Digital Background-Calibration Algorithm for "Split ADC" Architecture

John A. McNeill, *Senior Member, IEEE*, Michael C. W. Coln, *Member, IEEE*, D. Richard Brown, *Member, IEEE*, and Brian J. Larivee, *Member, IEEE*

***Abstract*—The "split ADC" architecture enables continuous digital background calibration by splitting the die area of a single ADC design into two independent halves, each converting the same input signal. The two independent outputs are averaged to produce the ADC output code. The difference of the two outputs provides information for a background-calibration algorithm. Since both ADCs convert the same input, when correctly calibrated, their outputs should be equal, and the difference should be zero. Any nonzero difference provides information to an error-estimation algorithm, which adjusts digital-calibration parameters in an adaptive process similar to a least mean square algorithm. This paper describes the calibration algorithm implemented in the specific realization of a 16-bit 1-MS/s algorithmic cyclic ADC. In addition to correcting ADC linearity, the calibration and estimation algorithms are tolerant of offset error and remove linear scale-factor-error mismatch between the ADC channels. Simulated results are presented confirming self-calibration in approximately 10 000 conversions, which represents an improvement of four orders of magnitude over previous statistically based calibration algorithms.**

***Index Terms*—Adaptive systems, analog–digital conversion, calibration, digital background calibration, mixed analog–digital integrated circuits, self-calibrating.**

I. Introduction

A. Goals

THE TREND in submicrometer CMOS ADC design is toward all-digital self-calibrating ADC architectures. The goal of the work described in this paper was to develop a self-calibrating ADC architecture specifically designed to exploit the advantages of scaling in deep-submicrometer CMOS. Three criteria were defined for the desired architecture.

1) Digital implementation: In submicrometer CMOS, the preferred tradeoff is to move complexity into the digital domain whenever possible. The desired architecture should implement all calibration and error correction digitally while relaxing required performance and complexity of analog circuitry.
2) Background calibration: The calibration process should operate continuously in the background, without interrupting the foreground processing of the ADC input signal.
3) Deterministic calibration procedure: The time constant of calibration adaptation should be short enough to track out parameter variations due to environmental influences such as temperature. In this paper, the term "deterministic" is used to draw a contrast with statistical methods, which are poorly suited to calibration of high-resolution ADCs.

Manuscript received June 15, 2006; revised August 25, 2007. First published July 22, 2008; current version published February 11, 2009. This work was supported in part by Analog Devices and the National Science Foundation. This paper was recommended by Associate Editor I. Bell.

J. A. McNeill and D. R. Brown are with the Electrical and Computer Engineering Department, Worcester Polytechnic Institute, Worcester, MA 01609 USA (e-mail: mcneill@ece.wpi.edu).

M. C. W. Coln and B. J. Larivee are with Analog Devices, Wilmington, MA 01887 USA.

Digital Object Identifier 10.1109/TCSI.2008.2001830

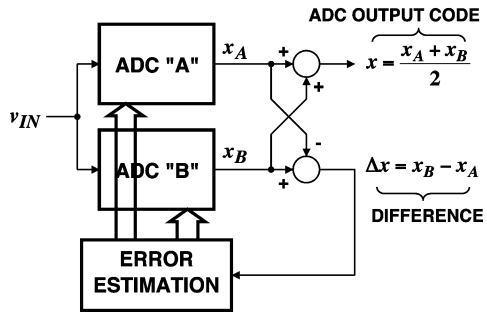


Fig. 1. "Split ADC" architecture.

This paper describes a calibration algorithm for a 16-bit 1-MS/s algorithmic ADC [1], [2] implementing the "split ADC" architecture. In contrast to most previous techniques [3]–[13], the split ADC architecture enables a deterministic digital-calibration procedure, operating continuously in the background, with a minimal impact on analog complexity. Use of parallel ADC paths to cancel input signal contribution was independently developed and presented in [14]–[16], while that approach is similar to the work presented in this paper; the approaches differ in important details of implementation.

This paper is organized as follows. A general overview of the split ADC architecture as implemented in this paper is provided in Section I-B. Sections II and III describe details of the least mean square (LMS) calibration algorithm and error-estimation theory, respectively, with implementation specific to the 16-b 1-MS/s cyclic ADC in [1] and [2]. Compared with the circuit-level design work presented in [1] and [2], the novel content of this paper is in Sections IV–VI, which describe the effect of offset and scale-factor errors, the iterative matrix-solution technique included in the LMS algorithm, and results showing how ADC performance is affected by choice of system parameters.

B. Split ADC Concept

The concept of the split ADC architecture is shown in Fig. 1. The ADC is split into two channels, each converting the same input and producing individual output codes x_A and x_B. The average of the two outputs is reported as the ADC output code x. The background-calibration signal is developed from the difference Δx between codes x_A and x_B. If both ADCs are correctly

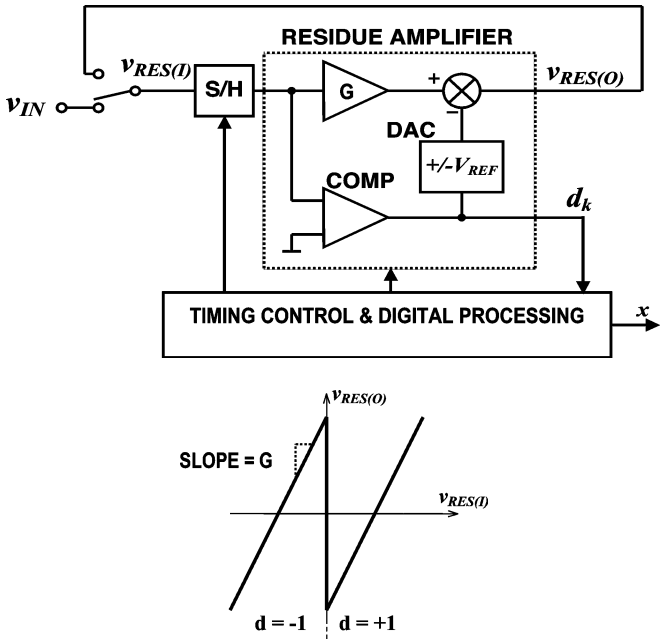


Fig. 2. (a) Cyclic ADC block diagram. (b) Residue-amplifier plot.

calibrated, the two outputs will agree, and the difference Δx will be zero. In the presence of nonzero differences, the pattern of "disagreements" in Δx can be examined in an error-estimation process to adjust calibration parameters in each ADC and drive the difference and the ADC errors to zero.

Comparison with statistical background-calibration techniques shows the advantage of using the difference Δx for the calibration signal. In [3], for example, extracting the DAC mismatch signal in the background requires that a sufficiently large number of samples have been collected to decorrelate the unknown signal at the ADC input. In contrast, for the split ADC approach, the magnitude of the unknown ADC input signal is greatly reduced by the subtraction in the calibration signal path. Moreover, it is not necessary to accumulate a large number of conversions to decorrelate the input signal.

This concept is referred to as a "split ADC" (rather than a "dual ADC"), since it essentially splits the analog area of a single ADC design and, as shown in [1] and [2], has negligible impact on analog complexity in terms of overall area, power, bandwidth, or noise performance.

II. Split ADC Cyclic Calibration

A. Cyclic ADC Review

For the implementation of the split ADC concept described in this paper, an algorithmic (or cyclic) converter was chosen for simplicity, since the only parameter needed to calibrate ADC linearity is the gain of the residue amplifier. To show how the residue-amplifier gain is involved in the calibration process, consider the example of a conventional cyclic ADC shown in Fig. 2(a). At the start of the conversion, the S/H switch is in the $v_{\rm IN}$ position. The cyclic converter samples the input, compares it to a threshold, applies the cyclic gain G, and adds or subtracts a reference depending on the comparator decision. The input-to-residue-output characteristic implemented is shown in Fig. 2(b). In the first cycle, the input $v_{\rm IN}$ is mapped to a residue $v_{\rm RES}(1)$ by

$$v_{\rm IN} \Rightarrow \underbrace{G \cdot v_{\rm IN} - d_1 \cdot V_{\rm REF}}_{\rm RESIDUE} = v_{\rm RES(1)}. \quad (1)$$

For all following cycles, the S/H input is switched to the residue-amplifier output. For the second cycle, the first cycle residue $v_{\rm RES(1)}$ is the input, so by the same process we get $v_{\rm RES(2)}$

$$v_{\rm RES(2)} = G\underbrace{(G \cdot v_{\rm IN} - d_1 \cdot V_{\rm REF})}_{v_{\rm RES(1)}} - d_2 \cdot V_{\rm REF}. \quad (2)$$

Moreover, for the third cycle

$$v_{\rm RES(3)} = G \cdot \underbrace{[G \cdot (G \cdot v_{\rm IN} - d_1 \cdot V_{\rm REF}) - d_2 \cdot V_{\rm REF}]}_{v_{\rm RES(2)}} - d_3 \cdot V_{\rm REF}. \quad (3)$$

Pausing here to rearrange, we obtain

$$v_{\rm RES(3)} = G^3 (v_{\rm IN}) - \left[G^2 d_1 + G^1 d_2 + G^0 d_3\right](V_{\rm REF}). \quad (4)$$

Rearranging to solve for $v_{\rm IN}/V_{\rm REF}$ (what we want for the output code of an ADC) gives

$$\frac{v_{\rm IN}}{V_{\rm REF}} = \frac{1}{G} d_1 + \frac{1}{G^2} d_2 + \frac{1}{G^3} d_3 + \frac{1}{G^3} \frac{v_{\rm RES(3)}}{V_{\rm REF}}. \quad (5)$$

In the general case of N cycles, the converter provides a sequence of N decisions d_k which satisfy

$$\frac{v_{\rm IN}}{V_{\rm REF}} = \underbrace{\sum_{k=1}^{N} \left(\frac{1}{G}\right)^k d_k}_{\substack{OUTPUT \\ CODE\ x}} + \underbrace{\frac{1}{G^N} \frac{v_{\rm RES(N)}}{V_{\rm REF}}}_{\substack{QUANTIZATION \\ ERROR}}. \quad (6)$$

Of the two terms on the right-hand side of (6), the first term represents the ideal output code x of the ADC which is determined from the comparator decisions d_k. Digital calibration and correction is implemented by estimating the cyclic gain G to the converter accuracy, calculating the powers of $1/G$ for use as decision weights in a lookup table (LUT), and accumulating the output code x as the decisions d_k become available in the cyclic conversion process. To provide margin for errors such as comparator offset and noise [9], a gain $G < 2$ is usually chosen.

