A Cost-Effective Histogram Test-Based Algorithm for Digital Calibration of High-Precision Pipelined ADCs

Xin Dai, Degang Chen, and Randall Geiger
Department of Electrical and Computer Engineering
Iowa State University
Ames, IA 50011, USA

Abstract—**This work presents a self-calibration algorithm that corrects the linearity errors of pipelined ADCs with a sub-radix architecture, based on the results of simple code density tests. The proposed algorithm identifies discontinuities in an ADC's output histogram data, calculates correction codes for transitions in pipeline stages, and digitally calibrates ADC's output codes. Simulation results show that the calibration algorithm can dramatically improve the linearity performance of ADCs. The INL can be reduced from about 1000 LSB to less than 1 LSB. Since this algorithm is based on conventional code density tests and uses only a few memory cells and simple logic circuits to carry out the calibration, this algorithm can be easily implemented on chip without introducing much area and cost overhead and serving as a self-calibration solution for high-speed high-precision pipelined ADCs.**

I. INTRODUCTION

The pipeline architecture is widely used in high speed, high resolution analog-to-digital converter (ADC) design. Particularly, the 1b/stage and 1.5b/stage configurations with over range protection are often used, because each stage has a very simple structure and the requirement on the op amp performance is relatively easy to achieve. However, issues like capacitor mismatch, comparator offset, charge injection, and finite gain and nonlinearity of op amps all limit the accuracy of ADC stages. Handling these issues directly is not favorable or even not doable. For example, the capacitor mismatch or comparator offset can only be reduced at the cost of large area consumption, while it is impossible to achieve infinite and completely linear op amp gain.

Over the years, great efforts have been made to improve the performance of pipelined ADCs. Error averaging [1] and analog calibration [2] techniques were proposed, but these techniques require elaborate calibration schemes and complex additional circuits which greatly increase the difficulty of circuit design. As compared to analog calibration, digital calibration is more favored because of its simpler calibration scheme, lower complexity, and smaller area consumption. Soenen and Geiger proposed an algorithm and architecture for digitally calibrating pipelined ADCs [3]. The circuit and the calibration scheme were so designed that the same hardware used in the calibration mode was used in the conversion mode. Karanicolas, Lee, and Bacrania also gave a simpler ADC architecture with an applicable digital self-calibration scheme [4]. All the previously reported digital calibration schemes inevitably need modification of the pipeline stages to allow external control during the calibration phase. This is not amenable, because calibrations which require disturbance of the pipeline may introduce errors by themselves, especially when the ADC's resolution is high, and architecture specific calibrations are not applicable to ADCs of different architectures.

This paper presents a new digital algorithm for calibrating pipelined ADCs. The calibration algorithm is based on results of input-output histogram tests so that it does not disturb the data path of an ADC during test and not require external control of pipeline stages. Correction codes are calculated from the discontinuity in the histogram data and the digital calibration can be done with a small number of memories, an adder and some simple control logics. Furthermore, this algorithm does not require a precision ramp input for the histogram test. All these features make this algorithm applicable for on-chip implementations. Simulations show that the INL of an ADC can be reduced from about 1000 LSB originally to less than 1 LSB after calibration, which is comparable to what the algorithm reported in [4] can achieve.

The rest of the paper is organized as follows. In section II, the pipelined architecture is reviewed with mathematical descriptions. Section III describes the principle of the digital calibration algorithm. Section IV discusses implementation issues and Section V gives the simulation results.

II. PIPELINE ARCHITECTURE AND MODELING

The pipeline architecture of 1-bit/stage is shown in Fig. 1. For each stage, the comparator compares the input voltage with 0, and gives a 1-bit digital output. The output voltage of the stage is determined by the input voltage and the comparison result, which can be described as follows.

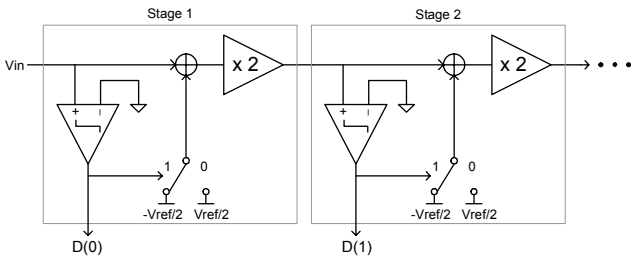


Figure 1. Pipeline ADC architecture

For any pipeline stage k, k=1, 2, 3…

$$d_k = \begin{cases} 0 & V_{ink} < 0 \\ 1 & V_{ink} > 0 \end{cases} \tag{1}$$

$$V_{outk} = 2 \times [V_{ink} + \frac{V_{ref}}{2} - V_{ref} \times d_k] \tag{2}$$

where V_{ink} and V_{outk} are the input and output voltages of stage k, respectively; d_k is the 1-bit digital output of stage k; and V_{ref} is the reference voltage.

