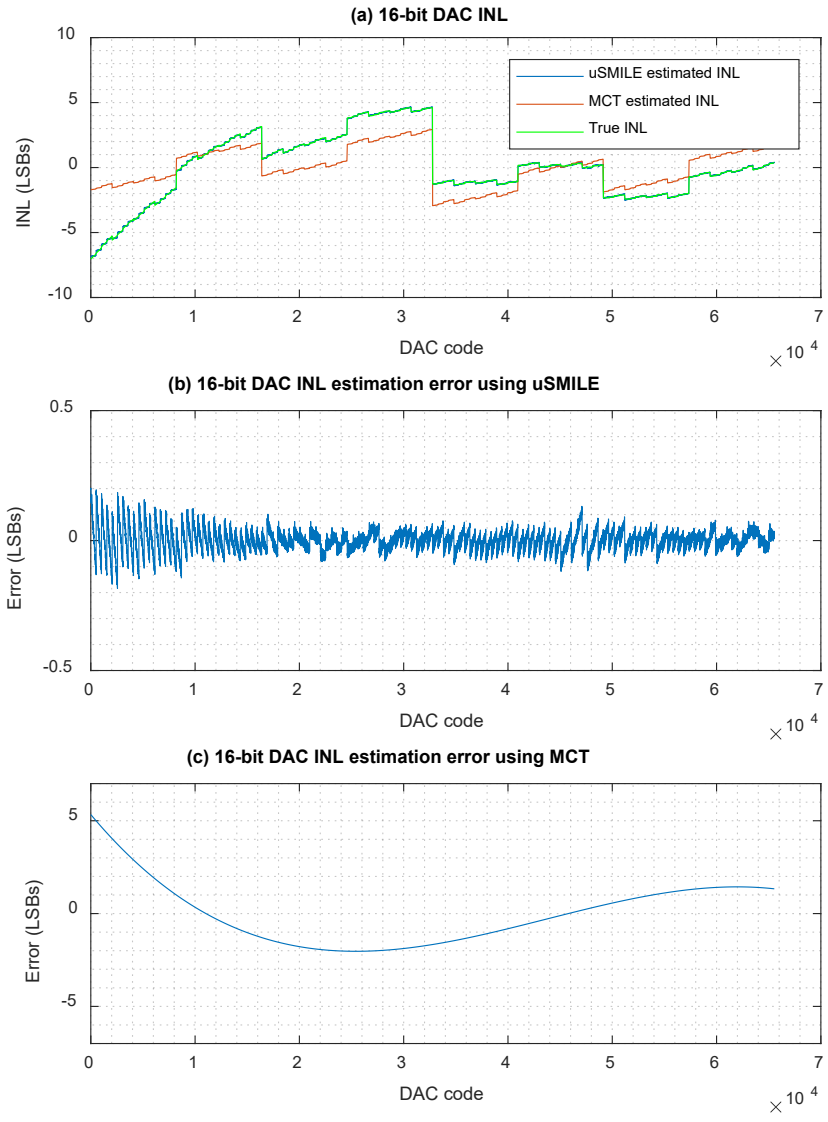


Figure 3.3. Simulation results of a 16-bit R-2R DAC with smooth nonlinearity
(a) INL estimations (b) INL estimation error using uSMILE
(c) INL estimation error using major carrier test

It can clearly be seen that MCT is unable to capture the DAC's smooth nonlinearity well and has significant estimation error, whereas uSMILE INL overlaps and agrees excellently with the true INL. If the number of MSB segments is smaller, then the error in uSMILE due to the smooth nonlinearity will be higher, but it will lead to better noise averaging. Proper segmentation can be chosen considering this tradeoff.

3.6 Measurement Results

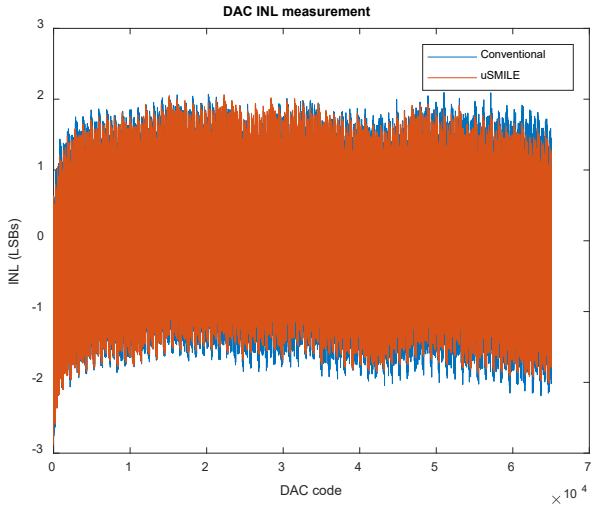
The effectiveness of the uSMILE and uISMILE algorithms are demonstrated by measurement results. First, the linearity of a 16-bit R2R DAC (TI x0508) was tested. The DAC was controlled using an Altera DE2 – 115 FPGA board. The output voltages of the DAC were measured using Audio Precision 2722 analog analyzer. Figure 3.4 shows the bench test setup which was used to take the measurements.

Figure 3.4. Bench test setup for 16-bit R2R linearity measurement

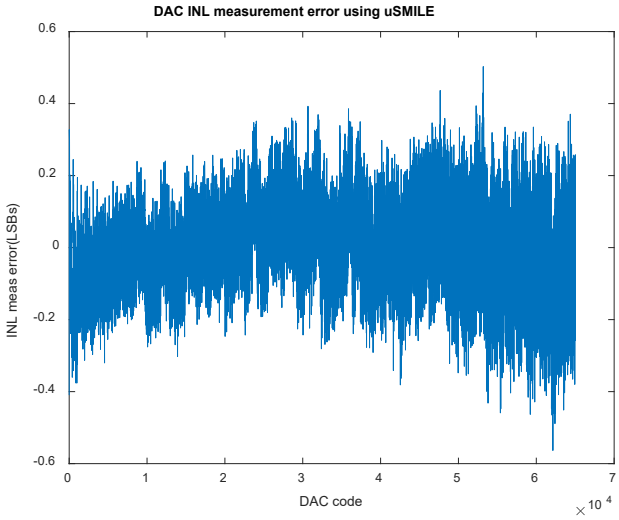
First, we measured the noise in the system by fixing the DAC code and measuring the spread of the output voltage. This gave us a noise rms value of 0.6LSB. Next, to measure the full code INL using the conventional method, 330 samples per DAC code were taken. To reduce the effect of flicker noise, the input code sequence was in the form of 330 up/down ramps with 1 measurement per code per ramp. The best fit INL was then calculated from these output voltage measurements. Next, the data of one of the ramps was used for the uSMILE algorithm with 7-5-4 segmentation. This gives us a value of $tsf_i \approx 372$. The estimated INLs using these 2 methods have been plotted in Figure 3.5(a).

This DAC actually has a buffered output. The nonlinearity of the buffer is clearly visible for the first few codes. Figure 3.5(b) shows the code-by-code INL estimation error of the uSMILE algorithm when compared with the conventional method. The estimation error band is mostly contained to within +/-0.5LSB. The uSMILE algorithm is doing a pretty good job at capturing the smooth nonlinearity which is likely caused by the output buffer. These errors are absorbed into the MSB segment errors. Increasing the number of MSB segments would probably have helped even further, but it would have come at the cost of worse noise averaging. Hence, 7-5-4 seems to be the optimal segmentation.

The uSIMILE algorithm was also implemented to measure the INL/DNL of a 12-bit interpolated DAC on a TI production chip. Figure 3.6 shows the reference and uISMILE estimated INLs for one such DAC, as well as the estimation error against the reference INL.



(a)



(b)

Figure 3.5. (a) INL measurements of a 16-bit R-2R DAC

(b) INL estimation error compared to conventional

The $tsf_i = 20$, so the uISMILE method used 20x less samples compared to the conventional while achieving good accuracy. Figure 3.7 shows the correlation between the maximum absolute INLs estimated using uISMILE against the reference for ~30,000 DACs in a production test

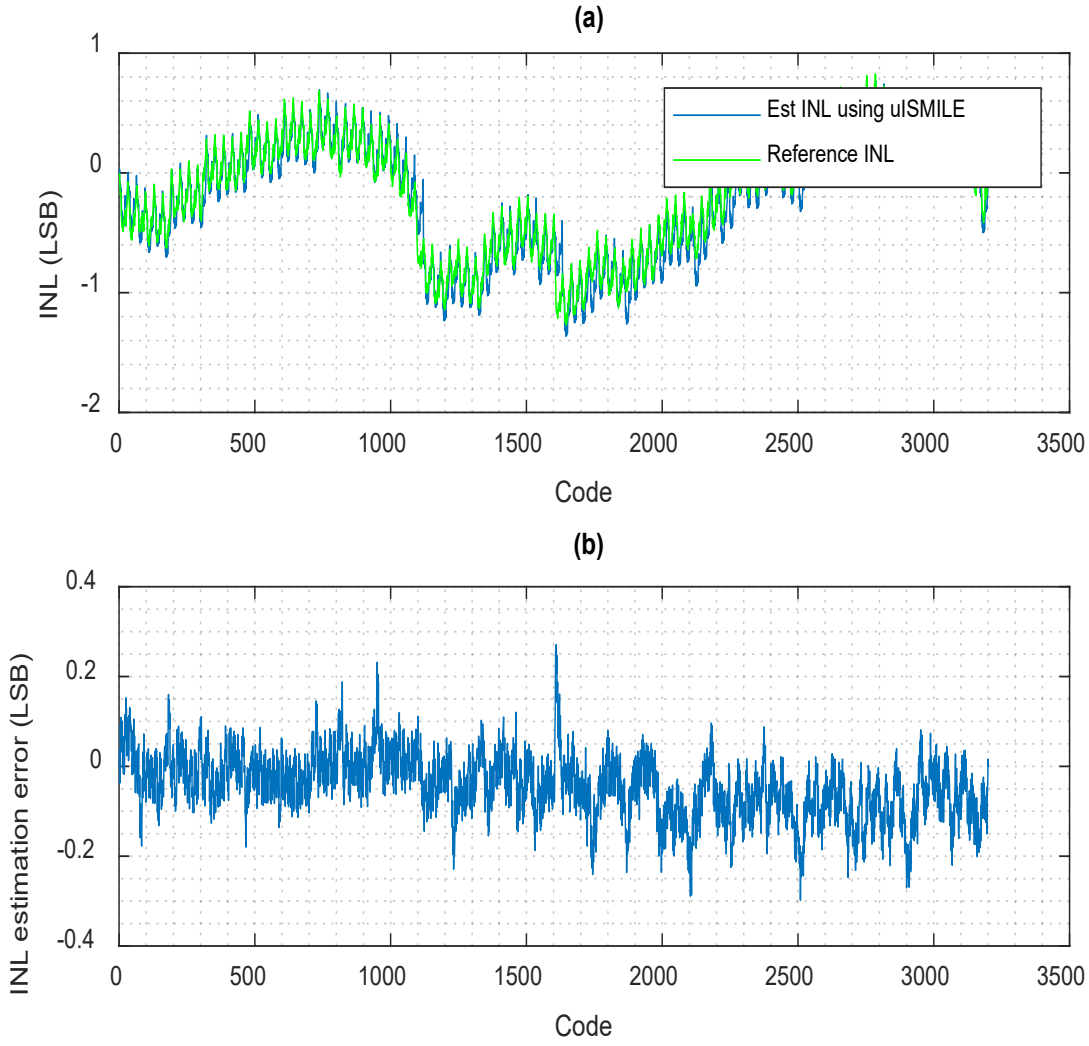


Figure 3.6. INL measurements for a 12-bit interpolated DAC

environment. Ideally, all the points should lie along the y=x line. We can see that the estimation errors are well within the tolerance band.

3.7 References

[1] W. Kester and D. Sheingold, “TESTING DATA CONVERTERS,” .

[2] “IEEE Standard for Terminology and Test Methods of Digital-to-Analog Converter Devices,” *IEEE Std 1658-2011*, pp. 1–126, Feb. 2012, doi: 10.1109/IEEESTD.2012.6152113.

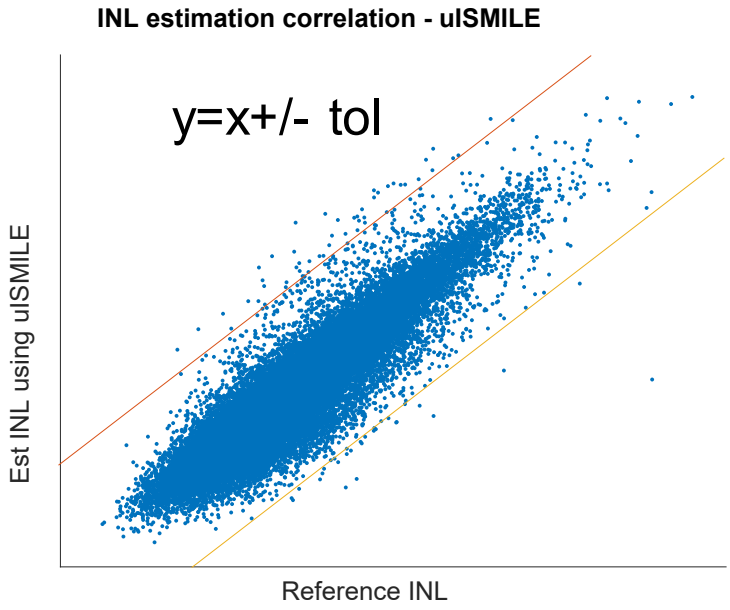


Figure 3.7. Production test results for a 12-bit interpolated DAC

[3] P. P. Fasang, "An optimal method for testing digital to analog converters," in *Proceedings. Tenth Annual IEEE International ASIC Conference and Exhibit (Cat. No.97TH8334)*, Sep. 1997, pp. 42–46, doi: 10.1109/ASIC.1997.616975.

[4] J. T. Doyle, Young Jun Lee, and Yong-Bin Kim, "An accurate DAC modeling technique based on wavelet theory," in *Proceedings of the IEEE 2003 Custom Integrated Circuits Conference, 2003.*, Sep. 2003, pp. 257–260, doi: 10.1109/CICC.2003.1249399.

[5] E. Awada, "A REDUCED-CODE LINEARITY TEST FOR DAC USING WAVELET ANALYSIS," *INTERNATIONAL JOURNAL OF ENGINEERING RESEARCH*, vol. 2, no. 1, p. 8, 2010.

[6] K. Arabi, B. Kaminska, and J. Rzeszut, "A built-in self-test approach for medium to high-resolution digital-to-analog converters," in *Proceedings of IEEE 3rd Asian Test Symposium (ATS)*, Nov. 1994, pp. 373–378, doi: 10.1109/ATS.1994.367203.

[7] Jiun-Lang Huang, Chee-Kian Ong, and Kwang-Ting Cheng, "A BIST scheme for on-chip ADC and DAC testing," in *Proceedings Design, Automation and Test in Europe Conference and Exhibition 2000 (Cat. No. PR00537)*, Mar. 2000, pp. 216–220, doi: 10.1109/DATE.2000.840041.

[8] Hanqing Xing, Degang Chen, and R. Geiger, "On-chip at-speed linearity testing of high-resolution high-speed DACs using DDEM ADCs with dithering," in *2008 IEEE International Conference on Electro/Information Technology*, May 2008, pp. 117–122, doi: 10.1109/EIT.2008.4554278.

[9] X. L. Huang and J. L. Huang, "ADC/DAC Loopback Linearity Testing by DAC Output Offsetting and Scaling," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 19, no. 10, pp. 1765–1774, Oct. 2011, doi: 10.1109/TVLSI.2010.2063443.

[10] Y. Zhuang, B. Magstadt, T. Chen, and D. Chen, "High-Purity Sine Wave Generation Using Nonlinear DAC With Predistortion Based on Low-Cost Accurate DAC–ADC Co-Testing," *IEEE Transactions on Instrumentation and Measurement*, vol. 67, no. 2, pp. 279–287, Feb. 2018, doi: 10.1109/TIM.2017.2769238.

[11] L. Jin, K. Parthasarathy, T. Kuyel, Degang Chen, and R. L. Geiger, "Accurate testing of analog-to-digital converters using low linearity signals with stimulus error identification and removal," *IEEE Transactions on Instrumentation and Measurement*, vol. 54, no. 3, pp. 1188–1199, Jun. 2005, doi: 10.1109/TIM.2005.847240.

[12] Z. Yu and D. Chen, "Algorithm for dramatically improved efficiency in ADC linearity test," in *2012 IEEE International Test Conference*, Anaheim, CA, USA, Nov. 2012, pp. 1–10, doi: 10.1109/TEST.2012.6401561.

[13] S. K. Chaganti, A. Sheikh, S. Dubey, F. Ankapong, N. Agarwal, and D. Chen, "Fast and accurate linearity test for DACs with various architectures using segmented models," in *2018 IEEE International Test Conference (ITC)*, Oct. 2018, pp. 1–10, doi: 10.1109/TEST.2018.8624753.

[14] T. Chen, X. Jin, R. L. Geiger, and D. Chen, "USER-SMILE: Ultrafast Stimulus Error Removal and Segmented Model Identification of Linearity Errors for ADC Built-in Self-Test," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 65, no. 7, pp. 2059–2069, Jul. 2018, doi: 10.1109/TCSI.2017.2775632.

[15] W. Kester and Analog Devices, inc, Eds., *Data conversion handbook*. Amsterdam ; Boston: Elsevier ; Newnes, 2005.

CHAPTER 4. USMILE-ROME: ULTRAFAST SEGMENTED MODEL IDENTIFICATION OF LINEARITY ERRORS AND REMOVAL OF MEASUREMENT ERRORS

Abstract

The Digital–to–Analog-Converter (DAC) is one of the fundamental components of Analog and Mixed-signal circuits. Static linearity testing of high resolution high performance DACs traditionally requires a long time and is very expensive. In this paper, a low-cost ultrafast method of testing DACs is presented. The method utilizes a low cost on-board measurement device for capturing the output of the DAC, instead of a precise digital voltmeter. By using a segmented non-parametric model for the DAC's INL curve and thus reducing the number of unknowns, the test time is also drastically reduced. Additionally, the linearity requirement on the measurement device is significantly relaxed by removing its non-linearity. The combination of these two methods results in drastic reduction in linearity test cost for DACs. Extensive analysis has also been done to evaluate the effect of noise on the algorithm as well as the amount of shift and the shift non-constancy.

4.1 Introduction

As one of the most fundamental blocks of analog and mixed signal (AMS) circuits, the digital-to-analog converter (DAC) is widely used in many areas such as audio, high definition television and cellular telephones. The DAC is usually deeply embedded in a system-on-chip (SoC). With the growth of the Internet-of-things, the DAC volume and performance requirements have improved significantly, while the test cost keeps increasing. Thus, there is an urgent need to develop low-cost methods for characterization and testing of DACs.

DAC testing includes the measurement of integral nonlinearity (INL) and differential nonlinearity (DNL), offset, gain error, spurious free dynamic range (SFDR), signal-to-noise ratio

(SNR), total harmonic distortion (THD) etc. [2], [3]. It is challenging to accurately test the DAC in a cost-effective way for various reasons. Traditional testing of DAC static linearity is done using a digital voltmeter (DVM) or a digital waveform recorder [3]. The equipment is required to have significantly better accuracy and resolution than the specifications of the DAC itself. Moreover, multiple samples are needed to average out the noise. The testing time is long and the test equipment is expensive.

In the past, many researchers have proposed methods to reduce the cost of DAC testing. The proposed method in [4] applied stimulus error identification and removal (SEIR) [5] to obtain the ADC linearity first and estimate DAC INL/DNL with the ADC. However, the accuracy of the DAC INL/DNL estimation is limited and the test time is long. In [6], the authors developed a circuit with deterministic dynamic element matching (DDEM) ADC and a dithering DAC to test the DAC. It is capable of testing a 14-bit DAC with ADC at 6-bit linearity. But it has to use the proposed ADC circuit and it also has the long test time problem. In [7], Huang, et al, improved the test accuracy with DAC scaling using local histogram test with DAC to test the ADC performance. Then the ADC is used to test the DAC with voltage scaling. It is architecture dependent and it takes long testing time. In [8], Ting, et al, tested the current-steering DAC by measuring the major transition current difference with a current-controlled oscillator and counter. This method is fast and low-cost but it is highly architecture-dependent.

In this paper, a new testing method and algorithm, named uSMILE-ROME, are introduced for accurate linearity testing of DACs with dramatically reduced test time and cost. This is done in two ways. Firstly, the algorithm exploits the fact that the number of truly independent error sources is much smaller than the number of DAC codes at which linearity has to be tested. This enables linearity testing with much fewer samples than is traditionally required, and so, saves on

test time. Secondly, the method proposes use of an on-board digitizer for measurement of the DAC output, instead of the traditionally used high accuracy digital voltmeter. This results in less test time per sample. This reduction in test time directly translates to reduction in test cost. Moreover, the algorithm relaxes the stringent linearity requirement on the measurement device by removing the errors introduced due to the nonlinearity of the device. This results in further cost savings. Rigorous analysis is performed for the various error sources for uSMILE-ROME and worst case bounds are calculated. These are verified by extensive simulation results. The method is also validated by INL measurement results of a 16-bit DAC using a 16-bit ADC and a 12-bit DAC using a 12-bit ADC.

4.2 Problem Statement

To test the linearity of a DAC, the digital stimulus needs to be generated and the analog response needs to be captured. In the conventional method, the DAC input code is swept from 0 to the maximum code, and the output voltage at each code is measured using a digital voltmeter (DVM). To average out the noise, multiple samples are taken for each DAC code. These voltages are compared to the ideal expected voltage for each code, and the INL and DNL are calculated.

As the resolution of the DAC grows, the number of input codes and therefore the number of INLs/DNLs to be estimated grows exponentially. For an n-bit DAC, if H samples per code are needed for noise averaging, then a total of $H \times 2^n$ output voltages will have to be measured. Measuring these $H \times 2^n$ voltages takes a long time, and so the test cost is also high. For a 16-bit DAC, with H=64, over 4 million samples would be required. If the DAC has a sampling rate of 500KSPS, the data acquisition time alone would be around 10 seconds. Multi-site testing will reduce this time, but this still corresponds to a significant test cost.

Conventional wisdom also dictates that the measuring device must be much more accurate and precise than the device under test. In the case of DAC testing, the digitizer in the DVM must be at least 10 times more linear than the DAC under test. If the INL of the DAC is at the +/-2 LSB level, then the digitizer's non-linearity must be in the range of +/-0.2 LSB. This stringent requirement on the linearity means that a highly accurate DVM is required, thus increasing the test cost even further.

4.3 The proposed method

4.3.1 Segmented model of DAC's INL

The conventional method essentially treats the INL/DNL error at each code as unrelated to each other, and so, the number of variables to be estimated is equal to the number of DAC codes. In reality, especially for high resolution DACs, the number of truly independent error sources is much smaller than the number of codes. For example, take a 16-bit R-2R DAC. The number of resistor mismatches is just $2\times16-1=31$ which is dramatically less than $2^{16}=65,536$. Although there will be many more error sources, it is true that the non-idealities (mismatches, voltage coefficients, etc.) of a limited number of analog components determines the errors in the input output transfer curve of the DAC. In other words, all the INL/DNL errors are highly correlated and are deterministic functions of a much smaller number of independent errors.

This correlated nature of the INL/DNL DAC errors makes a strong case for a model based approach to DAC linearity testing. Instead of basing the model on circuit laws, the proposed method takes a fundamentally different approach. It models the DAC's INL curve with a segmented non-parametric model. The idea is very similar to the uSMILE algorithm developed previously for ADCs [9], and is described in detail in the following paragraphs.

The INL curve of the DAC is broken into many MSB segments according to the MSB (Most Significant Bits) value of the DAC input code. Take a 16-bit DAC for example. If 6 bits are used as the MSB, then the INL curve is divided into 64 different segments. Each of these segments has an error term associated with it. Let's call this error as $E_M(C_{MSB})$. The error terms associated with the MSB segments will then be $E_M(0)$, $E_M(1)$, $E_M(2)$ ..... $E_M(63)$ corresponding to the MSB code. Each of these segments in turn can be further divided into smaller segments. Say the next 5 bits are used as ISB (Intermediate Significant Bits), then each MSB segment gets divided into 32 ISB segments, each of which has an error term associated with it, denoted as $E_I(C_{ISB})$. If we stop the segmentation here, the variations within each ISB segment away from the ISB average values are captured by the 32 LSB errors (5 LSB bits). The error term associated with each LSB code is denoted as $E_L(C_{LSB})$. The final INL value for code C will be:

$$INL(C) = E_M(C_{MSB}) + E_I(C_{ISB}) + E_L(C_{LSB}) \tag{4.1}$$

Most DAC architectures are inherently segmented in this fashion, like binary weighted, R-2R, mDAC etc., and so, this segmented non-parametric model can be applied to the INL curve. Note that this segmented model is not valid for string or thermometer-coded type architectures. For example, if you had a segmented 15-bit DAC implemented as a 7-bit thermometer coded resistor DAC and an 8-bit R-2R DAC, then the segmentation of the INL curve must be carefully chosen such that the MSB bits are greater than or equal to 7, since the thermometer coded part does not have a segmented architecture. For example, a 7-4-4 segmentation of the INL curve is valid, and so is an 8-3-4 segmentation, but a 6-5-4 segmentation is not valid.

This segmented model of the DAC's INL drastically reduces the number of variables to be estimated, thus enabling us to estimate the INL/DNL of the DAC at each code with a much-reduced number of samples.

4.3.2 Removal of error due to measurement device

As mentioned previously, conventional DAC outputs are captured with an accurate and precise digital voltmeter, which is costly and time-taking. In the proposed method, a non-linear on-board digitizer is used for this purpose. We say "non-linear" here because the error introduced due to the nonlinearity of the digitizer is completely removed by the method and algorithm proposed. This will be explained in detail later.

The basic idea of the Removal Of Measurement Error (ROME) method used here is similar to the USER-SMILE [10] method which has been proposed for accurate linearity testing of ADCs with non-linear input sources. For ADC static linearity testing, the ramp generator (stimulus) can be non-ideal and have non-linearities whereas the output is digital and thus assumed to have no error while being captured. For DAC testing, the input is ideal (sweep of digital code from all 0s to all 1s) but there can be errors in the measurement device, which can be non-linear. Hence, in USER-SMILE, the stimulus error is removed, whereas in ROME, the measurement error is removed. The details of the method are described below.

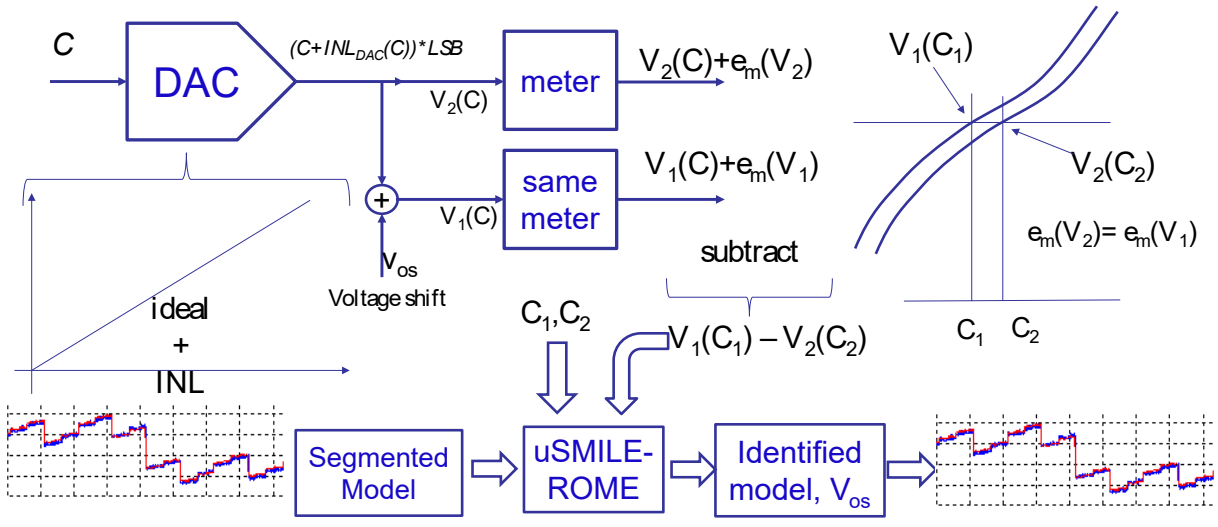


Figure 4.1. Test Setup and Overview of uSMILE-ROME

The test setup and overview is shown in Figure 4.1. The DAC output voltage is sent to an on-board digitizer/voltage meter whose linearity can be much less than the linearity of the DAC. This digitizer should have sufficiently small quantization errors, meaning it cannot have large "dead-zones", where a very large voltage range gives the same output code. This is an easy condition to guarantee even if the digitizer has bad linearity performance. It should have no, or sufficiently small kickback to the DAC output. For static linearity test, this should be very easy to meet, and means that the measurement device's transient settles faster than the DAC settling time. This is not an extra requirement. High performance measurement devices also need to satisfy this. But this might be easier to satisfy for low cost measurement devices since the accuracy requirement is more relaxed.

As an example, an ADC is used as the digitizer in this paper, but the method is not limited to using an ADC. The reference voltages for the DAC and the ADC are such that the output range of the DAC is less than the input range of the ADC. For each DAC code, two ADC output codes are obtained. The first ADC code is obtained when a positive voltage shift Vos or α is added to the output voltage of the DAC and then sent to the ADC for digitization. The second ADC code is obtained when the output voltage of the DAC is sent directly to the ADC. We assume that this shift α is constant for all the measurements. The value of this shift is not required to be accurate or known, but must be reasonable. There are several methods to generate this constant shift. Many of the constant shift generation methods developed for the SEIR algorithm [11], [12] can be used for this purpose. It is important to note that the ADC must not have wide codes or dead-zones which will result in many DAC codes giving the same ADC output code.

4.3.3 Derivation of equations

For a given DAC code $C1_{DAC}$, let's say that the output voltage after addition of shift is $V1$. This voltage can be expressed as:

$$V1 = \left(C1_{DAC} + INL_D\left(C1_{DAC}\right)\right)l_{DAC} + o_{DAC} + \alpha + w1_{DAC} \tag{4.2}$$

where INL_D is the INL function of the DAC, l_{DAC} is the actual LSB of the DAC, o_{DAC} is the offset of the DAC, α is the added voltage shift, and $w1_{DAC}$ is the additive noise. This same $V1$ becomes the input voltage to the ADC to give $C1_{ADC}$. The following equation can be written, with all variables corresponding to the ADC this time:

$$V1 = \left(C1_{ADC} + INL_A\left(C1_{ADC}\right)\right)l_{ADC} + o_{ADC} - w1_{ADC} - q1_{ADC} \tag{4.3}$$

Where $C1_{ADC}$ is the ADC output code, o_{ADC} is the offset of the ADC, $w1_{ADC}$ is the additive noise, and $q1_{ADC}$ is the quantization noise of the ADC.

Similarly, for any DAC code $C2_{DAC}$, let's say that the DAC output voltage, without shift, is $V2$, which can be expressed as:

$$V2 = \left(C2_{DAC} + INL_D\left(C2_{DAC}\right)\right)l_{DAC} + o_{DAC} + w2_{DAC} \tag{4.4}$$

This voltage, when sent directly to the ADC for measurement, gives ADC code $C2_{ADC}$. The voltage $V2$ can again be expressed as:

$$V2 = \left(C2_{ADC} + INL_A\left(C2_{ADC}\right)\right)l_{ADC} + o_{ADC} - w2_{ADC} - q2_{ADC} \tag{4.5}$$

Subtracting equations (4.2) and (4.4) and then equating that to the difference of equations (4.3) and (4.5), we get:

$$\begin{aligned}&\left[\left(C1_{DAC} - C2_{DAC}\right) + \left(INL_D\left(C1_{DAC}\right) - INL_D\left(C2_{DAC}\right)\right)\right]l_{DAC} + \alpha = \\ &\left[\left(C1_{ADC} - C2_{ADC}\right) + \left(INL_A\left(C1_{ADC}\right) - INL_A\left(C2_{ADC}\right)\right)\right]l_{ADC} \\ &+\left(w2_{DAC} - w1_{DAC} + w2_{ADC} - w1_{ADC} + q2_{ADC} - q1_{ADC}\right)\end{aligned} \tag{4.6}$$

Equation (4.6) can finally be re-arranged to get:

$$\begin{aligned} &INL_D\left(C1_{DAC}\right)-INL_D\left(C2_{DAC}\right)+\frac{\alpha}{l_{DAC}}= \\ &\frac{\left(C1_{ADC}-C2_{ADC}\right)}{g}-\left(C1_{DAC}-C2_{DAC}\right) \\ &+\frac{\left(INL_A\left(C1_{ADC}\right)-INL_A\left(C2_{ADC}\right)\right)}{g}+wq \end{aligned} \tag{4.7}$$

where $g = l_{DAC} / l_{ADC}$, and wq is the variable with all the additive and quantization noise terms. The reason for writing equation (4.7) in this manner will become apparent very soon. Now, if (i) the 2 ADC codes $C1_{ADC}$ and $C2_{ADC}$ are the same or close to each other, and (ii) they belong to the same "LSB segment" of the ADC, then the term $er = \left(INL_A\left(C1_{ADC}\right)-INL_A\left(C2_{ADC}\right)\right)/g$ will be either 0 or negligible. The underlying assumption in the above stated conditions is that the linearity within the LSB segment of an ADC is good, and so, the difference in INLs will be negligible compared to the noise. Or in other words, if the LSB segment of the ADC is linear, then the difference in input voltages to the ADC is approximately equal to the difference in ADC codes multiplied by the slope. The number of ADC codes in one ADC LSB segment can be set according to the architecture of the ADC or reasonable expectation of ADC INL. In order to guarantee that conditions (i) and (ii) are true, we need to carefully select from our (DAC input code, ADC output code) pairs for the shifted and non-shifted measurements. There is a strong possibility that there are jumps in INL from one ADC LSB segment to the other. Hence, if the codes are from different LSB segments, then the error term will be significant. This needs to be avoided.

When taking the measurements, we have two ADC output codes $k1_{ADC}, k2_{ADC}$ for each DAC input code k_{DAC}. We can create two matrices with rows of the form (DAC input code, ADC output code). One matrix for the shifted version ($A1$ with rows of the form $(k_{DAC}, k1_{ADC})$) and the

other matrix for the non-shifted version ($A2$ with rows of the form $(k_{DAC}, k2_{ADC})$). First, sort the rows of $A1$ in the increasing order of $k1_{ADC}$ codes. The whole row is sorted, so DAC codes will get rearranged too. Similarly, sort rows of $A2$ in the increasing order of $k2_{ADC}$ codes. Next, group together the rows in $A1$ in which the ADC codes belong to the same ADC LSB segment. Similarly, group together the rows in $A2$ which belong to the same ADC LSB segment. For those rows in $A1$ and the rows in $A2$ which belong to the same ADC LSB segment group, we can identify corresponding rows in which the ADC codes are almost the same or differ by a few codes. Some excess rows, and rows near the transition between LSB segments can be removed. Since the codes are already sorted, this task is easy to do. Figure 4.2 and Figure 4.3 give a clearer visual representation of the preceding operations and what happens to the ADC columns of the matrices.