The second term in (6) from the last (Nth) residue corresponds to the quantization error of the ADC. Choosing $G < 2$ implies that N cycles will provide fewer than N bits of resolution; the second term in (6) can be used to determine how many cycles are required to achieve a desired resolution.

B. Split ADC Cyclic Implementation

A simplified system block diagram of the split ADC implementation is shown in Fig. 3. The analog portion of each cyclic ADC consists of an input S/H, a 16-bit linear gain block (nominal gains $G_A = G_B = 1.92$), comparators, and a three-level DAC. The output of the analog subsystem is a three-level $(-1/0/+1)$ decision for each cycle of the conversion process.

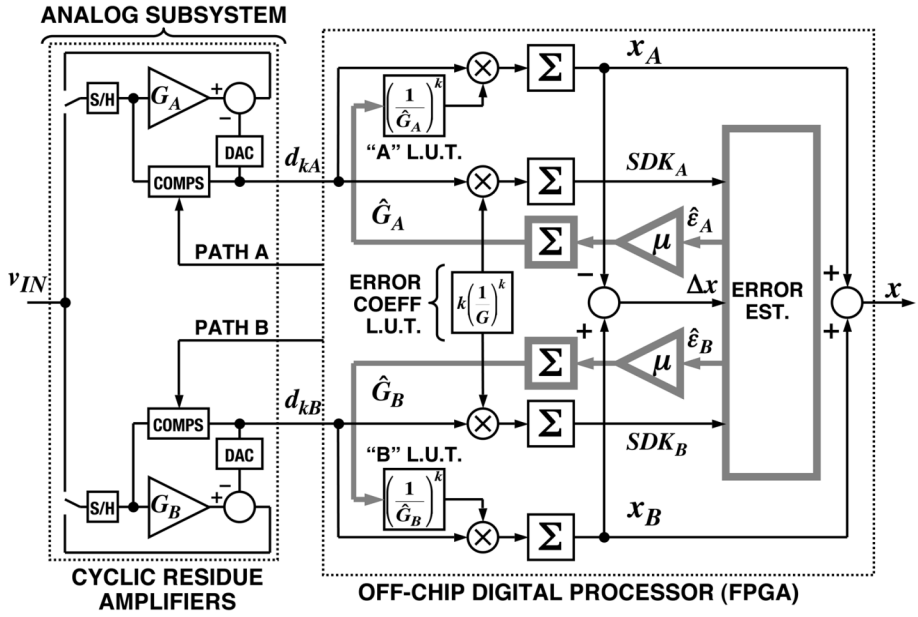


Fig. 3. Converter block diagram.

For each side of the split, digital outputs x_A and x_B are accumulated from the comparator decisions using a LUT containing the cycle decision weights, which are calculated from the gain estimates $\hat{G}_A$ and $\hat{G}_B$.

The calibration process, indicated by the thick gray line as shown in Fig. 3, operates in the digital domain so that $\hat{G}_A$, $\hat{G}_B$, and their associated LUTs are correct to within converter accuracy. It should be emphasized that this process operates continuously and in the background; foreground conversion of the input signal is not interrupted.

Note that the block diagram in Fig. 3 shows separate S/H circuitry for each channel. Depending on the maximum input $\mathrm{d}v_{\mathrm{IN}}/\mathrm{d}t$ and the mismatch in timing between channels, the dynamic error from sampling v_{IN} through two separate paths may limit system performance. Although this was not a problem in the case in [1] and [2], if necessary, the effects of dynamic errors can be mitigated by modifying the S/H sampling circuitry so that the critical timing path is common to both channels.

C. LMS Calibration Loop

To consider the operation of the LMS calibration loop, we first define ϵ_A and ϵ_B as the fractional error in estimates $\hat{G}_A$ and $\hat{G}_B$ relative to the true analog gains G_A and G_B

$$\begin{aligned}\hat{G}_A &= G_A(1+\varepsilon_A)\\ \hat{G}_B &= G_B(1+\varepsilon_B).\end{aligned} \tag{7}$$

The error-estimation process, described in Section III as follows, provides continuously updated zero-bias estimates $\hat{\varepsilon}_A$ and $\hat{\varepsilon}_B$ of the errors in the estimated gains $\hat{G}_A$ and $\hat{G}_B$. Estimates $\hat{\varepsilon}_A$ and $\hat{\varepsilon}_B$ are used in a negative-feedback LMS procedure, shown in the thick gray line in Fig. 3, to update $\hat{G}_A$ and $\hat{G}_B$

$$\hat{G}_A^{(\mathrm{new})} = \hat{G}_A^{(\mathrm{old})} - \mu_G \hat{\varepsilon}_A \tag{8a}$$

$$\hat{G}_B^{(\mathrm{new})} = \hat{G}_B^{(\mathrm{old})} - \mu_G \hat{\varepsilon}_B. \tag{8b}$$

As the $\hat{G}_A$ and $\hat{G}_B$ are periodically updated, the decision-weight LUT is recalculated. Equilibrium is reached at the correct G_A and G_B values when the estimated errors $\hat{\varepsilon}_A$ and $\hat{\varepsilon}_B$ reach zero (on average). The LMS coefficient μ_G controls the time constant of the calibration adaptation and is subject to a tradeoff between accuracy and speed of adaptation. Note an advantage of the LMS technique: The estimates $\hat{\varepsilon}_A$ and $\hat{\varepsilon}_B$ need not be accurate, as long as they are zero-bias and (on average) point the convergence of (8) in the correct direction. The general LMS technique is well established [4], [11], [19]; the novel contributions of this paper are in the error-estimation technique described in Sections III–VI.

III. Error Estimation

A. Basic Error-Estimation Theory

To simplify the development of the error-estimation theory, we will temporarily ignore offset and scale-factor mismatches between the A and B ADCs. The effect of these errors will be discussed in Section IV.

For each side of the split, digital outputs x_A and x_B are accumulated from the comparator decisions using

$$x_A = \sum_{k=1}^{N} \left(\frac{1}{\hat{G}_A}\right)^k d_{kA} \tag{9a}$$

$$x_B = \sum_{k=1}^{N} \left(\frac{1}{\hat{G}_B}\right)^k d_{kB} \tag{9b}$$

in which d_k is the comparator decision in the kth cycle, and the required decision weights are stored in a LUT calculated from powers of the gain estimates $\hat{G}_A$ and $\hat{G}_B$. To consider the effect of errors in $\hat{G}_A$ and $\hat{G}_B$, we suppose that the estimates

in (9) are in error as defined in (7). Substituting (7) into (9) and approximating $1/(1+\varepsilon)^k \approx 1 - k\varepsilon$ gives

$$x_A = \underbrace{\sum_{k=1}^{N}\left(\frac{1}{G_A}\right)^k d_{kA}}_{\text{CORRECT}x} - \underbrace{\varepsilon_A \overbrace{\sum_{k=1}^{N} k\left(\frac{1}{G_A}\right)^k d_{kA}}^{\text{SDK}A}}_{\text{"}A\text{"}\,ERROR}$$
$$= x - \varepsilon_A\left[\text{SDK}_A\right] \tag{10a}$$

$$x_B = \underbrace{\sum_{k=1}^{N}\left(\frac{1}{G_B}\right)^k d_{kB}}_{\text{CORRECT}x} - \underbrace{\varepsilon_B \overbrace{\sum_{k=1}^{N} k\left(\frac{1}{G_B}\right)^k d_{kB}}^{\text{SDK}B}}_{\text{"}B\text{"}\,ERROR}$$
$$= x - \varepsilon_B\left[\text{SDK}_B\right]. \tag{10b}$$

The expressions for the actual ADC output codes each consist of two terms. Comparing with (6) shows that the first term (indicated with "CORRECTx") corresponds to the correct output code. Since both ADCs are converting the same analog input, these must be equal (to within quantization error) even if the decision trajectories d_{kA} and d_{kB} are different. The second term corresponds to the errors in the "A" and "B" output codes. Note that the error is simply proportional to ϵ_A and ϵ_B, the fractional error in the gain estimates $\hat{G}_A$ and $\hat{G}_B$. The proportionality terms, designated as error coefficients SDK_A and SDK_B, are determined by the decision trajectories d_{kA} and d_{kB} and weighting terms $k\,(1/G)^k$. As shown in Fig. 3, the SDK_A and SDK_B values can be accumulated during each conversion using the comparator decisions and a separate error-coefficient LUT. Note that, since the errors ϵ_A and ϵ_B are ultimately forced to zero by the calibration loop, the accuracy requirements for SDK_A and SDK_B in the error-estimation process are not as stringent as for the x_A and x_B code LUTs. Thus, a single LUT stored in ROM is used with values precalculated using the nominal gain $G = 1.92$.

When the two output codes are averaged to obtain the overall ADC output code, the result is

$$\frac{x_A + x_B}{2} = x - \frac{\varepsilon_A\left[\text{SDK}_A\right]}{2} - \frac{\varepsilon_B\left[\text{SDK}_B\right]}{2}. \tag{11}$$

Therefore, if the errors ϵ_A and ϵ_B can be driven sufficiently close to zero, the overall ADC output code will be the correct value x.