Due to issues such as capacitor mismatch, charge injection, comparator offset and finite op amp gain, the output voltage of one stage may exceed the input range of the next stage, which will cause missing decision levels. This over-range problem can be avoided by intentionally setting the nominal gain of each stage to be less than 2 (radix<2). The modified ideal transfer curve of one stage is shown in Fig. 2(a). With an appropriate gain reduction, the output voltage of one stage can be ensured to be within the input range of the next stage in existence of errors such as capacitor mismatch and comparator offset. Since the gain is less than two, a pipeline consists of n stages will not give 2^n digital output codes for n bit resolution. Some more pipeline stages are usually added to provide enough redundancy in decision levels. The less-than-two gain also introduces missing codes in the output, causing the output codes to be discontinuous. The discontinuity in the digital output causes unreasonably large INL and DNL and thus must be removed.

Suppose a pipeline stage, say, stage k, has

1) Two gains g_{1k} and g_{2k}. g_{1k}'s nominal value is less than 2, say g_{10}=1.93, for over-range protection, and $g_{20}=g_{10}/2$;

2) A comparator offset V_{osk} with nominal value 0;

3) Two reference voltage V_{r1k} and V_{r2k}, whose nominal values are $-V_{ref}$ and $+V_{ref}$, respectively.

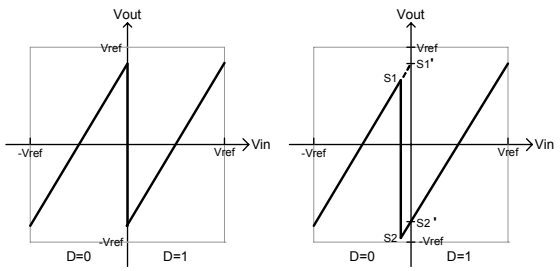
The actual transfer curve of stage k can be expressed as

$$V_{outk} = \begin{cases} g_{1k} \cdot V_{ink} - g_{2k} \cdot V_{r1k} & d_k = 0 \\ g_{1k} \cdot V_{ink} - g_{2k} \cdot V_{r2k} & d_k = 1 \end{cases} \tag{3}$$

where

$$d_k = \begin{cases} 0 & V_{ink} < V_{osk} \\ 1 & V_{ink} > V_{osk} \end{cases} \tag{4}$$

Solving V_{ink} from (3) and (4) gives



(a) (b)

Figure 2. Ideal and actual transfer curve of a radix<2 pipeline stage

$$V_{ink} = \frac{V_{outk}}{g_{1k}} + \frac{g_{2k}}{g_{1k}} V_{r1k} + d_k \frac{g_{2k}}{g_{1k}} (V_{r2k} - V_{r1k}) \tag{5}$$

For the 1st stage, we get

$$V_{in} = \frac{V_{out1}}{g_{11}} + \frac{g_{21}}{g_{11}} V_{r11} + d_1 \frac{g_{21}}{g_{11}} (V_{r21} - V_{r11}) \tag{6}$$

Since the 1st stage's output is also the 2nd stage's input,

$$V_{out1} = V_{in2} = \frac{V_{out2}}{g_{12}} + \frac{g_{22}}{g_{12}} V_{r12} + d_2 \frac{g_{22}}{g_{12}} (V_{r22} - V_{r12}) \tag{7}$$

Substituting (7) to (6) and doing this iteratively for all the n stages give

$$V_{in} = \sum_{i=1}^{n} d_i \frac{g_{2i}}{\prod_{j=1}^{i} g_{1j}} (V_{r2i} - V_{r1i}) + \sum_{i=1}^{n} \frac{g_{2i}}{\prod_{j=1}^{i} g_{1j}} V_{r1i} + \frac{V_{outn}}{\prod_{i=1}^{n} g_{1i}} \tag{8}$$

The last term on the right hand side of (8) is the residue of the last stage, which is very small and can be neglected. The second term is a constant offset. The first term is the most important one, which shows that V_{in} can be accurately interpreted from the digital output if $\frac{g_{2i}}{\prod_{j=1}^{i} g_{1j}} (V_{r2i} - V_{r1i})$ is known for all stages.

