A Generic Foreground Calibration Algorithm For ADCs with Nonlinear Impairments

Armia Salib, Mark F. Flanagan and Barry Cardiff
School of Electrical and Electronic Engineering, University College Dublin, Ireland
Email: armia.salib-farag@ucdconnect.ie, mark.flanagan@ieee.org, barry.cardiff@ucd.ie

***Abstract*—This paper presents a generic foreground calibration algorithm which compensates for memoryless nonlinear impairments in pipeline, SAR or hybrid ADC architectures. Amplifier nonlinearity, comparator offsets, capacitance mismatch and settling time errors are considered. During the calibration process, each element of a look up table is computed by mapping each raw ADC output value to an estimate of the corresponding input, and the most likely input corresponding to each raw ADC output is computed and stored in the table; this table is then used during normal operation to map the raw values to the calibrated ADC outputs. Complexity reduction techniques are presented to facilitate an in-circuit hardware implementation in order to reduce foreground calibration time. The algorithm's performance is evaluated using a SAR ADC model suffering from various nonlinear impairments. Results are presented for settling time errors, capacitor mismatch scenarios, and a wide range of nonlinear amplifier parameters, demonstrating a significant performance improvement in all cases.**

***Index Terms*—ADC, transfer function, calibration, nonlinearity.**

I. Introduction

ADCs typically contain many sub-blocks, each of which can suffer from nonlinear impairments; together these can limit the performance of the ADC. For example, sample-and-hold (S/H) circuits suffer from nonlinear behavior due to the gain variation which occurs when the transistors move to the triode region [1]. The relationship between the S/H input x and output y can be mathematically modeled as

$$y = \sum_{i=1}^{\infty} c_i x^i, \tag{1}$$

where c_i are the non-linearity coefficients. In [1], those coefficients are estimated, and the estimates are used to compensate the effect of the nonlinearity through post-processing.

Architecture-dependent sources of nonlinearity also exist, which include, but are not limited to,

1) *Inter-stage operational amplifier nonlinearity in pipeline ADCs:* This effect can be modeled using a polynomial expression as in (1). For example, the authors of [1] use this model but limit their focus to the estimation of the first and second order terms.
2) *Capacitor mismatch in SAR ADCs:* To compensate this type of impairment, [2] and [3] suggest to use a linear combination of the comparator decisions with weights proportional to the capacitance (called *ideal weights*). In [3], these weights are obtained by sensing the histogram of the ADC's output.
3) *Insufficient settling time in SAR ADCs:* After each bit trial, a capacitor is switched and the voltage presented to the comparator converges to its new value with an RC time constant τ. If the time interval τ_s allocated for this is not sufficient, then errors may occur [3]. Increasing the sampling rate f_s (i.e., reducing τ_s/τ) will exacerbate this issue.

Many algorithms have been proposed to address the nonlinearity problem through characterizing the ADC. For example, [4] targets the calibration of pipeline ADCs using an unconstrained sinusoidal input; curve fitting and least-mean-square (LMS) adaptation techniques are used to identify the weights used for the linear combination. However, a large memory is required to store the input samples to be processed by the curve fitting procedure.

When multiple nonlinear impairments exist simultaneously, it is inefficient to calibrate each one independently. Based on this idea, in [5] a look up table (LUT) is used to store the INL errors to be used in correcting the ADC output. The INL errors are evaluated by measuring the power spectrum harmonic amplitudes, which are related to the INL errors. However, these harmonic components can have small amplitudes and are sensitive to noise. Also, the implementation in [5] involves matrix manipulations, which are excessively complex for real-time operation.

In this paper, we propose a unified calibration algorithm which can work in the presence of several sources of nonlinearity. The algorithm works by populating a LUT which stores the most likely input corresponding to each possible raw output of the ADC. This saves post-processing computational power and reduces latency, at the cost of additional storage. A sinusoidal input is used for foreground calibration. To avoid the complexity of the sinusoidal curve fitting procedure required in [4], we use a sinusoid with a known frequency, where both the sampling clock and the input signal are generated using a single source to guarantee their frequency synchronization. It is demonstrated in Section VI that the proposed LUT-based approach is superior to any linear combination based technique in the presence of strong nonlinearity.

This publication has emanated from research supported in part by S3group, Dublin, Ireland, and in part by a research grant from Science Foundation Ireland (SFI), and is co-funded under the European Regional Development Fund under Grant Number 13/RC/2077.