When we subtract to get the difference Δx, the "CORRECTx" terms cancel, and the result is

$$\Delta x = x_B - x_A = \varepsilon_A\left[\text{SDK}_A\right] + \varepsilon_B\left[-SDK_B\right]. \tag{12}$$

Thus, the difference Δx depends only on the estimate errors ϵ_A and ϵ_B, with error coefficients SDK_A and SDK_B.

In principle, we could solve a 2×2 matrix equation using Δx, SDK_A, and SDK_B values from two conversions to estimate ϵ_A and ϵ_B. This is indicated in the equation as follows in which the "1" and "2" subscripts indicate values from two different ADC conversions:

$$\begin{aligned}\Delta x_1 &= \varepsilon_A\left[\text{SDK}A_1\right] + \varepsilon_B\left[-\text{SDK}B_1\right]\\ \Delta x_2 &= \varepsilon_A\left[\text{SDK}A_2\right] + \varepsilon_B\left[-\text{SDK}B_2\right]\\ \Rightarrow \begin{bmatrix}\Delta x_1\\ \Delta x_2\end{bmatrix} &= \begin{bmatrix}\text{SDK}A_1 & -\text{SDK}B_1\\ \text{SDK}A_2 & -\text{SDK}B_2\end{bmatrix}\begin{bmatrix}\varepsilon_A\\ \varepsilon_B\end{bmatrix}.\end{aligned} \tag{13}$$

A difficulty in using this approach with the simple cyclic ADC shown in Fig. 2 is that the decision paths d_{kA} and d_{kB} (and, therefore, the coefficients SDK_A and SDK_B) are determined entirely by the input signal. In the case of a dc input, the coefficients SDK_A and SDK_B are unchanged from conversion "1" to "2", the 2×2 matrix in (13) is singular, and no information about ϵ_A and ϵ_B can be determined.

B. Need for Multiple Residue-Mode Amplifier

To avoid matrix singularity, the coefficients SDK_A and SDK_B must be varied on a conversion-to-conversion basis. This is achieved by varying the decision trajectories using a multiple residue-mode amplifier, shown in Fig. 4(a) and described in more detail in [1] and [2]. The redundancy enabled by choosing $G < 2$ allows multiple diversity of valid decision sequences corresponding to a given input.

Fig. 4(b) shows input–output plots of the four possible residue modes, as well as for each plot of the resulting SDK coefficients versus output code over the ADC full-scale (FS) range. The residue mode is determined by the 2-bit "PATH" digital control, which allows a choice of different residue modes for the A and B converters, as well as variation of residue mode on a conversion-to-conversion basis. By varying residue modes, it is possible to force different decision trajectories d_{kA} and d_{kB} and, thereby, vary the SDK_A and SDK_B coefficients.

For additional insight into the choice of residue modes, consider the examples shown in Table I. For each residue mode, the calculated SDK coefficient values are given for representative input levels of 0.1 and 0.9 of FS. For an input of 0.1 FS, for example, the value of the SDK coefficient ranges from a maximum of $+0.37$ in the "HIGH" residue mode to a minimum of -0.42 in the "LOW" mode. When both channels are exercised through different combinations of residue modes, the result is sufficient variation of the SDK_A and SDK_B coefficients that the matrix in (13) is nonsingular, even in the case of a dc input. Therefore, a "busy" input is not required to extract calibration information for ϵ_A and ϵ_B. Intuitively, we see from (12) that if SDK_A and SDK_B are varied in an independent fashion, then the only way for Δx to always be zero is for ϵ_A and ϵ_B to both be zero.

Note from the column in Table I for an input of 0.9 FS that the variation between maximum and minimum coefficient values is reduced as the input nears FS. This is due to the limited diversity of available decision trajectories for larger inputs. As will be seen in Section IV, the reduced min-to-max variation will reduce the value of matrix coefficients used to solve for ϵ_A and ϵ_B. The dependence of matrix coefficients on signal level is the reason for the signal-related change in the rate of calibration convergence for the entire ADC which is seen in Section VI-C.

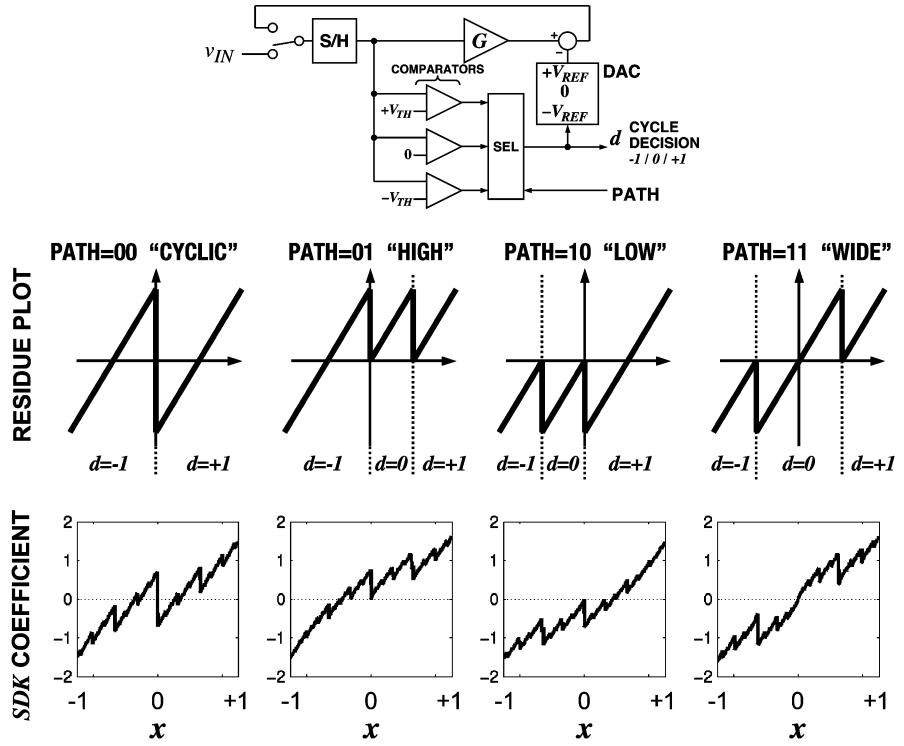


Fig. 4. (a) Multiple residue-mode cyclic amplifier. (b) Residue modes and SDK coefficients.

TABLE I
EXAMPLE SDK COEFFICIENT VALUES

		INPUT	
		0.1 FS	0.9 FS
RESIDUE MODE	CYCLIC	- 0.36	+ 1.23
	HIGH	+ 0.37	+ 1.40
	LOW	- 0.42	+ 1.14
	WIDE	+ 0.36	+ 1.37
	Max - Min	**0.79**	**0.26**

C. Modifications Required to Basic

In practice, there are several reasons to consider in modifying the simple approach embodied in (13).

1) In the presence of offset and scale-factor errors in the A and B converters, there will be systematic variations in the Δx values which are not related to the true values of ϵ_A and ϵ_B. To separate out these effects requires information from additional conversions and modification of the matrix equation, as described in Section IV as follows.
2) In the presence of noise in the A and B conversions, there will be additional random noise in the Δx values which are not related to the true values of ϵ_A and ϵ_B. Therefore, the results of solving the matrix equation should be considered as estimates $\hat{\varepsilon}_A$ and $\hat{\varepsilon}_B$.
3) To simplify digital hardware, matrix inversion should be avoided; rather, an iterative matrix approximation procedure is used to develop the estimates $\hat{\varepsilon}_A$ and $\hat{\varepsilon}_B$. As will be described in Section V, the procedure is merged into the iterative nature of the LMS algorithm described previously in Section II-C. Dealing with the matrix in this way has two advantages: The digital complexity of matrix inversion is avoided, and the effects of random noise are effectively averaged out over the time constant of the LMS adaptation.

IV. OFFSET AND SCALE-FACTOR ERRORS

A. Difficulty Due to Offset and Scale-Factor Errors

The previous analysis ignored offset errors in each of the A and B channels, as well as the possibility of a scale-factor gain error between channels. Intuitively, it can be anticipated that offset and scale-factor errors pose a challenge for the "split ADC" calibration procedure: The fundamental principle is that the difference Δx is zero when both ADCs are correctly calibrated and that any nonzero Δx is an indication that ADC linearity needs to be corrected. The problem is that, even if ADC linearity is correctly calibrated, any mismatch in offset or scale-factor errors between the ADC channels will give rise to a nonzero difference Δx, which will corrupt the linearity calibration procedure. If offset and scale-factor errors are not somehow taken into account, the ADC will fail to converge to correct cyclic gain estimates G_A and G_B; rather, in trying to drive Δx to zero, the calibration loops incorrectly attempt to compensate for offset and scale-factor errors by adjusting the cyclic gain estimates, which results in ADC nonlinearity.

It must be emphasized that the procedure described in this paper only makes calibration of ADC linearity *insensitive* to the presence of offset and scale-factor errors; absolute offset and linear scale-factor errors in the ADC output codes are *not* removed or corrected. In many applications, absolute accuracy correction is not necessary; end users can often tolerate offset and scale-factor errors as long as ADC linearity is maintained. For applications which require absolute accuracy, additional analog complexity is in general required, for example, by taking the ADC offline to apply zero and reference voltage inputs.

B. Modeling Offset and Scale-Factor Errors

Before considering how offset and scale-factor errors affect the Δx difference and calibration procedure, we first model these effects by modifying (10a) and (10b) as follows:

$$x_A = (1+g_A)\,x + x_{OSA} - \varepsilon_A\,[\mathrm{SDK}A] \tag{14a}$$
$$x_B = (1+g_B)\,x + x_{OSB} - \varepsilon_B\,[\mathrm{SDK}B] \tag{14b}$$

where x_{OSA}, x_{OSB} and g_A, g_B represent output-referred offset and scale-factor errors.

