Histogram Based Correction of Matching Errors in Subranged ADC

Jonas Elbornsson
Jan-Erik Eklund

Division of Communication Systems
Department of Electrical Engineering
Linköpings universitet, SE-581 83 Linköping, Sweden
WWW: http://www.comsys.isy.liu.se
Email: jonas@isy.liu.se

5th November 2001

Report No.: LiTH-ISY-R-2400
Submitted to ESSCIRC’01

Technical reports from the Communication Systems group in Linköping are available by anonymous ftp at the address ftp.control.isy.liu.se. This report is contained in the file 2400.pdf.
Abstract

A method for correcting matching errors in an ADC is presented. The method uses the unknown input data from the application and does not require any test signal. Two algorithms for implementing the method are compared. One algorithm is general and works on any type of ADC. The other algorithm utilizes the subranged architecture of a specific ADC and is very simple to implement in hardware. The signal quality is similar for both algorithms.

Keywords: A/D converter, estimation, measurement, histogram
Histogram Based Correction of Matching Errors in Subranged ADC

J. Elbornsson
Linköpings universitet
Department of Electrical Engineering
SE-581 83 Linköping, SWEDEN
jonas@isy.liu.se

J.-E. Eklund
Ericsson Microelectronics AB
SE-581 17 Linköping, SWEDEN
jan-erik.eklund@mic.ericsson.se

Abstract
A method for correcting matching errors in an ADC is presented. The method uses the unknown input data from the application and does not require any test signal. Two algorithms for implementing the method are compared. One algorithm is general and works on any type of ADC. The other algorithm utilizes the subranged architecture of a specific ADC and is very simple to implement in hardware. The signal quality is similar for both algorithms.

1. Introduction
In modern communication systems, such as xDSL modems and radio base stations, a greater part of the signal processing is done in the digital domain. This increases the demands on high performance of the analog to digital converter, ADC. Calibration of an ADC is time-consuming and expensive. Therefore a lot of costs can be saved if the errors in the ADC can be automatically estimated and removed under drift.

Several methods for compensating for these errors have been presented, [1, 2]. These methods are based on equalization of the histogram of the output signal. But all of them assume either a training signal with known probability density or an additional ADC with higher accuracy than the ADC-under-test to estimate the probability density.

In this paper, we will present a method for estimating matching errors in subranging AD converters, that does not assume any knowledge of the input signal, except that its amplitude distribution function should be smooth. We will also compare two algorithms for implementing the method. The first algorithm does not assume any knowledge of the subranging structure of the ADC. In the second algorithm, the subranging structure of the test ADC is explicitly used for simplifying realization in hardware.

2. ADC Description
The algorithms are tested on an ADC of the type subranged Successive Approximation-ADC. The reference level generation is implemented as a combination of voltage sharing in resistor ladders and charge redistribution in capacitors. Each resistor ladder together with a capacitor form one subranging stage, Fig. 1, [3, 4]. The analog signal, \( v_a \), is sampled in the capacitor \( C_1 \) by bottom plate sampling. The binary search algorithm is then implemented as: select \( a, b, c \) so that

\[
\min(v_a C_1 - (V_1 a C_1 + V_2 b C_2 + V_3 c C_3)) > 0 \quad (1)
\]

The digital output code from the ADC, \( D \), is then formed from \( a, b, c \) which are represented by 4, 4, 5 bits respectively. The subranging stages are overlapping [5] and a redundant code is formed as

\[
D = (a \cdot 12 + b) \cdot 24 + c \quad (2)
\]

where the bits in \( a \) are the most significant. (A non-redundant code would be formed as \( D = (a \cdot 2^4 + b \cdot 2^5 + c) \). In [5] the redundancy is used for dynamic error correction. Here, we will use it for matching error correction.

The test ADC is implemented in a 0.6 \( \mu \)m CMOS 2M2P process, where the resistors are made of poly-silicon. The clock frequency during the test was 10MHz (sampling frequency 10/16MHz) and the circuit consumed 2mW from a 3.3V supply.

![Fig. 1. Structure of test setup.](image-url)
3. Theory

In this section, a method for estimation and correction of mismatch errors in ADCs is described. Two algorithms for implementing the method, one general and one subrange specific, are also described.

3.1. Common properties and definitions

The basic ADC operation is summarized as: Sample an analog signal, compare the sample with a set of analog reference levels and assign the output to a digital code which corresponds to a selected reference level.

The matching error origin in that we cannot produce reference levels with enough accuracy. Thus, we do not know exactly what a digital output code from the ADC means. The correction methods use a look-up table with the content \( LUT \) for adjusting the digital output codes from the ADC, \( D(2) \), to corrected values, \( C \), in such a way that the transfer function from the analog input signal, \( v_a \), to the value \( C \) is linear (except for quantization).