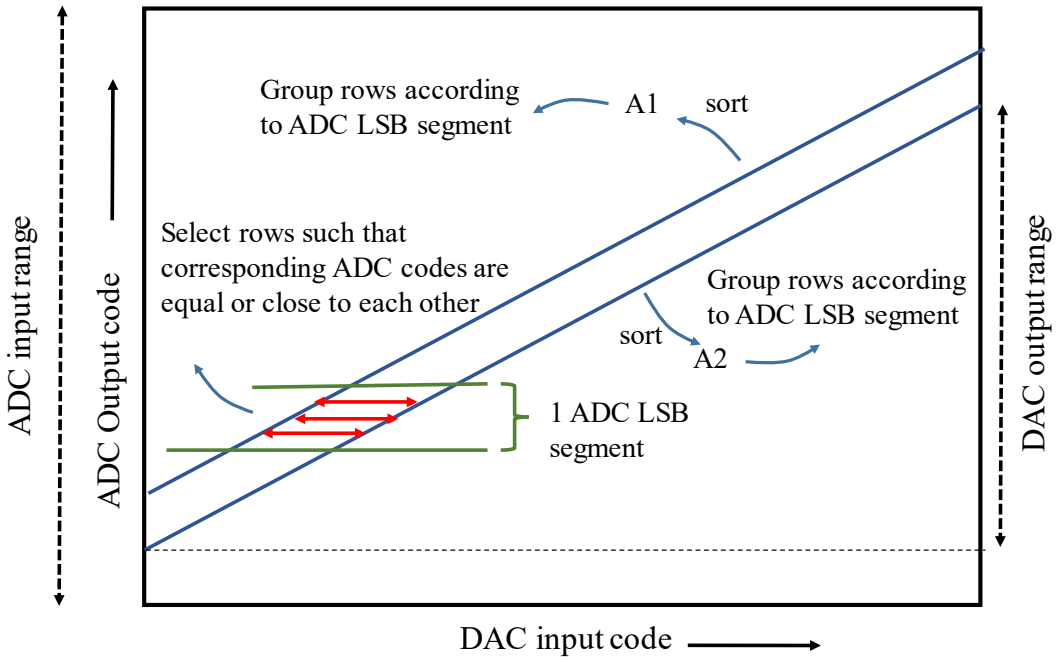
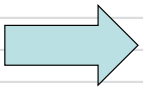


Figure 4.2. Visual representation of various operations performed

Input DAC Code	ADC o/p with shift	Input DAC Code	ADC o/p without shift
40	82	40	39
41	84	41	42
42	83	42	45
43	86	43	44
.	.	.	.
.	.	.	.
.	.	.	.
81	123	81	84
82	123	82	82
83	126	83	83
84	125	84	85
.	.	.	.
.	.	.	.
.	.	.	.



Input DAC Code	ADC o/p with shift	Input DAC Code	ADC o/p without shift
		40	39
		41	42
		43	44
		42	45
.	.	.	.
.	.	.	.
.	.	.	.
40	82	82	82
42	83	83	83
41	84	81	84
43	86	84	85
.	.	.	.
.	.	.	.
.	.	.	.
81	123		
82	123		
84	125		
83	126		

Figure 4.3. Example tabular representation of various operations performed

The result is that we now have two matrices $A1$ and $A2$, with each corresponding row of the form $(C1_{DAC}, C1_{ADC})$ and $(C2_{DAC}, C2_{ADC})$ respectively. For the same row number, $C1_{ADC}$ and $C2_{ADC}$ are either exactly equal or within a few codes of each other.

Our goal is to form an equation like (4.7) for each row of the final matrices $A1$ and $A2$ with unknowns on the left side and knowns on the right side. On the right side, if we ignore the error terms er and wq, then g is the only other "unknown". We know that $g = l_{DAC} / l_{ADC}$ is nothing but the gain from the DAC codes to the ADC codes. Within a specific ADC segment, the actual LSB of the DAC can be written as:

$$l_{DAC} = \frac{V1|_{\max\_DAC\_code} - V1|_{\min\_DAC\_code}}{\max\_DAC\_code - \min\_DAC\_code} \tag{4.8}$$

The actual LSB of the ADC can be written as:

$$l_{ADC} = \frac{V1|_{\max\_DAC\_code} - V1|_{\min\_DAC\_code}}{ADC\_code|_{\max\_DAC\_code} - ADC\_code|_{\min\_DAC\_code}} \tag{4.9}$$

From (4.8) and (4.9), g can be estimated as :

$$g = \frac{ADC\_code|_{\max\_DAC\_code} - ADC\_code|_{\min\_DAC\_code}}{\max\_DAC\_code - \min\_DAC\_code} \tag{4.10}$$

For the case of a general digitizer, equation (4.7) still applies. We can simply form the equations only when the output codes are equal. This might lead to a lot discarded rows. To avoid this loss of information, equations can also be formed when the codes are nearby. The gain term required on the right-hand side can be approximated by l_{DAC} divided by the weight of the digitizer. How far the codes can be before they are discarded, is dependent on how linear the digitizer is within an LSB segment. For example, if a digitizer LSB segment spans 16 codes, and we are fairly confident that as long as the two ADC codes are within a specific LSB segment and are within, say, 4 codes from each other, then the INL difference is small, then any code pairs where digitizer code difference is greater than 4 can be discarded. This is further discussed in the error analysis section

The discussion above is relevant mostly for cases where the digitizer's resolution is equivalent or higher than the DAC. When the digitizer's resolution is lower, we can be reasonably sure that we will be able to find DAC codes from the shifted and non-shifted ramp which give us the same digitizer code. In this case, each digitizer output code is its own segment, and matching up ADC codes will be easy after sorting.

4.3.4 uSMILE-ROME

Now that the measurement error has been removed in the previous section, equation (4.1) can be combined with equation (4.7) to get the final equation:

$$\begin{aligned} & E_M(C1_{DAC,MSB}) + E_I(C1_{DAC,ISB}) + E_L(C1_{DAC,LSB}) \\ & -E_M(C2_{DAC,MSB}) - E_I(C2_{DAC,ISB}) - E_L(C2_{DAC,LSB}) + \beta \\ & = \frac{(C1_{ADC} - C2_{ADC})}{g} - (C1_{DAC} - C2_{DAC}) \\ & + er + wq \end{aligned} \tag{4.11}$$

Where $\beta = \alpha / l_{DAC}$ is treated as an unknown. We can form one such equation for each row in the matrices. er is the error term which is negligible, and wq can be treated as random noise (the quantization noise can also be approximated to be whitened). Let's say we do $n_M - n_I - n_L$ segmentation of the DAC's INL. Since we are estimating end-point fit INL, $E_M(0) = E_I(0) = E_L(0) = 0$. We should also add an extra equation $E_M(2^{nMSB} - 1) + E_I(2^{nISB} - 1) + E_L(2^{nLSB} - 1) = 0$ for the last DAC code. If we take just 2 samples per DAC code (one with and one without shift), the number of equations will be $2^{nDAC} - r + 1$ where r is the total number of discarded rows. The number of unknowns will be $2^{n_M} + 2^{n_I} + 2^{n_L} - 3 + 1$ ($2^{n_M} - 1$ MSB segment errors, $2^{n_I} - 1$ ISB errors, $2^{n_L} - 1$ LSB errors and 1 shift term β). For a high resolution DAC and correct INL segmentation, the number of unknowns will be much less than the number of equations. For this over-determined system, the method of least squares can be applied to estimate the unknown vectors E_M, E_I, and E_L, and the shift. The noise term will be effectively averaged out. The full code INL of the DAC can be reconstructed after the least squares estimation.

A complete flow chart of the method is shown in Figure 4.4. The effect of noise and other sources of error are discussed in detail in the error analysis section.

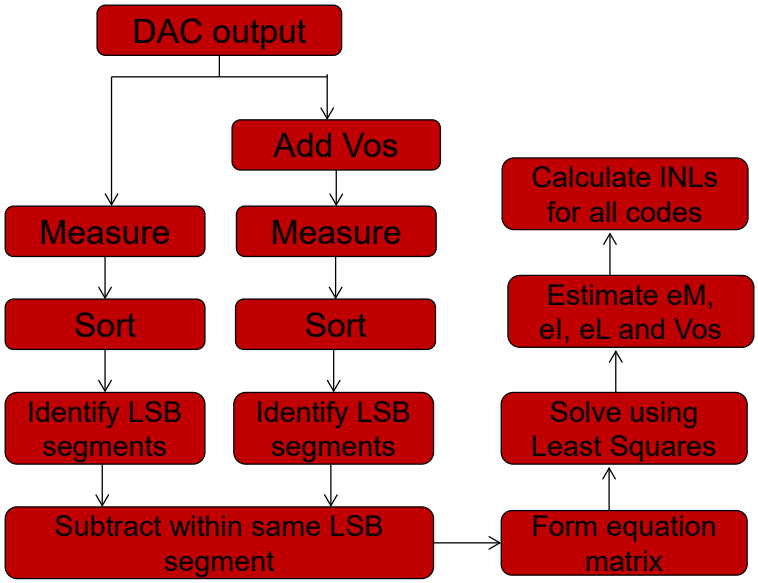


Figure 4.4. Flow chart for uSMILE-ROME

4.4 Error and Noise Analysis

Now, let us analyze the different sources of error in the uSMILE-ROME algorithm. In order to simplify the error analysis, let's ignore the ISB and LSB segment errors. Any estimation error in the MSB segment error now becomes the INL estimation error for all codes in that MSB segment. As we will derive, most errors arise from "accumulation" of small errors during the estimation of the MSB segment errors. For simplicity, we will also assume that the shift between the two ramps is around 1 DAC MSB segment. For example, if the segmentation is 6-5-5 for a 16-bit DAC, the shift is around $2^5 \times 2^5 = 1024$ LSBs. Also assume that we take h number of measurements per DAC code in each ramp. For convenience, we shall denote $n_{IL} = n_I + n_L$.

Now, since we are ignoring ISB and LSB segment errors, equation (4.11) becomes

$$\begin{aligned} & E_M(C1_{DAC,MSB}) - E_M(C2_{DAC,MSB}) \\ & = -\beta + \frac{(C1_{ADC} - C2_{ADC})}{g} - (C1_{DAC} - C2_{DAC}) + er + wq \end{aligned} \tag{4.12}$$

Since we will have one such equation per DAC-ADC code pair, we will get a bunch of equations:

$$\left.\begin{aligned}
&\hat{E}_M(1)-\hat{E}_M(0)=m_1+er_{1,1}\\
&\hat{E}_M(1)-\hat{E}_M(0)=m_1+er_{1,2}\\
&\qquad\vdots\\
&\hat{E}_M(1)-\hat{E}_M(0)=m_1+er_{1,h\times 2^{nIL}}
\end{aligned}\right\}\ h\times 2^{nIL}\ equations$$
$$\begin{aligned}
&\hat{E}_M(2)-\hat{E}_M(1)=m_2+er_{2,1}\\
&\hat{E}_M(2)-\hat{E}_M(1)=m_2+er_{2,2}\\
&\qquad\vdots
\end{aligned}\qquad (4.13)$$

Where $m_i=E_M(i)-E_M(i-1)$ is the true value and all the error terms, including noise, error in shift estimation, shift non-constancy, gain error etc., are lumped into the er terms. For every MSB segment pair, we will have $h\times 2^{n_{IL}}$ number of equations, which can be averaged to get

$$\begin{aligned}
&\hat{E}_M(1)-\hat{E}_M(0)=m_1+er_1\\
&\hat{E}_M(2)-\hat{E}_M(1)=m_2+er_2\\
&\hat{E}_M(3)-\hat{E}_M(2)=m_3+er_3\\
&\qquad\vdots\\
&\hat{E}_M(k)-\hat{E}_M(k-1)=m_k+er_k
\end{aligned}\qquad (4.14)$$

If end-point fit is assumed, then $E_M(0)=0$ and $E_M(2^{n_M}-1)=0$. So, the k'th MSB segment error estimation and the error in its estimation can be derived by adding up k equations:

$$\begin{aligned}
&\hat{E}_M(k)=\sum_{i=1}^{k}m_i+\sum_{i=1}^{k}er_i\\
&\Rightarrow e_M(k)=\hat{E}_M(k)-\sum_{i=1}^{k}m_i=\hat{E}_M(k)-E_M(k)=\sum_{i=1}^{k}er_i=\frac{1}{h\times 2^{nIL}}\sum_{i=1}^{k}\left(\sum_{j=1}^{h\times 2^{nIL}}er_{i,j}\right)
\end{aligned}\qquad (4.15)$$

where $e_M(k)$ is the estimation error of the $k'th$ MSB segment error. We shall now investigate what the error is for different types of error sources.

4.4.1 Effect of noise

Let's say the error term is just from additive thermal noise which follows a normal distribution $N\left(0,\sigma_n^2\right)$. Since each equation in (4.13) comes from 2 DAC-ADC code pairs, the

error term will follow the distribution $er_{i,j} \sim N\left(0, 2\sigma_n^2\right)$. The variance of estimation error at any code k is thus equal to

$$Var\left(e_M\left(k\right)\right) = \frac{k}{h \times 2^{n_{IL}}} \times 2\sigma_n^2 \tag{4.16}$$

Because of end-point fit, the worst case variance of the estimation error will be at the mid-MSB code.

$$Max\left(Var\left(e_M\right)\right) = \frac{2^{n_M}/2}{h \times 2^{n_{IL}}} \times 2\sigma_n^2 = \frac{\sigma_n^2}{h \times 2^{n-2n_M}} \tag{4.17}$$

If the shift amount is more than 1 MSB segment, and the variance of estimation errors due to ISB and LSB errors are assumed to be added, then the worst case variance can similarly be roughly derived as

$$Max\left(Var\left(e_M\right)\right) \approx \frac{\sigma_n^2}{h \times 2^n}\left(\frac{2^{2n_M}}{n_s} + 2^{2n_I} + 2^{2n_L}\right) \tag{4.18}$$

where h is the number of hits/measurements per DAC code and n_s is the shift amount in number of MSB segments.

4.4.2 Effect of shift non-constancy

The shift amount between the 2 ramps has been assumed to be constant till now. In practice, the shift amount will rarely be constant across the voltage range. It is often affected by the DAC code itself, or by the DAC output buffer, or by voltage coefficients of resistors, switch Ron mismatches etc. We will assume that the shift varies as a function of the DAC code. Since the average shift will be estimated as part of the least squares solution, only the deviation from the average shift becomes part of the error term in each equation. Say α_{avg} is the average shift amount and the shift deviation from the average shift amount for the $k'th$ MSB segment is α_k. By definition,

$$\sum_{i=1}^{2^{nM}} \alpha_i = 0 \tag{4.19}$$

According to (4.15), the estimation error of the $k'th$ MSB segment errors

$$e_M(k) = \sum_{i=1}^{k} \alpha_i \tag{4.20}$$

4.4.2.1 Worst Case Bound

Let's say that the absolute maximum value of α_i across all MSB segments is α_e LSBs. Then, an upper bound for the estimation error due to shift non-constancy can be calculated. The worst case shift curve is shown in Figure 4.5. This curve is proved to be the worst case in 0

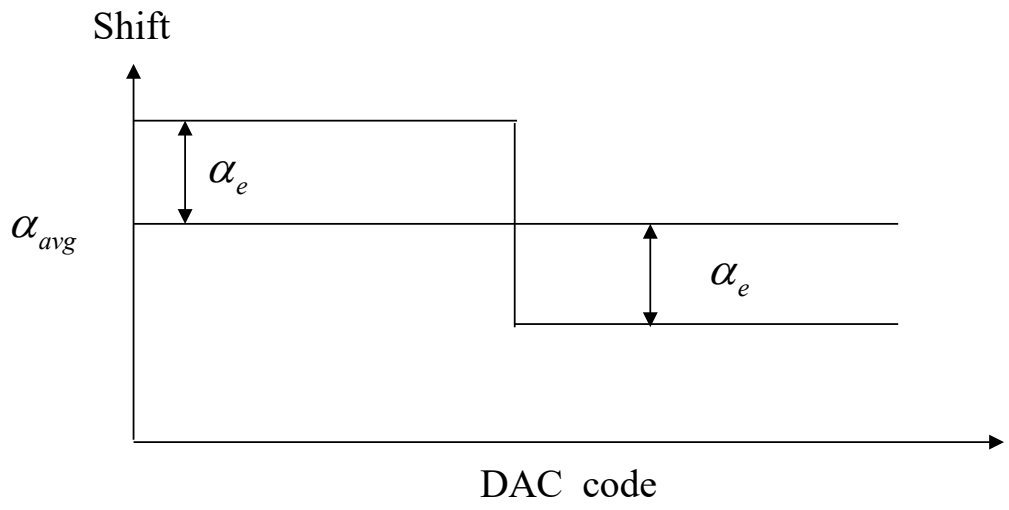


Figure 4.5. Worst case shift curve

Hence, the maximum estimation error will occur for the mid-MSB segment, and can be derived as:

$$Max(e_M) = \sum_{1}^{2^{nM}/2} \alpha_e = \alpha_e \times \frac{2^{nM}}{2} = 2^{nM-1} \times \alpha_e \tag{4.21}$$

4.4.2.2 Gain Error between ramps

If there is a gain error mismatch between the shifted and non-shifted ramps, i.e. $l_{DAC,shifted} \neq l_{DAC,non-shifted}$, this will manifest itself as a shift non-constancy in the equations. Gain error mismatches between the shifted and non-shifted ramps usually arise from voltage dependencies or switch mismatches in the shift mechanism. The average gain error mismatch becomes part of the estimated shift, and the deviation from the average can be treated as shift non-constancy which will look like Figure 4.6. If the difference of gain errors of the shifted and non-shifted DAC ramps in units of LSBs is G_s, as before, the maximum error will be at mid code and can be derived as:

$$Max(e_M) = \frac{1}{2} \times \frac{G_s}{2} \times \frac{2^{nM}}{2} = 2^{nM-3} \times G_s \tag{4.22}$$

4.4.2.3 Gain Error mismatch in ADC LSB segments

The value of the actual ADC lsb within an ADC LSB segment ($l_{ADC\_local\_segment}$) might not be equal to the actual lsb measured across all ADC codes (l_{ADC}). If g_{avg} is the estimated gain

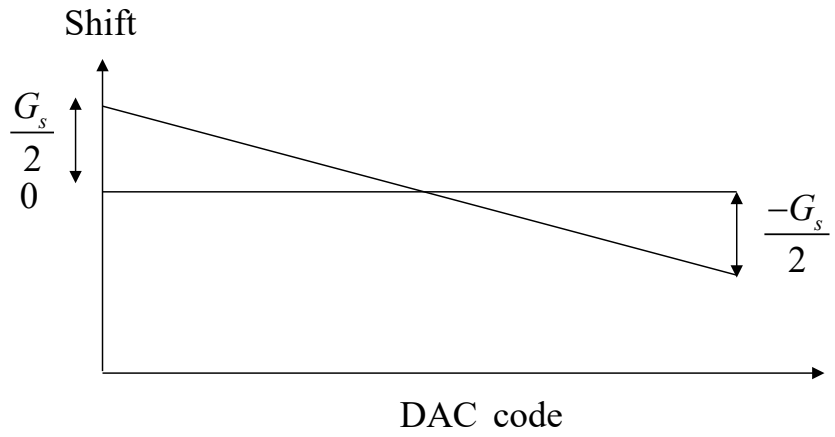


Figure 4.6. Gain error between ramps

between the DAC and the ADC, and Δg is the deviation of gain in an ADC LSB segment, then, for DAC codes in that LSB segment, substituting in (4.7), we get:

$$\begin{aligned} &INL_D\left(C1_{DAC}\right)-INL_D\left(C2_{DAC}\right)+\frac{\alpha}{l_{DAC}} \\ &=\left(C2_{DAC}-C1_{DAC}\right)+\frac{\left(C1_{ADC}+INL_A\left(C1_{ADC}\right)-\left(C2_{ADC}+INL_A\left(C2_{ADC}\right)\right)\right)}{g_{avg}+\Delta g} \end{aligned} \tag{4.23}$$

The equivalent shift deviation in a DAC MSB segment can be approximated as:

$$er_i \approx \frac{\Delta\left(C_{ADC}+INL(C_{ADC})\right)}{g_{avg}}\times\frac{\Delta g}{g_{avg}} \tag{4.24}$$

Where $g_{avg}=\frac{l_{ADC}}{l_{DAC}}$ and $\Delta g=\frac{l_{ADC,local\ segment}-l_{ADC}}{l_{DAC}}$. The worst case bound can be calculated by substituting $\alpha_e = abs(\max(er_i))$ across all DAC MSB segments.

$$Max(e_M) \approx \frac{2^{nM}}{2}\times\frac{\Delta C_{ADC}}{g_{avg}}\times\frac{\Delta g}{g_{avg}} \tag{4.25}$$

In actual implementation, the above analysis gives a bound which is usually small enough that it can be ignored, but this is still a very pessimistic bound, because many of the errors within the ADC LSB segment actually cancel each other out. If they do not, we can ensure that they do. When the ADC codes are equal, there is no error and we include the equation. For the equations in which ADC code difference is not 0, we discard some equations within the same ADC LSB segment such that the number of equations for which the ADC code difference is +1 is equal to the number of equations for which the ADC code difference is -1. A similar exercise can be performed for +2/-2, +3/ -3 and so on. This way, all the errors will cancel out.

4.5 Simulation results

To verify the effectiveness of the proposed method, extensive MATLAB simulations were performed with different DAC architectures and INL levels. The R-2R DAC is particularly studied due to its wide usage, high resolution and low power. A 16-bit R-2R DAC, modeled with resistor

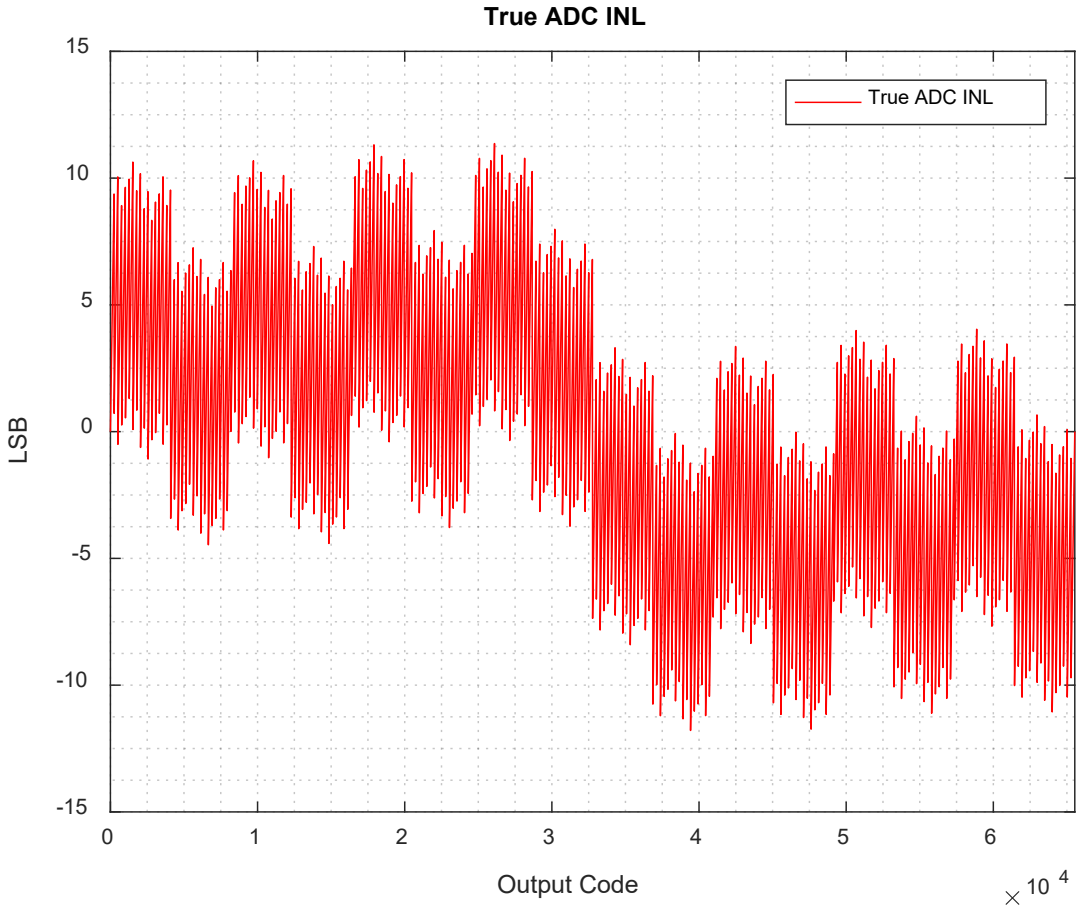


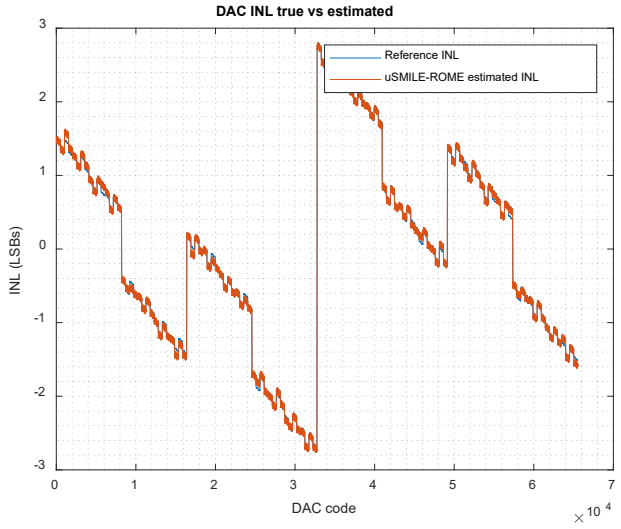
Figure 4.7. INL of ADC used for measurement

mismatches, was used as the device under test. A 16-bit SAR ADC, modeled with capacitor mismatches, was used as the measurement device. The SAR ADC had a scale down capacitor after 8 MSB bits. It was ensured that the ADC did not have any wide codes or dead zones. The additive noise added to the output of the DAC was set to around 0.3 LSB level.

First, to show that the method reduces the linearity requirement on the measurement device, a relatively low-linearity ADC is taken. The INL of the ADC is plotted in Figure 4.7. As can be seen in the figure, the INL of the ADC is around the +/-10 LSB level, which means that the ADC is only around 12-bit linear.

While estimating the INL of the DAC using the uSMILE-ROME method, the INL curve is segmented as 7-5-4. The number of unknowns is 176. Say the number of equations that remain after the discarded rows is around 80% of 2^16. The average number of equations per variable is

(a)



(b)

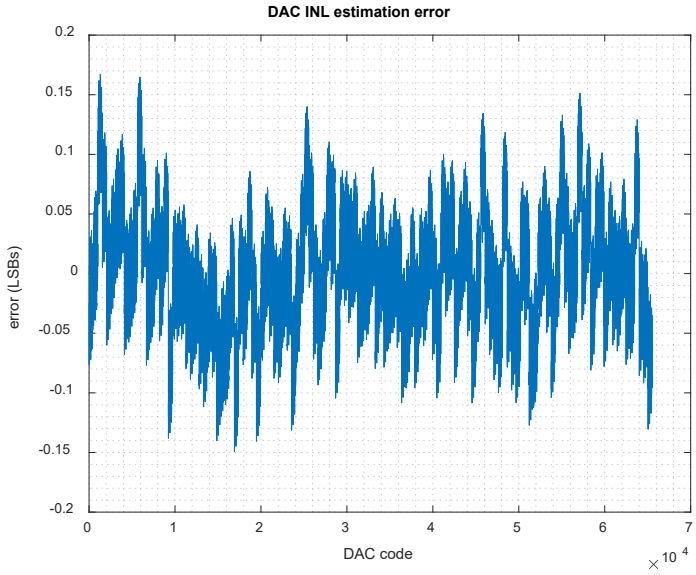


Figure 4.8. (a) True and estimated DAC INL (b) Error in INL estimation

around 298. The true and estimated INL of the DAC, along with the error in estimation for each DAC code, is shown in Figure 4.8.

It can be seen that the DAC INL estimation is very accurate in spite of the ADC used for measurement being highly non-linear (~12 bit linear). In comparison, the conventional method would have required an ADC with higher resolution and effective linearity of around 2 to 3 bits more than that of the DAC itself.

To further test the robustness of the method, 100 simulations were run with 100 randomly generated 16-bit DACs and 16-bit ADCs. The maximum and minimum of the DAC INL estimation errors for each run is shown in Figure 4.9. Here, Maximum error = max(INL_est – INL_tru) and Minimum error = min(INL_est – INL_tru).

All the estimation errors are within ± 0.44 LSBs. The bound for 3σ predicted by (4.18) comes out to be ~0.47 LSBs, which matches the simulation results well.

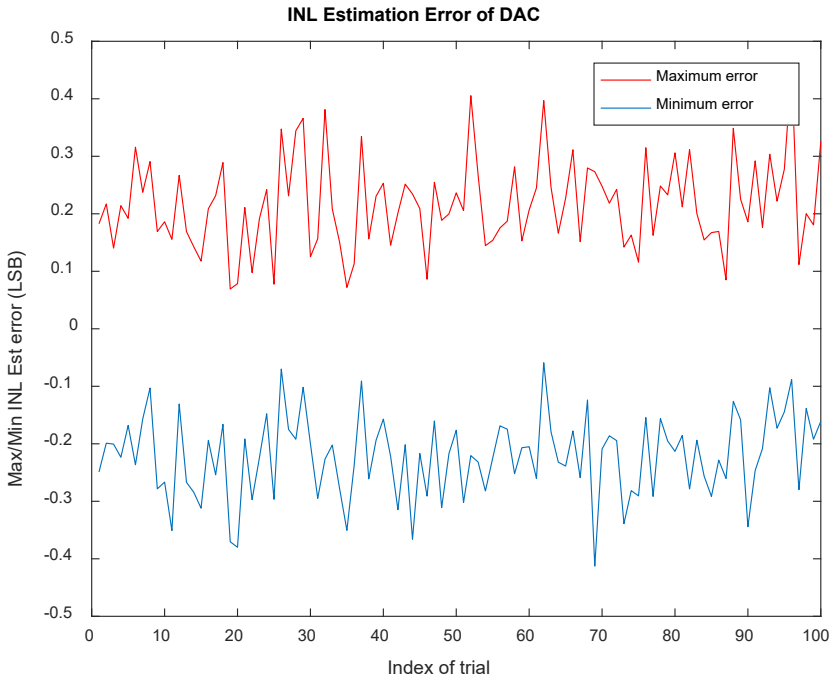


Figure 4.9. Maximum and minimum INL estimation errors over 100 runs

A different view of the estimation accuracy is presented in Figure 4.10. The X-axis is the true maximum absolute INL and the Y-axis is the estimated maximum absolute INL.

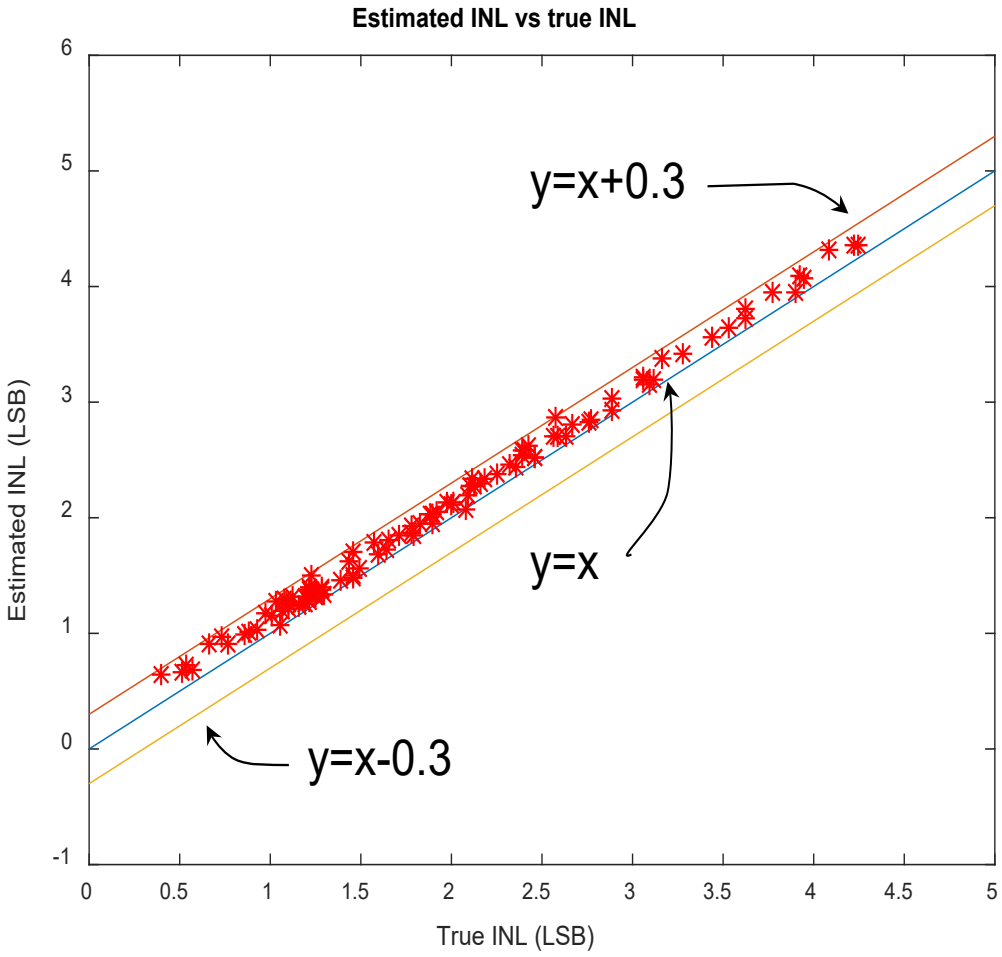


Figure 4.10. Estimated INL vs True INL over 100 runs

Ideally, all the points should lie on the $y = x$ line. ± 3 LSBs bands around $y = x$ are also plotted. All the points are contained within this band, which implies that the accuracy of the uSMILE-ROME algorithm is very high. The simulation results show that the proposed method is robust over different DAC linearity levels.

4.5.1 Effect of non-constant shift

To demonstrate the effect of non-constant shift on the algorithm, the same DAC and ADC are taken and an additional artificial non-constant shift is added between the DAC output ramps. This shift non-constancy is plotted in Figure 4.11.

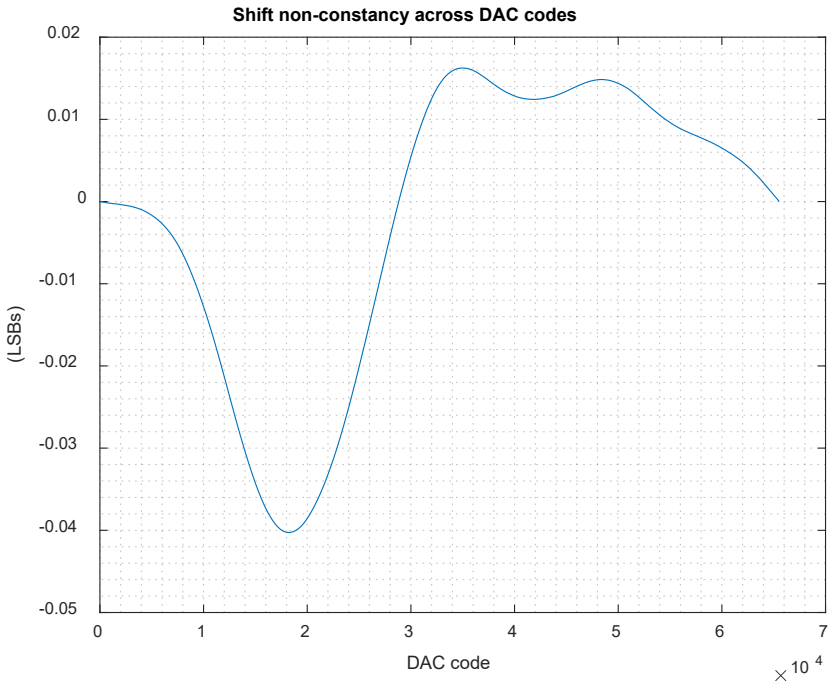
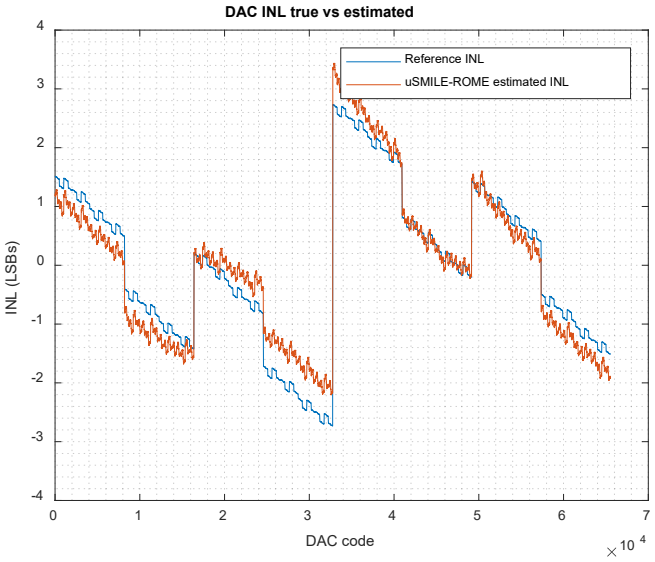


Figure 4.11. Artificially added shift non-constancy

The additive noise was set to 0 to demonstrate only the effect of shift non-constancy. The same 7-5-4 segmentation was used for the DAC and uSMILE-ROME was run to estimate the INL of the DAC. The simulation results are shown in Figure 4.12. Figure 4.12 (a) shows the true and estimated INL. Figure 4.12 (b) shows the INL estimation error in blue. Superimposed over it is the predicted estimation error per MSB segment as predicted by equation (4.20). To get the shape, first, the average shift non-constancy per DAC MSB segment was calculated. Then, a cumulative sum of these errors was calculated, and this sum was repeated for all codes within the respective

MSB segment to get the error for every DAC code. As can be seen, the simulated INL estimation error tracks the predicted estimation error very well, validating the theory described above.