Taking the average of (14a) and (14b) to obtain the overall ADC output code gives

$$\frac{x_A+x_B}{2} = x\left(1+\frac{g_A+g_B}{2}\right) + \frac{x_{OSA}+x_{OSB}}{2} - \frac{\varepsilon_A\,[\mathrm{SDK}_A]}{2} - \frac{\varepsilon_B\,[\mathrm{SDK}_B]}{2}. \tag{15}$$

Since the we do not need or intend to correct offset and scale-factor errors in the overall ADC output code, we define "$\mathrm{CORRECT}x$" including the effect of offset and scale-factor errors

$$x' = x\left(1+\frac{g_A+g_B}{2}\right) + \frac{x_{OSA}+x_{OSB}}{2}. \tag{16}$$

With this definition and defining mismatch variables

$$g = g_B - g_A \tag{17}$$
$$x_{OS} = x_{OSB} - x_{OSA}. \tag{18}$$

It can be shown that, assuming that second-order terms are negligible, (14a) and (14b) can be expressed as

$$x_A = x' - \left(\frac{g}{2}\right)x - \frac{x_{OS}}{2} - \varepsilon_A\,[\mathrm{SDK}A] \tag{19a}$$
$$x_B = x' + \left(\frac{g}{2}\right)x + \frac{x_{OS}}{2} - \varepsilon_B\,[\mathrm{SDK}B]\,. \tag{19b}$$

Averaging (19a) and (19b) in accordance with (11) gives

$$\frac{x_A+x_B}{2} = x' - \frac{\varepsilon_A\,[\mathrm{SDK}_A]}{2} - \frac{\varepsilon_B\,[\mathrm{SDK}_B]}{2}. \tag{20}$$

Therefore, if the errors ϵ_A and ϵ_B can be driven sufficiently close to zero, the overall ADC output code will be the new "correct" x', since the definition of x' includes the effects of nonzero offset and scale factor.

When we subtract to get the difference Δx, however, the result is

$$\Delta x = x_B - x_A = \varepsilon_A\,[\mathrm{SDK}A] + \varepsilon_B\,[-\mathrm{SDK}B] + x_{OS} + g\cdot x. \tag{21}$$

The difference Δx is influenced by the offset and scale-factor errors and no longer depends only on the estimate errors ϵ_A and ϵ_B. The effect of the additional x_{OS} and $g\cdot x$ terms, not due to errors ϵ_A and ϵ_B, must be eliminated to avoid corruption of the error estimation from Δx.

C. Modifying Estimation Process

If we were able to estimate the gain error g and offset error x_{OS}, then we could correct x_A and x_B as follows:

$$x_A = x' - \left(\frac{g}{2}\right)x - \frac{x_{OS}}{2} - \varepsilon_A\,[\mathrm{SDK}A] + \left(\frac{\hat{g}}{2}\right)x + \frac{\hat{x}_{OS}}{2} \tag{22a}$$
$$x_B = x' + \left(\frac{g}{2}\right)x + \frac{x_{OS}}{2} - \varepsilon_B\,[\mathrm{SDK}B] \underbrace{- \left(\frac{\hat{g}}{2}\right)x - \frac{\hat{x}_{OS}}{2}}_{\text{CORRECTION}} \tag{22b}$$

where $\hat{x}_{OS}$ and $\hat{g}$ are the estimated offset and scale-factor errors. Define errors in these estimates as

$$\varepsilon_g = \hat{g} - g \tag{23}$$
$$\varepsilon_{OS} = \hat{x}_{OS} - x_{OS}. \tag{24}$$

Taking the difference of (22a) and (22b), in similar fashion to (12), and substituting (23) and (24) gives

$$\Delta x = x_B - x_A = [-1]\,\varepsilon_{OS} + [-x]\,\varepsilon_g + [\mathrm{SDK}A]\,\varepsilon_A + [-\mathrm{SDK}B]\,\varepsilon_B. \tag{25}$$

Note from (16) and (20) that correcting the ADC output code for linearity does not necessarily require estimation or correction of the errors x_{OS} and g, since the definition of x' includes the effects of nonzero offset and scale factor. We only need to make the estimation of the cyclic gain errors ϵ_A and ϵ_B insensitive to x_{OS} and g. In the procedure described as follows, scale-factor error is estimated and corrected and ϵ_g driven to zero; offset is not estimated but the effect of nonzero offset is removed from the estimation process.

To develop the modified estimation process, consider expanding the set of conversions used in the matrix definition in (13) to six conversions, using the expression in (25)

$$\begin{bmatrix}\Delta x_1\\ \Delta x_2\\ \Delta x_3\\ \Delta x_4\\ \Delta x_5\\ \Delta x_6\end{bmatrix} = \begin{bmatrix}-1 & -x_1 & \mathrm{SDK}A_1 & -\mathrm{SDK}B_1\\ -1 & -x_2 & \mathrm{SDK}A_2 & -\mathrm{SDK}B_2\\ -1 & -x_3 & \mathrm{SDK}A_3 & -\mathrm{SDK}B_3\\ -1 & -x_4 & \mathrm{SDK}A_4 & -\mathrm{SDK}B_4\\ -1 & -x_5 & \mathrm{SDK}A_5 & -\mathrm{SDK}B_5\\ -1 & -x_6 & \mathrm{SDK}A_6 & -\mathrm{SDK}B_6\end{bmatrix}\begin{bmatrix}\varepsilon_{OS}\\ \varepsilon_g\\ \varepsilon_A\\ \varepsilon_B\end{bmatrix}. \tag{26}$$

The effect of offsets can be eliminated by using a "difference of differences" approach, subtracting row 2 from 1, 4 from 3, and 6 from 5. This cancels a constant offset from the calibration signal path and eliminates ϵ_{OS} as a variable from the system of equations. The result is given in (27), shown at the bottom of the next page.

We now have a 3×3 matrix equation that can, in principle, be inverted (if the matrix is nonsingular) to solve for ϵ_A, ϵ_B, and ϵ_g. A potential difficulty is that the matrix is in fact singular in the

TABLE II
RESIDUE-MODE CHOICES

	RESIDUE MODE	
CONVERSION	"A"	"B"
1	CYCLIC	CYCLIC
2	CYCLIC	CYCLIC
3	HIGH	CYCLIC
4	LOW	CYCLIC
5	CYCLIC	HIGH
6	CYCLIC	LOW

case of a dc input; the first column goes to zero. However, this is not a problem since the singularity obscures only information needed to solve for ϵ_g. Given the activity in the SDK_A and SDK_B coefficients enforced by the selection of residue modes, the remaining columns of the matrix provide sufficient information to solve for ϵ_A and ϵ_B.

D. Need for Residue-Mode Selection

It should be noted that, ideally, we would only need four rows in the matrix (26), given that there are four unknowns. The residue modes selected for the conversions corresponding to the extra rows in (26) are selected to ensure good numerical properties and avoid singularity for the matrix in (27). Table II shows one possible pattern of residue-mode choices.

Although a full analysis is beyond the scope of this paper, some insight into the effect of residue-mode variation can be gained from a qualitative examination of the residue-mode plots shown in Fig. 4(b). Consider, for example, the choice of residue modes in conversions 3 and 4: Different residue modes are used for ADC "A" while the residue mode for ADC "B" is unchanged. These will affect the values of $\mathrm{SDK}A_3 - \mathrm{SDK}A_4$ and $\mathrm{SDK}B_4 - \mathrm{SDK}B_3$ in the second row of the matrix in (27). To the extent that $\mathrm{SDK}A_3 - \mathrm{SDK}A_4$ differs from $\mathrm{SDK}B_4 - \mathrm{SDK}B_3$, the portion of matrix in (27) needed to solve for ϵ_A and ϵ_B will be nonsingular.

It can be seen from the plots of the SDK coefficients in Fig. 4(b) that, for any given code x, the value of the SDK coefficient for the "HIGH" residue mode is greater than that for the "LOW" residue mode. Thus, it is plausible that $|\mathrm{SDK}A_3 - \mathrm{SDK}A_4| > |\mathrm{SDK}B_4 - \mathrm{SDK}B_3|$ always, even for dc inputs. Behavioral simulations have indicated that, for a wide range of input signal conditions (for example, dc, random noise, and periodic signals at multiples of the conversion frequency), the activity in matrix coefficients enabled by choosing different residue modes is sufficient to prevent singularity in the matrix coefficients corresponding to ϵ_A and ϵ_B.

Note from Table I and from Fig. 4(b) that, in general, signal levels closer to FS correspond to a reduced min-to-max variation in SDK coefficients. From (27), it can be seen that this will reduce the value of matrix coefficients used to solve for ϵ_A and ϵ_B, which increases the condition number [17], [18] of the matrix. This means that the matrix is closer to the undesirable case of singularity, which slows the iterative matrix solution algorithm and will be seen in Section VI-C to slow the rate of calibration convergence for the entire ADC.

V. ITERATIVE MATRIX SOLUTION

A. Jacobi Iteration Method Review

To simplify the digital hardware, an iterative procedure is used to avoid matrix inversion. It will be shown that the procedure used has the additional advantage of dealing gracefully with matrix singularity. The procedure is based on the Jacobi iteration method [17], [18], which is now briefly reviewed.

Suppose we are trying to solve a linear system

$$\mathbf{S}\cdot\mathbf{e}=\mathbf{d} \quad \Rightarrow \quad \begin{bmatrix} S_{11} & S_{12} & S_{13} \\ S_{21} & S_{22} & S_{23} \\ S_{31} & S_{32} & S_{33} \end{bmatrix} \begin{bmatrix} e_1 \\ e_2 \\ e_3 \end{bmatrix} = \begin{bmatrix} d_1 \\ d_2 \\ d_3 \end{bmatrix} \tag{28}$$

in which the d_i are observations of a system (in our case, the Δx differences), e_i are unknown parameters (the estimate errors) we are trying to determine based on the observations, and the S_{ij} are known coefficients in (27) describing how the various parameters e_i affect the observations d_i.