The information we use to calculate \( LUT \) is a histogram, \( H_M \), of the ADC output, \( D \). Fig. 2(a) shows an analog reference with matching errors. The height of each step in the reference corresponds to the probability that the analog signal is within this range and thus the number of occurrences of the corresponding code at the analog output, Fig. 2(b). The problem is that the density of a specific code is also dependent of the amplitude distribution of the analog input signal, \( v_a \), which is unknown. With the assumption that two neighboring codes should have approximately the same code density, an estimate of the expected histogram, \( H_E \), can be calculated from \( H_M \), by a filtering operation. The relation between \( H_E \) and \( H_M \) then give a correction parameter vector \( \Theta \). The address to the look-up table, \( A \), is calculated from \( D \) and the vector \( LUT \) is calculated as

\[
LUT(A) = \sum_{i=1}^{A} \Theta(i) + A \tag{3}
\]

Most of the codes will occur in the middle of the range, Fig. 4. The estimate will be best in the middle. If we have a repetitive structure, the errors in the ends can be extrapolated, but the result is not perfect. This can be seen in Fig. 5, where the performance is less ideal for large amplitudes.

3.2. General estimation algorithm

This general algorithm is based on the data word \( D(2) \), which will have a behavior as in Fig. 2(a)(b). The algorithm does not take any information about the subranging into account. The algorithm is described in more detail, and in a more general form in [6]. In this paper, we have modified this algorithm to handle only large errors. This modification greatly reduces the calculation burden. The algorithm is divided into two parts:

\( H_E \) is calculated as a low pass filtered version of \( H_M \). Here a zero-phase filter should be used to avoid drift towards higher levels. This is achieved by forward-backward filtering of a second order butterworth filter.

The next step is to find a mapping, \( D \rightarrow C \), that eliminates the matching errors in \( C \). First the codes corresponding to the largest errors are found by comparing \( H_M \) with \( H_E \).

\[
l = \{D \mid H_M(D) - H_E(D) > \mu \} \tag{4}
\]

Here \( l \) is a vector that contains the codes where the relative deviation between the measured and estimated histograms is larger than \( \mu \). A vector \( \theta = [\theta_1 \ldots \theta_N] \) is used to parameterize the errors in the ADC. The parameters corresponding to levels with large errors, \( \{\theta_i \mid i \in l\} \), are free parameters...
(In the general formulation all the parameters are free). The other parameters are fixed to the relative mean deviation between the measured and the estimated histogram. The histogram of the corrected output, $H_{Ei}(D, \Theta)$, is parameterized with the error vector by interpolation between adjacent histogram values. The error estimate, $\Theta$, is calculated as the minimizing argument of the loss function

$$V(\Theta) = \sum_{D=1}^{N} (H_M(D) - H_{Ei}(D, \Theta))^2$$

where $N$ is the number of levels in the ADC. The loss function is minimized numerically with the steepest descent method.

The address to the look-up table in Fig. 1 is $A = D$ and the corrected value is $C = LUT(A)$.

### 3.3. Implementation specific algorithm

The implementation specific algorithm is directly based on the separation into subranging stages, and is intended to be simple to implement in hardware.

The algorithm is illustrated by an example of how a matching error is found in a 2-stage subranging SA-ADC with 3 bits in both the first and second stage and a redundant code, Fig. 2(c-h). Fig. 2(c) shows how the stages map to each other. The output (in this example) is formed as

$$D = y \cdot 6 + z$$

The codes marked with bold face in Fig. 2(c-h) $\{y \in \{000 - 111\}, z \in \{001 - 011\}\}$ are unique, and ideally, the result should be a combination of these codes. The other combinations of codes occur when there are matching errors. In this example we assume that, if there are no matching errors, $H_M$ is a constant function, Fig. 2(d).

If the physical reference level, which is corresponding to the code $y = 011$, is a bit too high, Fig. 2(e), the code $\{y = 010, z = 111\}$ will start to appear. The mapping of $\{y = 010, z = 111\}$ to an output code (6) will, however, give $D = 19$, which is the same as for $\{y = 011, z = 001\}$. The number of occurrences of the code $D = 19$ is then too high, Fig. 2(f).

If the physical reference level, which is corresponding to the code $y = 011$, is a bit too low, Fig. 2(g), the code $\{y = 010, z = 110\}$ will almost not appear at all. The number of occurrences of the code $D = 18$ is then too low, Fig. 2(h).