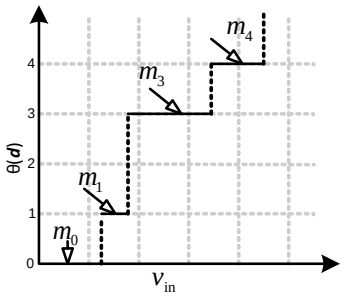

Fig. 1: An illustrative example of an ADC's transfer function.

II. Generic calibration algorithm

This section highlights the proposed calibration technique for a generic ADC whose raw output consists of N binary decisions denoted by an $N \times 1$ vector $\mathbf{d}$. The n^{th} binary output in $\mathbf{d}$ is d_n, for $0 \leq n < N$, where d_0 is the least significant bit.

We can define a mapping function to map any $\mathbf{d}$ to a unique integer via

$$\theta(\mathbf{d}) \triangleq \sum_{n=0}^{N-1} d_n 2^n. \tag{2}$$

For any $\mathbf{d}$, the nominal ADC output can be defined as

$$y \triangleq \sum_{n=0}^{N-1} d_n w_n, \tag{3}$$

where w_n is the nominal weight for the n^{th} binary output (e.g., in radix-2 architectures we have $w_n = 2^n$). However, in the presence of circuit imperfections, the value of y as given by (3) can be an inaccurate representation of the ADC's input.

A memoryless system can be described by a transfer function whose output at any time is a function of the input at that time and no other. In the absence of internal noise, the transfer function for an ideal ADC looks like a staircase as depicted in Figure 1; note that due to circuit impairments, this staircase may not have the same width for all steps. The step width reflects the voltage range assigned for each output by the ADC. For a given output $\mathbf{d}$, the ADC input can be modeled as a sum of

1) the *mid-level* input that is denoted by $m_{\theta(\mathbf{d})}$ as shown in Figure 1; and
2) uniformly distributed quantization noise with *zero mean*.

The goal of the proposed calibration algorithm is to estimate $m_{\theta(\mathbf{d})}$ for all possible $\mathbf{d}$. Figure 2 shows a block diagram for this algorithm. The estimated values for $m_{\theta(\mathbf{d})}$ are stored in the LUT. After calibration, the raw ADC output at the k^{th} time index, denoted by $\mathbf{d}^{(k)}$, is used to access the LUT directly, and the final calibrated output is $\tilde{y}[k]$, an estimate of $m_{\theta(\mathbf{d}^{(k)})}$.

To produce the estimate of $m_{\theta(\mathbf{d})}$ for each $\mathbf{d}$, a sinusoidal input with a known frequency is used. This test signal shall cover most of the input swing, in order to ensure that almost all possible raw outputs are exercised.

The calibration is done in two stages. In the first stage, the input signal is characterized in order to predict the input at any time index k; this stage is outlined in Section III. After characterizing the input, more samples are processed in the second stage to evaluate the average input corresponding to each output level, $m_{\theta(\mathbf{d})}$; Section IV describes this stage in more detail.

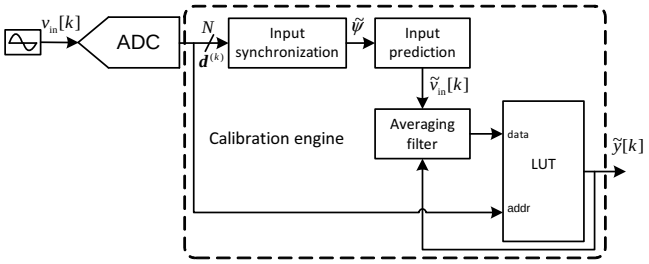

Fig. 2: Block diagram for the proposed generic calibration algorithm.

III. Input signal synchronization

To build the transfer function, we need to know both the output and an estimate for its corresponding input. Since a sinusoidal signal with known frequency is used, we can predict the input at any time after determining the input parameters, i.e., the amplitude and the initial phase.

The sinusoidal input frequency is selected to be $\frac{a}{K} f_s$ where K is a power of 2, f_s is the sampling frequency, and a is a selected positive integer less than $K/2$ such that a and K are relatively prime. We can write the input as follows:

$$v_{\text{in}}[k] = A\cos\left(\frac{2\pi a}{K}k + \phi_0\right) \tag{4}$$

$$= \frac{A}{2}\left(e^{j\left(\frac{2\pi a}{K}k+\phi_0\right)} + e^{-j\left(\frac{2\pi a}{K}k+\phi_0\right)}\right), \tag{5}$$

where A is the input amplitude and ϕ_0 is the initial phase.

