DIGITAL OFFSET COMPENSATION OF TIME-INTERLEAVED ADC USING RANDOM CHOPPER SAMPLING.

Jan-Erik Eklund
Ericsson Components AB
S-164 81 Kista, SWEDEN
jee@ifm.liu.se

Fredrik Gustafsson
Linköping University
S-581 83 Linköping, SWEDEN
fredrik@isy.liu.se

Abstract

An ADC using several parallel cells suffers from offset differences between the cells, which causes non-harmonic distortion. A method for removing the offset in the digital domain is proposed. The method is based on a PRBS-controlled chopper at the ADC input, which transforms any input signal to noise. The randomization controls the batch size required for removing the offset by a mean value calculation. The measured results are SFDR = 72dB and SNDR = 59.0dB @ 22MS/s, an improvement of 19dB and 10dB respectively.

1. INTRODUCTION

1.1. Parallel ADC

Parallel time-interleaved ADCs have become popular as a way of increasing the sampling frequency in pipe-lined [1] or Successive-Approximation ADCs, [2][3]. This is a study of a SA-ADC. The conversion of one sample requires k clock periods. The effective sampling rate of a single SA-ADC cell is thus $f_{Scell} = f_{clk} / k$. With m parallel time-interleaved cells, the sampling rate is increased to $f_S = m \cdot f_{clk} / k$. In our case $k = m = 16$ and $f_S = f_{clk}$.

1.2. Problem with offset

The 16 different ADC cells will have different offset values, $o_1, o_2, o_3, o_4 \ldots o_{16}$ respectively. Assume that the sampled signal is a sequence of values $\{s(1), s(2), s(3) \ldots\}$. The values $\{s(1), s(17), s(33) \ldots\}$ are A/D converted in cell 1 and $\{s(2), s(18), s(34) \ldots\}$ in cell 2 and so on. The offset values are added to the A/D converted sequence as

$$\begin{aligned} &s(1)+o_1, s(2)+o_2, s(3)+o_3, s(4)+o_4, \ldots s(16)+o_{16}, \\ &s(17)+o_1, s(18)+o_2, s(19)+o_3, s(20)+o_4, \ldots \end{aligned} \quad (1)$$

In the frequency domain, the offset variation causes tones at $p \cdot f_S / 16$ where p is an integer [0, 15], Fig. 3. In most applications, the offset must be removed.

There are analog techniques to handle offset compensation, [2][3]. However, the bottom plate sampling is not so easily implemented. The drawback is the requirement on analog components. The bandwidth of the sampling is limited by the comparator bandwidth, since it supplies the virtual ground for the sampling. It is therefore interesting to look for a digital compensation method.

One simple method would be to calculate the mean value a batch of N samples from each cell and subtract it from the signal. The first cell would have the mean value

$$\begin{aligned} &(s(1)+o_1 + s(17)+o_1 + s(33)+o_1 + \ldots) /N = \\ &o_1 + \mathrm{E}[s(16\, t)] \end{aligned} \quad (2)$$

The problems with this method is that there is no way of determine the required batch size N. It is depending on the input signal. In extreme cases e.g. for a continuous sine wave input at the frequency $f = p \cdot f_S / k$ the offset cannot be distinguished from the signal. Randomization in different ways is necessary in order to find the offset in a controlled way.

An example of randomization is found in [4], where simulations of a digital offset compensation method is presented. However, this method assume a pipe-lined ADC and it has not been implemented.

In [5], two methods for background offset compensation are described, however, these require a dedicated design of the entire ADC including analog calibration circuits.

We propose a method that transforms any input signal to noise with mean value = 0 before the offset is calculated and removed. In this way the required batch size N s limited and we determine the relation between N and Signal-to-Noise-and-Distortion-Ratio, SNDR. Since the analog part of the compensation is minimal, the method can easily be used for any type of ADC with differential input. We present a theoretical description of the method and verify it with measurements.