(a)



(b)

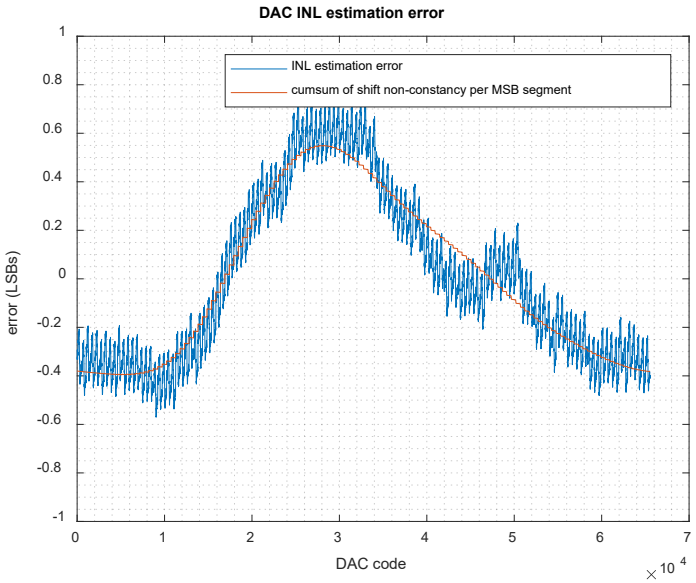


Figure 4.12. Simulation results (a) True and estimated DAC INL (b) Predicted and simulated error in INL estimation

4.6 Measurement Results

To validate the simulation results with actual measurement, both a 12-bit and 16-bit DAC were measured with a 16-bit SAR ADC using uSMILE-ROME. The shift was provided using an opamp on a breadboard. The circuit is shown in Figure 4.13.

The output voltage Vout is:

$$V_{out} = 2V_s - V_{DAC} \tag{4.26}$$

For every DAC code, the V_s signal was switched between V_{s1} and V_{s2}, where:

$$\begin{aligned} V_{s1} &= \frac{V_{ref}}{2} + \frac{\alpha}{4} \\ V_{s2} &= \frac{V_{ref}}{2} - \frac{\alpha}{4} \end{aligned} \tag{4.27}$$

where α is the shift amount between the ramps. The outputs will thus be:

$$\begin{aligned} V_{out1} &= V_{ref} - V_{DAC} + \frac{\alpha}{2} \\ V_{out2} &= V_{ref} - V_{DAC} - \frac{\alpha}{2} \end{aligned} \tag{4.28}$$

The chips used were a TI C2000 launchpad, a TI x0508 EVM and LMC6061 opamp. The C2000 microcontroller has 3 12-bit DACs and 4 ADCs (can be used in 12-bit/16-bit modes). TI x0508 is a 16-bit DAC. Pictures of the test setup are shown in figx.

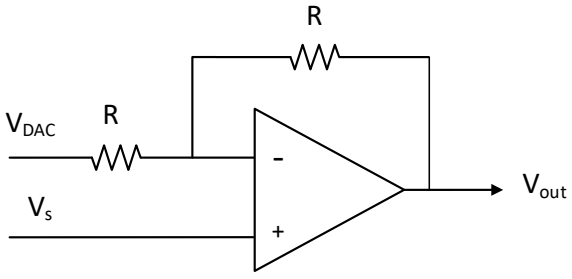


Figure 4.13. Shift using Opamp

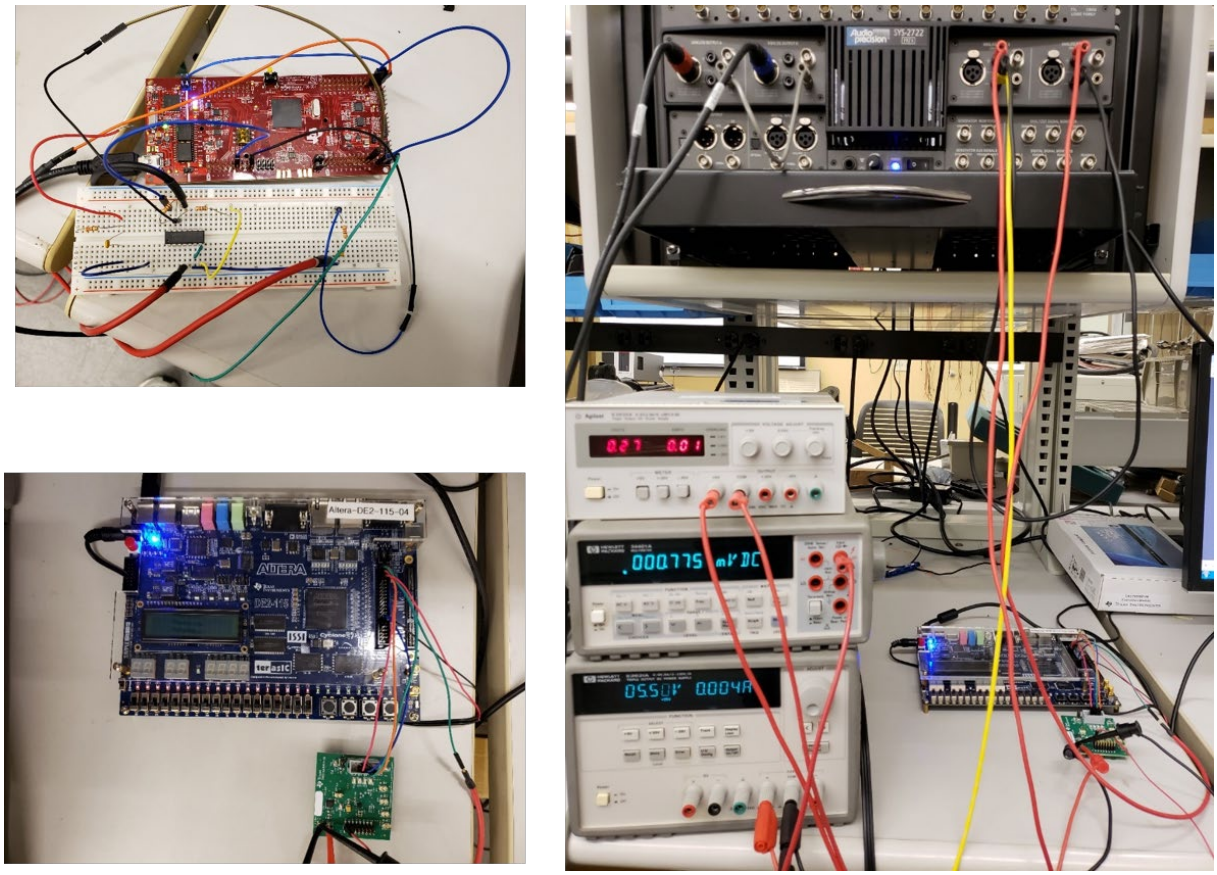
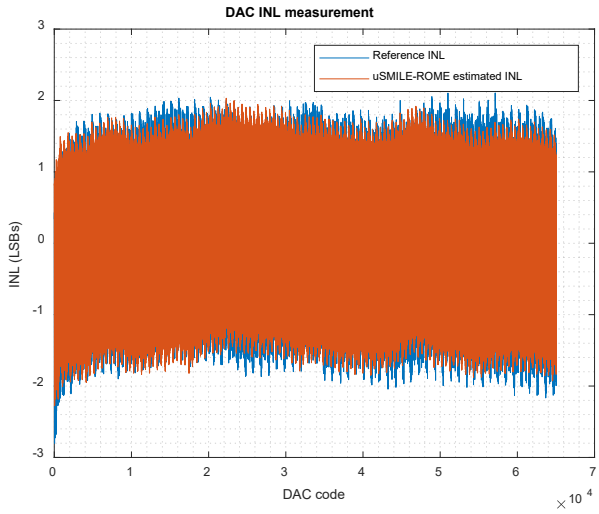


Figure 4.14. Measurement Test Setup

First the 16-bit DAC in TI x0508 was taken as the Device under test (DUT). The DAC was controlled using an Altera DE2 – 115 FPGA and the reference INL was measured using Audio Precision 2722, with 330 samples per DAC code to average out noise. Next, the 16-bit SAR ADC on C2000 was used as the measuring device and the circuit shown in Figure 4.13 was setup, with one of the 12-bit DACs from the C2000 chip used to provide the Vs signal. The sampling process was controlled using the C2000 microcontroller itself, with 4 samples taken per DAC code (shifted and non-shifted). uSMILE-ROME was then applied with 7-5-4 segmentation assumed for the 16-bit DAC. The measurement results are shown in Figure 4.15. Figure 4.15 (a) shows the reference INL of the 16-bit DAC in blue and the uSMILE-ROME estimated INL in red. Figure 4.15 (b)

(a)



(b)

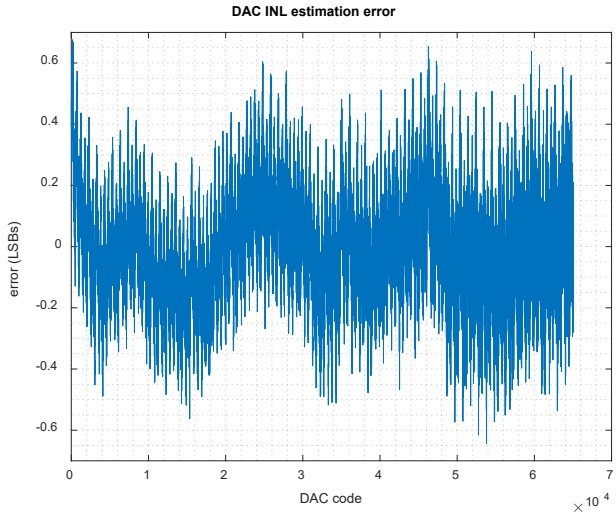
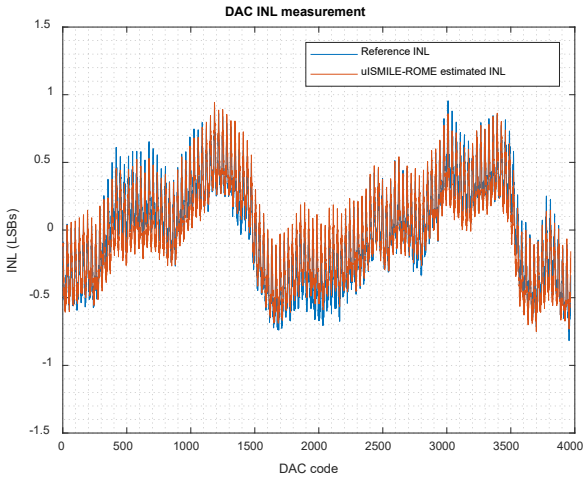


Figure 4.15. 16-bit DAC – 16-bit ADC Measurement results:
(a) Reference and uSMILE-ROME estimated DAC INL (b) Error in INL estimation

shows the error in DAC INL estimation. As can be seen, the error is mostly within ±0.6 LSBs. The DAC seems to be more nonlinear at lower codes, likely because of the output buffer nonlinearity. The segmented model is able to capture this smooth nonlinearity at lower codes reasonably well, though. The variance of estimation error also seems to be higher at higher DAC codes. This can be attributed to noise from the supply voltage.

(a)



(b)

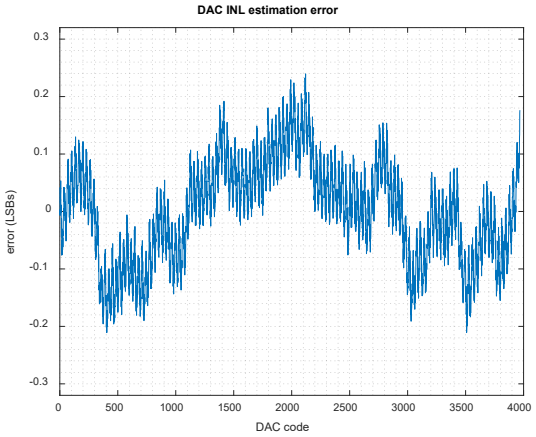


Figure 4.16. 12-bit DAC – 12-bit ADC Measurement results:
(a) Reference and uSMILE-ROME estimated DAC INL (b) Error in INL estimation

Next, one 12-bit DAC on the C2000 chip was taken as the Device under test (DUT). The DAC was controlled using the microcontroller itself and the reference INL was measured using the on-chip 16-bit ADC, with 256 samples per DAC code to average out noise. Next, the ADC on C2000 was used in 12-bit mode as the measuring device and the shift circuit using the opamp was setup, with another one of the 12-bit DACs used to provide the Vs signal. The sampling process

was controlled using the C2000 microcontroller itself, with 4 samples taken per DAC code (shifted and non-shifted). uSMILE-ROME was then applied with 7-5 segmentation assumed for the 12-bit DAC. The measurement results are shown in Figure 4.16. Figure 4.16 (a) shows the reference INL of the 16-bit DAC in blue and the uSMILE-ROME estimated INL in red. Figure 4.16 (b) shows the error in DAC INL estimation. As can be seen, the error is mostly within ± 0.25 LSBs.

4.7 Conclusion

A fast, low-cost method for static linearity testing of DACs is presented in this paper. The uSMILE-ROME method allows the testing of high linearity DACs with low-linearity on board digitizers, while considerably reducing the test time and the test cost. This is enabled by using a segmented non-parametric model for the INL curve of the DAC, which reduces the number of unknowns to be estimated. Additionally, the measurement error introduced due to the on-board digitizer is removed in the algorithm, thus allowing it to be orders of magnitude less linear than the DAC under test. All of this is combined in the uSMILE-ROME method, and results in significant reduction in time and cost for static linearity testing of DACs as compared to the traditional method. Measurement results on a 16-bit DAC – 16-bit ADC setup and a 12-bit DAC – 12-bit ADC setup demonstrate the validity of the algorithm in significantly reducing linearity test cost for DACs.

4.8 References

[1] S. K. Chaganti, T. Chen, Y. Zhuang, and D. Chen, “Low-cost and accurate DAC linearity test with ultrafast segmented model identification of linearity errors and removal of measurement errors (uSMILE-ROME),” in *2018 IEEE International Instrumentation and Measurement Technology Conference (I2MTC)*, May 2018, pp. 1–6, doi: 10.1109/I2MTC.2018.8409877.

[2] M. Burns, G. W. Roberts, and F. Taenzler, *An introduction to mixed-signal IC test and measurement*, vol. 2001. Oxford University Press New York, 2001.

[3] "IEEE Standard for Terminology and Test Methods of Digital-to-Analog Converter Devices," *IEEE Std 1658-2011*, pp. 1–126, Feb. 2012, doi: 10.1109/IEEESTD.2012.6152113.

[4] Y. Zhuang, B. Magstadt, T. Chen, and D. Chen, "High-Purity Sine Wave Generation Using Nonlinear DAC With Predistortion Based on Low-Cost Accurate DAC-ADC Co-Testing," *IEEE Transactions on Instrumentation and Measurement*, vol. PP, no. 99, pp. 1–9, 2017, doi: 10.1109/TIM.2017.2769238.

[5] L. Jin, K. Parthasarathy, T. Kuyel, D. Chen, and R. L. Geiger, "Accurate testing of analog-to-digital converters using low linearity signals with stimulus error identification and removal," *IEEE Transactions on Instrumentation and Measurement*, vol. 54, no. 3, pp. 1188–1199, Jun. 2005, doi: 10.1109/TIM.2005.847240.

[6] H. Xing, D. Chen, and R. Geiger, "On-chip at-speed linearity testing of high-resolution high-speed DACs using DDEM ADCs with dithering," in *2008 IEEE International Conference on Electro/Information Technology*, May 2008, pp. 117–122.

[7] X. L. Huang and J. L. Huang, "ADC/DAC Loopback Linearity Testing by DAC Output Offsetting and Scaling," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 19, no. 10, pp. 1765–1774, Oct. 2011, doi: 10.1109/TVLSI.2010.2063443.

[8] H.-W. Ting, S.-J. Chang, and S.-L. Huang, "A Design of Linearity Built-in Self-Test for Current-Steering DAC," *J Electron Test*, vol. 27, no. 1, pp. 85–94, Feb. 2011, doi: 10.1007/s10836-010-5187-2.

[9] Z. Yu and D. Chen, "Algorithm for dramatically improved efficiency in ADC linearity test," in *2012 IEEE International Test Conference*, Nov. 2012, pp. 1–10, doi: 10.1109/TEST.2012.6401561.

[10] T. Chen and D. Chen, "Ultrafast stimulus error removal algorithm for ADC linearity test," in *2015 IEEE 33rd VLSI Test Symposium (VTS)*, Apr. 2015, pp. 1–5, doi: 10.1109/VTS.2015.7116249.

[11] B. K. Vasan, J. Duan, C. Zhao, R. L. Geiger, and D. J. Chen, "Signal generators for cost effective BIST of ADCs," in *2009 European Conference on Circuit Theory and Design*, Aug. 2009, pp. 113–116, doi: 10.1109/ECCTD.2009.5274967.

[12] X. Jin and N. Sun, "Low-cost high-quality constant offset injection for SEIR-based ADC built-in-self-test," in *2014 IEEE International Symposium on Circuits and Systems (ISCAS)*, Jun. 2014, pp. 285–288, doi: 10.1109/ISCAS.2014.6865121.

4.9 Appendix: Proof of Worst Case Shift Non-constancy

Lemma: If a set of variables α_i and $e(k)$ satisfy the following constraints:

$$\sum_{i=1}^{n} \alpha_i = 0 \tag{4.29}$$

$$-\alpha_e \le \alpha_i \le \alpha_e \tag{4.30}$$

$$e(k) = \sum_{i=1}^{k} \alpha_i \qquad k = 1, 2, .., n \tag{4.31}$$

Then it can be proved that

$$-\frac{N}{2}\alpha_e \le e(k) \le \frac{N}{2}\alpha_e \tag{4.32}$$

Where k is a positive integer, N is a positive even integer, α_i's and $e(k)$'s are real numbers and α_e is a positive real number.

Proof:

We shall prove this by contradiction.

Let's assume that, for some $k_1 \le N$,

$$e(k_1) = \sum_{i=1}^{k_1} \alpha_i > \frac{N}{2}\alpha_e \tag{4.33}$$

From (4.30), we can say that, for any positive integers a and b, such that $a < b$,

$$-(b-a+1)\alpha_e \le \sum_{i=a}^{b} \alpha_i \le (b-a+1)\alpha_e \tag{4.34}$$

Substitute $a = 1$ and $b = k_1$ to get:

$$-k_1\alpha_e \le \sum_{i=1}^{k_1} \alpha_i \le k_1\alpha_e \tag{4.35}$$

From (4.29), we can say that:

$$\sum_{i=1}^{k_1} \alpha_i + \sum_{i=k_1+1}^{N} \alpha_i = 0 \tag{4.36}$$

Substitute (4.36) in (4.33) to get:

$$\begin{aligned} &-\sum_{i=k_1+1}^{N} \alpha_i > \frac{N}{2}\alpha_e \\ &\Rightarrow \sum_{i=k_1+1}^{N} \alpha_i < -\frac{N}{2}\alpha_e \end{aligned} \tag{4.37}$$

Combining 1.34 and 1.36, we get:

$$\begin{aligned} &-(N-k_1)\alpha_e \le \sum_{i=k_1+1}^{N} \alpha_i < -\frac{N}{2}\alpha_e \\ &\Rightarrow k_1\alpha_e < \frac{N}{2}\alpha_e \end{aligned} \tag{4.38}$$

From (4.31), (4.35), and (4.38), we get:

$$e(k) = \sum_{i=1}^{k} \alpha_i \le k_1\alpha_e < \frac{N}{2}\alpha_e \tag{4.39}$$

This is a direct contradiction of our initial assumption in (4.33), which means that our assumption is not possible given the constraints. It can similarly be proven $e(k)$ cannot be less than $-\frac{N}{2}\alpha_e$.

Also we know that $e\left(\frac{N}{2}\right) = \frac{N}{2}\alpha_e$ when

$$\begin{aligned} &\alpha_i = \alpha_e \quad for \;\; i = 1, 2, ..., \frac{N}{2} \\ &\alpha_i = -\alpha_e \quad for \;\; i = \frac{N}{2}, ..., N \end{aligned} \tag{4.40}$$

This satisfies the constraint that the average of α_i's should be 0, while giving the upper bound value for $e(k)$ at $k = \frac{N}{2}$.

Similarly, we can see that $e\left(\frac{N}{2}\right) = -\frac{N}{2}\alpha_e$ when

$$\begin{aligned} \alpha_i &= -\alpha_e \quad for \;\; i = 1,2,...,\frac{N}{2} \\ \alpha_i &= \alpha_e \quad for \;\; i = \frac{N}{2},...,N \end{aligned} \tag{4.41}$$

Hence, proved that $-\frac{N}{2}\alpha_e \le e(k) \le \frac{N}{2}\alpha_e$ and that the lower and upper bound can be achieved at $k = \frac{N}{2}$ when α_i's are as described in (4.40) and (4.41) respectively.

CHAPTER 5. ON-CHIP BUILT-IN SELF-TEST AND SELF-CALIBRATION OF DIGITAL TO ANALOG CONVERTERS USING USMILE-ROME

Abstract

The Digital–to–Analog-Converter (DAC) is one of the fundamental components of Analog and Mixed-signal circuits. Static linearity testing of high resolution high performance DACs traditionally requires a long time and is very expensive. In this chapter, a complete on-chip DAC BIST solution based on uSMILE-ROME is presented. By adapting the algorithm for efficient memory usage, the solution can be implemented on-chip and is especially useful for large SoCs like microcontrollers. Once DAC linearity is measured using the BIST implementation, the results can also be used for calibration of the DAC using pre-distortion of the input DAC codes, thus leading to a self-calibration method. This self-calibrated DAC can be used to generate a pure sine wave as long as the dominant sources of errors are the static nonlinearity of the DAC. This generated sine wave can also be used for spectral testing of ADCs.

5.1 Introduction

Testing of Digital-to-Analog Converters (DACs) has become ever more challenging in the semiconductor industry. Some of these challenges have been discussed in the previous chapters, like expensive test equipment and long test times. In chapter 4, the uSMILE-ROME algorithm was developed which enables testing of high resolution high linearity DACs with low-cost on-board/on-chip ADCs/digitizers, while reducing test time. The story here is not complete, though, without discussion of the challenges involved in the computations in the algorithm itself. The computation involved in the algorithm, as presented in chapter 4, is quite intensive and involves a linear least squares regression on a large numeric matrix. It was assumed that all the computations would be performed off-chip. In a production/test environment, this means that the output codes of the ADC would have to be sent out to the Automated Test Equipment (ATE), generally referred

to as the tester. Typically, the processor on the tester is powerful enough that these computations can be performed very quickly, so this is not a bottleneck. But in a high multi-site test environment, the real bottleneck is the throughput of data transfer from the ADC to the tester. If the ADC is off-chip and on the test board, then there is not much that can be done and the speed of data collection will be low. For this reason, and for others discussed below, it is preferable to have the ADC on-chip. This is very often already the case, especially for large SoCs, like microcontrollers etc., which have a large number of analog IPs and data converters along with a processor and memory.

For an n-bit DAC, typically, at least two samples per DAC code would need to be taken, one with shift and one without shift. This means 2×2^n ADC output codes would have to be stored on-chip, say in the RAM, and then transferred to the tester. The parallel efficiency for RAM reads of such a large amount of data is typically not high when there are a large number of sites. This leads to long test times. It is highly desirable that the computations be performed on the SoC itself. This would enable 100% parallel test efficiency as far as the computation is involved and only the final linearity test results (INLmax/INLmin and DNLmax/DNLmin) can be sent out to the tester.

In addition to this, testing deeply embedded analog/mixed-signal blocks in an SoC is becoming very challenging and expensive due to the lack of access to internal nodes and the difficulty of maintaining adequate signal integrity while driving an accurate signal on and off chip. A built-in self-test (BIST) capability is hence highly desirable since it requires no external signals, and the data can be processed using on-chip resources!

Once the DAC is able to be tested on-chip, the results can be used to calibrate out the nonlinearity using pre-distortion of the DAC input codes. Hence, this BIST will also enable self-healing. Not only can this be used for calibration in production, it can be used in the field, say for example during/after power-on to self-test the DAC and possibly self-calibrate to ensure optimal

performance throughout the end of life of the chip. This calibrated DAC can now also be used for online monitoring of other voltages on-chip using the concurrent sampling method [1].

Now, uSMILE-ROME already relaxes the linearity requirement on the ADC used to test the DAC, so the available on-chip ADC can be used for testing the DAC even if its resolution and linearity is comparable to/lower than the DAC. The only piece of the puzzle left is figuring out how to perform the data processing on-chip. As mentioned before, uSMILE-ROME requires storage of a large numeric matrix and performing matrix multiplications and inversions for least squares estimation. The first big challenge is to solve the memory requirement.

It should be noted that the on-chip implementation of uSMILE-ROME for DAC testing is a similar problem to that of on-chip implementation of USER-SMILE for ADC testing. A detailed on-chip implementation procedure has already been developed for USER-SMILE [2], and we shall follow a similar procedure for uSMILE-ROME, along with some necessary additions and modifications. We will also take up an example case study for memory efficient implementation of uSMILE-ROME for a sub-radix 2 DAC [3], and then describe memory optimization for the algorithm as well as pre-distortion procedures, with simulation results.

5.2 Review of uSMILE-ROME

Since the fundamentals of uSMILE-ROME have already been described in the previous chapter, we will not go much in detail here. Two equation are combined to give rise to the uSMILE-ROME equation, and both are stated here for convenience. The first is that the DAC's INL can be modeled as a combination of 2^{n_M} MSB, 2^{n_I} ISB and 2^{n_L} LSB code errors:

$$INL(C) = E_M(C_M) + E_I(C_I) + E_L(C_L) \tag{5.1}$$

The second equation is:

$$\begin{aligned} &(C2 - C1) + (C1_{ADC} - C2_{ADC})/g \\ &= INL_D(C1) - INL_D(C2) + \beta + noise + er \end{aligned} \tag{5.2}$$

where $(C1, C1_{ADC})$ and $(C2, C2_{ADC})$ are the (input DAC code, corresponding ADC output code) pairs from the shifted and non-shifted ramps respectively, chosen according to the procedure described in the chapter 4. g is the gain between the DAC and the ADC, β is the shift amount in units of DAC lsb's and er is the sum total of all measurement and modeling errors. How to select these code pairs during data acquisition in a memory efficient manner will be discussed as part of the example case study later on. For now, we will assume that we are able to get such code pairs on the fly during data acquisition. The above 2 equations can be combined to get:

$$\begin{aligned}&(C2-C1)+(C1_{ADC}-C2_{ADC})/g\\&=E_M(C1_M)+E_I(C1_I)+E_L(C1_L)\\&-E_M(C2_M)+E_I(C2_I)+E_L(C2_L)+\beta+noise+er\end{aligned} \tag{5.3}$$

Each set of code pairs has one such equation. We need to estimate 2^{n_M} MSB code errors, 2^{n_I} ISB code errors, 2^{n_L} LSB code errors and the 1 shift amount β. The number of unknowns is thus $K = 2^{n_M} + 2^{n_I} + 2^{n_L} + 1$.

With M sets of code pairs, all M equations can be written in matix form as:

$$y_d = H.\begin{bmatrix}E_M\\E_I\\E_L\\\beta\end{bmatrix}+noise+er \tag{5.4}$$

where y_d is a column matrix of length M, with:

$$y_d(i)=\left(C2(i)-C1(i)\right)+\left(C1_{ADC}(i)-C2_{ADC}(i)\right)/g \tag{5.5}$$

H is an $M \times K$ matrix which has three +1s and three -1s in each row in columns corresponding to the MSB, ISB and LSB codes of C1 and C2:

$$\begin{aligned}&H(i,C1_M(i))=H(i,2^{nM}+C1_I(i))=H(i,2^{nM}+2^{nI}+C1_L(i))=1\\&H(i,C2_M(i))=H(i,2^{nM}+C2_I(i))=H(i,2^{nM}+2^{nI}+C2_L(i))=-1\end{aligned} \quad i=1,..,M \tag{5.6}$$

and the last column of H is all +1s which corresponds to the shift. $E=[E_M \quad E_I \quad E_L \quad \beta]^T$ is the

column matrix of unknowns of size $K \times 1$. This is an over-determined system of equations and can be solved using the least squares method as:

$$\hat{E} = \left(H^T H\right)^{-1}\left(H^T y_d\right) \tag{5.7}$$

The noise and other measurement errors in the equations are effectively "averaged out", but if one wishes to calculate the final errors in the estimation of the INLs, the error term can be calculated following a similar procedure as in Chapter 3 Section 4, since the basic structure is very similar.

5.3 Memory efficient implementation

Now, as can be seen from (5.7) above, the estimation of unknowns, and then the calculation of INLs, requires the storage of an $M \times K$ matrix. M is of the order of $\sim 2^{n_{DAC}}$ and the value of K is of the order of ~100. As an example, for a 16-bit DAC, M will be 65536 if 1 measurement is taken per DAC code, and if segmented as 7-5-4, K will be 160. It is definitely not feasible to perform this calculation on-chip as presented in (5.7) because of the memory requirements. To facilitate on-chip computation, some modifications to the original equation need to be made.

First let's re-write (5.4) as:

$$y_d = \begin{bmatrix} H_M & H_I & H_L & S \end{bmatrix} . \begin{bmatrix} E_M \\ E_I \\ E_L \\ \beta \end{bmatrix} + \text{noise+er} \tag{5.8}$$

Where H_M is the submatrix from column 1 to column 2^{n_M}, H_I is the submatrix from column $2^{n_M} + 1$ to column $2^{n_M} + 2^{n_I}$ and H_L is the submatrix from column $2^{n_M} + 2^{n_I} + 1$ to column $2^{n_M} + 2^{n_I} + 2^{n_L}$. S is a column matrix with all rows as +1's. For convenience, we will ignore the noise term from now on. Next, let's multiply both sides by H^T:

$$H^T \cdot y_d = H^T \cdot H \cdot \begin{bmatrix} E_M \\ E_I \\ E_L \\ \beta \end{bmatrix} \tag{5.9}$$

If we can somehow directly compute and store store $H^T H$, which is a $K \times K$ matrix, and $H^T y_d$, which is a $K \times 1$ matrix, then the unknowns can simply be estimated using equation (5.7), with the memory required now being $K^2 + K$ decimal numbers. Since $H^T H$ is a symmetric square matrix, Cholesky decomposition can be used to compute the inverse and this can be done in place, which means no extra memory is required for the inverse calculations.

Let's investigate how $H^T H$ and $H^T y_d$ can be directly updated for every code pair. First let's look at the term $H^T H$ which can be written as:

$$\begin{aligned} H^T \cdot H &= \begin{bmatrix} H_M^T \\ H_I^T \\ H_L^T \\ S^T \end{bmatrix} \cdot \begin{bmatrix} H_M & H_I & H_L & S \end{bmatrix} \\ &= \begin{bmatrix} H_M^T H_M & H_M^T H_I & H_M^T H_L & H_M^T S \\ H_I^T H_M & H_I^T H_I & H_I^T H_L & H_I^T S \\ H_L^T H_M & H_L^T H_I & H_L^T H_L & H_L^T S \\ S^T H_M & S^T H_I & S^T H_L & S^T S \end{bmatrix} \end{aligned} \tag{5.10}$$

Note that we only need to fill up the upper triangular matrix since $H^T H$ is symmetric and the other entries can be calculated easily, and for computation of the inverse, Cholesky decomposition only requires either the upper triangular or the lower triangular matrix anyway. If really required, with smart indexing, the memory requirement can be reduced to $(K^2 + K)/2$ by storing only the upper triangular matrix of $H^T H$ as a single vector. For a 2 dimensional square matrix of size $K \times K$, if the indices start from 0, then for a given row a and column b, the linear index c of the 1-dimensional upper triangular matrix can be derived as:

$$c = \frac{K(K+1)}{2} - \frac{(K-a)(K-a+1)}{2} + b - a \tag{5.11}$$

The total memory required for $H^T H$ and $H^T y_d$, if this scheme is followed, will then be $(K^2 + K)/2 + K = (K^2 + 3K)/2$. It should be noted that indexing every time when the matrix needs updating might be more time than its worth. For convenience, the subsequent equations will continue to assume a 2 dimensional matrix.

First, all matrices should be initialized to 0, and then the various elements of the submatrices should be incremented/decremented in the following manner, for every $i'th$ equation. For every $\left(C1(i), C1_{ADC}(i)\right)$ and $\left(C2(i), C2_{ADC}(i)\right)$ pair:

$$\begin{aligned}
H_x^T H_y\left(C1_x(i), C1_y(i)\right) &= H_x^T H_y\left(C1_x(i), C1_y(i)\right) + 1 \\
H_x^T H_y\left(C2_y(i), C2_y(i)\right) &= H_x^T H_y\left(C2_y(i), C2_y(i)\right) + 1 \\
H_x^T H_y\left(C1_y(i), C2_y(i)\right) &= H_x^T H_y\left(C1_y(i), C2_y(i)\right) - 1 \\
H_x^T H_y\left(C2_y(i), C1_y(i)\right) &= H_x^T H_y\left(C2_y(i), C1_y(i)\right) - 1
\end{aligned} \tag{5.12}$$

where (x, y) are $(M, M), (M, I), (M, L), (I, I), (I, L), and\ (L, L)$. The rest of the entries are updated in the following way:

$$\begin{aligned}
H_x^T S\left(C1_x(i)\right) &= H_x^T S\left(C1_x(i)\right) + 1 \\
H_x^T S\left(C2_x(i)\right) &= H_x^T S\left(C2_x(i)\right) - 1
\end{aligned} \tag{5.13}$$

where x is M, I and L. Finally:

$$S^T S = M \tag{5.14}$$

Next, let's look at $H^T y_d$:

$$H^T y_d = \begin{bmatrix} H_M^T y_d \\ H_I^T y_d \\ H_L^T y_d \\ S^T y_d \end{bmatrix} \tag{5.15}$$

These submatrices can be updated in the following way:

$$\begin{aligned}
H_x^T y_d\left(C1_x(i)\right) &= H_x^T y_d\left(C1_x(i)\right) + y_d(i) \\
H_x^T y_d\left(C2_x(i)\right) &= H_x^T y_d\left(C2_x(i)\right) - y_d(i)
\end{aligned} \tag{5.16}$$

where x is M, I and L. Finally:

$$S^T y_d = S^T y_d + y_d(i) \tag{5.17}$$

The value of $S^T y_d$ can very quickly accumulate to a very large number. To avoid this, we can do a rough initial guess of the shift amount (β_{est}) and subtract that from each $y_d(i)$. In our least squares solution, we will then essentially estimate the deviation of the actual shift amount from this initial guess. So, instead of $y_d(i)$, $y'_d(i)$ can be used in all the previous equations where

$$y'_d(i) = \left(C2(i) - C1(i)\right) + \left(C1_{ADC}(i) - C2_{ADC}(i)\right)/g - \beta_{est} \tag{5.18}$$

It is also important to note that some of the other error sources, like gain error mismatch average etc., described in chapter 4, section 4.B also become part of the β that is estimated by the least squares solution. This is why one will get a more accurate estimation of the segment errors when the term β is included as an unknown in the equations, as a catch-all term for other sources of error.

In addition to the equations from the code pairs, we must also include boundary conditions discussed in chapter 3 in the least squares solution matrix. Choosing the end-point fit boundary conditions instead of the best-fit boundary conditions is actually advantageous because it leads to a reduction in the number of unknowns. The boundary conditions discussed in chapter 3, along with the INL at first and last code being set to 0 lead to the following equations:

$$\begin{aligned} &E_M(0) = E_I(0) = E_L(0) = 0 \\ &E_M\left(2^{n_M} - 1\right) + E_I\left(2^{n_I} - 1\right) + E_L\left(2^{n_L} - 1\right) = 0 \end{aligned} \tag{5.19}$$

This means that the corresponding rows and columns for $E_M(0), E_I(0), and\ \ E_L(0)$ can be entirely removed from the $H^T H$ matrix. The last code condition can also be easily translated to simple matrix updates. If these boundary conditions are not set, the matrix will actually be rank deficient. Another point to note is that we need to give more "weight" to the last code end-point fit equation

by multiplying it with a large number, say 2^n. Otherwise the estimated INLs will not be end-point fit and a fit will have to be done later.