Let ψ denote the average of the product of $v_{\text{in}}[k]$ with $e^{-j\frac{2\pi a}{K}k}$ over K samples, i.e.,

$$\psi = \frac{1}{K}\sum_{k=0}^{K-1} e^{-j\frac{2\pi a}{K}k} v_{\text{in}}[k] \tag{6}$$

$$= \frac{A}{2} e^{j\phi_0}, \tag{7}$$

where (7) holds due to the fact that averaging takes place over an integer number of cycles of the input. The value of ψ contains information about both A and ϕ_0; this is sufficient to predict the input at any time instant. Note that (6) resembles the calculations needed for the a^{th} component of the Discrete Fourier Transform of size K.

However, (6) may not be used directly to find ψ, since we do not know $v_{\text{in}}[k]$. Therefore, we substitute $v_{\text{in}}[k]$ with the value y calculated in (3); the corresponding estimate of ψ may then be written as

$$\tilde{\psi} \triangleq \frac{1}{K}\sum_{k=0}^{K-1}\left(e^{-j\frac{2\pi a}{K}k}\sum_{n=0}^{N-1} w_n d_n^{(k)}\right). \tag{8}$$

The averaging operation in (8) ensures that this serves as an accurate approximation to ψ which is robust to the effects of noise and nonlinear impairments.

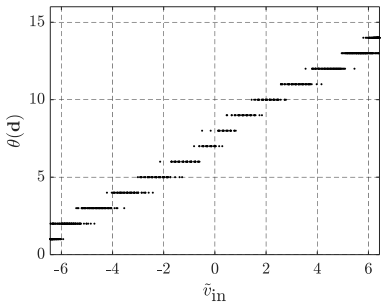

Fig. 3: Obtained plot for the transfer function in the presence of noise.

IV. Building the LUT

From (4) and (7), it follows that

$$\tilde{v}_{\text{in}}[k] = 2\Re\big(\tilde{\psi}e^{j\frac{2\pi\alpha}{K}k}\big). \tag{9}$$

Note that the input is periodic with period K samples, i.e., $\tilde{v}_{\text{in}}[k] = \tilde{v}_{\text{in}}[k \pmod K]$. Hence we are able to predict the input at any time index k after obtaining $\tilde{\psi}$ in the first calibration stage.

To obtain $\bar{m}_{\theta(\mathbf{d})}$, an estimate of $m_{\theta(\mathbf{d})}$ for each output level $\mathbf{d}$, we observe another F outputs indexed by k where $0 \le k < F$. For each output $\mathbf{d}^{(k)}$, we predict $\tilde{v}_{\text{in}}[k]$ using (9). Let $\mathcal{S}^{(\mathbf{d})}$ be a set that contains the time indices when the observed ADC output is $\mathbf{d}$; $\mathcal{S}^{(\mathbf{d})} = \{k \mid \mathbf{d}^{(k)} = \mathbf{d}, 0 \le k < F\}$. The estimate $\bar{m}_{\theta(\mathbf{d})}$ can be defined according to:

$$\bar{m}_{\theta(\mathbf{d})} \triangleq \frac{\sum_{s\in\mathcal{S}^{(\mathbf{d})}} \tilde{v}_{\text{in}}[s]}{|\mathcal{S}^{(\mathbf{d})}|}. \tag{10}$$

For demonstration purposes, we can collect many ADC outputs with their corresponding predicted input from (9), and we can plot the ADC output against its estimated input, where each pair is plotted as a *point*. If we collect enough samples, the obtained diagram closely resembles the ADC's transfer function. Figure 3 shows an example for the obtained plot with $N = 4$ for a radix-2 SAR ADC suffering from thermal noise. It can be seen that there is an overlap between the ADC's output levels on the $\tilde{v}_{\text{in}}$ axis. This overlap is a result of the noise inside the ADC; the ADC can produce different possible outputs for exactly the same input. Of course, the overlap depth is a direct indication of this noise level.

V. Implementation

The implementation for the key computations of the proposed algorithm is shown in Figure 4, which realizes both the input synchronization block (8) and prediction block (9) to support real-time calculations. Both equations require the calculation of $\beta[k] = e^{j\frac{2\pi\alpha}{K}k}$, which can be performed using a simple coordinate rotation digital computer (CORDIC) algorithm as shown in the figure. For each of the two nontrivial multipliers in the figure, only half of a full *complex* multiplier is required.

After processing the first K samples and obtaining $\tilde{\psi}$, both stages (described in Sections III and IV) can be run in parallel, so that $\tilde{\psi}$ is updated periodically to track small errors in the input frequency.