III. Digital Calibration Algorithm

Neglecting the last term in (8) and multiplying the two sides by $\prod_{i=1}^{n} g_{1i} / 2V_{ref}$ gives

$$\hat{V}_{in} \triangleq \sum_{i=1}^{n} d_i w_i + C \tag{9}$$

where

$$w_i = g_{2i} \prod_{j=i+1}^{n} g_{1j} \frac{V_{r2i} - V_{r1i}}{2V_{ref}} \tag{10}$$

and

$$C = \sum_{i=1}^{n} \left(g_{2i} \prod_{j=i+1}^{n} g_{1j} \frac{V_{r1i}}{2V_{ref}} \right) \tag{11}$$

Since $\hat{V}_{in}$ is a scaled version of V_{in}, it should have the same linearity performance as V_{in}. If we interpret the digital output using powers of 2, essentially we are assuming g_{1i}=2 and g_{2i}=1, and V_{r2i}-V_{r1i}=2V_{ref} for all i's, then (9) becomes

$$\hat{V}'_{in} = \sum_{i=1}^{n} d_i 2^{n-i} - \sum_{i=1}^{n} 2^{n-i-1} \tag{12}$$

Comparing (9)~(11) with (12), we can see that for radix<2 configuration, since the gain is intentionally set to be less than 2, the actual value of w_i is less than 2^{n-i}, which causes the discontinuity in the output codes.

To correct the discontinuity in the output, a good estimation of w_i is needed. Direct measurement of parameters such as g_i, V_{r1i} and V_{r2i} is infeasible. Fortunately, there exist digital calibration algorithms that can effectively calculate w_i without knowing the exact value of these parameters. The algorithm proposed in [4] tried to measure the jump in the transfer curve (S1-S2 in Fig. 2(b)) of each stage, which is essentially w_i. However, there are two potential problems associated with the algorithm. First, the algorithm actually doesn't measure S1-S2, it measures S1'-S2' instead. Because of the nonlinearity of the op amp, S1-S2 and S1'-S2' may not be the same and this will become more problematic when the ADC's resolution increases. Second, the pipeline is interrupted and externally controlled when doing calibration, which means the pipeline in calibration mode may not be the same as that in conversion mode. Thus the correction code may not be perfect for the pipeline working in conversion mode.

Looking at the discontinuity problem from another viewpoint may give us more insight about the relationship between the output codes and ADC characteristics. Notice that the discontinuities in the output codes show up as gaps composed of empty bins in the output histogram. The gap's width is the difference between ADC's two outputs for two adjacent inputs. Ideally, there should be no gaps since input with enough density should result in consecutive output codes. The ideal correction code for an output code should be the summation of the widths of all those gaps that happen before the output code. Furthermore, to get a better linearity performance after calibration, if the total count of the two codes on the two sides of a gap is too small, these two codes should be merged together and considered as one code after calibration. However, storing a correction code for each output code is too memory consuming. When the nominal gain is set to be less than two, (9)~(12) show that the difference between $\hat{V}'_{in}$ and $\hat{V}_{in}$ is caused by the difference between w_i and 2^{n-i} in each stage i whose digital output d_i=1, excluding a constant offset. Thus, the ideal correction code, which is a quantization of $\hat{V}'_{in}$-$\hat{V}_{in}$ is a linear combination of all those stages' contribution plus a constant offset. That is to say, if we can find a group of correction codes corresponding to the w_i-2^{n-i} of each stage (each bit), defined as

$$\boldsymbol{c}_b = \begin{pmatrix} c_{b1} & c_{b2} & \cdots & c_{bn} \end{pmatrix}^T \tag{13}$$

then the correction codes c_i for a specific output code $\boldsymbol{D}_i$ can be easily generated by adding up the bit correction codes corresponding to those stages whose digital output is 1, plus the offset. That is

$$c_i = \boldsymbol{D}_i \times \boldsymbol{c}_b + c_{os} \tag{14}$$

where

$$\boldsymbol{D}_i = \begin{pmatrix} d_{1i} & d_{2i} & \cdots & d_{ni} \end{pmatrix} \tag{15}$$