It is very often the case that the magnitude of the LSB segment errors is much smaller than the MSB and ISB code errors. In such cases, the LSB errors can be ignored. This will reduce the sizes of the matrices. Another approximation which will significantly reduce memory requirements and computation time is to assume that most entries in the $H^T H$ submatrices, other than $H_M^T H_M, H_I^T H_I, H_L^T H_L, and\ S^T S,$ are very small or just 0. This approximation holds as long as the DAC is reasonably linear. Then, equation (5.9) simply becomes:

$$\begin{bmatrix} H_M^T y_d \\ H_I^T y_d \\ H_L^T y_d \end{bmatrix} = \begin{bmatrix} H_M^T H_M & 0 & 0 \\ 0 & H_I^T H_I & 0 \\ 0 & 0 & H_L^T H_L \end{bmatrix} \begin{bmatrix} E_M \\ E_I \\ E_L \end{bmatrix} \tag{5.20}$$

In the above equation, we will have to assume that the shift amount is known and include it in the $H_x^T y_d$ matrices. But the true average shift is actually unknown during data acquisition. Regardless, we use the rough initial guess for updating the matrices. We keep track of the shift calculated in each equation and at the end of the data acquisition, we can get the actual average shift amount. At the end, we update the $H_x^T y_d$ matrix values accounting for the difference between the initial guess and the estimated actual shift value. The unknowns can then simply be separately estimated by:

$$\hat{E}_x = \left(H_x{}^T H_x\right)^{-1} \left(H_x{}^T y_d\right) \tag{5.21}$$

The memory requirement will also significantly reduce since only the three submatrices will need to be stored. The computation will also be significantly faster because inverse of much smaller matrices will need to be computed.

The above described computations can either be performed by the processor, if it is available on-chip or it can be hardened onto the SoC. There are advantages and disadvantages to both methods which have been discussed in prior literature [2].

5.4 Case Study

To describe some other challenges involved in on-chip implementation of the uSMILE-ROME algorithm, it will be easier to take a specific example as a case study. We will take the example of a 14-bit sub-radix 2 DAC, modeled in Matlab with resistor mismatch sigma of ~0.6% and additive noise sigma of 0.3 DAC LSBs. This DAC's linearity will be tested using a 12-bit SAR ADC with ~+/1.5LSB INL and additive noise sigma of 0.2 ADC LSBs. A sub-radix 2 DAC is desirable for pre-distortion because its output has no large positive code jumps. Large negative code jumps are intentionally introduced by adding extra resistance in series with the 2R resistor in a traditional R-2R DAC. This ensures that even in the presence of resistor mismatches, the output voltage will not have any large positive jumps. To understand why this is desirable for calibration, the INL curve of an R-2R DAC which has a mismatch at the MSB bit, such that its 2R resistor is smaller than twice of the R resistor, is shown in Figure 5.1.

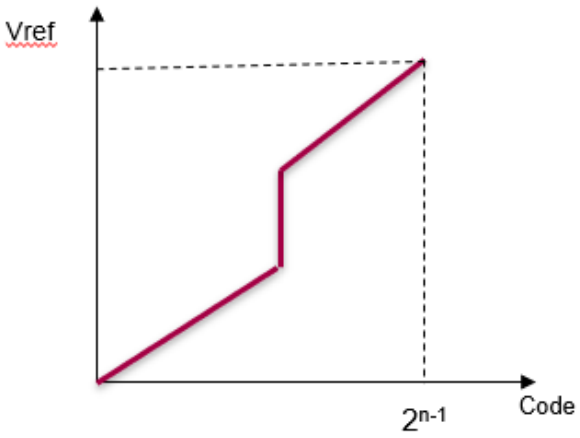


Figure 5.1. Output of R-2R DAC with resistor mismatch at MSB bit

This means that no input DAC code will give the voltages in the region of the jump. On the other hand, the output curve of a subradix 2 DAC will look like in Figure 5.2.

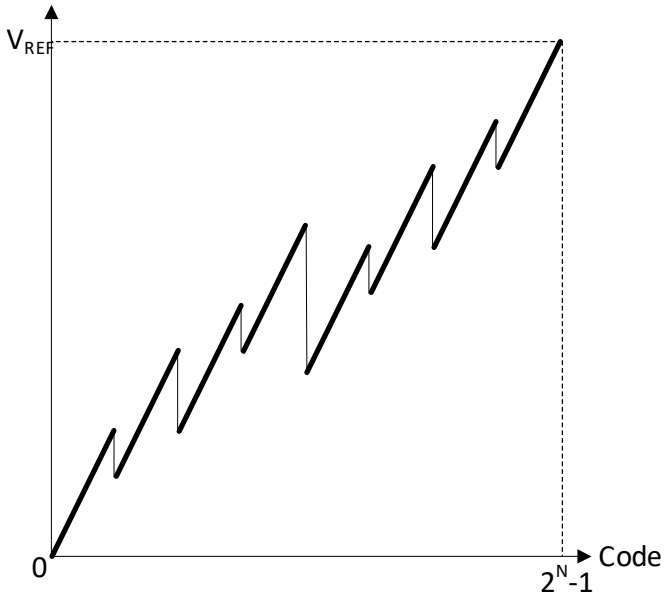


Figure 5.2. Output of subradix-2 DAC

This reduces the effective resolution slightly, but ensures that we will always be able to find some DAC code which will give an output close enough to any desired voltage. The shift required for the algorithm can be introduced using a shift resistor at the output of the subradix-2 DAC, as described in [4].

5.4.1 Buffer requirements for on-chip uSMILE-ROME

As discussed in chapter 4, to write the equations for uSMILE-ROME, we need to identify code pairs such that the output ADC codes are either equal or close to each other. In this case, since the resolution of the ADC (12-bit SAR ADC) is less than the resolution of the device under test (14-bit subradix-2 DAC), we can be reasonably sure that we will be able to find multiple DAC

codes from the shifted and non-shifted ramps which will give the same ADC output code. Let us now discuss how to obtain the codes pairs during the data acquisition process so that we can update the $H^T H$ and $H^T y_d$ matrices on the fly. This is not an issue in the USER-SMILE algorithm when testing ADCs because we fix the DAC code, and then get the ADC codes with and without shift for the same DAC code, which can immediately be used to update the matrices. For a DAC, this is not the case. This is understood better by observing Figure 5.3. Let's say that the approximate shift amount in units of DAC lsbs is C1. Then, the DAC codes for the non-shifted ramp which are less than C1 are useless. Similarly, codes greater than C2 in the shifted ramp cannot be used for the algorithm. In order to get relatively close ADC codes from the shifted and non-shifted ramps quickly, we will make use of the design target shift amount (say C1). Say the shift bit is 0 for the shifted-down ramp and 1 for the shifted-up ramp. The sequence of (DAC input code, shift-bit) should be:

$$(0,1)\rightarrow(C1,0)\rightarrow(1,1)\rightarrow(C1+1,0)\rightarrow(2,1)\rightarrow(C1+2,0)\rightarrow(3,1)\rightarrow(C1+3,0)\rightarrow\ldots$$

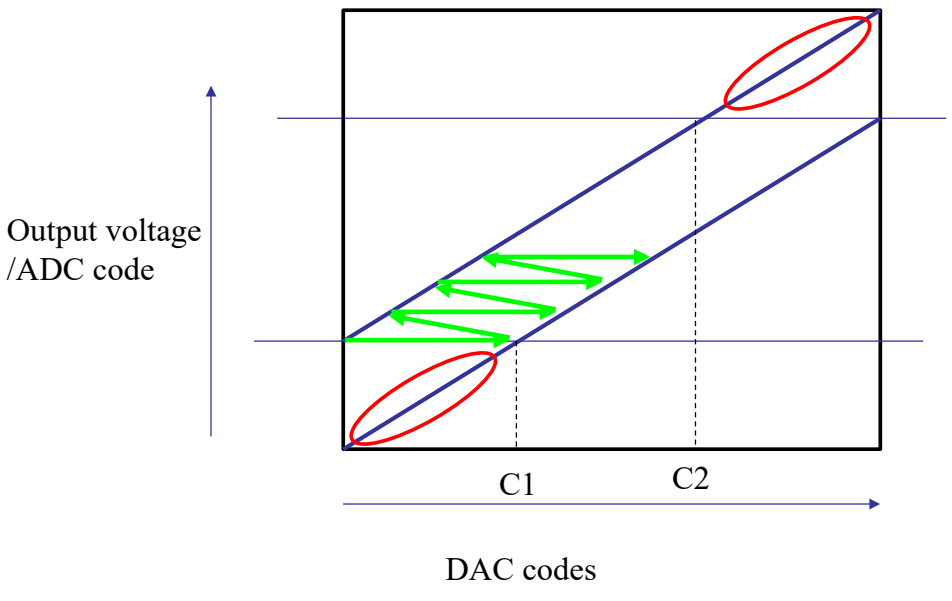


Figure 5.3. DAC code sequence for uSMILE-ROME

This will give us ADC output codes in the shifted and non-shifted ramps which are close to each other. Even if this sequence is followed, the only way to get the exact same ADC codes continuously is by storing the input DAC codes for a given ADC output code in a buffer. We need two buffers, one for the shifted ramp (BUF1), and one for the non-shifted ramp (BUF2). Each column of the buffer corresponds to an ADC code. The number of columns is determined by how much we expect the DAC output voltage to "jump around" in the worst case. For a subradix-2 DAC, the voltage will sometimes "jump down" by a large amount. The number of rows in the buffers are determined by how many DAC codes we expect will give the same output ADC code. Every time we get a certain ADC code, the corresponding input DAC code will be filled into a new row in the column corresponding to the ADC code. The various example states of the buffer are shown using a dummy example in Figure 5.4.

A state change happens every time shifted and non—shifted DAC codes are added to the buffers. In the figure, COL row denotes the buffer column numbers. The ADC row denotes the ADC code corresponding to that buffer column. To map an ADC code to a buffer column number, we calculate (ADC code) modulo (number of columns in the buffer). The entries which are filled in with DAC codes in the buffer are denoted by green. The cell which is currently being filled in the buffer in is denoted by red. Each buffer has "fill pointers" (FP1 and FP2) which are equal to the ADC codes/corresponding columns that are currently being filled in the respective buffers. FP is the minimum of FP1 and FP2. The TPB pointer ("To Be Processed") is always a fixed distance behind the FP pointer. In this example, it is always 8 ADC codes behind.

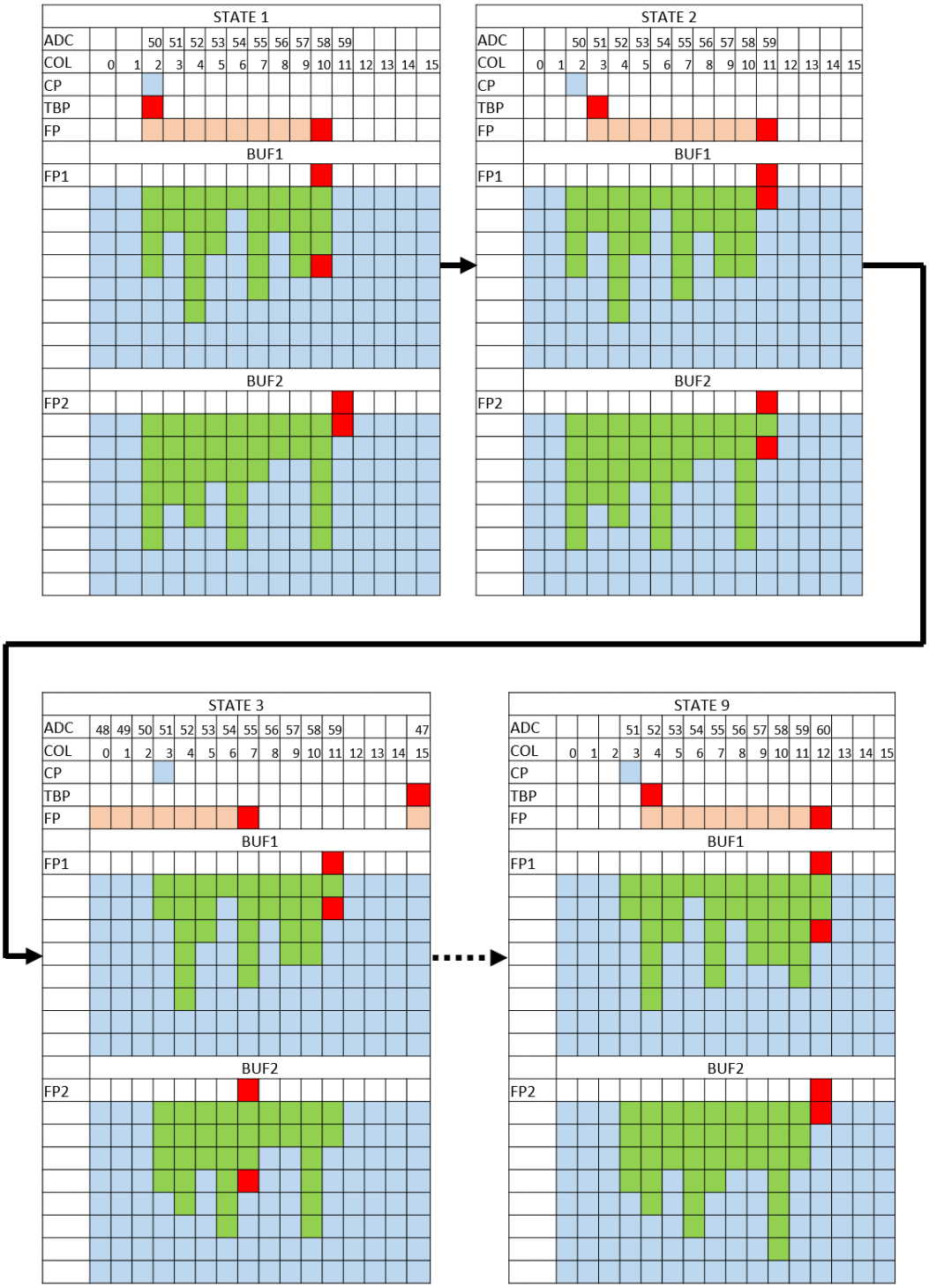


Figure 5.4. Example sequence of buffer states

There is a CP ("Currently Processing") pointer which points to the ADC code and the corresponding buffer column which will begin being processed to update the matrices the moment

CP falls behind TBP. As can be seen in the figure, from state 1 to state 2, FP has incremented by 1 (from 58 to 59) and so has TBP (from 50 to 51). CP, which is at 50, has now fallen behind TBP, so it starts processing ADC code 50/buffer column 2. In State 3, CP has finished processing ADC code 50 and is now ready to process code 51. But there has been a big negative jump in the output voltage of the non-shifted ramp. This leads to FP2 decrementing from 59 to 55 and so does FP. TBP goes from 51 to 47. CP will now wait till TBP exceeds CP, and only then begin processing and updating the matrices. As can be seen, this finally happens after 6 state changes, when TBP becomes 52 and CP can start processing ADC code 51/buffer column 3. Continuing on like this, the update of the $H^T H$ and $H^T y_d$will be complete just a few state cycles after the data acquisition itself.

5.4.2 DAC code pair selection criteria

Now that we know how to get the DAC input codes from the shifted and non-shifted ramps for a given ADC code, we have a few choices in selecting which code from buffer 1 is to be paired with which code from buffer 2. For a given column, say buffer 1 has x DAC codes and buffer 2 has y codes, with $y < x$. The codes are already in ascending order. One option is to select the first x codes from buffer 2 and ignore the rest (let's call this option 1). Another option is to select equally spaced out DAC codes from buffer 2 such that their number is equal to x (let's call ths option 2). Let us look at simulation results to see how they look. The 14-bit subradix 2 DAC is segmented as 5-5-4 and the shift amount is around 4 MSB segments. The simulation results from both these options actually look very similar so just one of them is shown in Figure 5.5.

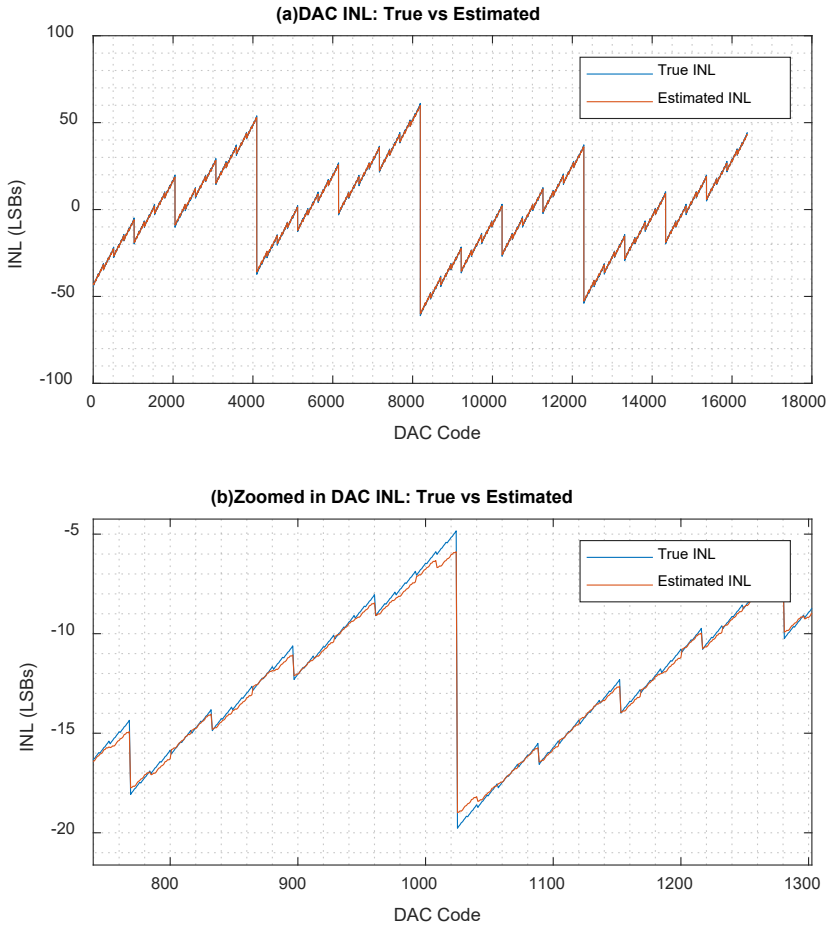


Figure 5.5. DAC INL estimation using uSMILE-ROME for a subradix-2 DAC: option 1/2

Figure 5.6. shows the INL estimation error which is around +/-1.5 LSBs. This is quite high and more than expected. The zoomed in plot in Figure 5.5 seems to indicate that there is a systematic error near major ISB/MSB code changes – the estimated INL just before the ISB/MSB code transition is lower than the true INL and the estimated INL just after the transition is higher than the true INL. This actually happens because of the subradix-2 architecture of the DAC. To help us understand better, let's look at example shifted and non-shifted ramps plotted in Figure 5.7.

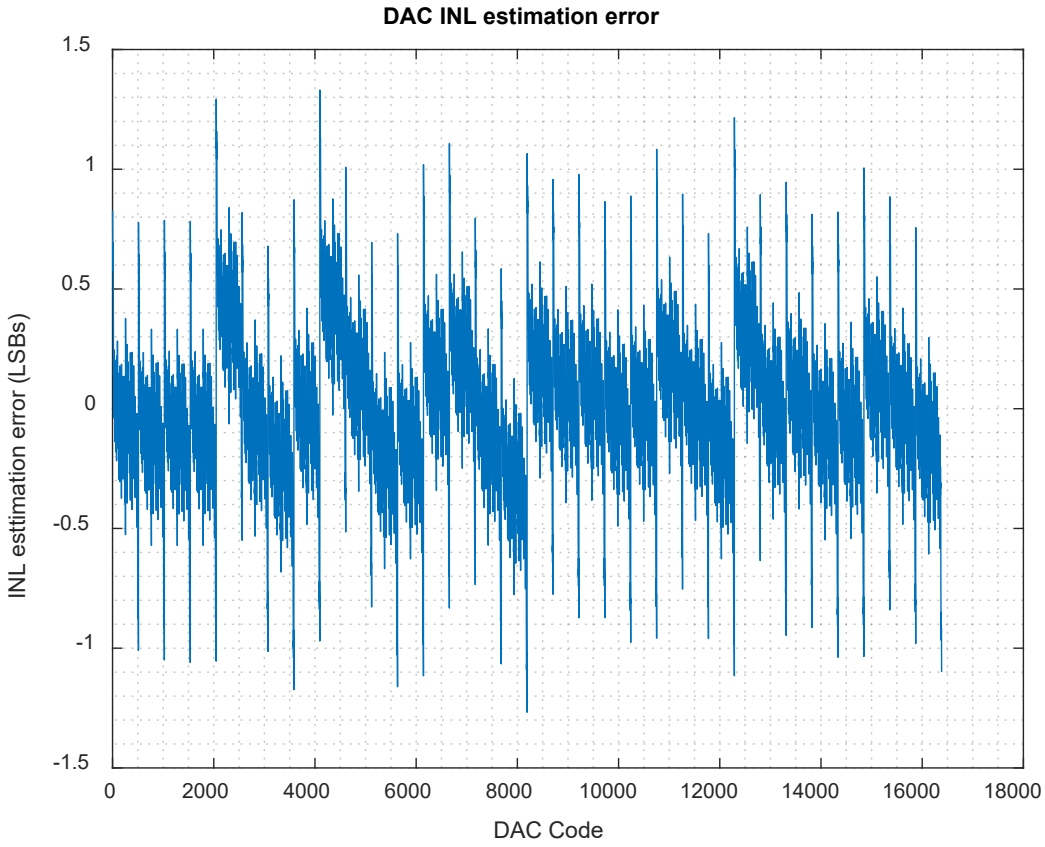


Figure 5.6. DAC INL estimation error for a subradix-2 DAC: option 1/2

Because of the large negative jumps in output voltage, DAC codes from one segment in the shifted get mapped to DAC codes from multiple segments from the non-shifted ramp, and vice-versa. Since the codes are in ascending order in the buffer, we are consistently mapping DAC codes in the non-shifted ramp just before the transition (shaded pink) to lower voltages/DAC codes from the shifted ramp. Similarly, we are consistently mapping DAC codes in the non-shifted ramp just after the transition (shaded green) to higher voltages/DAC codes from the shifted ramp. This leads to the INL estimation errors seen in Figure 5.6.

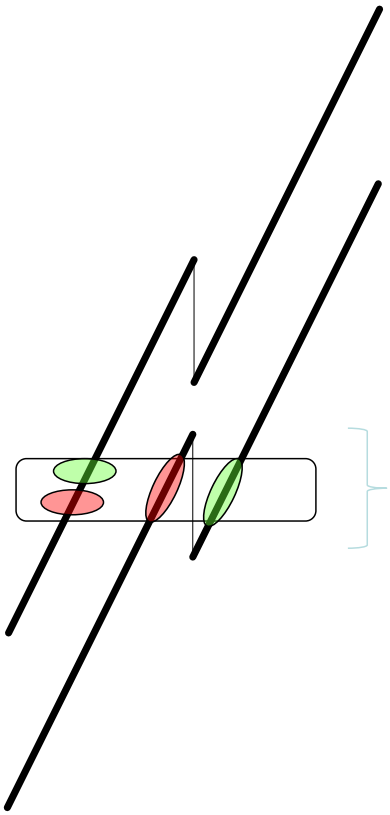


Figure 5.7. Output voltages of a subradix-2 DAC

Hence this is an artefact of the way we chose to pair the DAC codes and not the algorithm itself. To overcome this error, one option when codes from multiple segments are present is to always ensure that only DAC codes from only one segment are being mapped. But this could also lead to some kind of systematic errors. The best option (let's call it option 3) is to simply pair every single DAC code from buffer 1 to every single DAC code from buffer 2 i.e, we get $x \times y$ number of equation for every ADC code. This might also help in some sense to average out quantization errors. This is also easy to code instead of performing complex operations to mix and match. The simulation results using this option are presented in Figure 5.8 and Figure 5.9.

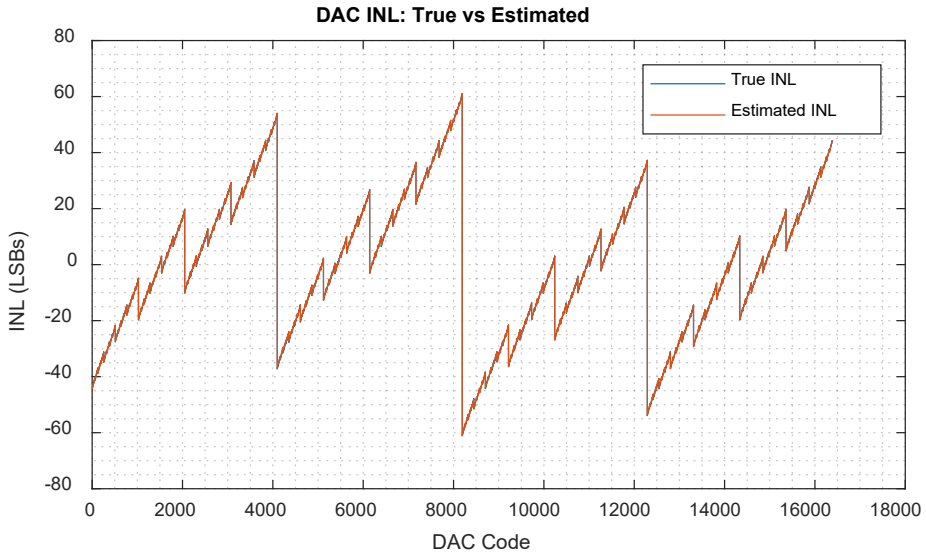


Figure 5.8. DAC INL estimation using uSMILE-ROME for a subradix-2 DAC: option 3

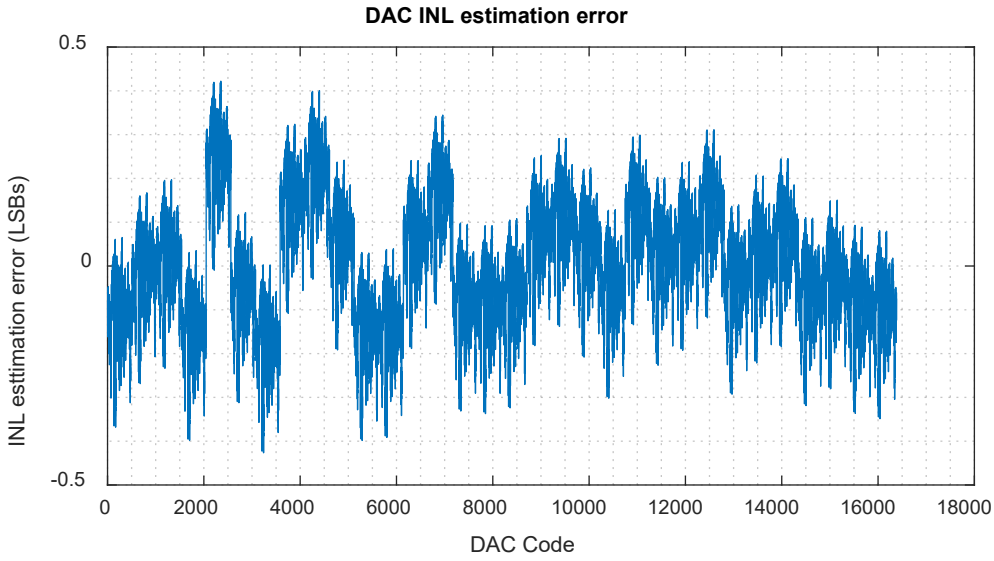


Figure 5.9. DAC INL estimation error for a subradix-2 DAC: option 3

The INL estimation error now seems to be bounded within +/-0.5 LSBs and the spikes at ISB and MSB code transitions have disappeared.

5.4.3 Further memory optimization

It was mentioned previously that the LSB segment errors are often small and can thus be ignored, which can help reduce the size of the matrices and thus reduce both memory size and computation time. This is not the case for a subradix-2 DAC because we are intentionally adding large negative DNL jumps. This is shown via simulations too. uSMILE-ROME was run on the same DAC as before with the LSB segment errors are ignored and the INL estimation error is plotted in Figure 5.10 (b). When compared to Figure 5.9, it can be seen that the LSB segment error contribution is quite significant. The DNL noise also seems to be high.

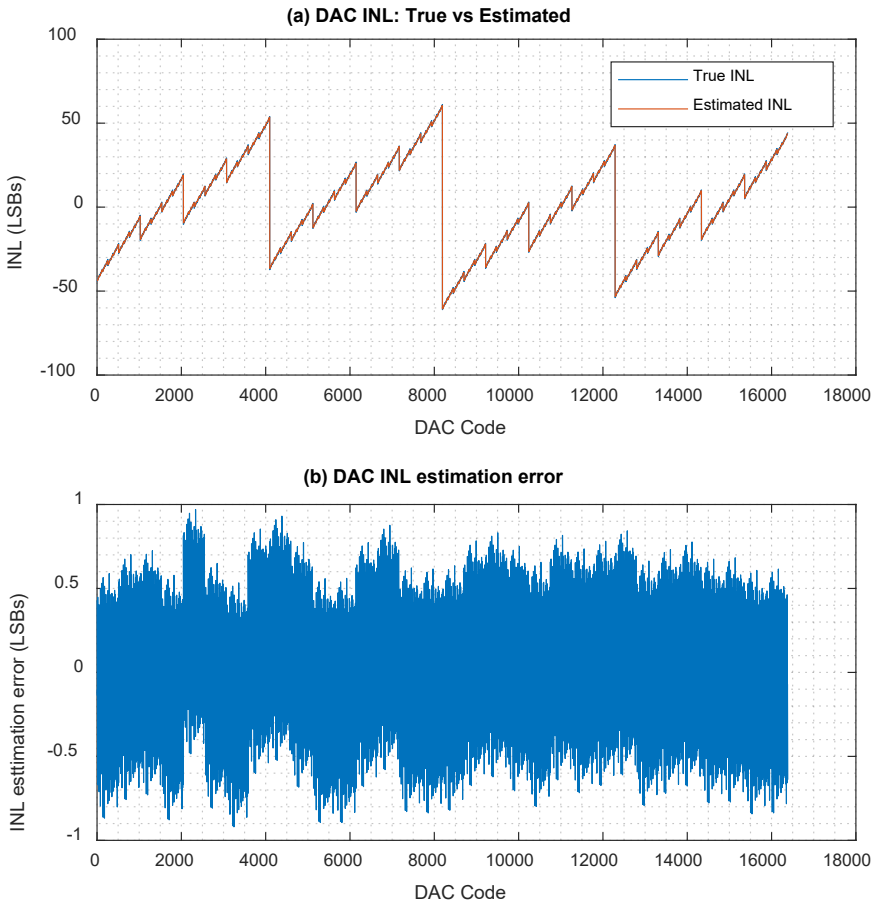


Figure 5.10. DAC INL estimation using uSMILE-ROME for a subradix-2 DAC with LSB segment errors ignored

While it is true that the intentionally added fixed mismatch is large, the contribution of errors due to random mismatches should actually still be negligible. We can actually tweak our segmented model a little to account for this. The extra resistance in the 2R branch basically leads to a gain error from the LSB portion of the DAC to the output. This gain error term can be accounted for in the segmented model in the following way:

$$INL(C) = E_M(C_M) + E_I(C_I) + g_L \times \left(C_L / 2^{n_L} \right) \tag{5.22}$$

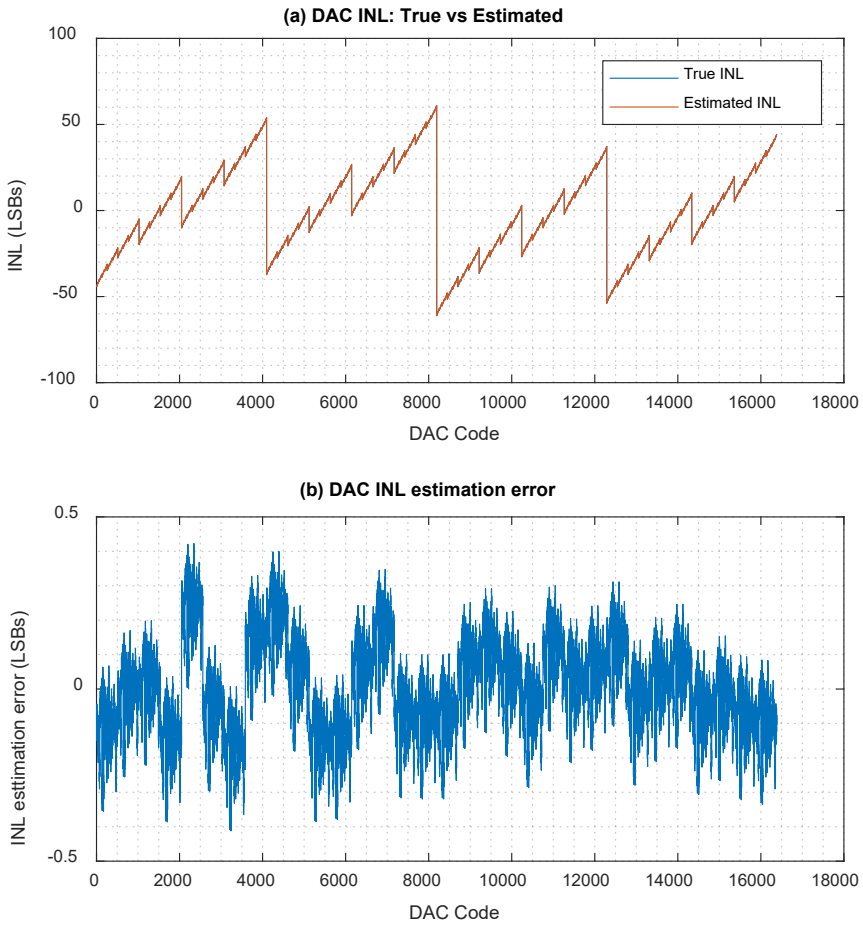


Figure 5.11. DAC INL estimation using uSMILE-ROME for a subradix-2 DAC with gain error term for LSB errors

where g_L is the gain error from the LSB DAC to the output because of the subradix-2 architecture. We now have $2^{n_M} - 1$ MSB segment errors, $2^{n_I} - 1$ ISB segment errors, 1 ISB segment gain error, and 1 shift variable. So, $K = 2^{n_M} + 2^{n_I}$. The simulation results with this new model for the subradix-2 DAC have been plotted in Figure 5.11. The estimated INL tracks the true INL very well and in fact, there seems to be hardly any difference between the plot in Figure 5.9 and the INL estimation error plot in Figure 5.11. This proves the validity of the modified segmented model.

5.4.4 Calibration using Pre-distortion

It has been shown that the static errors of a DAC can be calibrated using digital pre-distortion of the input codes[5]. For a desired output voltage C_{des} in units of ideal lsbs, one memory efficient option that has been presented previously is to approximate the pre-distorted DAC code in the following way:

$$C_{pd} = round\left(C_{des} - INL\left(C_{des}\right)\right) \tag{5.23}$$

Though time and memory efficient, this method will lead to large errors for the subradix-2 architecture because it ignores the INL difference between C_{pd} and C_{des}.

Of course, the method that will provide best performance is to do an exhaustive search through all DAC codes to find which one will give an output voltage that is closest to the desired voltage. If we want to be very precise, then we must account for the offset and gain error of the DAC too. Though well known, let us derive the exact relationship between the DAC input code and the output voltage here.