Expanding the matrix product for the first row gives

$$S_{11}e_1 + S_{12}e_2 + S_{13}e_3 = d_1. \tag{29}$$

If we already knew e_2 and e_3, we could solve for e_1. In the Jacobi iteration procedure, we develop the next (new) iteration for e_1 using the previous (old) values of e_2 and e_3

$$e_1^{(\mathrm{new})} = \frac{1}{S_{11}}\left(d_1 - S_{12}e_2^{(\mathrm{old})} - S_{13}e_3^{(\mathrm{old})}\right). \tag{30}$$

Analogous expressions can be derived for e_2 and e_3 using the other rows of the $\mathbf{S}$ matrix. Expressing the iteration in matrix form gives

$$\begin{bmatrix} e_1^{(\mathrm{new})} \\ e_2^{(\mathrm{new})} \\ e_3^{(\mathrm{new})} \end{bmatrix} = \begin{bmatrix} \frac{1}{S_{11}} & 0 & 0 \\ 0 & \frac{1}{S_{22}} & 0 \\ 0 & 0 & \frac{1}{S_{33}} \end{bmatrix} \times \left(\begin{bmatrix} d_1 \\ d_2 \\ d_3 \end{bmatrix} - \begin{bmatrix} 0 & S_{12} & S_{13} \\ S_{21} & 0 & S_{23} \\ S_{31} & S_{32} & 0 \end{bmatrix} \begin{bmatrix} e_1^{(\mathrm{old})} \\ e_2^{(\mathrm{old})} \\ e_3^{(\mathrm{old})} \end{bmatrix} \right). \tag{31}$$

$$\begin{bmatrix} \Delta x_1 - \Delta x_2 \\ \Delta x_3 - \Delta x_4 \\ \Delta x_5 - \Delta x_6 \end{bmatrix} = \begin{bmatrix} x_2 - x_1 & \mathrm{SDK}A_1 - \mathrm{SDK}A_2 & \mathrm{SDK}B_2 - \mathrm{SDK}B_1 \\ x_4 - x_3 & \mathrm{SDK}A_3 - \mathrm{SDK}A_4 & \mathrm{SDK}B_4 - \mathrm{SDK}B_3 \\ x_6 - x_5 & \mathrm{SDK}A_5 - \mathrm{SDK}A_6 & \mathrm{SDK}B_6 - \mathrm{SDK}B_5 \end{bmatrix} \begin{bmatrix} \varepsilon_g \\ \varepsilon_A \\ \varepsilon_B \end{bmatrix} \tag{27}$$

B. Difficulty With Basic Jacobi Iteration

Considering the iteration in (31), the need for reciprocal elements $1/S_{ii}$ is undesirable in this application for (at least) three reasons.

1) Digital-hardware complexity: To keep the digital hardware simple, we prefer to avoid division by anything other than a power of two (note that the multiplications required for the $S_{ij}e_j^{(\text{old})}$ terms do not impose an additional complexity burden; the multiplier used for calculating the $(1/G)^k$ LUTs is available as a shared resource).
2) Iteration stability depends on values of S_{ii}: If the original $\mathbf{S}$ matrix is dominated by its diagonal elements, the iteration in (31) is stable and is guaranteed to converge. If this is not the case, the iteration can be made stable as described in [18] by scaling the matrices in (31). Although we have some influence on the S_{ii} through residue-mode selection, to a large extent, the S_{ii} are signal-dependent, and there may be some input signals for which the iteration in (31) is unstable.
3) Numerical sensitivity to values of S_{ii}: For example, from (30), we see that if S_{11} is small, the large reciprocal applies a large "gain" to the $\left(d_1 - S_{12}e_2^{(\text{old})} - S_{13}e_3^{(\text{old})}\right)$ term in an effort to "squeeze" $e_1^{(\text{new})}$ information out of d_1. The calculation is therefore very sensitive to small errors in the value of S_{11}. However, from (29), we see that when S_{11} is small, it is very unlikely that there actually is any information to be found in d_1 regarding e_1. In the worst case of a singular matrix (for example, first column $= 0$ with a dc input), the reciprocal of S_{11} blows up completely. When S_{11} is small, we are better off realizing that the equation has little to tell us about e_1; it is better off to just ignore the result and wait until the next set of six conversions. The desired behavior is similar to that encountered in solving an overdetermined set of equations with singular-value decomposition [17], a technique which gracefully handles poor numerical conditions.

C. Modifications to Jacobi Iteration

The solution proposed in this paper is to adopt an LMS-style approach in modifying the Jacobi iteration. In the method of successive over-relaxation (SOR) [18], the new iteration of the e_i is obtained by combining a weighted sum of the matrix operation of (31) with the previous iteration, as shown in

$$\begin{bmatrix} e_1^{(\text{new})} \\ e_2^{(\text{new})} \\ e_3^{(\text{new})} \end{bmatrix} = [1-\mu_e] \begin{bmatrix} e_1^{(\text{old})} \\ e_2^{(\text{old})} \\ e_3^{(\text{old})} \end{bmatrix} + \mu_e \begin{bmatrix} \frac{1}{S_{11}} & 0 & 0 \\ 0 & \frac{1}{S_{22}} & 0 \\ 0 & 0 & \frac{1}{S_{33}} \end{bmatrix} \times \left(\begin{bmatrix} d_1 \\ d_2 \\ d_3 \end{bmatrix} - \begin{bmatrix} 0 & S_{12} & S_{13} \\ S_{21} & 0 & S_{23} \\ S_{31} & S_{32} & 0 \end{bmatrix} \begin{bmatrix} e_1^{(\text{old})} \\ e_2^{(\text{old})} \\ e_3^{(\text{old})} \end{bmatrix} \right). \tag{32}$$

The factor μ_e allows tailoring of the dynamics of the iteration. Exploring further, we expand the matrix product for the first row, giving

$$e_1^{(\text{new})} = [1-\mu_e]\, e_1^{(\text{old})} + \mu_e \frac{1}{S_{11}} \left(d_1 - S_{12}e_2^{(\text{old})} - S_{13}e_3^{(\text{old})} \right) \tag{33}$$

which still has the problems associated with determining the $1/S_{11}$. To adopt an LMS-style approach, we will abandon hope for an exact solution and just use the sign of S_{11}

$$e_1^{(\text{new})} = [1-\mu_e]\, e_1^{(\text{old})} + \mu_e \left[\text{sgn}\,(S_{11})\right] \left(d_1 - S_{12}e_2^{(\text{old})} - S_{13}e_3^{(\text{old})} \right). \tag{34}$$

Thus, rather than solve exactly for the e_i using (31), we just take a small step in the direction indicated by $\mu_e \left[\text{sgn}\,(S_{11})\right] \left(d_1 - S_{12}e_2^{(\text{old})} - S_{13}e_3^{(\text{old})} \right)$. The factor μ_e can be chosen small enough to guarantee stable convergence of the iteration.

When the approach of (34) is applied to the matrix (31), we have (35), shown at the bottom of the page. For hardware implementation, (35) is rearranged to (36), shown at the bottom of the page, which is in an LMS form of adding a small update to the previous iteration result [19]. Applying the approach of (36) with the data in (27) provides (37), shown at the bottom of the page, which is used to develop the estimates ϵ_A, ϵ_B, and ϵ_g. In

$$\begin{bmatrix} e_1^{(\text{new})} \\ e_2^{(\text{new})} \\ e_3^{(\text{new})} \end{bmatrix} = [1-\mu_e] \begin{bmatrix} e_1^{(\text{old})} \\ e_2^{(\text{old})} \\ e_3^{(\text{old})} \end{bmatrix} + \mu_e \begin{bmatrix} \text{sgn}\,(S_{11}) & 0 & 0 \\ 0 & \text{sgn}\,(S_{22}) & 0 \\ 0 & 0 & \text{sgn}\,(S_{33}) \end{bmatrix} \left(\begin{bmatrix} d_1 \\ d_2 \\ d_3 \end{bmatrix} - \begin{bmatrix} 0 & S_{12} & S_{13} \\ S_{21} & 0 & S_{23} \\ S_{31} & S_{32} & 0 \end{bmatrix} \begin{bmatrix} e_1^{(\text{old})} \\ e_2^{(\text{old})} \\ e_3^{(\text{old})} \end{bmatrix} \right) \tag{35}$$

$$\begin{bmatrix} e_1^{(\text{new})} \\ e_2^{(\text{new})} \\ e_3^{(\text{new})} \end{bmatrix} = \begin{bmatrix} e_1^{(\text{old})} \\ e_2^{(\text{old})} \\ e_3^{(\text{old})} \end{bmatrix} - \mu_e \left(\begin{bmatrix} 1 & S_{12} & S_{13} \\ S_{21} & 1 & S_{23} \\ S_{31} & S_{32} & 1 \end{bmatrix} \begin{bmatrix} e_1^{(\text{old})} \\ e_2^{(\text{old})} \\ e_3^{(\text{old})} \end{bmatrix} - \begin{bmatrix} \text{sgn}\,(S_{11}) & 0 & 0 \\ 0 & \text{sgn}\,(S_{22}) & 0 \\ 0 & 0 & \text{sgn}\,(S_{33}) \end{bmatrix} \begin{bmatrix} d_1 \\ d_2 \\ d_3 \end{bmatrix} \right) \tag{36}$$

$$\begin{aligned} \varepsilon_g^{(\text{new})} &= \varepsilon_g^{(\text{old})} - \mu_e \left(\varepsilon_g^{(\text{old})} + [\text{SDK}A_1 - \text{SDK}A_2]\, \varepsilon_A^{(\text{old})} + [\text{SDK}B_2 - \text{SDK}B_1]\, \varepsilon_B^{(\text{old})} - \text{sgn}\,(x_2 - x_1)\, [\Delta x_1 - \Delta x_2] \right) \\ \varepsilon_A^{(\text{new})} &= \varepsilon_A^{(\text{old})} - \mu_e \left([x_4 - x_3]\, \varepsilon_g^{(\text{old})} + \varepsilon_A^{(\text{old})} + [\text{SDK}B_4 - \text{SDK}B_3]\, \varepsilon_B^{(\text{old})} - \text{sgn}\,(\text{SDK}A_3 - \text{SDK}A_4)\, [\Delta x_3 - \Delta x_4] \right) \\ \varepsilon_B^{(\text{new})} &= \varepsilon_B^{(\text{old})} - \mu_e \left([x_6 - x_5]\, \varepsilon_g^{(\text{old})} + [\text{SDK}A_5 - \text{SDK}A_6]\, \varepsilon_A^{(\text{old})} + \varepsilon_B^{(\text{old})} - \text{sgn}\,(\text{SDK}B_6 - \text{SDK}B_5)\, [\Delta x_5 - \Delta x_6] \right) \end{aligned} \tag{37}$$