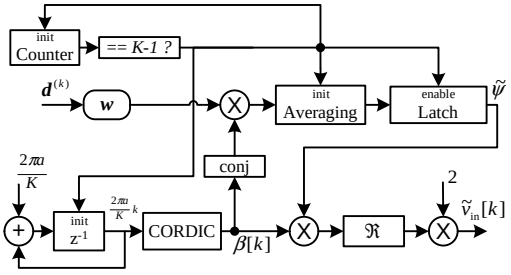

Fig. 4: Implementation for the input synchronization and prediction blocks.

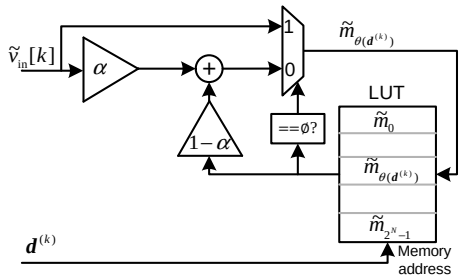

Fig. 5: One-pole low-pass filter used for the averaging process.

The LUT shown in Figure 2 is used to store $\tilde{m}_{\theta(\mathbf{d})}$, which is an approximation for $\bar{m}_{\theta(\mathbf{d})}$ in (10). This table has 2^N entries that are initialized to $\varnothing$, an invalid value that is used to detect any non-updated entries during the calibration process. The table width equals $N + b$, where b is a selected integer that is allocated for the fractional part of $\tilde{m}_{\theta(\mathbf{d})}$. This width can be further reduced if the table is used to store the *difference* between $\tilde{m}_{\theta(\mathbf{d})}$ and the output obtained by (3) with nominal weights.

To implement the averaging process in (10), we use the one-pole filter shown in Figure 5, where $0 < \alpha < 1$; for each $\mathbf{d}^{(k)}$ we update the content of the LUT according to

$$\tilde{m}_{\theta(\mathbf{d}^{(k)})} \leftarrow\!\!\shortmid \begin{cases} \tilde{v}_{\text{in}}[k], & \text{when } \tilde{m}_{\theta(\mathbf{d}^{(k)})} = \varnothing \\ (1-\alpha)\tilde{m}_{\theta(\mathbf{d}^{(k)})} + \alpha\tilde{v}_{\text{in}}[k], & \text{otherwise.} \end{cases} \tag{11}$$

Since we observe a limited number of samples, there may exist entries in the LUT which are not updated during the calibration process, i.e., these LUT entries remain equal to $\varnothing$. For such entries, we use the nearest updated entries to find an estimate for the non-updated entries using straightforward interpolation techniques.

Upon the completion of the foreground calibration, we can switch to normal mode where $\tilde{m}_{\theta(\mathbf{d})}$ stored in the LUT are used to map the raw values to the calibrated ADC outputs.

VI. Results

Although this algorithm is generic and can be applied to different ADC architectures, the results reported here use as example a differential radix-2 SAR ADC architecture which suffers from a variety of nonlinearity sources, specifically capacitor mismatch, S/H imperfection and settling time error.

Matlab simulations are used to verify the proposed calibration algorithm. We target a 10-bit radix-2 SAR ADC,

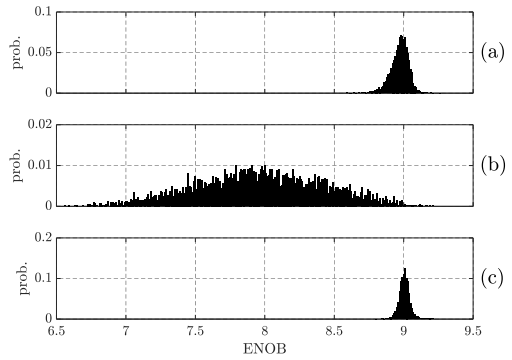

Fig. 6: ENOB distribution for SAR ADC with (a) linear combination using ideal weights, (b) without calibration, and (c) proposed calibration algorithm.

where the unit capacitance, c_u, suffers from mismatch having a Gaussian distribution with standard deviation $\sigma_u = 0.1c_u$. To model a realistic ADC, comparator noise is added to limit the ENOB to 9 bits. For the calibration algorithm we used $\alpha = 2^{-3}$, $a = 409$, $K = 2^{12}$ and $F = 61440$, so that the total number of samples used in calibration is $K + F = 2^{16}$. The final ADC output is truncated to $N = 10$ bits, while the LUT values are stored as $N + b = 15$ bits.

In the following tests, we measured the ENOB after foreground calibration using a sinusoidal signal with different frequencies to guarantee that the obtained LUT does not correlate with a specific input frequency.