2. DIGITAL OFFSET COMPENSATION

2.1. Principle of operation

The implementation of the proposed method is shown in Fig. 1. The main principle of operation is as follows:

1. Chop the input signal before the ADC by multiplying with a pseudo random sequence of -1 and +1.
2. Pass the signal through the A/D converter. It is now digi-

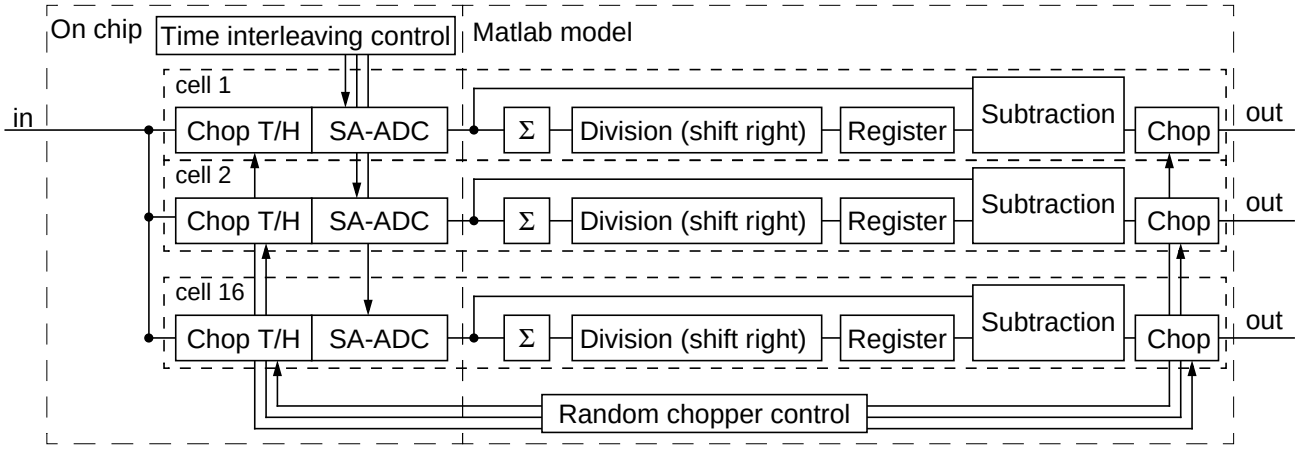

Fig. 1. Block diagram of the parallel ADC with offset compensation. 3 of 16 cells are drawn. The critical parts are implemented in the test chip and the rest is implemented in Matlab.

tized and has an offset pattern as in (1).

3. Estimate the offsets by calculating the mean values of one batch of digital outputs.
4. Store the estimated offset values in registers.

The the offset value in the register is used for correcting the ADC output values. The offsets, which are estimated from one batch of values are used for correcting the next batch. When the offset has been removed, the digital signal is chopped with the same sequence as the input and the signal value is restored.

A new estimate is calculated from each batch and the offset values are thus updated during operation.

2.2. Theory for the batchwise estimation

Let the signal be $s(t)$, the chopper control signal $u(t)$, the offsets o_i, i=1,2, ... , M and the quantization noise (and other disturbances) be denoted $e(t)$. The digital signal $\hat{s}(t)$, without offset compensation can then be written

$$\hat{s}(t) = u(t) \cdot (u(t) \cdot s(t) + o_{t_{modM}} + e(t))$$

Here t_{modM} is the modulo function which picks out the current offset value of the M possibilities. To utilize the parallel structure and simplify notation, we will restrict ourself to one ADC cell. From now on, the index t denotes samples to a particular cell. Offset estimation is performed before the second chopper, so the signal at hand after the A/D converter is

$$y(t) = u(t)s(t) + o + e(t)$$

The most natural strategy to estimate the offset is to average the signal value over a batch of data of size N,

$$\hat{o} = \frac{1}{N}\sum_{t=1}^{N} y(t) = \frac{1}{N}\sum_{t=1}^{N} (u(t)s(t) + o + e(t)) \qquad (3)$$

The estimate's mean and variance are by standard calculations given by

$$E[\hat{o}] = \frac{1}{N}\sum_{t=1}^{N} E[u(t)]E[s(t)] + E[o] + E[e(t)] = o$$

$$Var[\hat{o}] = \frac{1}{N}\sum_{t=1}^{N} Var[u(t)s(t) + o + e(t)]$$

$$= \frac{1}{N}(Var[u(t)]Var[s(t)] + Var[e(t)])$$

$$= \frac{1}{N}(Var[s(t)] + Var[e(t)])$$

Here we have only used an assumption of independence between the signal and chopper sequence, which is easy to satisfy in practice, and $\mathrm{Var}[u(t)] = \mathrm{E}[u^2(t)] = 1$.