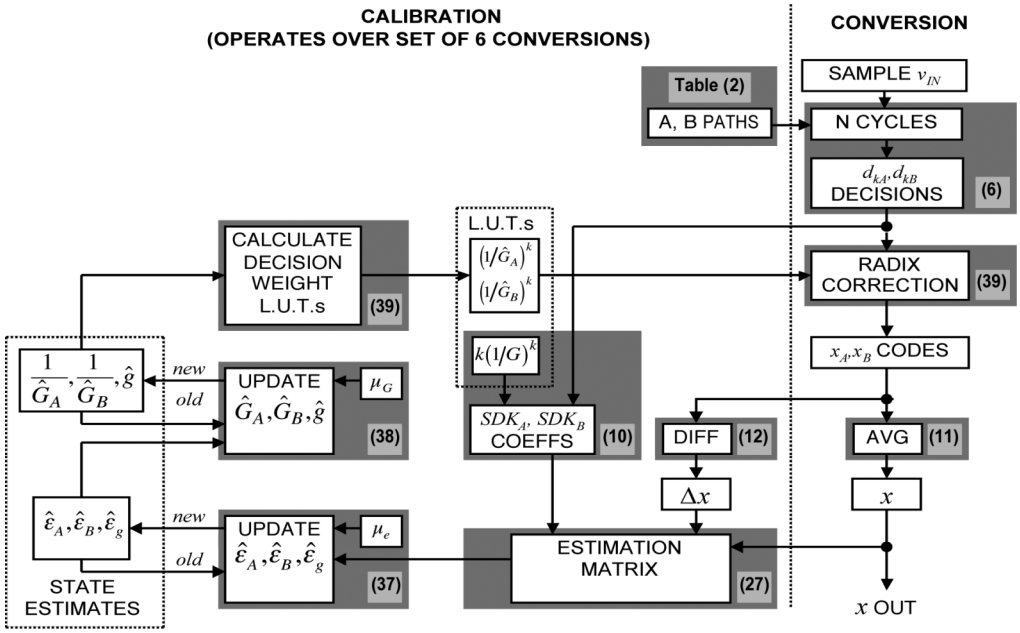


Fig. 5. Calibration and conversion flowchart.

accord with the LMS principle, the update approaches zero at equilibrium when the $\mathbf{e}$ and $\mathbf{d}$ are driven to zero.

In the full SOR method, the iteration in (37) could be repeated until the $\epsilon^{(\mathrm{new})}$ values converged to a sufficiently accurate solution. However, there is a limited time for iterating (37), since a new set of coefficients is generated every six conversions. It might seem that there is a tradeoff in which improving digital speed and complexity to perform more iterations would speed the convergence of ADC calibration; however, this is not really the case. Due to the approximate nature of the high-level LMS loop, improving accuracy in the $\epsilon^{(\mathrm{new})}$ values provides little improvement in the rate of convergence of system calibration parameters G_A, G_B, and g. Indeed, the advantage of the LMS technique is that accuracy is unnecessary, since the larger LMS loop eventually drives ϵ_A, ϵ_B, and ϵ_g to zero.

In the case of [1] and [2], for example, the digital implementation was simplified by basing the updated $\epsilon^{(\mathrm{new})}$ values on the results of only one iteration as carried out in (37). The resulting estimated values of ϵ were carried forward to be used as the starting point in the iteration corresponding to the next set of six conversions.

The estimates from (37) can be used in the LMS procedure to adjust estimates of G_A, G_B, and g used in calibration. However, from (6), we see that only the reciprocals of G_A and G_B are needed for calculating the decision-weight LUT, so we store an estimate of the reciprocal. Since our error estimates ϵ_A and ϵ_B are in terms of the gain, but we are updating the gain reciprocals, the sign of the LMS updating must be inverted to keep maintain negative feedback in the LMS procedure loop

$$\frac{1}{\hat{G}_A}^{(\mathrm{new})} = \frac{1}{\hat{G}_A}^{(\mathrm{old})} + \mu_G \hat{\varepsilon}_A \tag{38a}$$

$$\frac{1}{\hat{G}_B}^{(\mathrm{new})} = \frac{1}{\hat{G}_B}^{(\mathrm{old})} + \mu_G \hat{\varepsilon}_B \tag{38b}$$

$$g^{(\mathrm{new})} = g^{(\mathrm{old})} - \mu_G \hat{\varepsilon}_g. \tag{38c}$$

Correction for gain error g requires modifying (9a) and (9b); this correction is easily incorporated into calculation of the decision-weight LUTs as follows:

$$x_A = \underbrace{\left(1+\frac{\hat{g}}{2}\right)\sum_{k=1}^{N}\left(\frac{1}{\hat{G}_A}\right)^k}_{A\quad LUT} d_{kA}$$

$$x_B = \underbrace{\left(1-\frac{\hat{g}}{2}\right)\sum_{k=1}^{N}\left(\frac{1}{\hat{G}_B}\right)^k}_{B\quad LUT} d_{kB}. \tag{39}$$

D. Summary of Estimation and Calibration Algorithm

A flowchart of the entire calibration procedure is shown in Fig. 5, with shaded boxes indicating the calculations implemented and relevant equation numbers. The figure is partitioned into those calculations which occur every conversion (right side) and those which occur in updating calibration parameters every set of six conversion (left side).

Beginning at the top right of Fig. 5, the input v_{IN} is sampled, and N residue cycles are completed by each of the A and B ADCs. Comparator decision sequences d_{kA} and d_{kB} are produced in accordance with (6). Over each set of six conversions, the residue modes are varied according to the pattern described in Table II. From the decisions, radix correction is implemented using the updated decision-weight LUTs according to (39) to produce the individual ADC codes x_A and x_B for each side of the split ADC. The codes are averaged according to (11) to produce the overall ADC output code x.

For error-estimation and calibration purposes, the individual ADC codes x_A and x_B are also used to calculate the difference Δx according to (12). The error-estimation process also requires error coefficients SDK_A and SDK_B, which are calculated from the d_{kA} and d_{kB} decisions and the fixed error-coefficient-weight LUT according to (10). The output code x is

TABLE III
SYSTEM BEHAVIORAL SIMULATION PARAMETERS

PARAMETER	VALUE
Number of Cycles N	20
Cyclic Gain G_A	1.922
Cyclic Gain G_B	1.923
Initial Gain Estimates	1.92
Internal Digital Precision	24 bits
Gain LMS parameter μ_G	1/1024
Estimate Error LMS parameter μ_e	1/32
A-B Offset Error	0.2% FS
A-B Scale Factor Error	0.1% FS
Input Referred Cyclic Amplifier Noise	-96 dBFS

also used in assembling the error-estimation matrix as defined by (27).

The state estimates of the errors ϵ_A, ϵ_B, and ϵ_g are updated using the estimation matrix in accordance with the LMS procedure defined by (37), with the dynamics of the LMS matrix iteration determined by coefficient μ_e. The estimated errors are in turn used to update the scale-factor error-estimate g and the reciprocals of the gain estimates μ_e and in accordance with the LMS procedure defined by (38), with the dynamics of the LMS approximation determined by coefficient μ_G. Finally, the decision-weight LUTs are updated by recalculating the decision weights as powers of the gain reciprocal in accordance with (39). The calibration procedure operates in the background, periodically updating the decision-weight LUTs, and is transparent to the foreground operation of the converter.

VI. RESULTS

Experimental results for an integrated circuit implementing the procedure described earlier were presented in [1] and [2]. With LMS adaptation factors of $\mu_G = 1/1024$ and $\mu_e = 1/32$, successful self-calibration was demonstrated in approximately 10 000 conversions. In this section, simulation results are presented exploring the choice of μ_G and μ_e, as well as other aspects of the estimation and calibration procedures.

The ADC was simulated in behavioral fashion using MATLAB, with the code roughly configured as the system block diagram shown in Fig. 3, implementing the algorithm of Fig. 5. Parameters were included for describing analog block nonidealities such as finite op-amp gain, offset, noise, and nonlinearity. Unless otherwise indicated, system parameters were as shown in Table III. In [1] and [2], a cyclic gain of $G \approx 1.92$ was used, which according to (6) would require 18 cycles to achieve 16-bit resolution. In the actual implementation, two extra cycles were added for a total of $N = 20$ to reduce the impact of quantization error on the estimation process.

A. DNL and INL Improvement With Calibration

Fig. 6(a) and (b) show plots of DNL and INL (in LSB) before and after calibration. Once the LMS calibration loop has adapted and converged to its steady state, the simulated linearity is better than ± 1 LSB. These results show that improvement is similar to the results presented in [1] and [2], thus showing good correspondence between the behavioral simulation approach and the measured performance from the fabricated prototype IC.