A Monte Carlo simulation is used to verify the proposed algorithm's performance in compensating the capacitor mismatch problem, where 5000 different capacitor sets are used. Figure 6 shows the ENOB distributions when 1) a linear combination with ideal weights is used to obtain the output, 2) no calibration is used, 3) the proposed calibration algorithm is used. The average ENOB values are 8.97, 7.96 and 9.00 bits for these cases respectively. The proposed algorithm gives a slightly better performance compared to the linear combination with ideal weights (note that the latter has limited ability to deal with nonlinear circuit impairments).

For simulation of the settling time, we used the CDAC settling model available in [3]. Since the foreground calibration is done using the same sampling frequency as in normal operation, the LUT values $\tilde{m}_{\theta(\mathbf{d})}$ are specifically tailored to the appropriate value of τ_s/τ, which aids in providing a better performance compared to the linear combination based approach. However, the proposed algorithm is not able to avoid the problem of missing codes which will occur for sufficiently small values of τ_s/τ.

Figure 7 compares the measured ENOB obtained using the proposed algorithm with that obtained using the linear combination with ideal weights. It can be observed that the degradation of the ENOB due to reducing τ_s/τ is smaller when the proposed algorithm is used. This can allow an increase in the sampling frequency with a very minor loss in performance.

The proposed algorithm was also tested in the presence of S/H nonlinearity. Figure 8 shows the obtained results on changing the nonlinearity coefficients c_2 and c_3 in (1). It can be observed that the performance after calibration has far greater

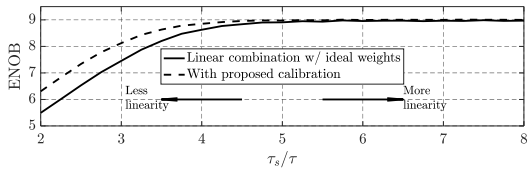

Fig. 7: Measured ENOB with varying τ_s/τ.

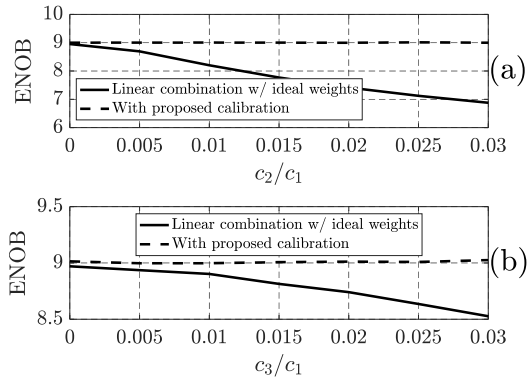

Fig. 8: Measured ENOB with varying S/H nonlinearity coefficients.

immunity to the nonlinearity of the S/H.

VII. Conclusion

In this paper, a generic foreground calibration technique has been presented to compensate the effect of various nonlinearity sources in SAR, pipeline and hybrid architecture ADCs. A sinusoidal input with known frequency is used to estimate the mid-level input for each raw ADC output, these values being stored in a LUT. The proposed technique avoids the complexity of post-processing often needed for calibration, replacing it with a simple memory access. We also proposed various simplifications to facilitate a real-time hardware implementation which reduces the calibration time. The proposed algorithm was verified using a SAR ADC model suffering from various nonlinear impairments. When compared to a genie-based linear combination approach, the algorithm showed superior capacitor mismatch calibration, increased tolerance to settling time reduction and significant improvements in the presence of second and third order nonlinear terms.

References

[1] C. Grace, P. Hurst, and S. Lewis, "A 12-bit 80-MSample/s pipelined ADC with bootstrapped digital calibration," *IEEE Journal of Solid-State Circuits*, vol. 40, no. 5, pp. 1038–1046, 2005.

[2] W. Liu, P. Huang, and Y. Chiu, "A 12-bit, 45-MS/s, 3-mW redundant successive-approximation-register analog-to-digital converter with digital calibration," *IEEE Journal of Solid-State Circuits*, vol. 46, no. 11, pp. 2661–2672, 2011.

[3] A. Salib, B. Cardiff, and M. F. Flanagan, "Blind SAR ADC capacitor mismatch calibration," *IEEE International Midwest Symposium on Circuits and Systems*, August 2017.

[4] A. J. Ginés, E. J. Peralías, and A. Rueda, "Black-box calibration for ADCs with hard nonlinear errors using a novel INL-based additive code: A pipeline ADC case study," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 64, no. 7, pp. 1718–1729, 2017.

[5] V. Kerzrho, V. Fresnaud, D. Dallet, S. Bernard, and L. Bossuet, "Fast digital post-processing technique for integral nonlinearity correction of analog-to-digital converters: validation on a 12-bit folding-and-interpolating analog-to-digital converter," *IEEE Transactions on Instrumentation and Measurement*, vol. 60, no. 3, pp. 768–775, 2010.