Given the output histogram, for any output code $\boldsymbol{D}_i$ with count >0, c_i can be calculated by summing gaps' widths up (including an unknown offset). Then ideally

$$\boldsymbol{D} \times \boldsymbol{c}_{bit} = \boldsymbol{C} \tag{16}$$

where

$$\boldsymbol{D} = \begin{pmatrix} \boldsymbol{D}_1 & 1 \\ \boldsymbol{D}_2 & 1 \\ \vdots & \vdots \\ \boldsymbol{D}_N & 1 \end{pmatrix} = \begin{pmatrix} d_{11} & d_{21} & \cdots & d_{n1} & 1 \\ d_{12} & d_{22} & \cdots & d_{n2} & 1 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ d_{1N} & d_{2N} & \cdots & d_{nN} & 1 \end{pmatrix} \tag{17}$$

$$\boldsymbol{c}_{bit} = \begin{pmatrix} c_{b1} & c_{b2} & \cdots & c_{bn} & c_{os} \end{pmatrix}^T \tag{18}$$

and

$$\boldsymbol{C} = \begin{pmatrix} c_1 & c_2 & \cdots & c_N \end{pmatrix}^T \tag{19}$$

N is the total number of output codes whose count is greater than 0. One column is added in $\boldsymbol{D}$ and one element is added in $\boldsymbol{c}_{bit}$ for correcting the offset. $\hat{\boldsymbol{c}}_{bit}$ should be chosen as

$$\hat{\boldsymbol{c}}_{bit} = \arg\min_{\boldsymbol{c}} \left\| \boldsymbol{C} - \boldsymbol{D} \times \boldsymbol{c} \right\|^2 \tag{20}$$

Solving $\dfrac{\partial \left\| \boldsymbol{C} - \boldsymbol{D} \times \boldsymbol{c} \right\|^2}{\partial \boldsymbol{c}} = 0$ gives

$$\hat{\boldsymbol{c}}_{bit} = \left(\boldsymbol{D}^T \boldsymbol{D} \right)^{-1} \boldsymbol{D}^T \boldsymbol{C} = \boldsymbol{D}^+ \boldsymbol{C} \tag{21}$$

where $\boldsymbol{D}^+$ is the pseudo inverse of matrix $\boldsymbol{D}$. Given the output histogram, $\boldsymbol{D}$ and $\boldsymbol{C}$ can be easily obtained and applying (21) will directly give $\hat{\boldsymbol{c}}_{bit}$, which is the correction code for each stage and the offset. After $\hat{\boldsymbol{c}}_{bit}$ is obtained, for any digital output $\boldsymbol{D}_i$, the corresponding correction code $\hat{c}_i$ can be calculated as

$$\hat{c}_i = \begin{pmatrix} \boldsymbol{D}_i & 1 \end{pmatrix} \times \hat{\boldsymbol{c}}_{bit} \tag{22}$$

IV. Implementation and Performance Issues

Equation (22) shows that instead of storing a correction code for each output code, we only need to store a small group of correction codes corresponding to each pipeline stage and an offset. The correction code for a particular output code can be easily calculated from $\hat{\boldsymbol{c}}_{bit}$. As a result, when implementing the digital correction on-chip, the memory requirement is dramatically reduced (from the order of 2^n to that of n, where n is the number of stages). And only

an adder is needed for calculating the correction code of a digital output.

Since $\hat{c}_{bit}$ is calculated based on $\boldsymbol{D}$ and $\boldsymbol{C}$, which are obtained from the output histogram, no external control of each pipeline stage is needed. The pipeline is exactly the same in the calibration mode as in the conversion mode. And since the algorithm uses the digital output, which changes when the input pass the trip point of the comparator, the algorithm should give a better estimation of the voltage jump in the transfer curve than the algorithm proposed in [4].

V. Simulation Results

The algorithm is applied to a 15-bit pipelined ADC in MATLAB simulation. The pipeline is composed of 17 stages, and each stage gives a 1-bit output. The nominal gains of the first 11 stages are set to be 1.93 and the rest 6 stages have nominal gains set to 2. The last bit is truncated after calibration to give a 15-bit digital output.

Fig. 3 shows the ADC's nonlinearity before calibration. Fig. 4 and Fig. 5 show the calibrated ADC's nonlinearity after applying the algorithm proposed in this paper and in [4], respectively. The un-calibrated ADC has peak INL about 1000 LSB and 1 LSB DNL. As shown in Fig. 4, the proposed algorithm reduces the INL to be less than 1 LSB and DNL to be about 0.3 LSB after calibration, which is comparable to the result of applying the algorithm proposed in [4], which is about 1 LSB INL and 0.5 LSB DNL as shown in Fig. 5.

VI. Conclusion

This paper presents an algorithm for the digital calibration of pipelined ADC's. The algorithm calculates the correction codes using the histogram result of a code density test. No disturbance of the pipeline or precision ramp input is needed for the calibration. The algorithm can significantly improve the linearity of a pipeline ADC by removing the discontinuities in the output. The digital calibration can be easily implemented on-chip with very small hardware overhead. The algorithm can be applied to pipeline or cyclic ADC architectures with 1-bit/stage or multi-bit/stage sub-radix configuration.