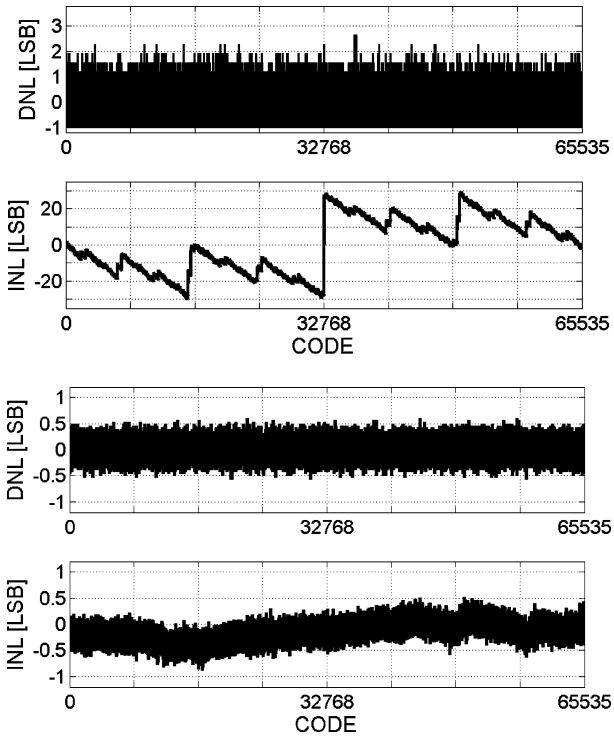


Fig. 6. (a) DNL and INL before calibration. (b) DNL and INL after calibration.

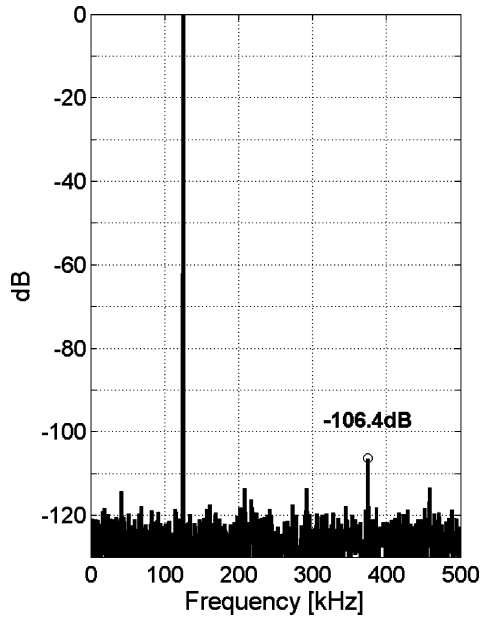
B. Frequency-Domain Performance

A modification that was necessary for the IC implementation described in [1] and [2] is related to the sequencing of residue modes shown in Table II. Nonidealities such as gain nonlinearity in the cyclic amplifier can lead to small differences (of order $< 10\text{ppm}$) in the optimal gain estimate G corresponding to each residue mode. Since the converters cycle through different residue modes, these differences average out and do not affect DC plots of DNL or INL. However, there is a problem if frequency-domain performance is evaluated with a periodic input: The same pattern of residue modes is constantly repeated, and the error pattern reinforces to produce spurs in a magnitude spectrum plot. This is shown in Fig. 7(a). The problem is solved by shuffling the sequence of residue modes in pseudorandom fashion, which breaks up any pattern of reinforcement. The spur energy is spread and disappears under the noise floor, as shown in Fig. 7(b).

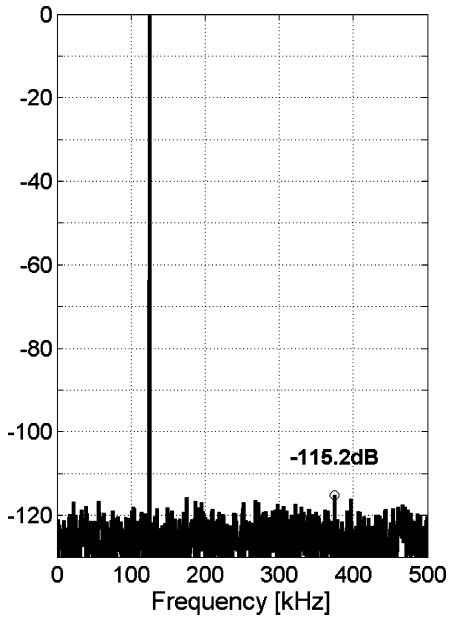
C. Adaptation For Various Input Signals

Convergence speed is affected by two factors: the nature of the input signal and the choice of LMS parameters. Input signal effects are explored in this section; the effects of LMS parameter choices are covered in the following section.

As described previously in Section III-B, a "busy" input is not required for calibration. As described in Section IV-D, however, the coefficients of the estimation matrix in (27) and (37) are affected by the input signal. The dynamics of the matrix iteration in turn depend on the matrix coefficients [18], so we expect to see some variation in convergence speed for different input signals. Fig. 8 shows a plot of ADC error in dB relative to full scale (dBFS) as a function of conversion for different input signals. For the random input, calibration is achieved to noise-limited



(a)



(b)

Fig. 7. Spectrum of ADC output, residue modes ordered versus shuffled. (a) Ordered. (b) Shuffled.

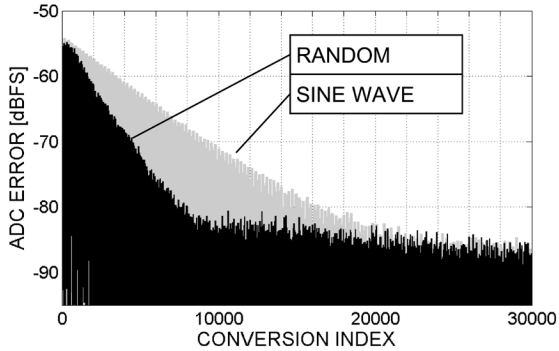


Fig. 8. ADC error convergence for sine and random inputs.

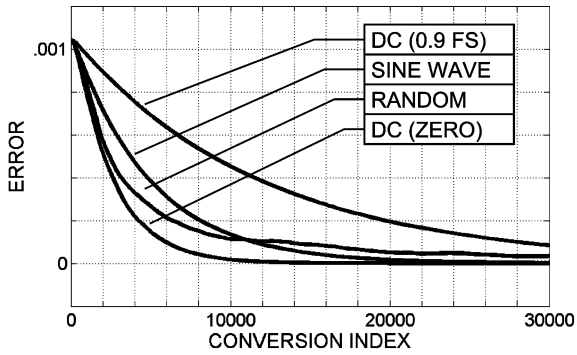


Fig. 9. Coefficient convergence for different input signals.

performance within 10 000 conversions; approximately 20 000 conversions are necessary with a 0.9-FS sine-wave input.

Fig. 9 shows a plot of the error in the G_A and G_B gain estimates as a function of time (in conversions) for various input signals. As shown in the figure, the time constant of the adaptation convergence depends somewhat on the nature of the input signal. Since a diversity of decision paths are allowed with redundancy and the multiple residue-mode approach, calibration information can be extracted even for a dc input; even, for example, in the case of a dc input at 0.9-FS calibration eventually does converges although more slowly than for a zero dc input. As the input nears FS, the decision sequences are more similar and the resulting *SDK* coefficients become closer in value, which can be seen qualitatively from the *SDK* coefficient plots shown in Fig. 4(b) and numerically from the example *SDK* coefficients in Table I as described in Section IV-D. Indeed, as described in [2], the only input pattern that presents a pathological difficulty is a dc input $\mathrm{at}+\mathrm{or}$—FS, which, despite redundancy, requires a nearly identical sequence of decisions regardless of the residue mode. As can be seen from (27), if all pairs of *SDK* coefficients are identical, then the entire matrix is zero and the Δx differences provide no visibility whatsoever to errors. Although calibration cannot proceed reliably during this input condition of a dc input $\mathrm{at}+\mathrm{or}$—FS, the condition can be identified and calibration suspended to avoid corruption of calibration parameters.

D. LMS Parameter Selection

Fig. 10 shows a plot of error in the G_A and G_B gain estimates versus conversion for selected values of the LMS parameters μ_G and μ_e. In addition, shown for each set of values is the associated steady-state rms variation σ in the G_A and G_B estimates after convergence. Results show the expected tradeoff in an LMS system between convergence and sensitivity to noise: Larger μ corresponds to faster convergence but more sensitivity to parameter variation from noise. Noise in each of the A and B ADCs causes a difference Δx which is unrelated to calibration errors; since this Δx contribution varies randomly on a conversion-to-conversion basis, its effect is reduced by the LMS loop, which effectively averages over the time constant of the adaptation process. For the prototype system in [1] and [2], values of $\mu_e = 2^{-5}$ and $\mu_G = 2^{-10}$ were chosen as a compromise.

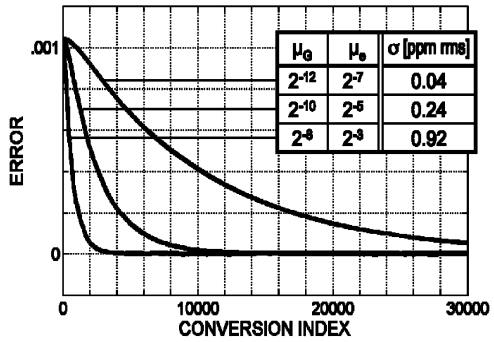


Fig. 10. Convergence of gain estimate versus μ_G coefficients.

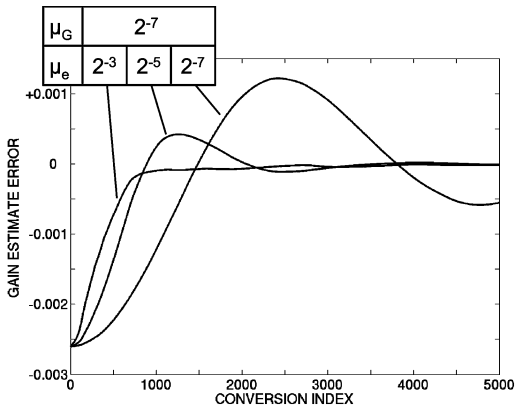


Fig. 11. Convergence of gain estimate versus μ_e and μ_G coefficients.