Say, we measured the output voltage of the DAC at code 0 and at all 1s code. Let's say these measurements are taken in volts.

$$lsb_i = V_{ref} / 2^n \tag{5.24}$$

The measured offset and gain error are:

$$\begin{aligned} off_{meas} &= \frac{V_{meas}(0)}{lsb_i} \\ g_{meas} &= \frac{\left[V_{meas}\left(2^n - 1\right) - V_{meas}\left(0\right)\right] - \left[V_{ref} - lsb_i\right]}{lsb_i} \end{aligned} \tag{5.25}$$

Let us define a gain correction term g_c as:

$$g_c = \frac{g_{meas}}{2^n - 1} \tag{5.26}$$

Note that if offset error is negative or gain error is positive, we will need to approximate based on extrapolation: off_{meas} = -(one less than the first DAC code which gives us non-zero output voltage). Similarly, g_{meas} = [(all 1's code)-(the first DAC code which gives us output voltage of $V_{ref} - lsb_i$)]- off_{meas}

Now, at any code k_{DAC}, the output voltage of the DAC, in volts, can be written as:

$$V_{out}\left(k_{DAC}\right) = V_{meas}\left(0\right) + \left(k_{DAC} + INL\left(k_{DAC}\right)\right) \times lsb_a \tag{5.27}$$

Where

$$lsb_a = \frac{V_{meas}\left(2^n - 1\right) - V_{meas}\left(0\right)}{2^n - 1} \tag{5.28}$$

For an ideal DAC with no offset, gain error or non-linearity, the desired output voltage at code k_{des} is

$$V_{des} = k_{des} \times lsb_i \tag{5.29}$$

We want to find which k_{DAC} will give us the same desired output voltage for a given k_{des}. So,

$$\begin{aligned} & k_{des} \times lsb_i = V_{meas}\left(0\right) + \left(k_{DAC} + INL\left(k_{DAC}\right)\right) \times lsb_a \\ \Rightarrow & k_{des} = \frac{V_{meas}\left(0\right)}{lsb_i} + \left(k_{DAC} + INL\left(k_{DAC}\right)\right) \times \frac{lsb_a}{lsb_i} \\ \Rightarrow & k_{des} = off_{meas} + \left(k_{DAC} + INL\left(k_{DAC}\right)\right) \times \frac{lsb_a}{lsb_i} \end{aligned} \tag{5.30}$$

Upon simplification, we get that the output voltage of the DAC, in units of ideal lsbs, can be written as:

$$k_{des} = off_{meas} + \left(k_{DAC} + INL\left(k_{DAC}\right)\right) \times \left(1 + g_c\right) \tag{5.31}$$

We will present two different methods for pre-distortion. Both of them demonstrate a tradeoff between time and memory. The first method is accurate, memory efficient but will require a per code computation to "search" for the code which will give the closest output to the desired voltage using (5.31). Now, we do not have to actually search through every single code for every single desired voltage level – this cannot be done on the fly. Instead, we will do a "local search" around k_{des}. For a reasonably linear or well modeled DAC, the region to search will be minimal. For the subradix-2 DAC, since the code jumps can be large, we will do a search within +/-128 codes. Further, since we assumed that there is only a gain error from the LSB DAC to the output voltage, our search can be further simplified.

$$\begin{aligned} k_{des} &= off_{meas} + \left(k_{MI} \times 2^{n_L} + k_L + E_M\left(k_M\right) + E_I\left(k_I\right) + g_L \times \frac{k_L}{2^{n_L}} \right) \times \left(1 + g_c\right) \\ \Rightarrow k_L &= \frac{\left(k_{des} - off_{meas}\right)/\left(1 + g_c\right) - \left(k_{MI} \times 2^{n_L} + E_M\left(k_M\right) + E_I\left(k_I\right)\right)}{\left(1 + g_L / 2^{n_L}\right)} \end{aligned} \tag{5.32}$$

Assuming 5-5-4 segmentation, we simply find the MISB code of the desired voltage, $k_{des,MI}$ and then sweep +/-128/16=+/8 MISB codes from $k_{des,MI}$ i.e. we sweep from $k_{des,MI} - 8$ to $k_{des,MI} + 8$. For each k_{MI}, we compute the value in equation (5.32). If it comes out to be <0 or >15, then we go to the next MISB code. If the value is >=0 and <=15, then we stop. Let's call the MISB code and the k_L value that we stopped at, as k'_{MI} and k'_L . Then, we can simply calculate our pre-distortion code as $k_{pd} = k'_{MI} \times 2^{n_L} + round(k'_L)$. The above equation can be further simplified if some of the terms are small and can be ignored. Note that all the INLs do not need to be stored in memory. The INL at any code can be calculated on the fly as long as the segment errors are

stored in memory. This local search and computation at MISB codes (~16 codes in our case study) can even be pipelined, in which case, we can calculate our pre-distortion code at the same rate as our input, with just a little latency. The results will not be any different than if we had carried out an exhaustive search which would have taken much more time.

The second method is extremely time efficient but takes up more memory. We simply create a lookup table based on equation (5.31). This lookup table can in fact be created while we are checking the INL values against the specification limits. As we loop through all 2^n codes for INL calculation, we fill up our look-up table, Vlookup, as follows:

$$\begin{aligned} k_{measured} &= off_{meas} + \left(k_{DAC} + INL\left(k_{DAC}\right)\right) \times \left(1 + g_c\right) \\ V_{lookup}\left(round\left(k_{measured}\right)\right) &= k_{DAC} \end{aligned} \tag{5.33}$$

We know that if the input is k_{DAC}, the output voltage is $k_{measured}$ in units of ideal lsb's. Hence, we simply store k_{DAC} in the memory location $round(k_{measured})$. When the user asks for a voltage k_{des}, we go to the memory location $round(k_{des})$in the lookup table and our pre-distorted code simply becomes $k_{pd} = V_{lookup}(round(k_{des}))$. The error will be within +/-0.5 LSBs for rows which are filled in this manner. For rows which are unfilled (filled with -1 by default, say), we actually do a loop through the lookup table to fill them up in the following manner:

$$V_{lookup}\left(k\right) = V_{lookup}\left(k-1\right) \quad if \; V_{lookup}\left(k\right) = -1 \tag{5.34}$$

With just two simple loops, we have a lookup table. For any voltage that the user desires, getting the pre-distorted code is a simple lookup. The drawback here of course is that this takes up 2^n memory locations. We can reduce the size of each entry by storing $k_{DAC} - round(k_{measured})$ in the lookup table instead of storing k_{DAC}. This is feasible in a production environment where the DAC needs to be used to generate a ramp or sine wave for testing other IPs and the memory is available for test and calibration.

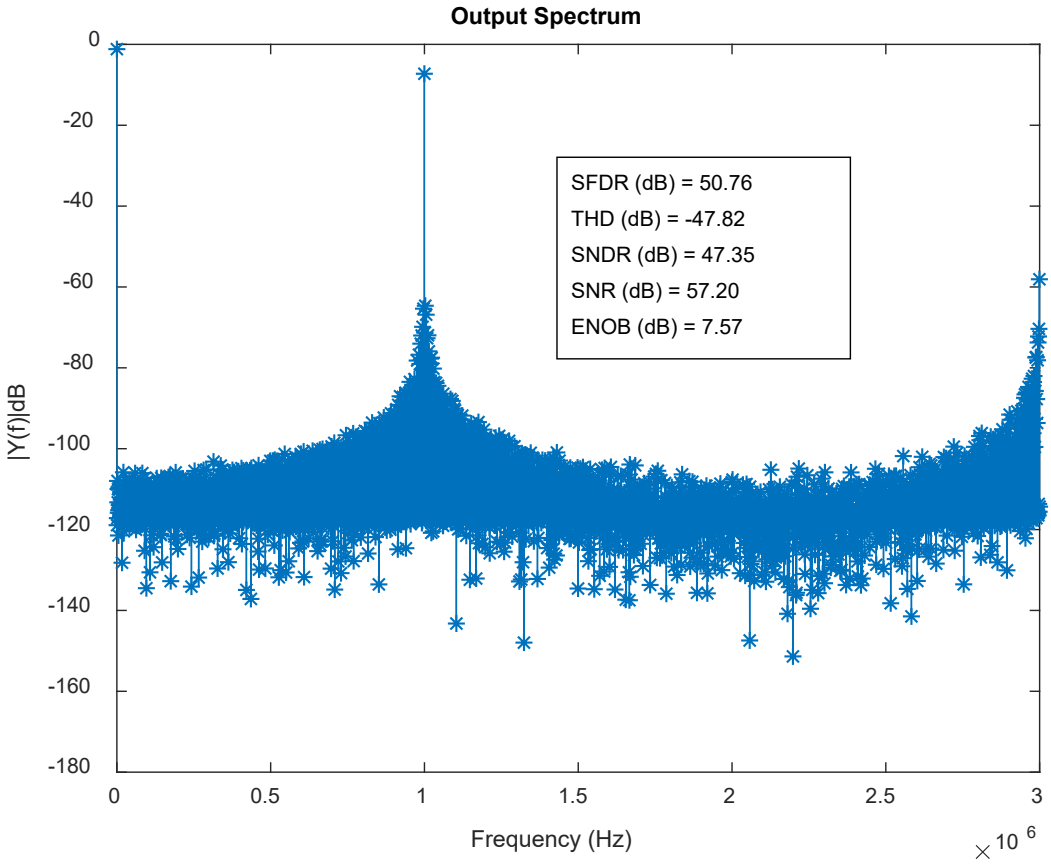


Figure 5.12. Output spectrum of DAC without calibration

The effectiveness of the described uSMILE-ROME based linearity measurement and pre-distortion methods will be demonstrated by generating a sine wave and looking at its spectral purity. We know that the INL of the DAC is extremely high. The spectrum of the sine wave generated by the DAC without any calibration is shown in Figure 5.12. The target frequency of the sine wave is 1MHz with the DAC code update rate as well as the sampling rate at 6MHz. As can be seen, the harmonic distortions are so significant that the ENOB of this 14-bit DAC is only 7.57.

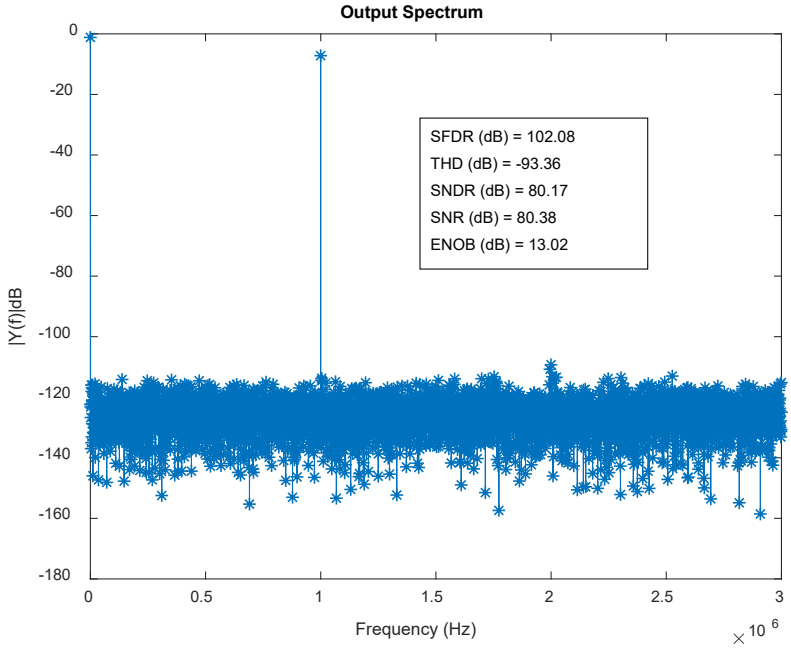


Figure 5.13. Output spectrum of DAC with uSMILE-ROME based digital pre-distortion using per-code search

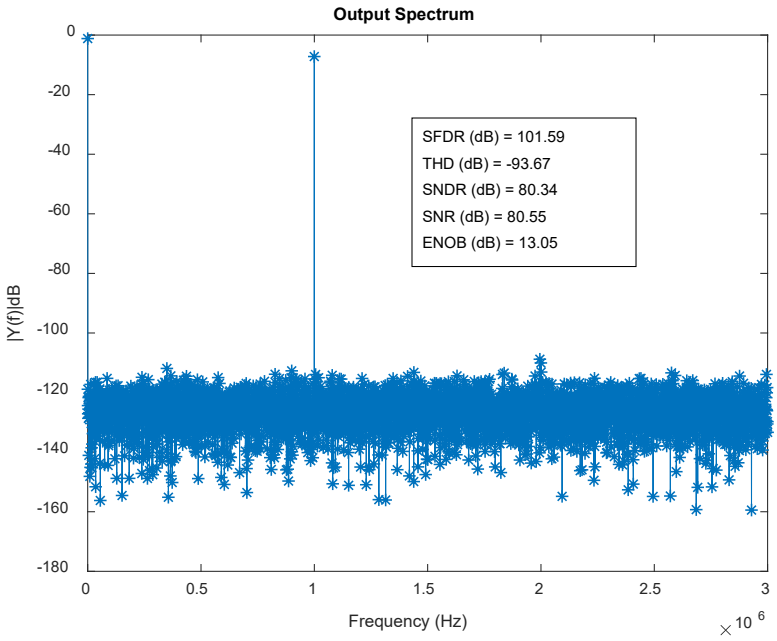


Figure 5.14. Output spectrum of DAC with uSMILE-ROME based digital pre-distortion using a fast lookup table

Next, we run memory optimized uSMILE-ROME to estimate the static nonlinearity of the DAC and then apply digital pre-distortion using Method 1 (exhaustive search). The output spectrum with this method is plotted in Figure 5.13. Next the same INL results are used and digital pre-distortion is done using Method 2 (fast lookup table). The results are plotted in Figure 5.14. The spectral performance has tremendously improved, with the ENOB now at 13.05. Note that 0.3 LSB of additive noise is also part of the DAC model. The SFDR of the output sine wave is also excellent and at the 100dB level. This demonstrates the effectiveness of the uSMILE-ROME algorithm as well as the digital pre-distortion methods to calibrate the DAC.

5.5 Conclusion

A complete on-chip DAC BIST solution based on uSMILE-ROME was detailed in this chapter. The computations involved in the algorithm were adapted so that it could be implemented on-chip in a highly memory optimized and time efficient manner. The intricate challenges of implementing the algorithm on-chip were discussed through an example case study of a subradix-2 DAC. Additionally, different methods for effective calibration of the DAC using digital pre-distortion were described. Every concept was verified via extensive simulations and the uSMILE-ROME based DAC Built-in Self-test and self-calibration methodology was shown to be highly effective on an example subradix-2 DAC. The work presented here can be leveraged to significantly cut down and test cost and test time. It also enhances in-field functional safety and reliability not only of the DAC itself, but also of other IPs when combined with the Concurrent sampling method described in chapter 8.

5.6 References

[1] N. Liu, S. K. Chaganti, Z. Liu, D. Chen, and A. Majumdar, "Concurrent Sampling with Local Digitization — An Alternative to Analog Test Bus," in *2018 IEEE International Symposium on Circuits and Systems (ISCAS)*, May 2018, pp. 1–5, doi: 10.1109/ISCAS.2018.8351555.

[2] X. Jin *et al.*, "An on-chip ADC BIST solution and the BIST enabled calibration scheme," in *2017 IEEE International Test Conference (ITC)*, Oct. 2017, pp. 1–10, doi: 10.1109/TEST.2017.8242032.

[3] T. Chen and D. Chen, "Built-in self-test and self-calibration for analog and mixed signal circuits," in *2019 IEEE International Test Conference (ITC)*, Nov. 2019, pp. 1–8, doi: 10.1109/ITC44170.2019.9000120.

[4] T. Chen *et al.*, "A Low-Cost On-Chip Built-In Self-Test Solution for ADC Linearity Test," *IEEE Transactions on Instrumentation and Measurement*, vol. 69, no. 6, pp. 3516–3526, Jun. 2020, doi: 10.1109/TIM.2019.2936716.

[5] Y. Zhuang, B. Magstadt, T. Chen, and D. Chen, "High-Purity Sine Wave Generation Using Nonlinear DAC With Predistortion Based on Low-Cost Accurate DAC-ADC Co-Testing," *IEEE Transactions on Instrumentation and Measurement*, vol. PP, no. 99, pp. 1–9, 2017, doi: 10.1109/TIM.2017.2769238.

CHAPTER 6. PURE SINE WAVE GENERATION USING DYNAMIC USMILE BASED DIGITAL PRE-DISTORTION FOR DACS TO ESTIMATE AND COMPENSATE FOR STATIC AND DYNAMIC ERRORS

Abstract

uSMILE has been shown to significantly reduce DAC static linearity test time and cost, but they can only be used to estimate and correct for static linearity errors. A pure sine wave can be generated using digital pre-distortion based on the static errors estimated using the algorithms, but beyond a certain sine wave frequency or DAC code update rate, dynamic errors start to become significant and even start dominating the effects of static non-linearity, leading to high levels of harmonic distortion. In this chapter, a dynamic version of uSMILE, which estimates dynamic segment errors based on the phase of the sine wave, will be introduced. The dynamic version of the algorithm can be used to generate digital pre-distortion codes for a sine wave for a given frequency, DAC code update rate and a given load. The effectiveness of the proposed method will be shown via simulations with different sine wave frequencies and DAC architectures. For cases when the dynamic errors are very high, the algorithm will be coupled with iterative pre-distortion to achieve the required sine wave purity level.

6.1 Introduction

In the previous chapters, uSMILE has been proven to be highly effective and efficient for estimating static linearity errors of DACs. The estimated static INLs have also been used to calibrate the DAC by means of digital pre-distortion of the input codes. By definition, the linearity test of DACs is pseudo-static by nature i.e., linearity testing is performed at low enough frequencies that the output voltage for an input DAC code has had sufficient time to settle. But when DACs are used to generate sine waves, it is not just the static linearity of the DAC that matters, but the dynamic errors too. For low frequencies, simply correcting for static errors is

sufficient. But for higher frequencies, settling errors become significant and lead to large harmonic distortions in the output spectrum.

Integrated systems often have both a DAC and an ADC available. A big advantage of this is that they can be used to co-test each other. As far as static performance is concerned, the DAC can be used to test the ADC using USER-SMILE[1], [2] and the ADC can be used to test the DAC using uSMILE-ROME[3], [4].

DACs are also often used as arbitrary waveform generators, and one of the most common applications of waveform generators is to generate a sine wave. For example, a pure sine wave is required for ADC spectral testing. In production testing, an external instrument on the tester is used to generate this sine wave. It is highly desirable to instead use an available on-chip DAC. The dynamic performance of these DACs is often lacking though, and the generated sine wave is not pure enough to perform spectral testing of the ADC.

The goal of developing a dynamic version of uSMILE is to somehow compensate for the dynamic errors in the DAC due to settling errors etc., in addition to the static errors. By using the dynamic version of the algorithm, a DAC's dynamic performance, which is originally poor, can be significantly improved to the point of being able to generate a high purity sine wave for a given load. Once the pre-distortion codes have been estimated for a given frequency and given load, the DAC can be used to generate a pure sine wave with the same frequency for similar loads.

6.2 Static uSMILE and pre-distortion review

The best way to explain the basic idea of dynamic uSMILE based pre-distortion is through examples and simulations. We start from the static version. Since the fundamentals of static uSMILE have already been described in previous chapters, we will not go into detail here, but simply re-state the fundamental equations for convenience. The segmented model for static

linearity assumes that the static nonlinearity or the INL at any code can be modeled as a combination of 2^{n_M} MSB, 2^{n_I} ISB and 2^{n_L} LSB code errors:

$$INL(k) = E_M(k_M) + E_I(k_I) + E_L(k_L) \tag{6.1}$$

Let's say that we are generating a sine wave using the DAC instead of a ramp, and the sine wave has M samples/number of DAC codes. We will assume that the DAC code update rate is the same as the sampling rate of the sine wave. Once the output voltage is sampled at each code, the following equation can be written for every DAC input code:

$$P(i) = INL(k(i)) + m(i) \quad i = 0,1,2,...,M-1 \tag{6.2}$$

Where $P(i)$ is the difference of the measured and expected voltages:

$$P(i) = V_{meas}(i) - V_{expected}(i) \tag{6.3}$$

$INL(k(i))$ is the INL at code $k(i)$, and $m(i)$ is the measurement error due to noise etc. at code $k(i)$. All the equations can be written in matrix form and all the segment errors can be solved for using least squares. The INL at every code can be computed from the segment errors.

In chapter 5, calibration of the DAC static INLs was performed using digital pre-distortion. A simple equation to approximate the pre-distortion DAC code can be written as [5]:

$$C_{pd}(i) = round\left(C_{des}(i) - INL\left(C_{des}(i)\right)\right) \tag{6.4}$$

where $C_{des}(i)$ is the desired output voltage , which, in this case, is the $i'th$ sine wave code, and $C_{pd}(i)$ is the pre-distorted code to be sent to the DAC.

To illustrate how to modify this to catch dynamic errors, we will use an example. Let's apply this to a DAC which has both static and dynamic errors. A 14-bit DAC is modeled in Matlab as an example, with the architecture being segmented as a 7-bit R-string DAC and a 7-bit R-2R DAC which interpolates between the voltages of the string DAC. The DAC is modeled with 1st order settling errors at the output of the string DAC. The unit resistance of the MSB DAC is taken as 1KOhm and the resistor mismatch sigma is 2%. The switch resistance is 100ohm. The unit

resistor of the LSB DAC is 200Ohm with a resistance mismatch sigma of 2.5%. Only first order RC settling is assumed, with R as 10KOhm and C as 0.02pF. Additive noise with a sigma of 0.4LSBs is added to the output, which means that even without any dynamic or static errors, the maximum achievable ENOB is ~13. A sine wave of frequency 500KHz (f_{in}) is generated with the DAC code update rate set and the output sampling rate (f_s) set at 2MHz. The spectrum of the sine wave without any pre-distortion is shown in Figure 6.1. The THD is -56.04dB, the SFDR is 58.91dB and the ENOB is 9.73.

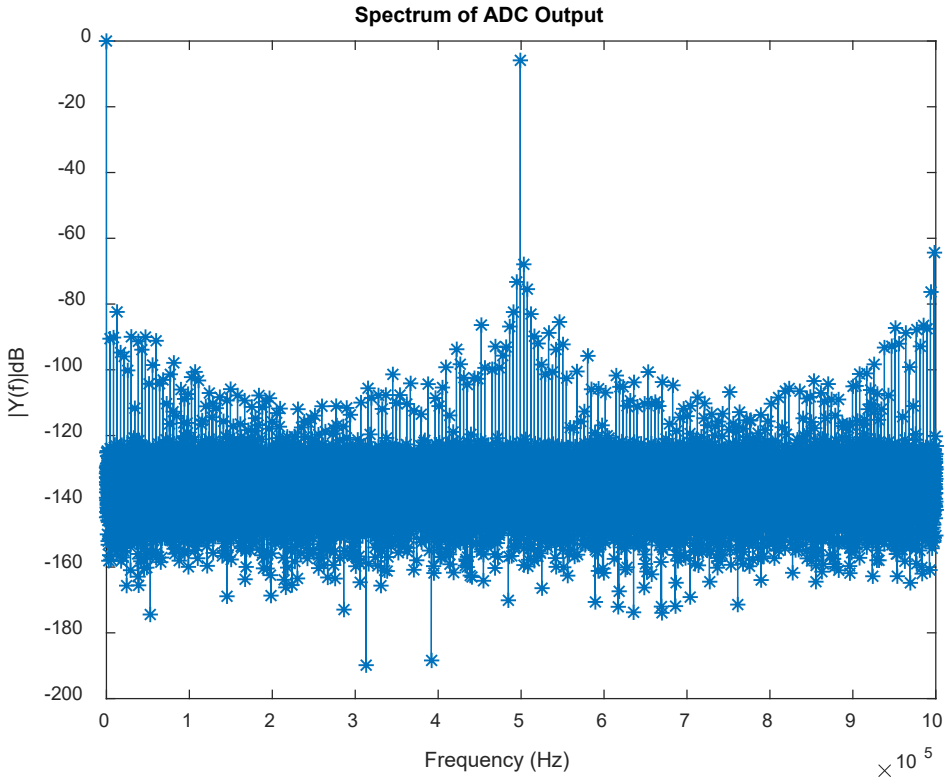


Figure 6.1. Output of 14-bit DAC without pre-distortion

Next, since the architecture is interpolated, we run uISMILE with 7-4-3 segmentation, obtain the pre-distortion codes and then get the output spectrum. This spectrum is shown in Figure 6.2.

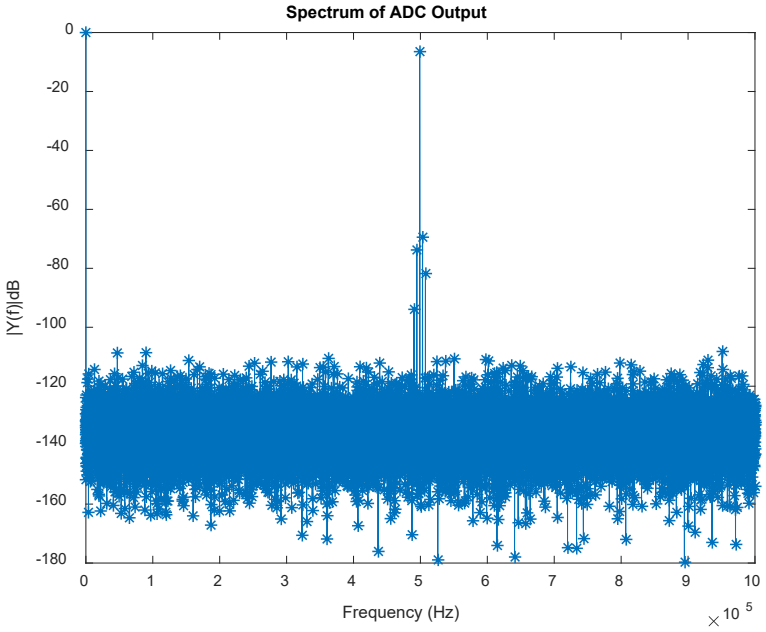


Figure 6.2. Output of 14-bit DAC with static uSMILE based pre-distortion

The Spectral performance has improved because the static errors have been calibrated out. The THD is now -61.44dB, the SFDR is 62.94dB and the ENOB is 9.9. This is of course not close to the maximum achievable ENOB of ~13 because the dynamic errors cause large harmonic distortions.

6.3 Dynamic uSMILE model

To develop the model for dynamic uSMILE, let's look at a sine wave curve plotted in Figure 6.3 with the sampling instances denoted by the black vertical lines.

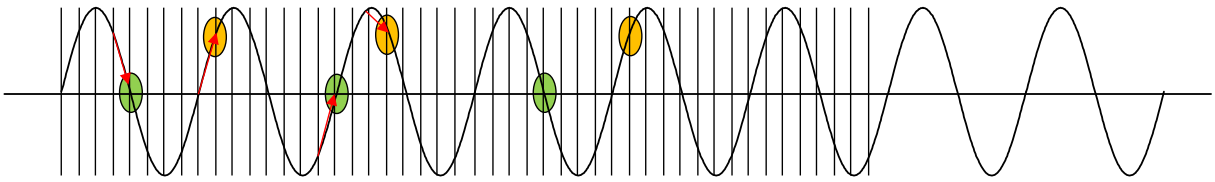


Figure 6.3. Sine wave with sampled voltages denoted by vertical lines

An oval on the curve denotes the MSB segment to which the input DAC code belongs to. First, we can assume that the dynamic errors for codes falling inside the same oval will have almost the same settling error. But this assumption is only valid if the codes from the same segment also belong to the same "phase" of the sine wave. For example, if we applied normal uSMILE, both the codes which are in the first two green ovals will belong to the same MSB segment and have the same E_M. But if we consider dynamic errors, the settling error at those 2 codes are actually different. As an example, for the code in the first green oval, if first order settling is assumed, the measured voltage will likely be higher than the desired voltage because it is on the falling edge, whereas, for the code in the second green oval, the measured voltage will likely be lower than the desired voltage because it is on the rising edge. We basically need to consider the error for each phase segment of the sine wave separately. This basically translates to treating the MSB segment errors for rising and falling edges differently. As far as dynamic errors are concerned, ISB and LSB segment errors do not really make sense, but we can still keep them, possibly to catch some of the static errors.

Therefore, for dynamic uSMILE, the number of unknowns is now twice that of static uSMILE. Typically, when doing calibration, we ignore the ISB and LSB segment error terms for static uSMILE, and we can do the same for dynamic uSMILE. Every time we write the equation for dynamic uSMILE, we need to determine whether the code belongs to the rising edge or the falling edge. If the sine wave equation is $Asin\left(\frac{2\pi iJ}{M}+\phi\right)$, then the sign of its slope is given by the sign of the $\cos\left(\frac{2\pi iJ}{M}+\phi\right)$ term. If the slope is positive, then we include the rising edge MSB segment error in the equation, and if it is negative, we include the falling edge MSB segment error in the equation. The rising and falling edge segment errors can still be solved for using least squares

to average out noise (or simply averaged if only MSB segment errors are considered), and pre-distortion codes can be calculated separately for rising and falling edge codes.

The output spectrum with this new dynamic uSMILE model based pre-distortion is plotted in Figure 6.4.

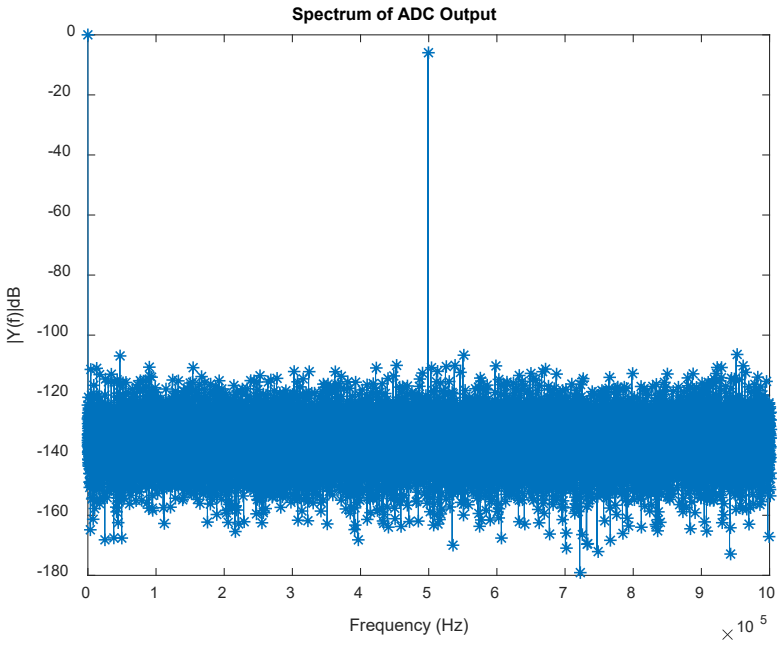
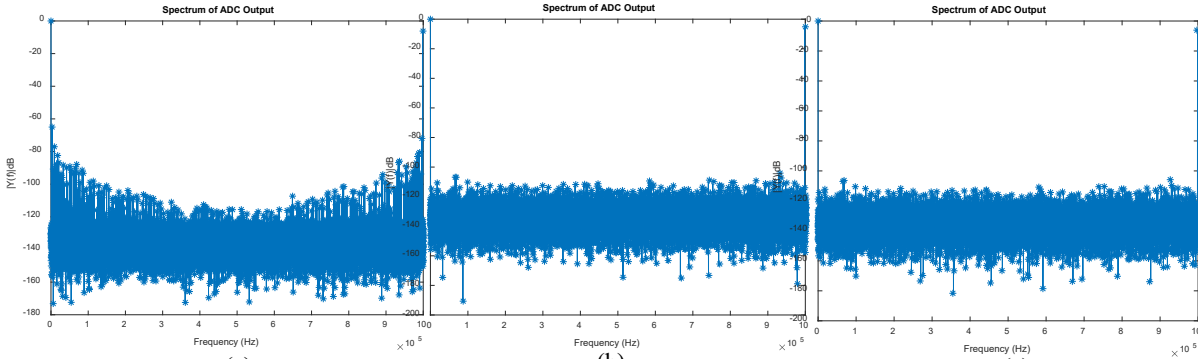


Figure 6.4. Output of 14-bit DAC with dynamic uSMILE based pre-distortion

Most of the large harmonic spurs have now been removed and the spectrum actually looks relatively clean. The THD is now -90.36, the SFDR is 101dB and the ENOB is 12.87. The ENOB is very close to the maximum achievable ENOB (~13). This means that the dynamic uSMILE estimates the dynamic errors with a very good accuracy.

The preceding simulations were performed with $f_s = 4f_{in}$. Let's see what results we get if we keep f_s the same but change f_{in} so that $f_s = 2f_{in}$ (Nyquist rate). Of course, we ensure that coherent sampling is maintained. All the output spectra, along with the spectral performance parameters are shown in Figure 6.5.

Info		ENOB	THD (dB)	SFDR (dB)
fs = 2MHz, fin = fs/2 = 1MHz, R =10k, Csw = 0.02pF	w/o PD	9.215067	-57.2996	58.89122
	PD with normal uISMILE	12.8907	-91.7931	99.50398
	PD with modified uISMILE	12.88096	-91.8113	99.70076



(a) (b) (c)

Figure 6.5. 14-bit DAC output spectrum with $f_s = 2f_{in}$
(a) Without pre-distortion
(b) With static or normal uSMILE based pre-distortion
(c) With dynamic or modified uSMILE based pre-distortion

The results are surprising – static uSMILE seems to perform as well as dynamic uSMILE. The reason for this can be understood by observing the sine wave curve sampled at near Nyquist rate plotted in Figure 6.6.

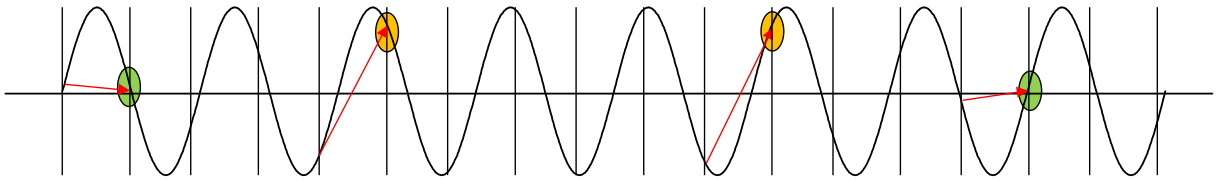


Figure 6.6. Sine wave sampled at near Nyquist rate

As can be seen in the figure, if the sampling is at near Nyquist rate, the voltage difference from the previous sample is nearly equal regardless of whether one is on the rising edge or the falling edge. This means that the rising edge and falling edge segment errors are nearly equal. This is why static uSMILE also results in good spectral performance. A sine wave sampled at $4f_{in}$ is shown in Figure 6.7.

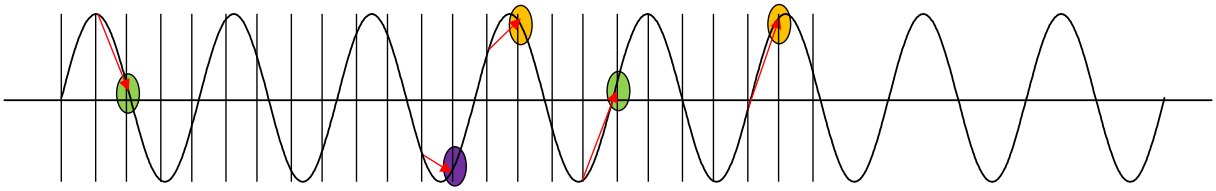


Figure 6.7. Sine wave sampled at $4f_{in}$

It is apparent that the rising and falling edge segment errors will be the most different for this case. So, static uSMILE will show the worst performance when $f_s = 4f_{in}$, whereas dynamic uSMILE should continue to good performance. When we observe the purple oval and the first yellow oval and the the green ovals, the red arrows seem to be mirror images of each other. This means that the voltage jumps for code k and code $2^n - k$ seem to be negative of each other. This would mean that there is a symmetry between the rising and falling edge errors and only one of them might be enough to estimate the other. This in fact hold true for any sampling frequency as long as we assume constant RC settling. If RC settling is not the same at all voltages, which is often the case (for example, when the output resistance is code dependent or the switch Ron is code dependent), then this symmetry will not exist and it is better to treat rising and falling edges separately.

6.4 Iterative dynamic uSMILE

When the sampling frequency is increased so much that the output voltage is being sampled many time constants before the voltage would have settled, even dynamic uSMILE's performance will begin to degrade. In order to get good spectral performance, we will have to do multiple iterations of dynamic uSMILE. The reason for this is best explained by using Figure 6.8.