References

[1] B.-S. Song, M. F. Tompset, and K.R. Lakshmikumar, "A 12-b 1Msample/s capacitor error-averaging pipelined A/D converter," IEEE J. Solid-State Circuits, vol. 23, no. 6, pp. 1324-1333, Dec. 1988.

[2] H. Ohara et al., "A CMOS programmable self-calibrating 13-b eight-channel data acquisition peripheral," IEEE J. Solid-State Circuits, vol. SC-22, pp. 930-938, Dec. 1987.

[3] E. G. Soenen and R. L. Geiger, "An architecture and an algorithm for fully digital correction of monolithic pipelined ADCs," IEEE Trans. Circuits Syst. II, vol. 42, pp. 143–153, Mar. 1995.

[4] A. N. Karanicolas, H.-S. Lee, and K. L. Bacrania, "A 15-b 1-Msample/s digitally self-calibrated pipeline ADC," IEEE J. Solid-State Circuits, vol. 28, pp. 1207-1215, Dec. 1993

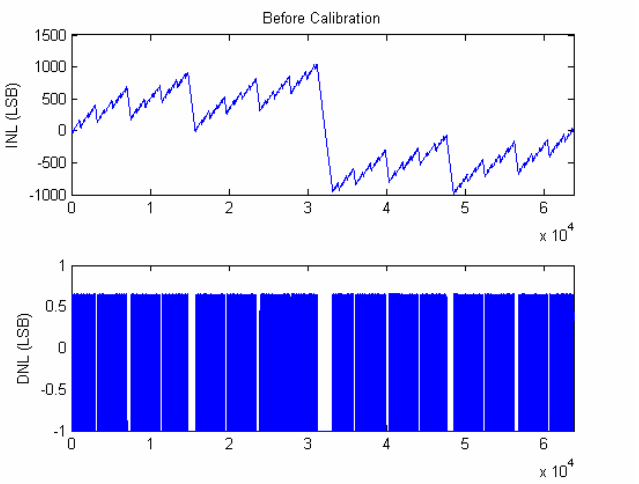


Figure 3. ADC's linearity before calibration

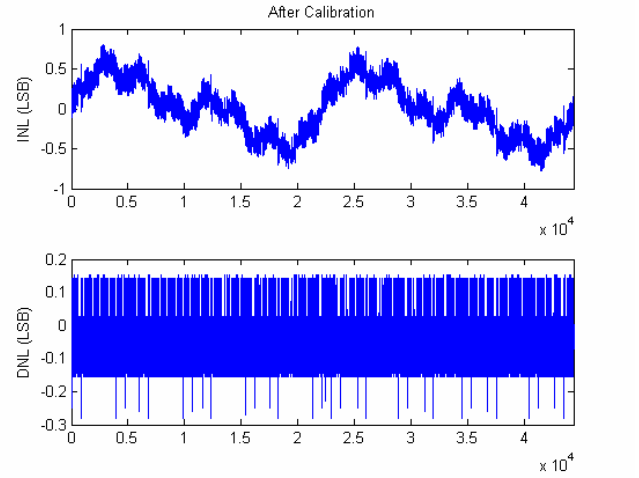


Figure 4. ADC's linearity after calibrated using the proposed algorithm

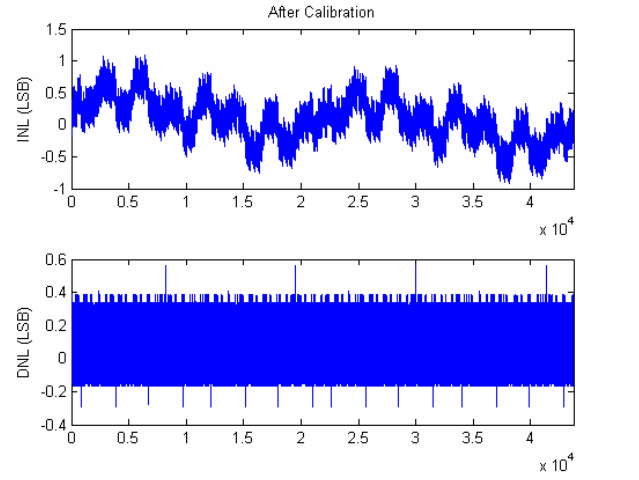


Figure 5. ADC's linearity after calibrated using the algorithm in [4]