As seen in Fig. 2(f)(h), the positions of the matching errors in the histogram is known. This information is used to simplify the algorithm:

Divide $H_M$ into fractions around the positions of the errors, $H_M = \{H_M(D), D = 32 \cdot i, \ldots, 32 \cdot (i + 1) - 1\}$ (+ in Fig. 3). (The value 32 comes from 5 bits in the fine reference in the implemented ADC.) For each segment, $i$, calculate $H_{Ei}$ as an LMS curve fit of a straight line to $H_M$, (dashed in Fig. 3). (This can be compared to the filtering operation in 3.3.2). Since $H_{Ei}$ is heavily dependent of the error, this first estimate is not very good. Therefore, select the values close to $H_{Ei}$ (o in Fig. 3) and calculate a new estimate $H_{Ei}$ (solid line in Fig. 3). The size of the matching error is then given as the relative error as:

$$\Theta(i) = \sum_{k=1}^{32} \frac{H_M(k) - H_{Ei}(k)}{H_{Ei}(k)}$$

From (2), we get the address to the look-up table in Fig. 1 as $A = (\alpha \cdot 12 + b)$ and the corrected data as $C = LUT(A) \cdot 24 + c$.

### 4. Implementation

The algorithms was tested as post processing in MATLAB of measured data. The cost of implementing them in hardware is of great importance for the usefulness and is therefore investigated. Table 1 shows the number of operations and an estimate of required area and power, which is needed to do an implementation in a 0.25 $\mu$m CMOS process. Only the collection of the histograms and the look-up table access are done at the sampling frequency. The correction algorithms are calculated batch wise and thus several millions of clock periods are available. This part can be implemented as a very small processor, which also can handle the control.

### Table 1. Implementation cost

<table>
<thead>
<tr>
<th>Operation</th>
<th>Algorithm General Specific</th>
<th>Units</th>
</tr>
</thead>
<tbody>
<tr>
<td>Histogram</td>
<td>64 K</td>
<td>0.3 bits of RAM</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1 mW/(MSamples/s)</td>
</tr>
<tr>
<td>Additions</td>
<td>1 M</td>
<td>74 k Gates</td>
</tr>
<tr>
<td>Multiplications</td>
<td>1 M</td>
<td>51 k mm²</td>
</tr>
<tr>
<td>Divisions</td>
<td>13 k</td>
<td></td>
</tr>
<tr>
<td>HW including control</td>
<td>20 k</td>
<td>20 k Gates</td>
</tr>
<tr>
<td>control</td>
<td>0.5 mm²</td>
<td></td>
</tr>
<tr>
<td>Memory for calculations</td>
<td>15.6 mm²</td>
<td></td>
</tr>
<tr>
<td>Look-up table</td>
<td>182 k bits of RAM</td>
<td></td>
</tr>
<tr>
<td>Total area</td>
<td>20 mm²</td>
<td></td>
</tr>
<tr>
<td>Total power</td>
<td>1 mW/(MSamples/s)</td>
<td></td>
</tr>
</tbody>
</table>

### 5. Measurements

In this section we show results from measurements on the ADC described in section 2. Both estimation algorithms described in section 3 are evaluated here.

#### 5.1. Data acquisition

A sequence of Discrete Multi Tone (DMT) symbols was used as input signal to the ADC when calculating the histogram. In Fig. 4, an example with $5 \cdot 10^5$ samples is shown.
5.2. Evaluation

The correction algorithms have been evaluated with sinusoidal signals. The signal power has been varied and the Signal to Noise and Distortion Ratio (SNDR) and Spurious Free Dynamic Range (SFDR) has been measured before and after correction for each signal, Fig. 5. Both methods show similar results and the peak improvement is about 8dB for SNDR and 14dB for SFDR.

6. Discussion

6.1. Reference signals

The assumption was that two neighboring codes would have similar densities. For the large data sets in this test, this is true enough. The results in this paper are based on a histogram from a DMT signal, which has a nice distribution. A more nasty set of signals, with sequences of one strong and one weak (<32 LSBs) sine wave was also tested. The correction works as good as for DMT, but the data set has to be increased about 5 times.

6.2. Validation of the algorithms

The results show that matching errors in an ADC can be corrected on the fly. There are mainly two differences between the algorithms that was tested.

First, the general algorithm requires a large memory for storing the Jacobian of the histogram, and will thus require a large area. The subrange specific algorithm requires only 1.8mm² and 1 mW/MSamples/s) which makes it feasible.

The second difference considers overflow in the histogram accumulators. The filtering operation in the general algorithm rely on the entire histogram. Handling of overflow (reset or scaling) must thus be done on the entire histogram simultaneously. The ratio between high and low density will thus be preserved, which limits the precision in the ends of the histogram. In the subrange specific algorithm, the filtering is done on each segment separately, Fig. 3. Overflow within one segment can be handled as divide by 2 or 4 (shift operation) within the segment only. The histogram will thus be compressed. In principle, this means that data is acquired for a longer time in the low density regions and precision is increased for these codes.

7. Acknowledgment

The authors would like to thank Dr. Dake Liu at Linköping University for help with estimating the implementation cost.

8. References