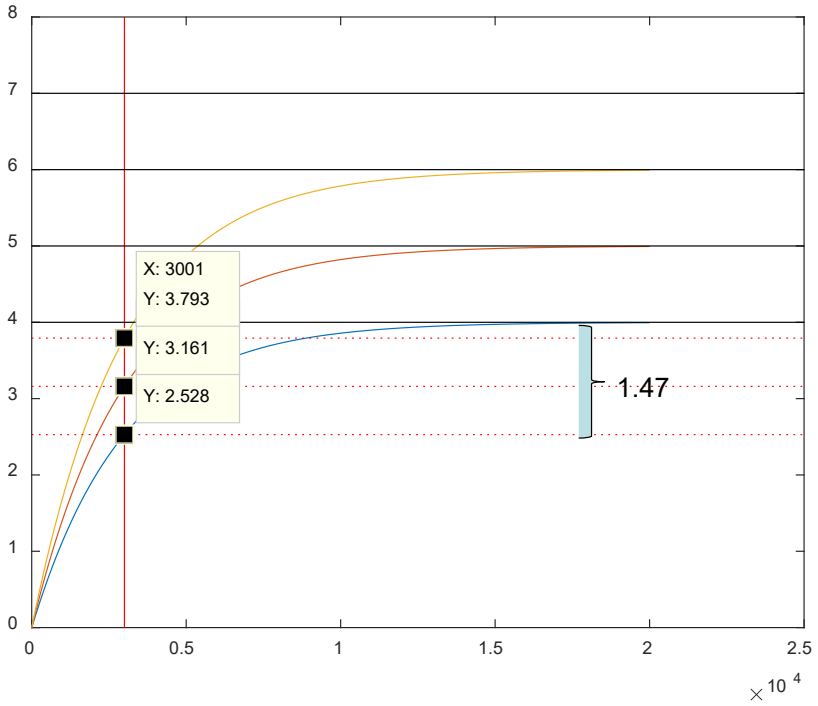


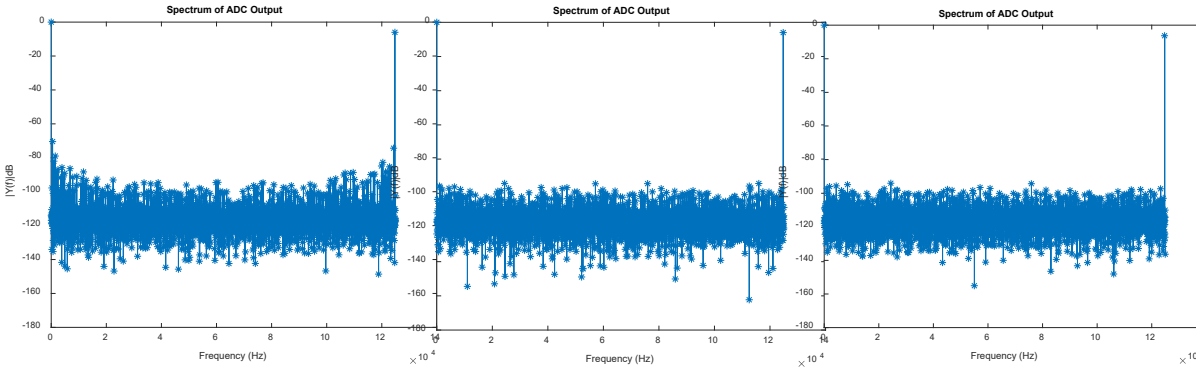
Figure 6.8. Need for iterative dynamic uSMILE

The y axis is the DAC output voltage in units of LSBs. The red vertical line denotes the sampling instant. We have increased the sampling frequency so much that the sampling now happens, say, at around one time constant. The desired input DAC code changes from 0 to 4, say. The blue line denotes the output voltage curve when the DAC's code changes from 0 to 4. The sampled voltage is 2.53. The dynamic error is equal to $2.53 - 4 = -1.47\ LSBs$. To compensate for this, the pre-distortion code calculated in the first iteration of dynamic uSMILE will be $Round\big(4 - (-1.47)\big) = 5$. The red line denotes the output voltage curve when the DAC's code changes from 0 to 5. The sampled voltage is 3.16. The dynamic error is equal to $3.16 - 4 = -0.84$, which is still not within the ± 0.5 error level which we desire. Hence, we will have to do a 2nd iteration of dynamic uSMILE in order to get the pre-distortion code of 6, the output voltage curve at which is shown in yellow.

A new 12-bit 12-bit DAC model is used to illustrate the need for dynamic uSMILE. A 12-bit DAC with 9-bit resistor string and 3-bit thermometer encoded architecture is modeled in Matlab. Because of the resistor string, the output resistance of the DAC is high, which leads to large settling times.

Figure 6.9 shows the simulation results at a relatively low frequency.

Info		ENOB	THD (dB)	SFDR (dB)
fs =	w/o PD	9.73	-61.23	64.33
0.25MHz,	PD with normal uSMILE	11.20	-78.39	88.15
fin = fs/2 =	PD with dynamic uSMILE	11.17	-78.52	87.93
0.124MHz,				
M = 2*2^12				



(c)

Figure 6.9. 12-bit DAC output spectrum at a low sampling frequency of $f_s = 250KHz \approx 2f_{in}$
(a) Without pre-distortion
(b) With static or normal uSMILE based pre-distortion
(c) With dynamic or modified uSMILE based pre-distortion

In this case, because of the low frequency (250KHz), and it being Nyquist rate sampling, both static and dynamic uSMILE give good performance.

Figure 6.10 shows the output spectra when the sampling frequency is high (2MHz).

Info		ENOB	THD (dB)	SFDR (dB)
fs = 2MHz,	w/o PD	4.44	-28.49	28.59
fin = fs/18 =	PD with normal uSMILE	4.45	-28.58	28.65
109.62KHz,	PD with dynamic uSMILE	5.88	-37.18	38.46
M = 2*2^12	MSB code PD with iterative dynamic uSMILE (iter 10)	10.97	-78.40	91.96

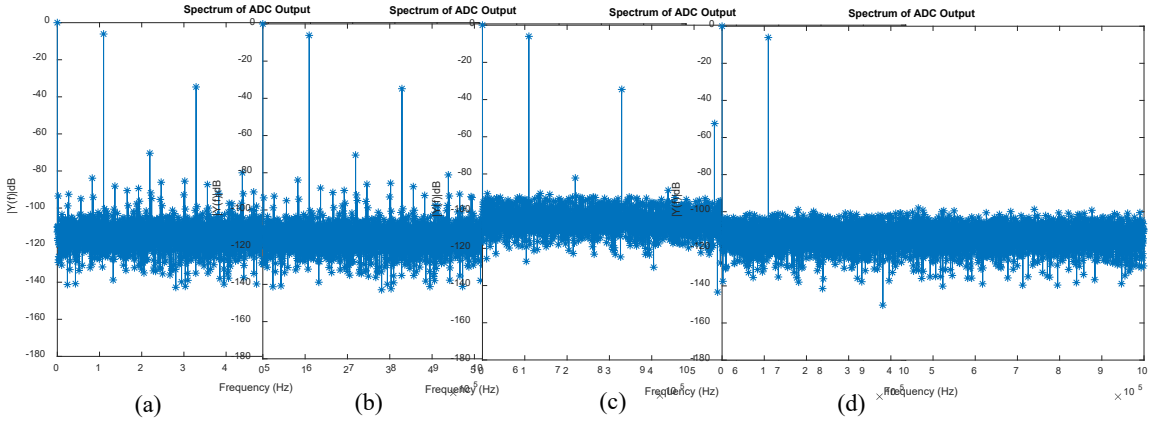


Figure 6.10. 12-bit DAC output spectrum at a high sampling frequency of $f_s = 2MHz \approx 18f_{in}$
(a) Without pre-distortion
(b) With static uSMILE based pre-distortion
(c) With dynamic uSMILE based pre-distortion
(d) 10th iteration of dynamic uSMILE based pre-distortion

In this case, it is only after 10 iterations that the harmonic distortions disappear, and we get an ENOB of 11.

We can take this to an extreme and increase the frequency extremely high. Figure 6.11 shows the output spectra with $fs = 20MHz \approx 60f_{in}$. When the sampling frequency is so high, it takes nearly 60 iterations for the ENOB to reach a respectable 10.34!

Now, obviously 60 iterations is not practical every time, but it should be noted that these pre-distortion codes need to be calculated only once for a given load. For the same device, the pre-distortion procedure will not have to be repeated. The baseline can be the pre-distortion codes obtained in the 60th iteration and then 1 or 2 iterations to account for chip to chip variation can be performed.

Info		ENOB	THD (dB)	SFDR (dB)
fs = 20MHz,	w/o PD	2.78	-18.48	18.78
fin = fs/60 =	PD with normal uSMILE	2.46	-16.57	17.29
334.47KHz,	PD with dynamic uSMILE	2.02	-13.93	15.26
M = 2*2^12	MSB code PD with iterative dynamic uSMILE (iter 70)	10.34	-77.95	83.51

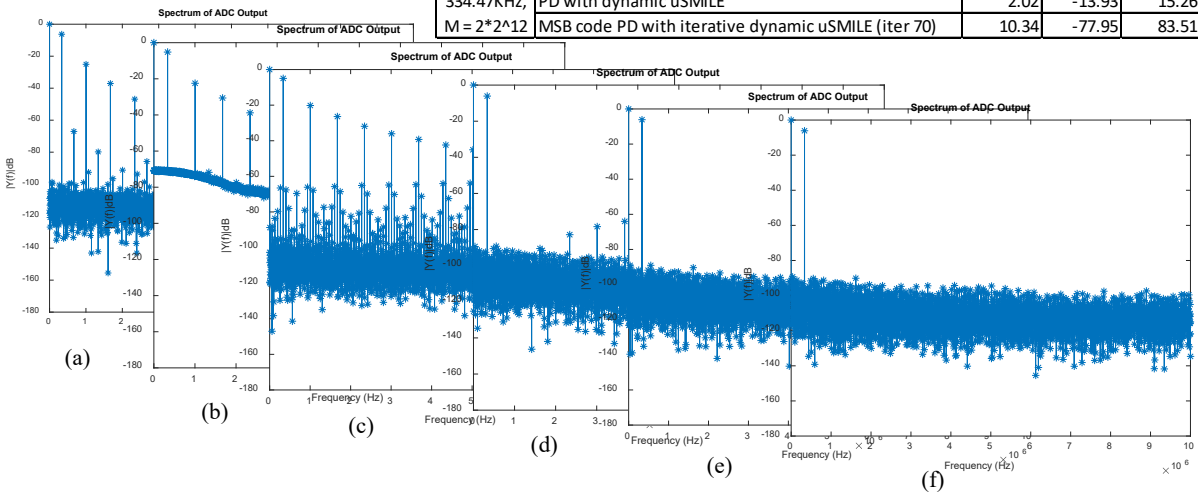


Figure 6.11. 12-bit DAC output spectrum at a very high sampling frequency of $f_s = 20MHz \approx 60f_{in}$
(a) Without pre-distortion
(b) With static or normal uSMILE based pre-distortion
(c) 1st iteration of dynamic uSMILE based pre-distortion
(d) 30th iteration of dynamic uSMILE based pre-distortion
(e) 60th iteration of dynamic uSMILE based pre-distortion
(f) 90th iteration of dynamic uSMILE based pre-distortion

It is interesting to see how the output voltage residue errors reduce over these many iterations. The voltage errors in units of DAC LSBs across all sine wave codes for different iterations are plotted in Figure 6.12.

It is also interesting to observe the plot of the final DAC input codes after, say the 60th iteration. This is plotted in Figure 6.13.

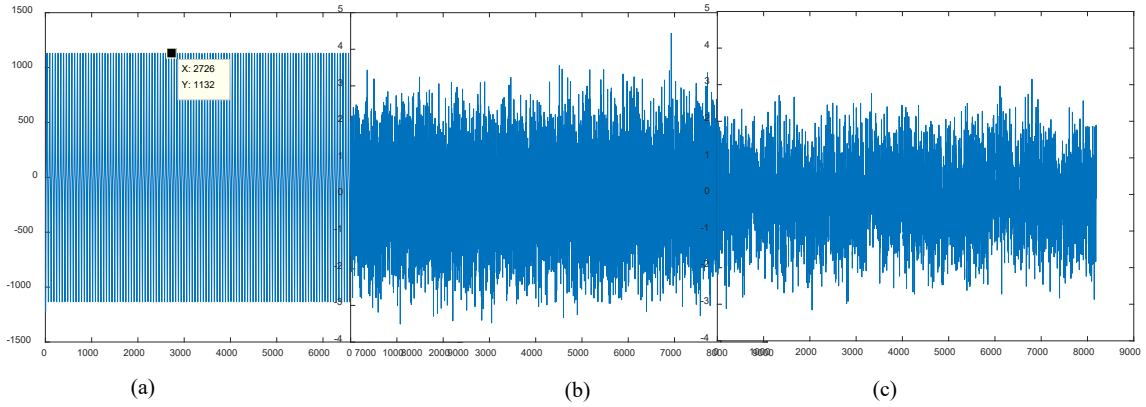


Figure 6.12. 12-bit DAC residue errors in units of LSBs across all sine wave codes with a very high sampling frequency of $f_s = 20MHz \approx 60f_{in}$ before
(a) 1st iteration of dynamic uSMILE based pre-distortion
(b) 30th iteration of dynamic uSMILE based pre-distortion
(c) 90th iteration of dynamic uSMILE based pre-distortion
The X axis is time, and the Y axis is the residue error in LSBs

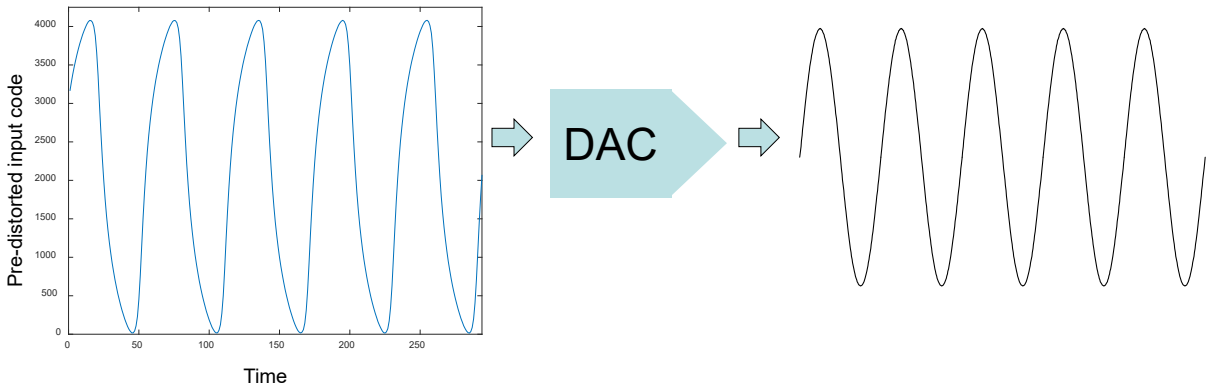


Figure 6.13. Final pre-distorted input codes for a 12-bit DAC to generate a pure sine wave at a very high sampling frequency of $f_s = 20MHz \approx 60f_{in}$

Figure 6.13 seems to hint that there might exist a simple transformation from the original sine wave input codes to the pre-distorted input codes, which could be derived using possibly just

the knowledge of the architecture, the resistor and capacitor values, and the required frequency and sample rate. Exploring this can be a future topic of research.

A dynamic version of the uSMILE-ROME algorithm may also be developed, as long as the measurement device, the ADC, only has static errors. If the ADC also has dynamic errors, then the combined effect of the dynamic errors of the DAC and the ADC will both be removed during the pre-distortion process such that the output spectrum will appear clean. If the goal was to test the dynamic performance of the ADC, then dynamic uSMILE-ROME does not seem to help. Regardless, the concept is promising and might find use for a niche application.

6.5 Conclusion

A dynamic version of the uSMILE algorithm was developed in this chapter, which can be used to estimate digital pre-distortion codes for a DAC such that both the static and dynamic errors are calibrated out. The calculated pre-distorted codes were used to generate a pure sine wave at a given frequency, sampling rate, and output load. Simulation results showed that the THD can be improved by around 30dB, and the ENOB by more than 3 bits for a 14-bit DAC for low sampling frequencies using dynamic uSMILE. For a 12-bit DAC, the THD improved by upto 50dB, and the ENOB by more than 7 bits for very high sampling frequencies using iterative dynamic uSMILE.

6.6 References

[1] T. Chen and D. Chen, “Ultrafast stimulus error removal algorithm for ADC linearity test,” in *2015 IEEE 33rd VLSI Test Symposium (VTS)*, Apr. 2015, pp. 1–5, doi: 10.1109/VTS.2015.7116249.

[2] X. Jin *et al.*, “An on-chip ADC BIST solution and the BIST enabled calibration scheme,” in *2017 IEEE International Test Conference (ITC)*, Oct. 2017, pp. 1–10, doi: 10.1109/TEST.2017.8242032.

[3] S. K. Chaganti, T. Chen, Y. Zhuang, and D. Chen, "Low-cost and accurate DAC linearity test with ultrafast segmented model identification of linearity errors and removal of measurement errors (uSMILE-ROME)," in *2018 IEEE International Instrumentation and Measurement Technology Conference (I2MTC)*, May 2018, pp. 1–6, doi: 10.1109/I2MTC.2018.8409877.

[4] S. K. Chaganti, A. Sheikh, S. Dubey, F. Ankapong, N. Agarwal, and D. Chen, "Fast and accurate linearity test for DACs with various architectures using segmented models," in *2018 IEEE International Test Conference (ITC)*, Oct. 2018, pp. 1–10, doi: 10.1109/TEST.2018.8624753.

[5] Y. Zhuang, B. Magstadt, T. Chen, and D. Chen, "High-Purity Sine Wave Generation Using Nonlinear DAC With Predistortion Based on Low-Cost Accurate DAC-ADC Co-Testing," *IEEE Transactions on Instrumentation and Measurement*, vol. PP, no. 99, pp. 1–9, 2017, doi: 10.1109/TIM.2017.2769238.

CHAPTER 7. A LOW-COST METHOD FOR SEPARATION OF ADC NOISE, APERTURE JITTER, AND CLOCK JITTER, AND ACCURATE ADC SPECTRAL TESTING WITHOUT REQUIRING COHERENT SAMPLING

Timing jitter is a crucial factor for high speed and high performance ADCs. Random clock jitter and the intrinsic aperture jitter of the ADC raise the noise floor and make it difficult to accurately estimate ADC specifications from the output spectrum. The stringent requirement of coherent sampling imposes further constraints on the test equipment. The proposed method significantly relaxes clock jitter and coherent sampling requirements by utilizing a dual channel test setup. The algorithm can efficiently separate and estimate noise, intrinsic ADC aperture jitter and random clock jitter, while allowing for arbitrary non-coherency in sampling. Simulation results of ADCs of different resolutions and sub-picosecond jitter levels validate the functionality and accuracy of the method.

7.1 Introduction

The rising demand for mixed signal devices is driving the need for high performance and high speed Analog-to-digital converters (ADCs). Accurate characterization and testing of these ADCs is a very challenging and costly task [1]-[3]. At high frequencies, timing jitter in the sampling process can be the ultimate limiting factor for ADC performance. Timing jitter can be defined as the relative time difference between the actual sampling instant and the ideal sampling instant, relative to the input signal. Its effect on the sampled voltage is shown in Figure 7.1.

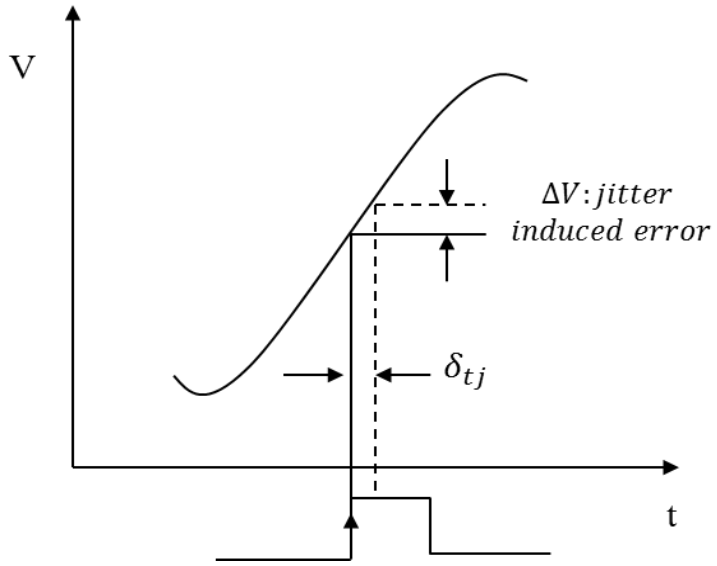


Figure 7.1. Jitter induced error in sampling

As the frequency of the input signal to the ADC increases, the resultant voltage error due to jitter increases proportionally. This increases the noise floor in the output spectrum of the ADC, thus making it very difficult to accurately estimate ADC specifications like the Signal-to-Noise Ratio (SNR).

Total timing jitter for an ADC can be separated into clock jitter and aperture jitter. The clock jitter along with the jitter introduced due to the board routing are external sources of jitter. On the other hand, aperture jitter is caused due to the clock distribution and the sample and hold circuits inside the ADC itself [4]. Aperture jitter and clock jitter have exactly the same effect on the output spectrum. Since the aperture jitter is intrinsic to the ADC and contributes to the noise floor, it must be included when we calculate the SNR of the ADC. On the other hand, the clock jitter which is from an external source must be removed from the ADC output. Hence, especially at frequencies where jitter becomes a significant contributing factor to the noise floor, it becomes

essential to separate clock jitter from the aperture jitter in order to accurately estimate ADC specifications.

In the current state of the art, there is a lack of low cost techniques to separate clock jitter from ADC noise and aperture jitter. Conventional methods for high performance ADC testing require an ultra-low jitter clock signal so that the noise contribution due to the clock jitter is insignificant and can be ignored [4]. The coherent sampling requirement further complicates this task. Generating and maintaining such a precise and pure clock signal is difficult and expensive.

Conventional methods apply two inputs with sufficiently separate frequencies to the ADC under test to calculate the total jitter [2]. The dual frequency method has a high ATE test cost due to the signal generator and synthesizers. For SoC test, the cost is high because for low frequency, on chip tests need a large die area for capacitors. A fast jitter and noise measurement method with one frequency test signal was proposed in [5]. By setting some harmonics of the ADC output to zero in the frequency domain, the residues of the ADC output were separated into two sets with different jitter powers. The RMS of jitter was obtained by processing the two sets of data. However, it requires knowledge of the harmonics, and non-harmonic spurs also affect the test result. Several other methods have been proposed to estimate the total jitter.

Xu et al. [6] proposed a low cost method to accurately estimate the total jitter by dividing the ADC output into 2 segments and subtracting one segment from the other to give the residue containing the total jitter and noise information. This method was improved to allow for non-coherent sampling in [7]. Both these methods essentially focus on estimating the total jitter (clock + aperture). They are targeted towards low speed applications where the clock jitter (in picosecond range) has a significant effect on the output, and the aperture jitter (in the hundreds of

femtoseconds range) has negligible effect on the noise floor. None of the methods mentioned above can separate ADC aperture jitter from the total jitter.

An analytic signal method was used by Yamaguchi et al. [8], [9] to measure the alias-free aperture jitter waveform. However, this method requires a signal generator with ultra-low phase noise to provide an almost jitter-free clock signal which, as mentioned previously, is expensive. Kim et al [3] separated aperture jitter, noise and clock jitter using a spectral loopback scheme with a DAC. However, this method requires the measurement of off-chip clock jitter using external equipment. This simply transfers the cost from the signal generator to the test equipment.

A low cost jitter separation and ADC spectral testing method, which neither requires an ultra-low jitter clock generator nor an expensive jitter measurement equipment, without requiring coherent sampling, is proposed in this chapter. It utilizes a dual channel test setup which has been successfully used in other works [10]–[12]. The same clock and input signals are applied to 2 ADCs. Two segments are carefully selected and subtracted to get the residue for each ADC. The difference of these two residues in turn gives us another residue equation. Processing of these three residues combined with ADC spectral testing algorithms which allow for non-coherency [13] gives us an estimate of the RMS of aperture jitter of each ADC, noise power and the RMS of the clock jitter. The proposed method is robust to non-coherent sampling, does not require prior knowledge of harmonics, and can tolerate significant clock jitter. Thus, by relaxing the stringent requirements on the sampling clock and the input signal, the test cost is considerably reduced.

7.2 ADC Spectral Testing and Aperture Jitter

7.2.1 Jitter modelling

Traditional ADC testing is performed with a pure sine wave as input, sampled coherently at frequencies above the Nyquist rate. The input signal to the ADC can be expressed as

$$V_{in}(t) = A\sin(2\pi ft + \varphi) \tag{7.1}$$

where A is the amplitude of the sine wave, f its frequency, and φ its initial phase. The total jitter is defined as the variation in the sampling instance from the ideal sampling instance. As mentioned previously, this jitter is the sum of the external clock jitter and the intrinsic ADC aperture jitter. Hence, the output of the ADC can be written as

$$x_n = V_{in}(nT_s + \delta c_n + \delta a_n) + hd_n + w_n \quad n = 0,1,...,M_t - 1 \tag{7.2}$$

where M_t is the total number of sampling points, T_s is the sampling period of the clock, δc_n is the clock jitter, δa_n is the aperture jitter, hd_n is the effect of the higher order distortions, and w_n is the summation of the additive and quantization noise. The clock and aperture jitter can be modelled as Gaussian distributed random variables, respectively: $\delta c_n \sim N(0,\sigma_c^2)$, $\delta a_n \sim N(0,\sigma_a^2)$. Similarly, w_n, which will be referred to generically as "noise" from here on, separate from the noise contribution due to jitter, is also modelled approximately as $w_n \sim N(0,\sigma_q^2)$

.

7.2.3 Effect of jitter and non-coherent sampling

Since the RMS of jitter is usually small, a Taylor series expansion of the output can be written:

$$x_n = A\sin(2\pi fnT_s + \varphi) + 2\pi fA\cos(2\pi fnT_s + \varphi)(\delta c_n + \delta a_n) + hd_n + w_n \tag{7.3}$$

It can be seen from equation (7.3) that the voltage error due to the jitter is proportional to the slope, and thus, the frequency of the input signal. So, for the same jitter, the error in the output is larger for higher input signal frequencies. If we want to measure the jitter, we give an input signal with frequency almost half of the clock sampling frequency (near Nyquist rate) so that this error is significant. The error due to jitter is seen as an increase in the noise floor in the output spectrum of the ADC. If the output is coherently sampled, i.e. $f / f_s = C / M_t$ where M_t and C are both integers and M_t is the total number of sampling points and C is the number of cycles of the input in the data length, then the fundamental power, total noise power etc. can be calculated directly from the DFT of the output.

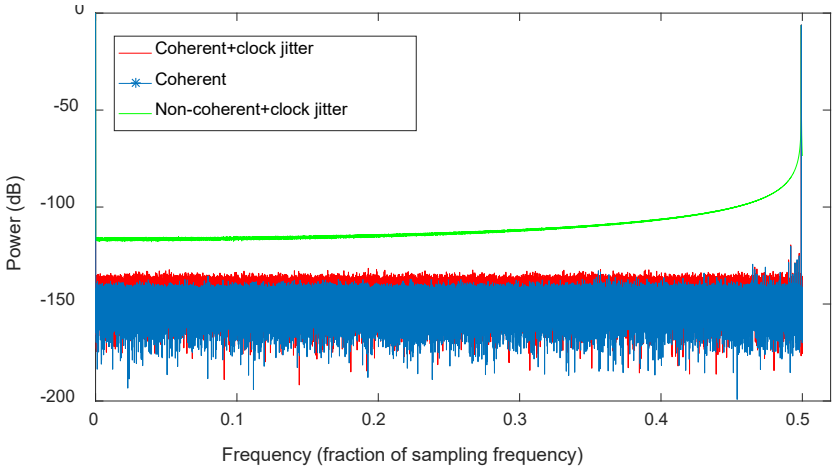


Figure 7.2. ADC spectrum
(a) Coherently sampled without jitter (blue)
(b) Coherently sampled with clock jitter (red)
(c) Non-coherently sampled with clock jitter (green)

In Figure 7.2, three plots have been superimposed on top of each other: 1) The ADC is coherently sampled and without clock jitter (blue), which gives us accurate ADC specifications 2) Coherently sampled and with clock jitter (red) and 3) Non-coherently sampled and with clock jitter (green). It can be clearly seen that in the case of coherent sampling, when there is clock jitter, the

noise floor is higher than when there is no clock jitter. Additionally, when there is non-coherent sampling, the spectral leakage of the fundamental covers the harmonic, noise and jitter information. Hence, we see that calculation of ADC specifications is a challenging task under the non-ideal conditions of clock jitter and non-coherent sampling.

7.3 Proposed method

A method which can accurately separate and estimate clock jitter, ADC aperture jitter and noise, without the need for coherent sampling, is proposed in this section. A dual channel ADC test setup is used, as shown in Figure 7.3.

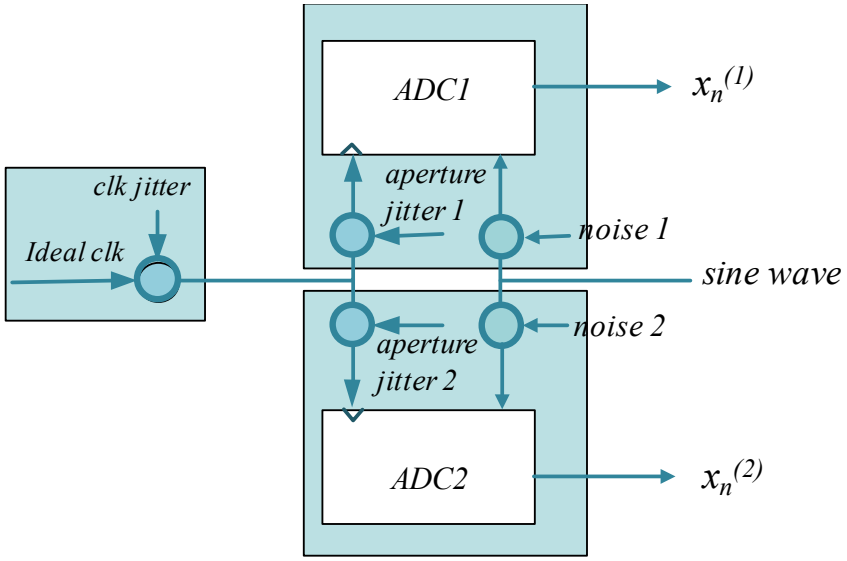


Figure 7.3. Dual ADC test setup model

The same clock and a pure sine wave input is applied to 2 ADCs. The test setup must be designed such that the clock and input signals to both the ADCs are the same. For example, when designing the board for testing 2 ADC chips, care must be taken such that the traces of the clock and input split just near the ADC pins to go to the 2 ADCs. This ensures that the jitter due to the

clock and routing is the same for both the ADCs, and also that there is minimal phase difference or jitter difference between the input signals to the 2 ADCs.

7.3.1 Segment selection

If we can ensure strict coherent sampling, then segment selection is simple – collect twice the usual number of samples for each ADC. The two consecutive segments are each coherently sampled. This is shown in Figure 7.4.

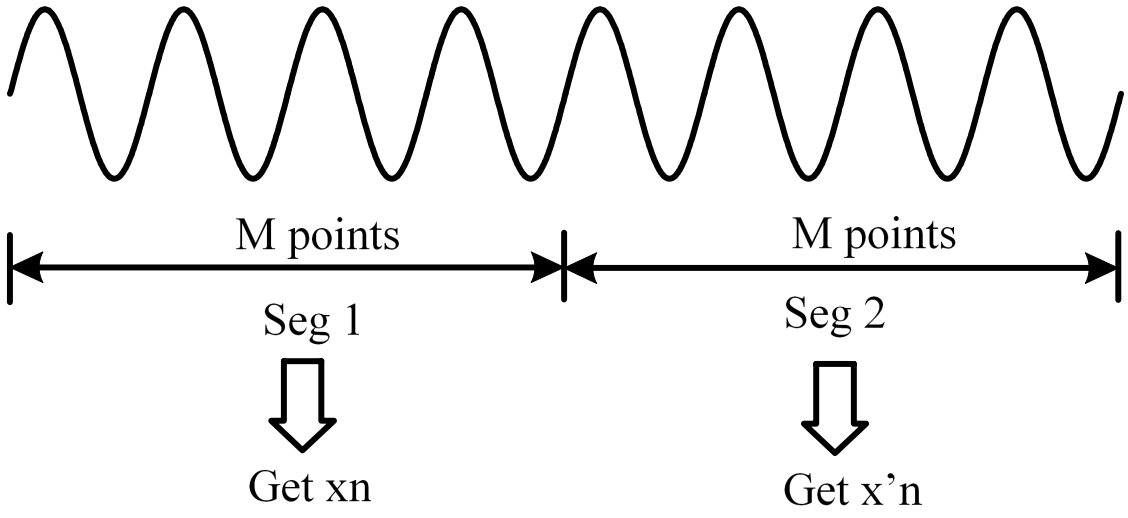


Figure 7.4. Segment selection strategy 1

If coherent sampling cannot be ensured, which is typically the case in real measurements, a more inventive strategy for segment selection can be employed. Collect at least 2 to 3 times the number of samples that you would normally collect for spectral testing of an N-bit ADC, and let this number be M_t. Out of these M_t points for each ADC, we need to obtain 2 segments with almost identical initial phase (for simplicity, say 0), and both segments having an almost integer number of periods.

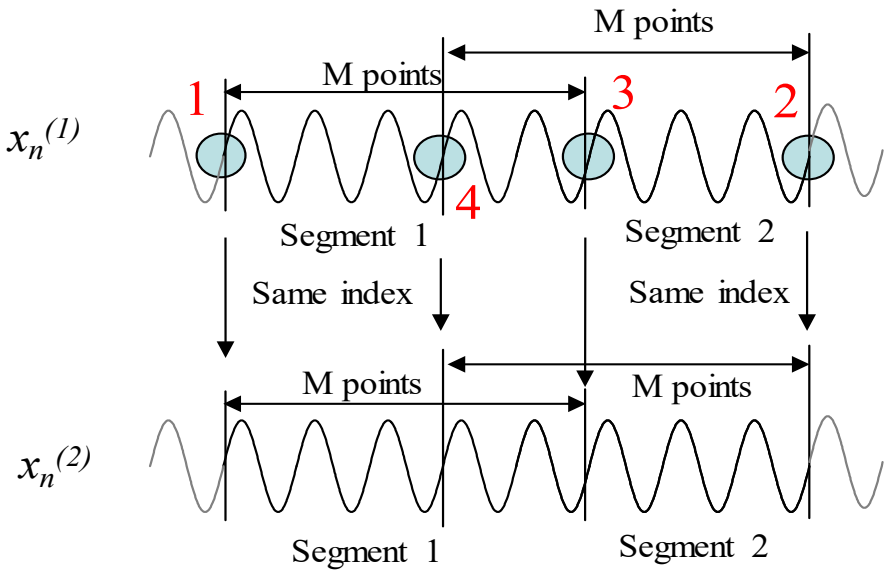


Figure 7.5. Segment selection strategy 2

To do this, for the first ADC, from the initial few points, we select those which are nearest the zero crossing (point 1 in Figure 7.5). This is where the first segment begins. Then we select points from the last few such that the phase is closely matched to 0. This gives us the end of the 2nd segment (point 2). We go backwards from here to again find the closest phase match to 0. This gives us the end of the 1st segment (point 3). Say there are M points now, in the 1st segment. We go backwards M points from point 2 to get the beginning of the 2nd segment (point 4). We use the exact same indices/points to get the 2 segments for the 2nd ADC output sequence. This is so that the clock jitter is the same, point-to-point, in the segments of the 2 ADCs. We have thus obtained two segments for each ADC, with M sample points each, with near identical initial phase and near integer number of periods.