The block diagram shown in Fig. 5 shows that the series connection of the μ_e and μ_G LMS loops effectively makes the entire calibration loop analogous to a second-order system, which is prone to instability if the time-constants and LMS parameters of the μ_e and μ_G loops are chosen inappropriately. Fig. 11 shows a plot of gain estimation error as a function of conversion for different values of μ_e with μ_G fixed at 2^{-7}. If μ_e is not chosen to be at least 16 $\times$ greater than μ_G, the adaptation transient exhibits overshoot and oscillatory behavior.

Other techniques from adaptive and nonlinear filtering were considered. Estimates of the errors ϵ_A, ϵ_B, and ϵ_g are most prone to noise, since they are derived from a small set of conversions and filtered with a shorter time constant, since $\mu_e > \mu_G$. Nonlinear techniques of noise-reduction, such as median filtering and limiting, were examined but showed no significant improvement in performance in this application. To break the convergence versus accuracy tradeoff, it is possible to change the LMS parameters μ_e and μ_G as a function of error: Large μ when error is large for fast convergence; changing to smaller μ when error is small for better accuracy once convergence is achieved. Although these techniques ultimately were not necessary in the work presented in this paper, they may prove useful in future work with self-calibrating ADCs.

VII. Conclusion

An algorithm has been presented for the "split ADC" architecture, which is tolerant of offset and scale-factor errors, has an efficient digital implementation, and enables continuous digital background calibration for high-resolution ADCs. Unlike statistical techniques which require long decorrelation times to separate calibration information from the input signal, the split ADC approach uses the difference of ADC output codes to cancel the unknown input and rapidly extract calibration information. In this paper, the split ADC concept is realized in an algorithmic converter; it is anticipated that future work will demonstrate applicability in other ADC architectures such as successive approximation, pipelined, and interleaved ADCs. For the specific realization presented of a 16-bit 1-MS/s algorithmic ADC, self-calibration in approximately 10 000 conversions is demonstrated. It is anticipated that future work will demonstrate applicability in other ADC architectures such as successive approximation, pipelined, and interleaved ADCs.

Acknowledgment

The authors would like to thank S. Gong, R. Adams, and C. Lyden, the members of the Precision Nyquist Converter and High-Speed Converter groups at Analog Devices; B. Murmann of Stanford University; and the National Science Foundation.

References

[1] J. McNeill, M. Coln, and B. Larivee, "A Split-ADC architecture for deterministic digital background calibration of a 16b 1 MS/s ADC," in *Proc. ISSCC Dig. Tech. Papers*, Feb. 2005, pp. 276–598.

[2] J. McNeill, M. Coln, and B. Larivee, "'Split ADC' architecture for deterministic digital background calibration of a 16-bit 1-MS/s ADC," *IEEE J. Solid-State Circuits*, vol. 40, no. 12, pp. 2437–2445, Dec. 2005.

[3] I. Galton, "Digital cancellation of D/A converter noise in pipelined A/D converters," *IEEE Trans. Circuits Syst. II, Analog Digit. Signal Process.*, vol. 47, no. 3, pp. 185–196, Mar. 2000.

[4] B. Murmann and B. E. Boser, "A 12b 75 MS/s pipelined ADC using open-loop residue amplification," in *Proc. ISSCC Dig. Tech. Papers*, Feb. 2003, pp. 328–329.

[5] H. Liu *et al.*, "A 15b 20 MS/s CMOS pipelined ADC with digital background calibration," in *Proc. ISSCC Dig. Tech. Papers*, Feb. 2004, pp. 454–455.

[6] K. Nair and R. Harjani, "A 96 dB SFDR 50 MS/s digitally enhanced CMOS pipeline A/D converter," in *Proc. ISSCC Dig. Tech. Papers*, Feb. 2004, pp. 456–457.

[7] S. Ryu *et al.*, "A 14b-linear capacitor self-trimming pipelined ADC," in *Proc. ISSCC Dig. Tech. Papers*, Feb. 2004, pp. 464–465.

[8] H.-S. Lee, "A 12-b 600 ks/s digitally self-calibrated pipelined algorithmic ADC," *IEEE J. Solid-State Circuits*, vol. 29, no. 4, pp. 509–515, Apr. 1994.

[9] A. Karanicolas *et al.*, "A 15-b 1-MSample/s digitally self-calibrated pipeline ADC," *IEEE J. Solid-State Circuits*, vol. 28, no. 12, pp. 1207–1215, Dec. 1993.

[10] O. E. Erdogan, P. J. Hurst, and S. H. Lewis, "A 12-b digital-background-calibrated algorithmic ADC with −90-dB THD," *IEEE J. Solid-State Circuits*, vol. 34, no. 12, pp. 1812–1820, Dec. 1999.

[11] Y. Chiu, C. W. Tsang, B. Nikolic, and P. R. Gray, "Least mean square adaptive digital background calibration of pipelined analog-to-digital converters," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 51, no. 1, pp. 38–46, Jan. 2004.

[12] Y. Chiu, "A 1.8 V 14b 10 MS/s pipelined ADC in 0.18 μm CMOS with 99 dB SFDR," in *Proc. ISSCC Dig. Tech. Papers*, Feb. 2004, pp. 458–459.

[13] P. Bogner, F. Kuttner, C. Kropf, T. Hartig, M. Burian, and H. Eul, "A 14b 100 MS/s digitally self-calibrated pipelined ADC in 0.13 μm CMOS," in *Proc. ISSCC Dig. Tech. Papers*, Feb. 2006, pp. 832–841.

[14] J. Li and U. Moon, "Background calibration techniques for multistage pipelined ADCs with digital redundancy," *IEEE Trans. Circuits Syst. II, Analog Digit. Signal Process.*, vol. 50, no. 9, pp. 531–538, Sep. 2003.

[15] J. Li, G. Ahn, D. Chang, and U. Moon, "0.9 V 12 mW 2MSPS algorithmic ADC with 81 dB SFDR," in *Proc. IEEE Symp. VLSI Circuits*, Jun. 2004, pp. 436–439.

[16] J. Li, G. Ahn, D. Chang, and U. Moon, "A 0.9-V 12-mW 5-MSPS algorithmic ADC with 77-dB SFDR," *IEEE J. Solid-State Circuits*, vol. 40, no. 4, pp. 960–969, Apr. 2005.

[17] W. H. Press, S. A. Teukolsky, W. T. Vetterling, and B. P. Flannery, *Numerical Recipes in C: The Art of Scientific Computing*. Cambridge, U.K.: Cambridge Univ. Press, 1992.

[18] G. Golub and C. F. van Loan, *Matrix Computations*, 3rd ed. Baltimore, MD: Johns Hopkins Univ. Press, 1996.
[19] S. S. Haykin, *Adaptive Filter Theory*, 3rd ed. Upper Saddle River, NJ: Prentice-Hall, 1996.

John A. McNeill (M'90–SM'05) was born in Syracuse, NY, in 1961. He received the A.B. degree from Dartmouth College, Hanover, NH, in 1983, the M.S. degree from the University of Rochester, Rochester, NY, in 1991, and the Ph.D. degree from Boston University, Boston, MA, in 1994.

From 1983 to 1990, he worked in industry in the design of high-speed high-resolution analog-to-digital converters and low-noise interface electronics used in high-speed wide-dynamic-range imaging systems. Since 1994, he has been with the Electrical and Computer Engineering Department, Worcester Polytechnic Institute (WPI), Worcester, MA, where he is currently an Associate Professor and, since 1998, has been the Director of the New England Center for Analog and Mixed Signal Integrated Circuit Design, a consortium of industry sponsors supporting undergraduate projects and graduate research in the area of analog- and mixed-signal integrated-circuit design.

Dr. McNeill was the recipient of WPI's campus-wide award for Outstanding Teaching in 1999 and in 2007 was one of the first recipients of WPI's Exemplary Faculty award recognizing accomplishment in teaching, research, and service. In 2006, he and coauthors Coln and Larivee were the recipients of the Lewis Winner Award for Outstanding Paper at the 2005 IEEE International Solid-States Circuits Conference.

Michael C. W. Coln (M'80) received the B.S. degree from California Institute of Technology, Pasadena, in 1976 and the M.S. and Ph.D. degrees in electrical engineering from Massachusetts Institute of Technology, Cambridge, in 1979 and 1985, respectively.

From 1985 to 1988, he was with Hewlett-Packard Laboratories, Palo Alto, CA, where he was a Member of technical staff in the area of high-performance converters. Since 1988, he has been with Analog Devices, Wilmington, MA, where he is currently a Design Manager for Precision Nyquist Converters group and a Fellow. His research and development interests include monolithic high-resolution Nyquist converters, high-accuracy analog integrated-circuit design, and low-noise architectures for instrumentation. He is the author or coauthor of three technical papers in the area of integrated-circuit design. He is the holder of 14 U.S. patents.

D. Richard Brown (M'97) received the B.S. and M.S. degrees in electrical engineering from the University of Connecticut, Storrs, in 1992 and 1996, respectively, and the Ph.D. degree in electrical engineering from Cornell University, Ithaca, NY, in 2000.

He is currently an Associate Professor with the Department of Electrical and Computer Engineering, Worcester Polytechnic Institute, Worcester, MA, where since 2002, he has also been the Director of the Signal Processing and Information Networking Laboratory. His research interests include signal-processing applications, cooperative communication in networks, and adaptive channel equalization.

Brian J. Larivee (M'04) was born in Troy, MI, on June 12, 1980. He received the B.S.E.E. and M.S.E.E. degrees from the University of Michigan, Ann Arbor, in 2002.

He spent three summers working for General Motors, Pontiac, MI, from 1999 to 2001. In 2002, he was with the Center for Wireless Integrated Microsystems, University of Michigan. Since 2003, he has been working for Analog Devices, Wilmington, MA, where he is currently a Design Engineer for the Precision Nyquist Converters group. His development interests are in the areas of Nyquist rate converter design and low-power analog integrated-circuit design techniques.