7.3.2 Residue equations

The two segments of ADC1 can be written similar to equation (7.3):

$$x_n^{(1)} = A\sin(2\pi fnT_s + \varphi) + hd_n^{(1)} + 2\pi fA\cos(2\pi fnT_s + \varphi)(\delta c_n + \delta a_n^{(1)}) + w_n^{(1)} \quad (7.4)$$

$$x'^{(1)}_n = A\sin(2\pi fnT_s + \varphi') + hd'^{(1)}_n + 2\pi fA\cos(2\pi fnT_s + \varphi')(\delta c'_n + \delta a'^{(1)}_n) + w'^{(1)}_n \quad (7.5)$$

where $x_n^{(1)}$ is the 1st segment of ADC1 and $x'^{(1)}_n$ is the 2nd segment of ADC1. The clock jitter is normally distributed $\delta c_n, \delta c'_n \sim N(0,\sigma_c^2)$ and so is the aperture jitter $\delta a_n^{(1)}, \delta a'^{(1)}_n \sim N(0,\sigma_a^{2\,(1)})$ and the noise $w_n^{(1)}, w_n'^{(1)} \sim N(0,\sigma_q^{2\,(1)})$. The superscripts $^{(1)}$ and $^{(2)}$ will be used from here on to refer to the variables belonging to ADC1 and ADC2 respectively, with $'$ indicating the second segment. We subtract these two segments to get:

$$e_n^{(1)} = F_{res} + 2\pi fA\cos\left(2\pi fnT_s + \varphi\right)\{(\delta c_n - \delta c'_n) + (\delta a_n^{(1)} - \delta a'^{(1)}_n)\} + \left(w_n^{(1)} - w'^{(1)}_n\right) \tag{7.6}$$

Note that the fundamental and harmonics are almost completely removed since both the segments are outputs from the same excitation. Since the initial phases of the 2 segments (φ and φ') are not perfectly matched, there will be a small fundamental component (F_{res}) left in $e_n^{(1)}$. We will now show that this fundamental component has been significantly reduced. From equations (7.4) and (7.5), we know that F_{res} can be written as:

$$\begin{aligned} F_{res} &= A\left(\sin(2\pi fnT_s + \varphi) - \sin(2\pi fnT_s + \varphi')\right) \\ &= 2A\sin\left(\frac{\varphi - \varphi'}{2}\right)\cos\left(2\pi fnT_s + \frac{\varphi + \varphi'}{2}\right) \end{aligned} \tag{7.7}$$

If the phase difference between the two segments is small, then,

$$F_{res} \approx A(\varphi - \varphi')\cos\left(2\pi fnT_s + \frac{\varphi + \varphi'}{2}\right) \tag{7.8}$$

This implies that the amplitude of the fundamental component in $e_n^{(1)}$ is approximately equal to the amplitude of the original sine wave multiplied by the phase difference $(\varphi - \varphi')$. For example, if the phase difference between the two segments is 0.01%, then the peak of the fundamental gets reduced by 80dB in $e_n^{(1)}$. This by itself is very small. This peak can easily be identified in the spectrum of $e_n^{(1)}$ and removed. What about the spectral leakage of this fundamental component due to non-coherent sampling? Let us try to get a rough upper bound for the power that is leaked

to the other bins, compared to the power of the fundamental, for a sine wave that is non-coherently sampled. Let's say that the sine wave is sampled at almost Nyquist rate, i.e. the input signal frequency is set to little less than half of the sampling frequency, such that $f / f_s = C / M$ where M is the number of samples and $C = J_{int} + J_{frac}$. Here, J_{int} is an integer, and J_{frac} is the fractional part. $J_{frac} \neq 0$ means that the input is non-coherently sampled.

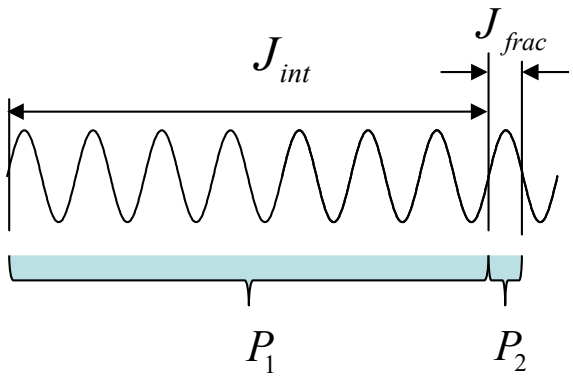


Figure 7.6. Estimation of upper bound of leakage power due to non-coherent sampling

In Figure 7.6, say region 1 constitutes an integer number of cycles of the sine wave and region 2 is the excess fractional part that is sampled due to non-coherent sampling. Since the total "time" is the same, the ratio of the powers is equal to the ratio of the energies, i.e., the areas of the squares of the regions of the sine wave:

$$\begin{aligned}\frac{P_2}{P_1} &= \frac{\int_0^{2\pi J_{frac}} (A\sin\theta)^2 d\theta}{\frac{A^2}{2}\times 2\pi J_{int}} \\ &= \frac{2\pi J_{frac} - \frac{1}{2}\sin\left(4\pi J_{frac}\right)}{2\pi J_{int}} \\ &= \frac{J_{frac}}{J_{int}} - \frac{\sin\left(4\pi J_{frac}\right)}{4\pi J_{int}}\end{aligned} \tag{7.9}$$

Since we are only interested in a rough upper bound:

$$\frac{P_2}{P_1} \le \frac{J_{frac}}{J_{int}} + \left| \frac{\sin\left(4\pi J_{frac}\right)}{4\pi J_{int}} \right| \le \frac{J_{frac}}{J_{int}} + \frac{4\pi J_{frac}}{4\pi J_{int}}$$
$$\Rightarrow \frac{P_2}{P_1} \le \frac{2J_{frac}}{J_{int}} \tag{7.10}$$

The maximum value of J_{frac} is 0.5. So we get $P_2/P_1 \le 1/J_{int}$. Say the number of samples $M = 2^{14}$, then for near Nyquist rate, $J_{int} \approx 2^{13}$, which means that P_2 will be around 78dB less than P_1. We already estimated that the fundamental component gets reduced by around 80dB if the phase difference between the two segments is 0.01%. The leakage power of the reduced fundamental component is a further 78dB below this, which means it is less than around -160dB! This implies that for a typical case, the leakage power of the fundamental component in $e_n^{(1)}$ due to non-coherent sampling is well below the noise floor. Hence, F_{res} can easily be identified from the DFT of $e_n^{(1)}$ as a single peak, and removed in the time domain to get "Residue1":

$$er_n^{(1)} = 2\pi fA\cos\left(2\pi fnT_s + \varphi\right)\{(\delta c_n - \delta c'_n) + (\delta a_n^{(1)} - \delta a'^{(1)}_n)\} + \left(w_n^{(1)} - w'^{(1)}_n\right) \tag{7.11}$$

We can also obtain the phase information from $x_n^{(1)}$ using the FIRE algorithm [13]. Next, as shown in [6], since there are nearly an integer number of periods in each segment, 2 equations can be obtained from this Residue1:

$$\sum er_n^{2(1)} \approx 4M\left(\pi fA\right)^2\left(\sigma_a^{2(1)} + \sigma_c^2\right) + 2M\sigma_q^{2(1)} \tag{7.12}$$

$$\sum\left(er_n^{(1)}\cos\left(2\pi fnT_s + \varphi\right)\right)^2 \approx 3M\left(\pi fA\right)^2\left(\sigma_a^{2(1)} + \sigma_c^2\right) + M\sigma_q^{2(1)} \tag{7.13}$$

Applying the same procedure for ADC2 (Take the difference of the two segments and remove the fundamental component), we obtain "Residue2" and 2 equations from it:

$$er_n^{(2)} = 2\pi fA\cos\left(2\pi fnT_s+\varphi\right)\{(\delta c_n-\delta c'_n)+(\delta a_n^{(2)}-\delta a'^{(2)}_n)\}+\left(w_n^{(2)}-w'^{(2)}_n\right) \tag{7.14}$$

$$\sum er_n^{2^{(2)}} \approx 4M\left(\pi fA\right)^2\left(\sigma_a^{2(2)}+\sigma_c^2\right)+2M\sigma_q^{2^{(2)}} \tag{7.15}$$

$$\sum\left(er_n^{(2)}\cos\left(2\pi fnT_s+\varphi\right)\right)^2 \approx 3M\left(\pi fA\right)^2\left(\sigma_a^{2^{(2)}}+\sigma_c^2\right)+M\sigma_q^{2^{(2)}} \tag{7.16}$$

Note that the clock jitter terms are exactly the same in the respective segments of ADC1 and ADC2, which is why they do not have superscripts. Next, we subtract Residue2 from Residue1 (Difference of equations (7.11) and (7.14)) to get "Residue3":

$$\begin{aligned} e_n = {} & 2\pi fA\cos\left(2\pi fnT_s+\varphi\right)\{(\delta a_n^{(1)}-\delta a'^{(1)}_n)-(\delta a_n^{(2)}-\delta a'^{(2)}_n)\} \\ & +\left(w_n^{(1)}-w'^{(1)}_n\right)-\left(w_n^{(2)}-w'^{(2)}_n\right) \end{aligned} \tag{7.17}$$

The clock jitter terms will cancel out and we are left with only the aperture jitter and noise of the 2 ADCs in equation (7.17). We can again obtain 2 different equations from residue3:

$$\sum e_n^2 \approx 4M\left(\pi fA\right)^2\left(\sigma_a^{2(1)}+\sigma_a^{2(2)}\right)+2M\left(\sigma_q^{2^{(1)}}+\sigma_q^{2^{(2)}}\right) \tag{7.18}$$

$$\sum\left(e_n\cos\left(2\pi fnT_s+\varphi\right)\right)^2 \approx 3M\left(\pi fA\right)^2\left(\sigma_a^{2(1)}+\sigma_a^{2(2)}\right)+M\left(\sigma_q^{2^{(1)}}+\sigma_q^{2^{(2)}}\right) \tag{7.19}$$

We have thus obtained 6 equations, 2 for each residue (equations (7.12), (7.13), (7.15), (7.16), (7.18), and (7.19)). We have 5 variables to estimate $(\sigma_c^2, \sigma_a^{2(1)}, \sigma_a^{2(2)}, \sigma_q^{2(1)}, \sigma_q^{2(2)})$. Since this is an overdetermined system, the method of least squares can be used to estimate the required variables. Thus, we have finally estimated the clock jitter, the aperture jitters and the noise of the two ADCs. These can now be used to accurately estimate the specifications of the 2 ADCs.

7.3.4 Accurate estimation of ADC specifications

The RMS of the aperture jitters of the two ADCs and the RMS of the clock jitter can be calculated as:

$$RMS_{AJ}^{(1)} = \sqrt{\sigma_a^{2(1)}} \tag{7.20}$$

$$RMS_{AJ}^{(2)} = \sqrt{\sigma_a^{2(2)}} \tag{7.21}$$

$$RMS_{CJ} = \sqrt{\sigma_c^2} \tag{7.22}$$

The SNRs of the 2 ADCs can now be estimated using the equations:

$$SNR_{est}^{(1)} = 10\log\left(\frac{P_{sig}}{P_{TotalNoise}^{(1)} - P_{ClockJitter}}\right) \tag{7.23}$$

$$SNR_{est}^{(2)} = 10\log\left(\frac{P_{sig}}{P_{TotalNoise}^{(2)} - P_{ClockJitter}}\right) \tag{7.24}$$

where P_{sig}, $P_{TotalNoise}^{(1)}$ and $P_{TotalNoise}^{(2)}$ can be estimated by applying the FIRE algorithm [13] on $x_n^{(1)}$ and $x_n^{(1)}$, and $P_{ClockJitter}$ is calculated as $2\pi^2 f^2 A^2 \sigma_c^2$. Similarly, true values for SNDR, ENOB etc. can be calculated.

7.3.5 Summary of the proposed method

The entire process can be summarized in the steps shown in Figure 7.7.

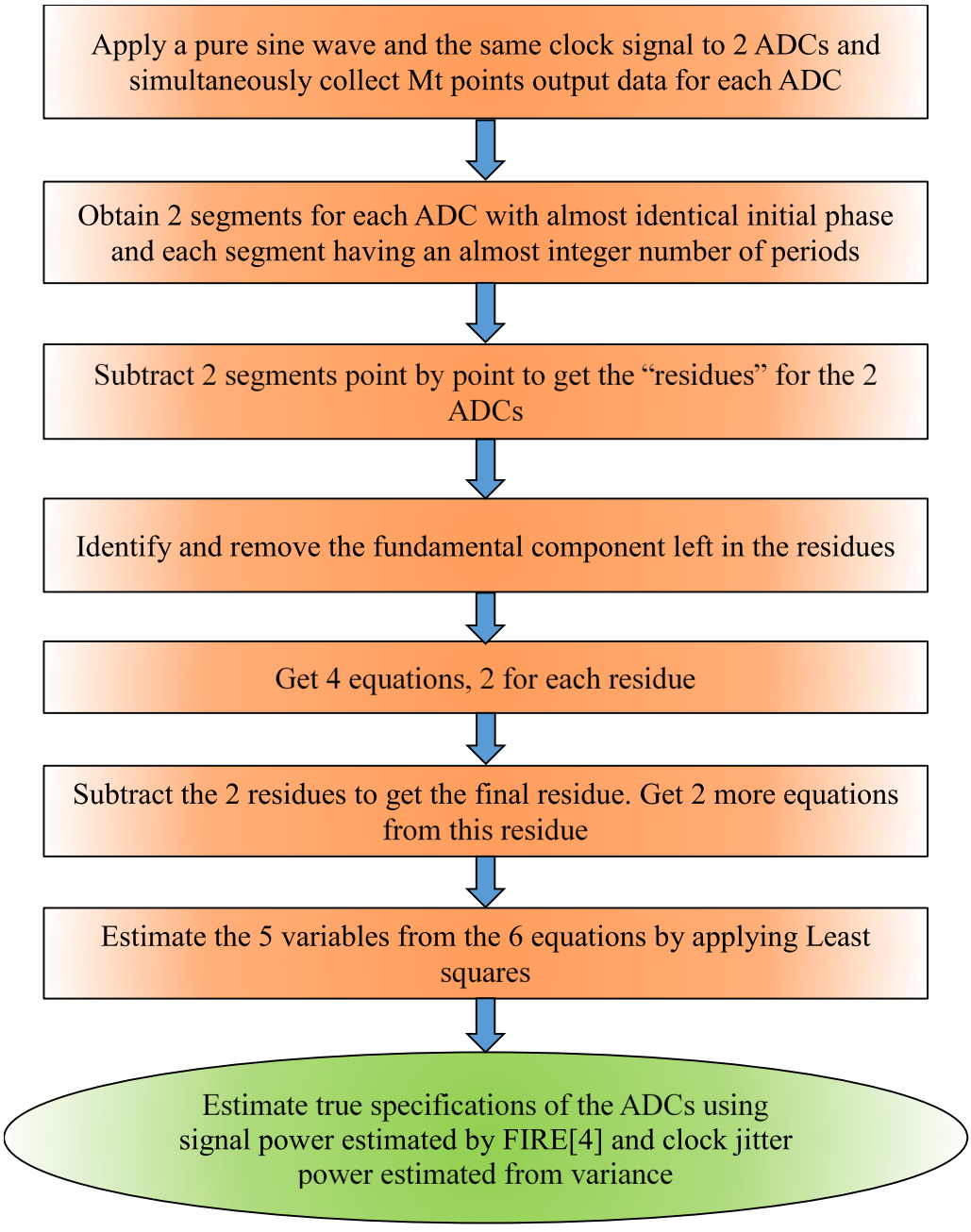


Figure 7.7. Summary flowchart

7.4 Simulation results

The proposed method is validated by MATLAB modelling and simulations. Different SAR ADCs are modelled with capacitor mismatches. The input sine wave amplitude is set such that the signal covers 99% of the full range of the ADC. The input signal frequency is set to little less than half of the sampling frequency, such that $f / f_s = C / M_t$ where M_t is the total number of samples and $C = J_{int} + J_{frac}$, where J_{int} is an integer, and J_{frac} is the fractional part. $J_{frac} = 0$ means that the input is coherently sampled and $J_{frac} \neq 0$ means that it is non-coherently sampled. ADC additive noise, clock jitter and ADC aperture jitters are randomly generated with their respective variances. The proposed method is applied and the jitter values and the SNRs of the ADCs are calculated. To get the true specifications of the ADC, a sine wave is sent as the input, with clock jitter as zero and $J_{frac} = 0$. This means that the input is coherently sampled.

First, simulations are run under perfect coherent sampling conditions, i.e. with clock jitter but $J_{frac} = 0$. Strategy 1 is used for segment selection, and the rest of the method is as described in the flowchart. The results are shown in Table 7.1.

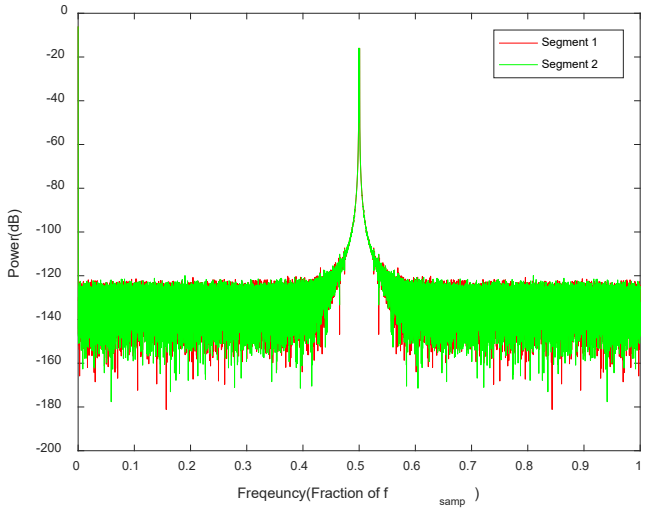
The values reported in the table are for one run for each column. The “SNR true” is the true SNR of the ADC, estimated from the test with no clock jitter. “SNR est1” is the SNR estimated directly from the spectrum of the ADC tested with clock jitter. “SNR est1” is much lower than the true SNR because of the effect of the clock jitter. “SNR est2” is the SNR estimated by removing the total jitter. “SNR est2” is higher than the true SNR because the aperture jitter has also been removed in addition to the clock jitter. “SNR est3” is the SNR estimated using the algorithm described in this paper i.e. by removing only the clock jitter, and including the aperture jitter as part of the noise. Additionally, in order to test the standard deviation of the jitter estimates, each column was tested 100 times with different random distributions of clock jitter and aperture jitter.

Table 7.1 Estimation of Jitter and ADC specifications with coherent sampling

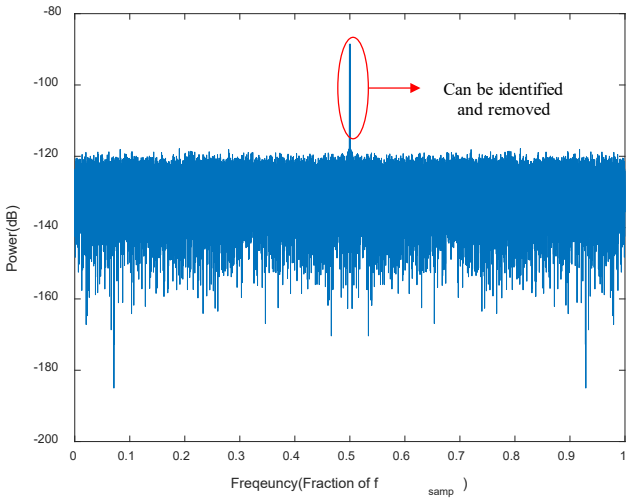
Resolution	**12 bit**	**14 bit**	**16 bit**	**16 bit**
Sampling frequency	2GHz	500MHz	125MHz	200MHz
0.5 LSB jitter	55fs	55fs	55fs	35fs
M	2^16	2^16	2^16	2^16
CJ added	250fs	250fs	250fs	300fs
CJ est	250.1fs	250.5fs	250.2fs	299.6fs
Std of CJ est	0.7fs	0.65fs	1.05fs	0.8fs
AJ1 added	50fs	50fs	50fs	50fs
AJ1 est	51.6fs	50.8fs	50.5fs	48.4fs
Std of AJ1 est	4.8fs	4.0fs	4.4fs	5.1fs
AJ2 added	60fs	60fs	60fs	60fs
AJ2 est	60.4fs	59fs	61.1fs	59.7fs
Std of AJ2 est	3.6fs	2.3fs	2.35fs	4.2fs
SNR1 true	65.28dB	77.5dB	89.6dB	87.2dB
SNR1 est1	55.57dB	67.6dB	79.7dB	74.2dB
SNR1 est2	66.98dB	79.35dB	91.5dB	90.5dB
SNR1 est3	65.77dB	77.8dB	90.1dB	87.8dB
SNR2 true	64.81dB	76.7dB	88.5dB	86.8dB
SNR2 est1	55.52dB	67.4dB	79.6dB	74.2dB
SNR2 est2	67.25dB	79.12dB	90.9dB	91.8dB
SNR2 est3	65.14dB	77.4dB	89.1dB	86.9dB

The standard deviations of the different jitter estimates over these 100 runs are given in the rows labelled "Std of…". It can be seen from the table that the estimation of the clock jitter, aperture jitter and SNR of the two ADCs is accurate and repeatable. The standard deviation of estimation is of the order of only a few femto seconds.

Next, to test the method which can account for non-coherent sampling, the ADC is then tested with non-zero clock jitter and $J_{frac} = 0.5$. The RMS of additive noise is set to 0.33LSB. Strategy 2 is used for segment selection, and the algorithm is applied to get the residues and estimate the jitter values. As an example, Figure 7.8(a) shows the double-sided spectra of the 2 segments of ADC1. The spectral leakage due to non-coherency is clearly visible. Figure 7.8(b)



(a)



(b)

Figure 7.8. (a) Non-coherently sampled spectra of the 2 segments of ADC1
(b) Spectrum of the residue (point-by-point difference) of the 2 segments

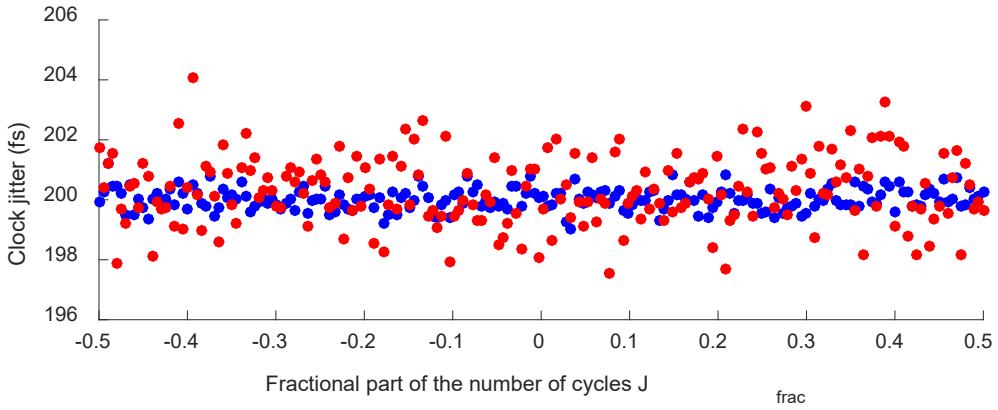
shows the spectrum of the residue of the 2 segments of ADC1 (equation (7.6)). As expected, because the phase between the 2 segments is not perfectly matched, a fundamental component remains, but its peak is reduced considerably. As mentioned previously, this is easily identified and removed.

Table 7.2 Estimation of Jitter and ADC specifications with non-coherent sampling

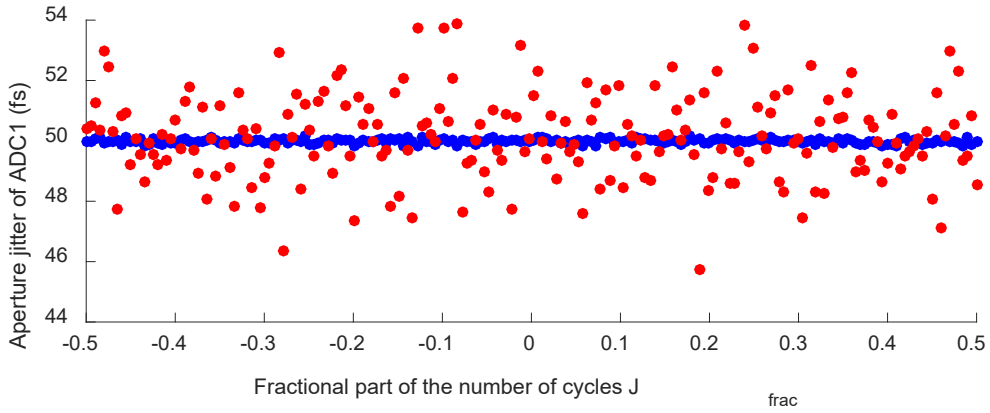
Resolution	**12 bit**	**14 bit**	**16 bit**
Sampling frequency	2GHz	500MHz	125MHz
Mt	3×2^{16}	3×2^{16}	3×2^{16}
CJ added	200fs	200fs	200fs
CJ estimated	200.4fs	197.3fs	198.9fs
AJ1 added	50fs	50fs	50fs
AJ1 estimated	50.2fs	50.0fs	48.6fs
AJ2 added	60fs	60fs	60fs
AJ2 estimated	59.6fs	59.8fs	61.4fs
SNR1 true	67.1dB	79.1dB	91.1dB
SNR1 estimated	67.2dB	78.7dB	90.9dB
SNR2 true	66.2dB	78.2dB	90.3dB
SNR2 estimated	66.4dB	78.0dB	90.0dB

The simulation results for ADCs of different resolutions, with non-coherent sampling, are summarized in Table 7.2. The values reported are for one run for each column. Values are reported for both ADCs. We can see that the estimated RMS values of clock jitter and the aperture jitters of the 2 ADCs are very close to the true values. Similarly, the SNR is also estimated accurately.

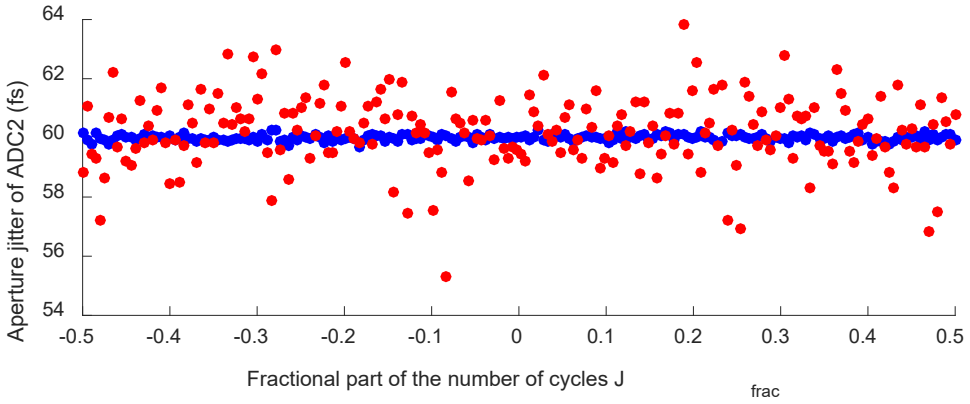
To check the robustness of the proposed method with respect to the level of non-coherency, 200 16-bit ADC pairs are generated. The RMS of the aperture jitter of ADC1 is 50fs and for ADC2, it is 60fs. The RMS of the additive noise is 0.33LSB. The fractional part of the number of cycles, J_{frac} is swept from -0.5 to 0.5 and so, the frequency of the input also changes, thus changing the amount of non-coherency. Figure 7.9 and Figure 7.10 show the results of the robustness test. The blue points are the reference jitter RMS values and SNRs for the 200 runs, and the red points are the estimated values. It can be seen that the method is fairly robust to the amount of non-coherency.



(a)

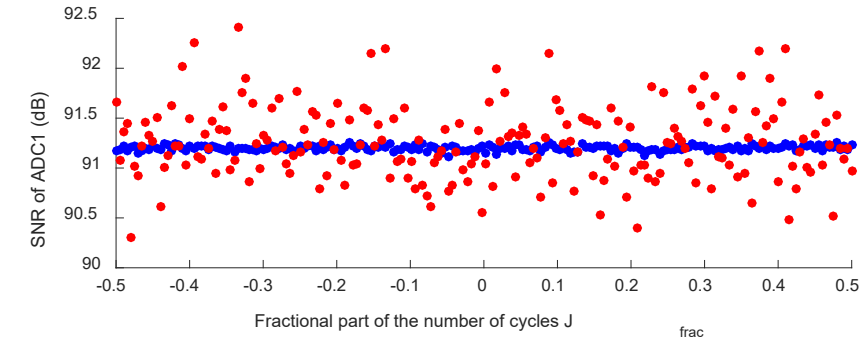


(b)

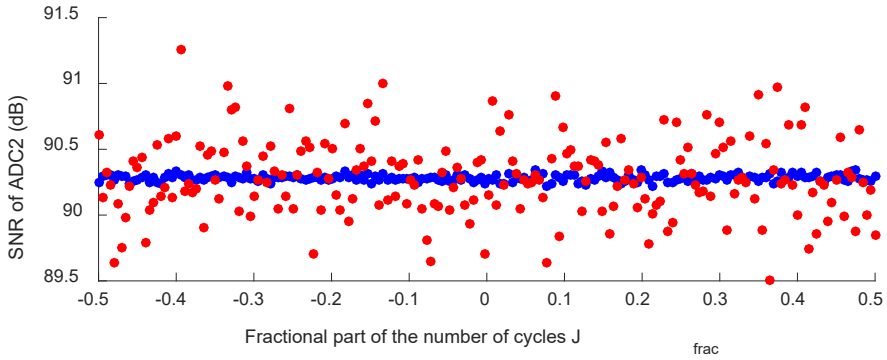


(c)

Figure 7.9. Blue: true values. Red: estimated values (a) Clock jitter (b) Aperture jitter of



(a)



(b)

Figure 7.10. Blue: true values. Red: estimated values (a) SNR of ADC1 (b) SNR of ADC2

7.5 Conclusion

A low-cost method was presented which can separate and accurately estimate clock jitter, ADC aperture jitter and noise, and thus the specifications of the ADC. Existing state of the art methods can estimate aperture jitter accurately only when the clock jitter is insignificant compared to the aperture jitter, or, alternatively, they require the clock jitter to be measured using expensive test equipment. This method not only alleviates the need for an ultra-low jitter clock signal, but relaxes the stringent coherency requirement, thus enabling accurate and low-cost characterization and testing of high speed ADCs.

7.6 References

[1] S. Chaganti, L. Xu, and D. Chen, "A low-cost method for separation and accurate estimation of ADC noise, aperture jitter, and clock jitter," in *2017 IEEE 35th VLSI Test Symposium (VTS)*, Apr. 2017, pp. 1–6, doi: 10.1109/VTS.2017.7928950.

[2] S. K. Chaganti, L. Xu, and D. Chen, "A low-cost jitter separation and ADC spectral testing method without requiring coherent sampling," in *2018 IEEE International Symposium on Circuits and Systems (ISCAS)*, May 2018, pp. 1–5, doi: 10.1109/ISCAS.2018.8351210.

[3] B. Kim and J. A. Abraham, "Efficient Loopback Test for Aperture Jitter in Embedded Mixed-Signal Circuits," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 58, no. 8, pp. 1773–1784, Aug. 2011, doi: 10.1109/TCSI.2011.2106030.

[4] W. Kester, *The Data Conversion Handbook*. Analog Devices Inc, 2004.

[5] M. Wu, D. Chen, and J. Duan, "Fast accurate algorithm for jitter test with a single frequency test signal," in *2011 IEEE International Conference on Electro/Information Technology*, May 2011, pp. 1–5, doi: 10.1109/EIT.2011.5978625.

[6] L. Xu and D. Chen, "Accurate and efficient method of jitter and noise separation and its application to ADC testing," in *2014 IEEE 32nd VLSI Test Symposium (VTS)*, Apr. 2014, pp. 1–5, doi: 10.1109/VTS.2014.6818743.

[7] L. Xu and D. Chen, "A low cost jitter estimation and ADC spectral testing method," in *2015 IEEE International Symposium on Circuits and Systems (ISCAS)*, May 2015, pp. 2277–2280, doi: 10.1109/ISCAS.2015.7169137.

[8] T. J. Yamaguchi, K. Degawa, M. Kawabata, M. Ishida, K. Uekusa, and M. Soma, "A new method for measuring alias-free aperture jitter in an ADC output," in *Test Conference (ITC), 2015 IEEE International*, Oct. 2015, pp. 1–6, doi: 10.1109/TEST.2015.7342384.

[9] T. J. Yamaguchi, M. Kawabata, M. Soma, M. Ishida, K. Sawami, and K. Uekusa, "A New Method for Measuring Aperture Jitter in ADC Output and Its Application to ENOB Testing," in *2008 IEEE International Test Conference*, Oct. 2008, pp. 1–9, doi: 10.1109/TEST.2008.4700639.

[10] Y. Langard, J. L. Balat, and J. Durand, "An improved method of ADC jitter measurement," in *Test Conference, 1994. Proceedings., International*, Oct. 1994, pp. 763–770, doi: 10.1109/TEST.1994.528023.

[11] J. M. Janik, D. Bloyet, and B. Guyot, "Measurement of timing jitter contributions in a dynamic test setup for A/D converters," *IEEE Transactions on Instrumentation and Measurement*, vol. 50, no. 3, pp. 786–791, Jun. 2001, doi: 10.1109/19.930455.

[12] G. Chiorboli, "Sub-picosecond aperture-uncertainty measurements [ADCs]," *IEEE Transactions on Instrumentation and Measurement*, vol. 51, no. 5, pp. 1039–1044, Oct. 2002, doi: 10.1109/TIM.2002.807799.

[13] S. K. Sudani and D. Chen, "FIRE: A Fundamental Identification and Replacement Method for Accurate Spectral Test Without Requiring Coherency," *IEEE Transactions on Instrumentation and Measurement*, vol. 62, no. 11, pp. 3015–3025, Nov. 2013, doi: 10.1109/TIM.2013.2267473.

CHAPTER 8. CONCURRENT SAMPLING WITH LOCAL DIGITIZATION – AN ALTERNATIVE TO THE ANALOG TEST BUS

Nanqi Liu*, Shravan K. Chaganti*, Zhiqiang Liu* and Degang Chen*, Amitava Majumdar

*Dept. of Electrical and Computer Engineering, Iowa State University, Ames, IA, 50010

Xilinx Inc., 2100 Logic Dr., San Jose, CA 95124

This chapter is adapted from a manuscript published in the proceedings of the IEEE International Symposium on Circuits and Systems 2018

Abstract

This chapter presents a Concurrent Sampling (CS) method for measuring a multitude of analog DC voltages concurrently using local digitization. Boolean results after digitization are routed in an IJTAG compatible fashion. Analog quantities are no longer routed across the die, thus overcoming several limitations of Analog Test Buses. Furthermore, the proposed method enables real-time measurement of analog voltages, thus addressing a growing need for Automotive test and reliability. The proposed method is applied to an analog circuit consisting of some widely used analog blocks such as a bandgap reference and an operational amplifier. Transistor level simulation results demonstrate that the proposed method is functional and the drawbacks of the ATB are no longer present.

8.1 Introduction

Quality and thus, test requirements for mixed-signal system on chips (SoCs) are becoming increasingly stringent, especially for automotive, medical, military and other mission-critical

applications. Testing for faults in the digital part of SoCs can be achieved with scan chains [1]. However, controllability and observability of analog circuits is not so straightforward. The Analog Test Bus (ATB) was introduced for this purpose over 30 years ago [2], and despite its various limitations, is still the only choice till today. The essence of an ATB are two global wires that carry an analog voltage and its ground reference to an analog-to-digital converter (ADC). Various IEEE standards (IEEE 1149.1, 1149.4) have been developed over the years to standardize test access for mixed signal circuits, the most recent being the IEEE 1687 [3]–[5].

Although the ATB has been improved for over 30 years, it has some major limitations which are summarized as follows.

1. Each ATB-ADC pair can measure only one node at a time. Measuring n nodes simultaneously requires n ATB-ADC pairs making it cost-ineffective for $n > 1$.
2. An ATB is essentially a large 1-hot analog multiplexor, with a leg (or probe point) for each analog node. Ensuring non-overlapping connection to the ATB is a difficult, and often intractable, verification problem.
3. The large mux connections to the ATB lead to large settling times of the ATB. This leads to large test times.
4. Ensuring minimum measurement fidelity requires upper layer metals for implementing ATBs, thus adding to demands on this scarce resource in large SoCs.
5. Switching from one probe-point to the next causes large swings in the node-under-test. Due to this reason, switching a node requires the chip to go through reset, eliminating the possibility of real-time node measurement.
6. Long global wires used for implementing ATBs invariably lead to degradation of signal integrity and measurement accuracy.

7. High measurement accuracy is required for a very small subset of nodes connected to an ATB-ADC pair. Due to this reason, the high-precision, high-cost ADC needed for measuring such nodes ends up being under-utilized.

In modern safety-critical systems, such as automotive SoCs, there are emerging safety and reliability requirements to monitor key system properties in real-time. Heffernan et. al, in [6] developed a non-invasive monitor solution guided by the ISO-26262 Standard [7]. Ciaran et. al, in [8] proposed a method to extend existing on-chip trace/test/debug modules to support runtime verification [9] monitoring. In these on-chip real time monitoring methods, the most critical part is to acquire the data within the embedded ICs. As explained earlier, existing ATBs cannot be used in real time and there are no IEEE standards on on-chip monitoring of analog circuits. This work will enable technology to realize real-time monitoring of analog circuits.

A Concurrent Sampling (CS) method for simultaneously measuring many analog DC voltages using local digitization is proposed. Comparator and flip flop based digitizers are placed close to the voltage nodes under test. Outputs of the digitizers located across the SoC are integrated at system level. The advantages of this method over those of ATBs are summarized as follows.

1. Multiple nodes can be measured concurrently which significantly reduces test time.
2. Similar nodes can share a digitizer to reduce area overhead which only requires small decoded MUXs.
3. Local routing between nodes and digitizers can be done with lower layer metals.
4. The cross-coupling of nodes between different blocks is totally eliminated.
5. Low cost strongARM latches or precision auto-zero comparators can be used for nodes of different resolutions.

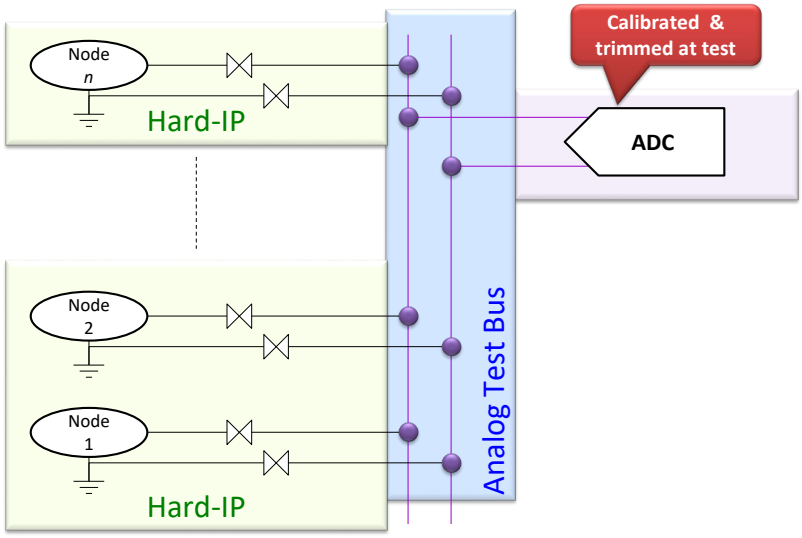


Figure 8.1 Architecture of the Analog Test Bus

The combined effect of these benefits is that *real-time monitoring of analog circuits* becomes realizable.

The remainder of the chapter is organized as follows. In section II, the general idea of the concurrent sampling method with local digitization is introduced. The IP and system level implementations of the proposed solution are also presented. Section III shows simulation results to validate the proposed method. Finally, conclusions are drawn in Section IV.

8.2 Proposed Concurrent Sampling Solution

Before discussing the proposed approach in detail, we would like to review the architecture of the conventional ATB, reveal its limitations and use that to motivate the proposed method.

The generic analog test bus architecture is shown in Figure 8.1. Analog and mixed-signal IPs in an SoC include but are not limited to op amps, comparators, analog switches, references, bias generators, and data converters. The nodes chosen for testing should ensure functionality of these IPs.

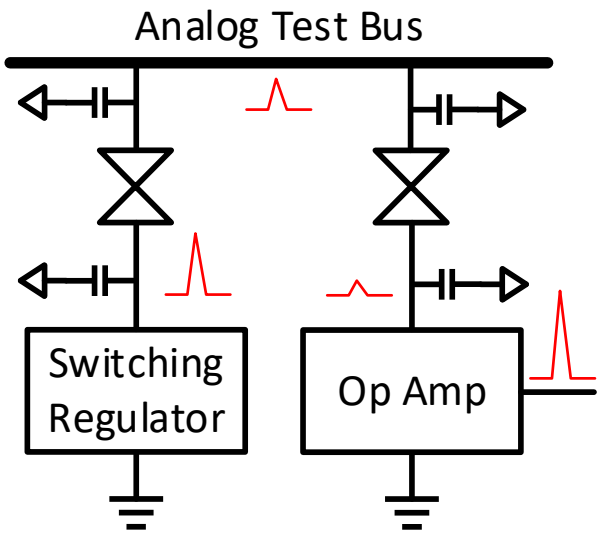


Figure 8.2 Cross-coupling between probe points in ATB

As shown in Figure 8.1, the nodes under test are all loaded onto two differential buses through switches. This results in large RC loading on the bus and kickback between probe points. The loading and kickbacks are one reason that real-time monitoring is challenging or nearly impossible with the ATB.

Some analog IPs, e.g., switching regulators, can generate high frequency voltage spikes. When the ATB is used, the voltage spikes can couple though switches and wiring parasitic capacitors to a noise sensitive block. This problem is illustrated in Figure 8.2, where a spike from the regulator is coupled to an operational amplifier (Op Amp) and may cause dramatic performance loss or possibly kill the operation of the Op Amp.

To overcome these limitations, local digitization is performed in the CS architecture as shown in Figure 8.3. Voltage nodes under test are first digitized with local digitizers and held in digital storage elements (FFs). Contents of FFs can be read out in one of many ways. Figure 8.3 suggests that the FF contents are shifted out through one or more scan chains, or through IJTAG. With the proposed local digitization, the cross coupling of nodes between different blocks is totally

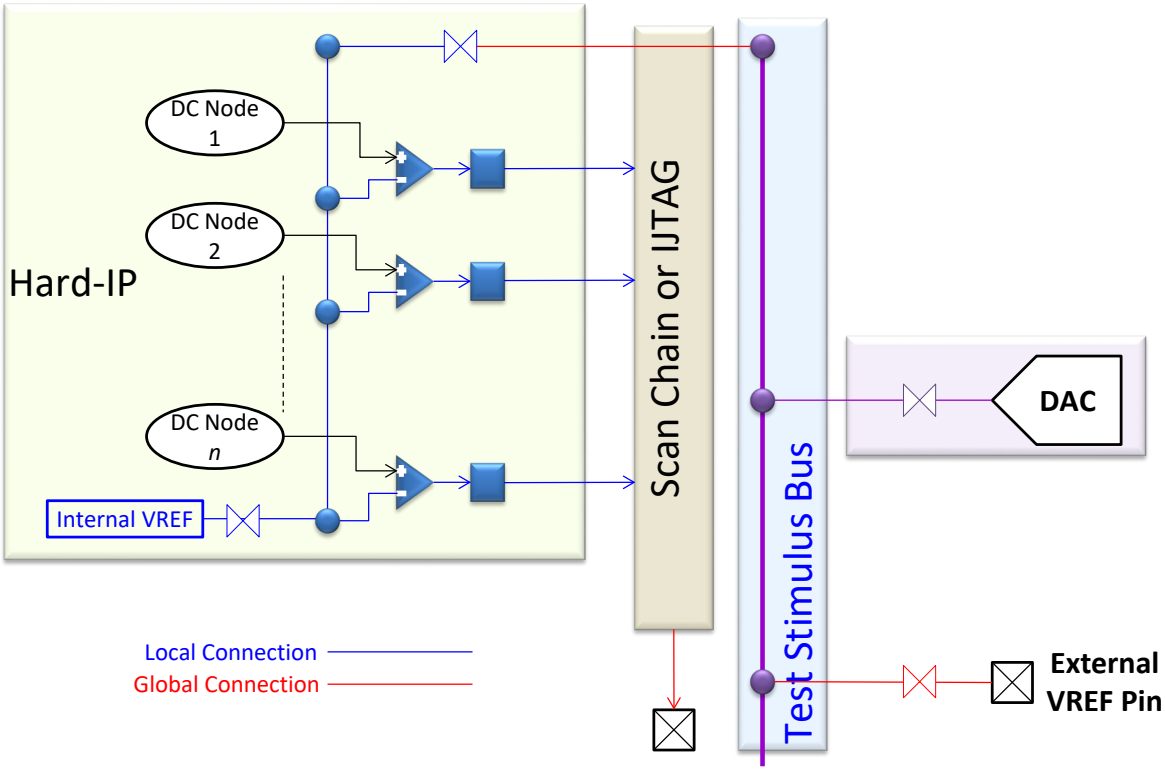


Figure 8.3 Architecture of the proposed Concurrent Sampling method

eliminated. With the shared VREF sweep, multiple voltage nodes can be measured at the same time which significantly reduces measurement time, especially in SoCs with a large number of nodes under test.

8.2.1 Local Digitization

The local digitization concept is shown in Figure 8.4. A DC node under test, which can be supply, bias or reference voltage, is sent to one comparator input. Another comparator input is a reference voltage VREF that is sequentially ramped up, either from a digital-to-analog convert (DAC) or from an external pin. The comparator will compare the DC value of the 'Node Under Test' with the VREF value and the output is latched into the flip-flop. The comparator is located near the node under test, and therefore requires only lower level metals for routing. This offers

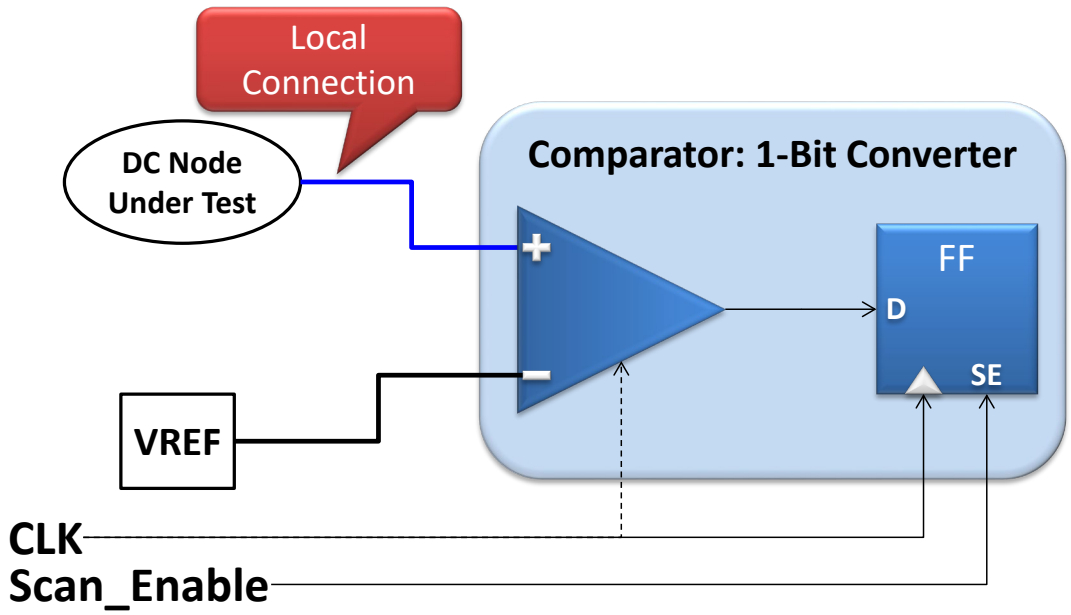


Figure 8.4 Local Digitization

significant savings in design effort and reduces die cost. The output of the flip-flop is further sent out through a scan chain or an IJTAG network [4] which will be discussed later in this section.

In local digitization, a comparator is used instead of a full ADC to avoid area and power overhead. Variety of nodes under test requires flexible measurement resolution. According to the authors' experience, in real implementation, these nodes can be roughly divided into three categories: 1) Most nodes only require 1-bit resolution, for example, switches controlling nodes in digital trimming circuits need only 1-bit resolution and can be monitored with just digital buffers and flip-flops. 2) Some nodes require medium resolution, e.g., biasing voltages of cascode transistors. Three or four bit resolution is enough to verify that the transistors are working in saturation. So simple comparators, e.g. strongARM latches [10], can be used. The strongARM

latch can work very fast and consumes very little quiescent current. While strongARM latches can suffer from relatively large offset voltages, they are still acceptable for 4-bit resolution.

3) Very few nodes may need to be tested with high resolution, e.g., supply voltage of analog blocks, especially for those circuits with poor power supply rejection (PSR). To reduce the offset voltage and improve gain, pre-amplifiers can be added before simple comparators. For nodes that need very high resolution, e.g. reference voltages of a 16-bit DAC, on-chip testing is difficult. However, for references of low-dropout regulators (LDO), which may require 8-bit resolution, the proposed idea is still applicable.

8.2.2 Concurrent Sampling

With a given ATB-ADC pair, only one node can be measured at a time. However, with the proposed method, since each node under test has its own test cell, multiple nodes can be measured at the same time with the shared VREF sweep. Note that the ground reference voltage inside a block is itself a node-under-test. An illustration example is given in Table 8.1.

Table 8.1 Linear VREF sweep with 100mV Resolution

	Nodes							
VREF(V)	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**
0.1	1	1	1	1	1	1	1	1
0.2	1	1	0	1	1	1	1	1
0.3	1	1	0	1	1	1	1	1
0.4	1	1	0	1	1	0	1	1
0.5	1	1	0	1	1	0	1	1
0.6	1	0	0	1	0	0	1	0
0.7	1	0	0	1	0	0	1	0
0.8	0	0	0	0	0	0	0	0
Node Voltage (V)	**0.8**	**0.6**	**0.2**	**0.8**	**0.6**	**0.4**	**0.8**	**0.6**

After a linear VREF sweep with 100 mV step size, voltages of all the eight nodes can be interpreted simultaneously.

In conventional analog and mixed-signal IP test, the settling of voltage under test also limits the speed of testing. The analog test bus (ATB) needs sophisticated switches (large size) to reduce the settling time of the bus. Local digitization only introduces small parasitic resistances and capacitances, which leads to much faster settling of the comparator input and significantly reduces conversion time. Comparators don't need to wait several clock cycles for its input to settle and then compare.

8.2.3 Concurrent Sampling in an IP-SoC Environment

A possible implementation of CS inside an IP is shown in Figure 8.5. As noted earlier, the ground-reference(s) inside an IP are themselves nodes-under-test. The proposed read-out mechanism in this figure is IJTAG-based [4]. Test data registers (TDR)s are connected in flexible length scan chains using segment insertion bits (SIB)s. Measurement resolution is controlled by

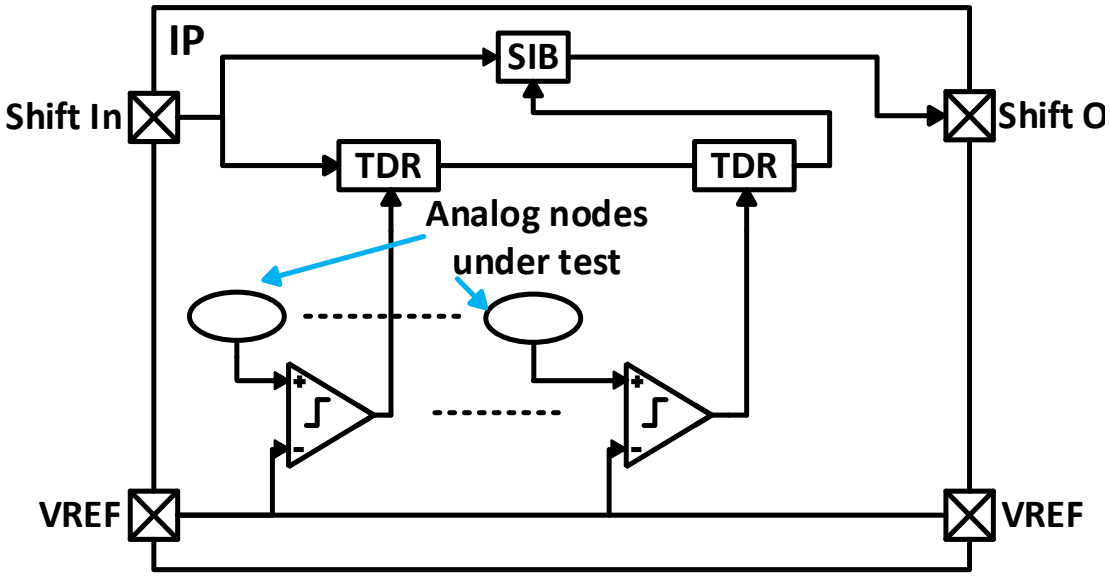


Figure 8.5 CS inside an IP

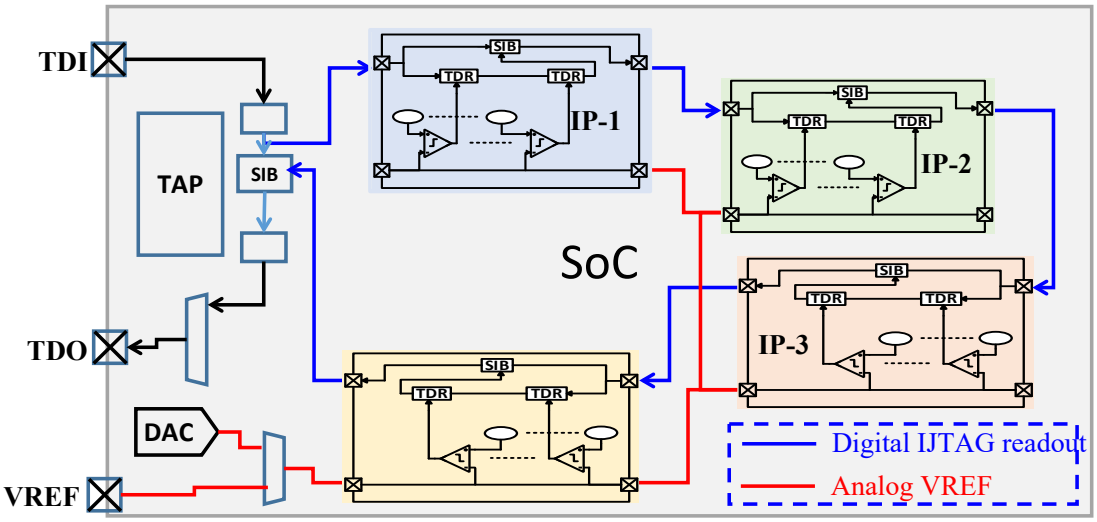


Figure 8.6 System level implementation of CS in SoC using IJTAG based control and readout

VREF step size. Therefore, we can achieve flexible resolution by controlling VREF sweep and select suitable MUXs and comparators. VREF is the only analog signal which is shared between test cells and needs to be distributed at the system level. To guarantee the resolution and sufficient settling of VREF at different locations across the SoC, analog buffers may need to be inserted on VREF path segments.

Figure 8.6 shows the integration of a CS network at an SoC level with multiple IPs. The test access port (TAP) controls the IP's test mode and provides connections for test data in and out. Note that VREF can be internally generated using an on-die DAC or, if higher precision is desired, it can be brought in from an external pin.

In general, more sophisticated the design of the comparator, the higher the accuracy of conversion and lower the influence of parasitics. However, there is a tradeoff between area and these figures of merits. One option to reduce area overhead is to share a single comparator across

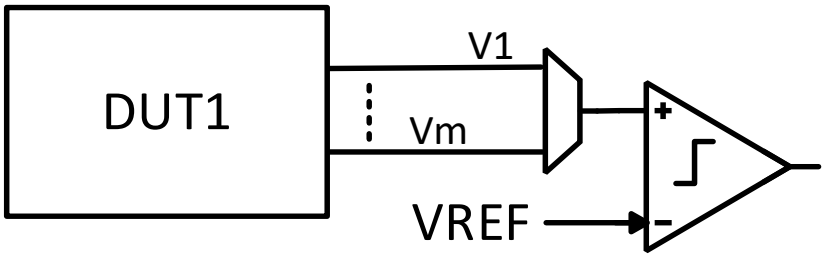


Figure 8.7 Sharing a comparator across multiple nodes under test

multiple nodes. This requires an analog MUX to select between the nodes-under-test as shown in Figure 8.7. Unlike ATBs, this MUX can be fully decoded since it is localized. While this method increases measurement time, designers can trade this off against area, accuracy and resolution. Several methods exist for reducing measurement time for shared comparators but are out of the scope of this work.

8.2.4 IP Design

In a SoC, analog and mixed-signal IPs can come from different companies or different product lines in the same company. The proposed method is suitable to follow an IP design methodology. As mentioned before, the nodes to be monitored have different resolution requirements. Comparators, MUXs, and buffers with various resolution can be designed as analog IPs and put under the concurrent sampling library. Combined with digital IPs including flip-flops, registers and finite state machines (FSM), CS test cells are created. When a design group provides an analog IP to the SoC application, CS test IPs should be placed locally at device under test (DUT). The only analog input to these IPs is the VREF voltage. Outputs of concurrently sampled IPs are all digital signals and hence integration at the SoC level is also completely digital. A new IEEE standard on on-chip monitoring for analog circuits needs to be developed so that IPs from all companies can follow the same test methodology.

Table 8.2 Similarity and differences between ATB and CS

	Analog Test Bus	Concurrent Sampling
Similarities	1. Target at analog and mixed signal IPs test 2. Test voltage nodes including supply / biases / references	
Differences	One node at a time	Multiple nodes concurrently
	Large 1-hot MUX	Small decoded MUXs
	Large variable RC load	Small static local RC load
	Upper layer metals	Lower layer metals
	Production test	Production test+Real-time monitoring

Similarities and differences between ATB and CS are summarized in Table 8.2

8.3 Simulation Results

The proposed method is verified in a 130nm CMOS process. Since op amps and voltage references are fundamental blocks and exist in almost any analog IP of SoCs, they are chosen as

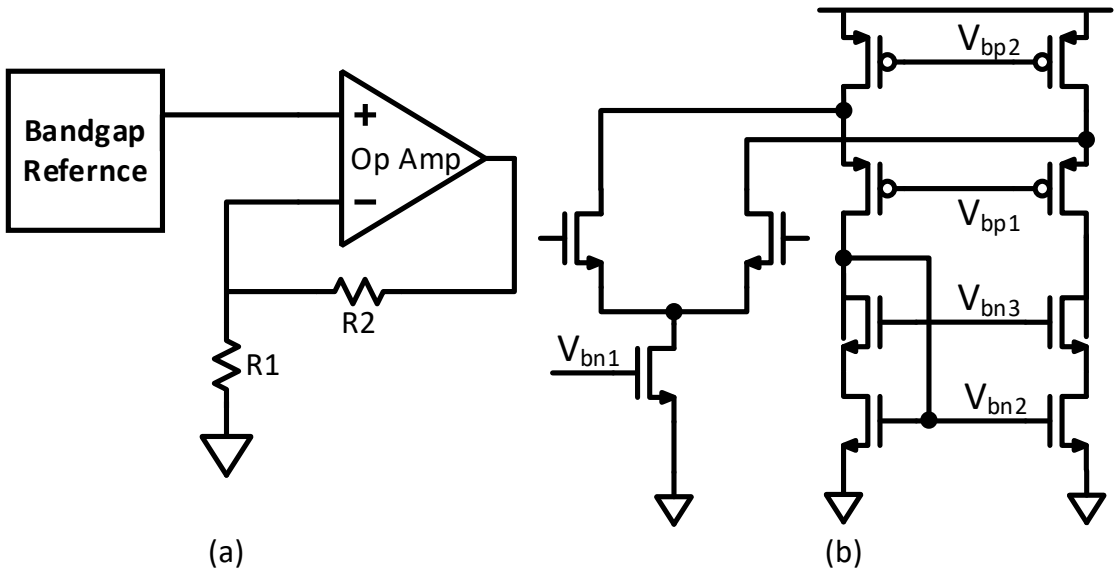


Figure 8.8 Example IPs under test (a) Block diagram, (b) Biasing nodes in Op Amp gain stage

the example devices under test. As shown in Figure 8.8, a bandgap reference generates a reference voltage and is scaled up by a gain stage based on a high precision op amp.

The folded cascode gain stage in the precision op amp is used as part of the schematic to show which nodes should be tested. Vbn1 and Vbp2 are tested as they decide the input pair tail current and the gain stage current, and therefore control the DC gain and speed of the op amp. Vbn2, Vbn3 and Vbp1 also need to be measured to ensure there is enough room for cascode transistors to work in saturation regions. All the nodes under test in the example IPs are listed in Table 8.3.

A 4-bit VREF sweep is performed to illustrate the proposed method assuming CLK is running at 1MHz. As shown in Figure 8.9, VREF ramps up in each cycle of CLK, and as it does, comparator outputs at different nodes switch from high to low indicating the voltage levels of the corresponding nodes. For example, simulation results tell us that $\frac{3}{16} * 3.3 = 618.75\,mV < Vbn1 < \frac{4}{16} * 3.3 = 825\,mV$.

Table 8.3 Nodes that are monitored

Resolution	**Num**	**Op Amp**	**Bandgap Ref**
1-bit	20	Control signals of digital trimming switches or enable switches	
4-bit	10	Biases of cascode transistors, biases in start-up circuits	
8-bit	5	Supply voltages, biases of tail current transistor and diode V_{BE}	
12-bit	1	Bandgap reference's output	

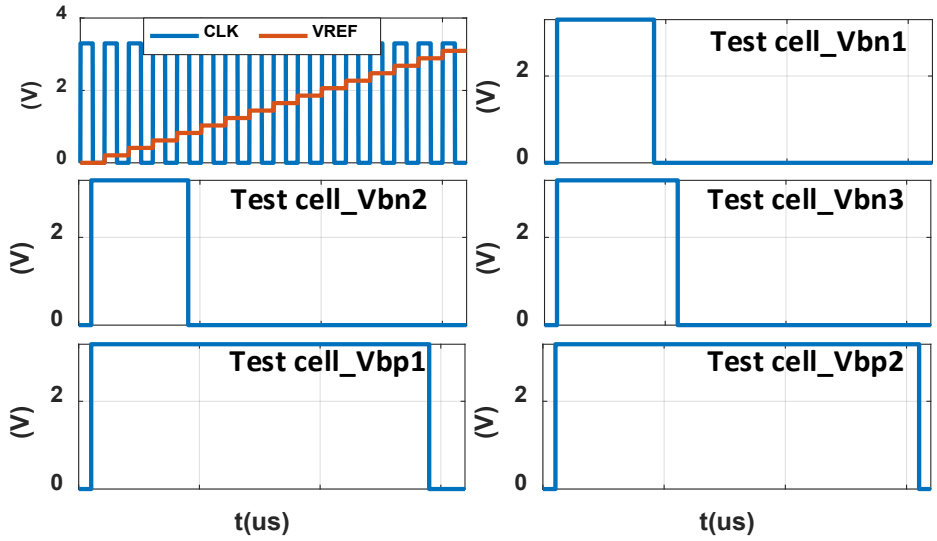


Figure 8.10 Outputs of test cells at different nodes

One problem here is that the output data is a chain of 1's and 0's which needs to be scanned out. Scheduling and making sense of the chain of 1s and 0s is also a major problem. A simple solution for this is to use a "gated counter" which increments/decrements as long as the comparator output is 1 and then stops when the comparator output becomes 0. The circuit is shown in Figure 8.10.

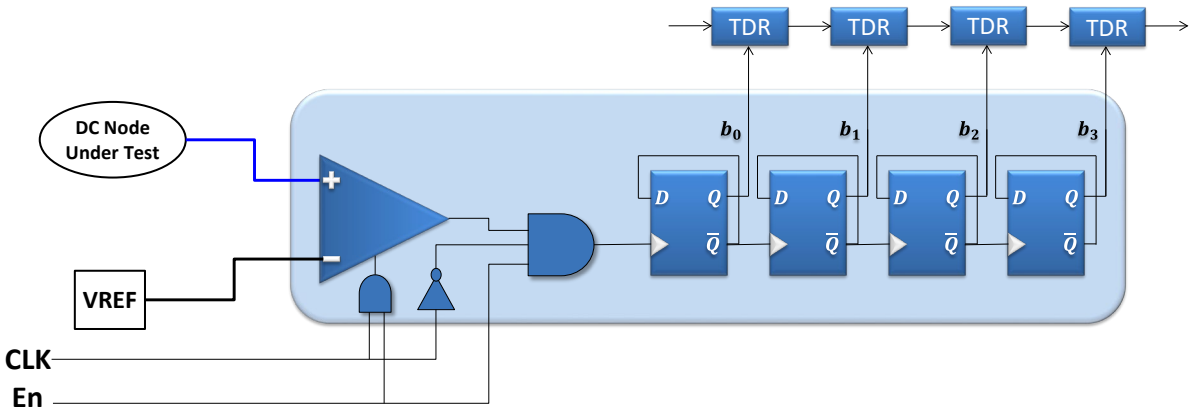


Figure 8.9. Gated counter

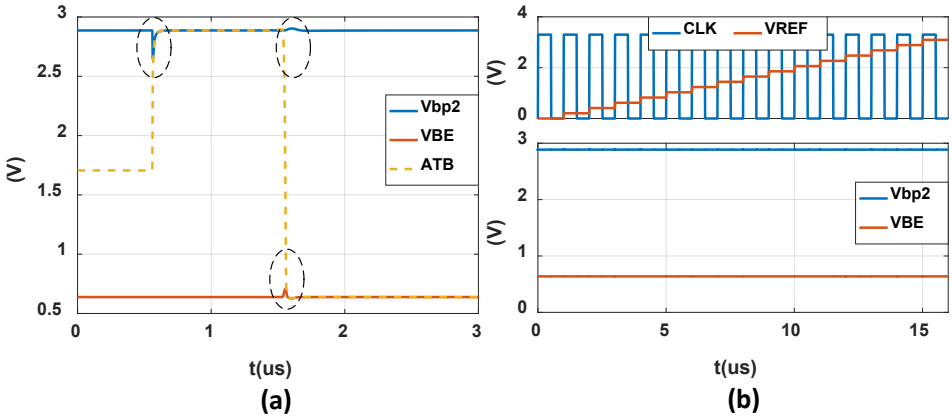


Figure 8.11 (a) Glitches in ATB during node switching, (b) No coupling in CS

By utilizing this, a direct digital number is stored in the register which is a faithful representation of the analog voltage of the node under test. This significantly reduces the number of bits that need to be stored/scanned out. After the end of the sweep, all the digital values can be scanned out using IJTAG.

Cross-coupling simulation results are given in Figure 8.11. With the ATB, as the node under test switches from Vbp2 to V_{BE}, glitches as large as 200 mV occur. While with CS, there are no coupling effects.

The Concurrent sampling method, combined with the concept of fault propagation graphs (FPGs), can be used to identify and monitor a set of analog voltage nodes which can enable near complete analog fault coverage [11].

8.4 Conclusion

An alternative solution to the Analog Test Bus has been proposed. Concurrent sampling with local digitization method has been shown to overcome several limitations of the ATB. In general, the method provides a fast, low cost and robust production test solution. Furthermore, the proposed method enables technology to realize real-time monitoring of analog circuits, thus addressing a growing demand for Automotive test and reliability.

8.5 References

[1] L.-T. Wang, C. E. Stroud, and N. A. Touba, *System-on-Chip Test Architectures: Nanometer Design for Testability*. Burlington : Elsevier Science, 2010.

[2] P. P. Fasang, D. Mullins, and T. Wong, "Design for testability for mixed analog/digital ASICs," in Proceedings of the IEEE 1988 Custom Integrated Circuits Conference, 1988, p. 16.5/1-16.5/4.

[3] "IEEE Standard for a Mixed-Signal Test Bus," IEEE Std 11494-2010 Revis. IEEE Std 11494-1999, pp. 1–116, Mar. 2011.

[4] "IEEE Standard for Access and Control of Instrumentation Embedded within a Semiconductor Device," IEEE Std 1687-2014, pp. 1–283, Dec. 2014.

[5] "IEEE SA - 1149.1-2013 - IEEE Standard for Test Access Port and Boundary-Scan Architecture," 02-Nov-2015. [Online]. Available: https://standards.ieee.org/findstds/standard/1149.1-2013.html. [Accessed: 02-Nov-2015].

[6] D. Heffernan, C. Macnamee, and P. Fogarty, "Runtime verification monitoring for automotive embedded systems using the ISO 26262 functional safety standard as a guide for the definition of the monitored properties," IET Softw., vol. 8, no. 5, pp. 193–203, Oct. 2014.

[7] "ISO 26262-1:2011(en), Road vehicles — Functional safety — Part 1: Vocabulary." [Online]. Available: https://www.iso.org/obp/ui/#iso:std:iso:26262:-1:ed-1:v1:en. [Accessed: 19-Dec-2016].

[8] C. MacNamee and D. Heffernan, "On-Chip Instrumentation for Runtime Verification in Deeply Embedded Processors," in 2015 IEEE Computer Society Annual Symposium on VLSI, 2015, pp. 374–379.

[9] L. Pike, S. Niller, and N. Wegmann, "Runtime Verification for Ultra-Critical Systems," in Runtime Verification, 2011, pp. 310–324.

[10] B. Razavi, "The StrongARM Latch [A Circuit for All Seasons]," in IEEE Solid-State Circuits Magazine, vol. 7, no. 2, pp. 12-17, Spring 2015.

[11] Z. Liu, "Design and verification approaches for reliability and functional safety of analog integrated circuits," Graduate Theses and Dissertations, Jan. 2018, [Online]. Available: https://lib.dr.iastate.edu/etd/17246.

CHAPTER 9. GENERAL CONCLUSION

In this dissertation, we presented cost-efficient solutions for some of the pertinent analog and mixed-signal test and calibration challenges, with a heavy focus on Digital to Analog Converters (DACs). The challenges associated with linearity testing of DACs are analyzed and then addressed by two classes of testing algorithms, adapted and suited for specific needs.

The first class of algorithms (uSMILE, ER, uISMILE, ER+I) significantly reduced DAC linearity test time and cost by reducing the number of measurements required. These methods were implemented on a production chip at Texas Instruments and a test time reduction of 15x-20x was seen in actual silicon measurement results for multiple 12-bit DACs, and >100x was seen in simulation case studies for many 16-bit DACs. The fundamentals of uSMILE were described in detail and rigorous theoretical analysis was performed on the various error sources, along with presenting a fast and memory efficient calculation procedure.

The second class of algorithms (uSMILE-ROME) not only reduced test time but significantly reduced test cost by eliminating the need for high precision measurement devices for linearity testing of high resolution DACs. A cheap on-board/on-chip digitizer with comparable resolution and worse linearity than the DAC under test can be used to get an accurate estimation of the DAC INL. The method was validated by measurement results of a 12-bit and 16-bit DAC by a 12-bit and 16-bit ADC respectively.

Furthermore, a complete on-chip DAC BIST solution based on uSMILE-ROME was detailed. The computations involved in the algorithm were adapted so that it could be implemented on-chip in a highly memory optimized and time efficient manner. The intricate challenges of implementing the algorithm on-chip, including a buffer update strategy, were discussed through an example case study of a subradix-2 DAC. Additionally, different methods for effective calibration

of the DAC using digital pre-distortion were described. This significantly cuts down DAC test cost and test time. It can also be used to enhance in-field functional safety and reliability not only of the DAC itself, but also of other IPs when combined with the Concurrent sampling method.

A dynamic version of the uSMILE algorithm was developed, which can be used to estimate digital pre-distortion codes for a DAC such that both the static and dynamic errors are calibrated out. The calculated pre-distorted codes were used to generate a pure sine wave at a given frequency, sampling rate, and output load. This pure sine wave can then be used for ADC spectral testing.

A low-cost method was presented which can separate and accurately estimate clock jitter, ADC aperture jitter and noise, and thus the specifications of the ADC. This method not only alleviates the need for an ultra-low jitter clock signal, but relaxes the stringent coherency requirement, thus enabling accurate and low-cost characterization and testing of high speed ADCs.

Finally, the last section presents a Concurrent Sampling (CS) method for measuring a multitude of analog DC voltages on-chip concurrently using local comparators and a calibrated DAC. Furthermore, the proposed method enables real-time measurement of analog voltages.

Many of the solutions presented here, like uSMILE and uSMILE-ROME, are already being implemented in the semiconductor industry, across different companies. uSMILE and its variants are currently used for production testing of multiple products at Texas Instruments and other semiconductor companies to reduce DAC linearity test time. A uSMILE-ROME based DAC built-in self-test and self-calibration scheme is currently being deployed on a chip at NXP semiconductors. Some SoCs which use these algorithms for production testing are already in wide use in the market. Hardware implementation of concurrent sampling, which builds on top of DAC BIST, for real time monitoring of analog circuits, is an untapped well which promises significant returns in terms of functional reliability of the device. The growing need for automotive test and

reliability is going to further spur the adoption of the BIST, self-calibration, and self-monitoring techniques presented in this work to ensure that the chips are reliable till the end-of-life of the device.

Although many low-cost methods were presented in this dissertation, the focus was heavily on data converter testing. Some of these methods can be extended to other circuits like DAC plus comparator subsystems. But there are many more analog circuits which require innovative low cost self-test and/or self-calibration solutions, like LDOs, buck/boost converters, bandgap circuits, Power-on-reset circuits, Brown-out circuits, oscillators and so on, in order to meet the stringent requirements for reliability and functional safety. The methods described in this work are just one step forward in the pursuit of high performance, reliable, and low-cost SoCs.