2.3. SNDR improvement

The SNDR is here rather naturally defined as the ratio of the energy in the input signal and the mean square error, $s(t)$-$\hat{s}(t)$, in the ADC output signal.

$$\mathrm{SNDR} = \frac{E[s^2(t)]}{E[(s(t) - \hat{s}(t))^2]}$$

Here we distinguish two cases. Without offset compensation, the signal estimate is given by $\hat{s}(t) = u(t)\, y(t)$, and the SNDR is

$$\mathrm{SNDR} = \frac{E[s^2(t)]}{o^2 + Var[e(t)]}$$

With compensation, the signal estimate is $\hat{s}(t) = u(t)\,(y(t)\text{-}\hat{o}\,)$, and

$$\text{SNDR} = \frac{E[s^2(t)]}{Var[\hat{o}]} = \frac{N \cdot Var[s(t)]}{Var[s(t)] + Var[e(t)]} \quad (4)$$

$$= \frac{N}{1 + \frac{Var[e(t))]}{Var[s(t)]}}$$

Thus, SNDR is approximately equal to the batch size N in linear scale.

As an example, resembling the measurements in section 4, let the signal be a sinusoid with amplitude 2000 (12 bits ADC) and the offset difference in the order of o = 5.4. The SNDR without compensation is then

$$\text{SNDR} = \frac{2000^2/2}{5.4^2} = 80000 = 49\text{dB}.$$

This is seen in Fig. 3. With compensation based on a batch of size $N = 10^6$, the SNDR is

$$\text{SNDR} \cong N = 10^6 = 60\text{dB}. \quad (5)$$

The effect of quantization is in this case negligible. Note that if $N < 8 \cdot 10^4$, the performance is degraded, so there is a point where compensation starts to pays off.

2.4. Adaptivity using forgetting factor estimator

The batch processing can of course be repeated every N sample. The design parameter N should be matched to the time variations of the offset. The drawback with using (3) is an offset compensator that takes abrupt jumps, which leads to unwanted tones in the ADC output. A softer compensator can be formed by using a first order filter, also known as a forgetting factor (λ) estimator. The time-variable estimate is computed by

$$\hat{o}(t) = \lambda\, \hat{o}(t\text{-}1) + (1 - \lambda)\, y(t) \quad (6)$$

Here $\lambda < 1$ should be chosen close to 1. It is easily checked that $E[\hat{o}] = o$ and $Var[\hat{o}] = (1\text{-}\lambda) / (1+\lambda)\ (\ Var[s(t)] + Var[e(t)]\ )$. Thus, N corresponds approximately to $1/(1\text{-}\lambda)$.

2.5. Numerical aspects on implementation

The batch average in (3) is conveniently performed in hardware if N is chosen to be 2^n for an integer n. The division is then only n shifts in binary logic. The forgetting factor in the algorithm (6) should be chosen as $\lambda = 1\text{-}\ 2^{-n}$. Multiplication with $1\text{-}\ \lambda$ is again just n shifts, and multiplication with λ is just a subtraction of the estimate with a shifted version of the same estimate. A standard rule of thumb is that the forgetting factor is matched to the batch method by $N = 1/(1\text{-}\ \lambda)$ which in both cases lead to 2^n. Thus both methods are comparable from a numerical point of view.

Finally, there might be practical aspects that lead to a combination of batch processing and forgetting factor. For instance, instead of using $N = 10^6$, one can take $N = 10^3$ and $\lambda = 1\text{-}2^{-10}$ and get the same SNDR but with a softer ADC output signal.

3. IMPLEMENTATION

3.1. Parallel SA-ADC overview

The critical parts of the offset compensation is implemented in a 16 cells 12 bits parallel SA-ADC. The basic principle for this ADC cell was presented in a single cell implementation, [6].

3.2. Random chopper and offset compensation

The digital offset compensation consists of two parts: the analog random chopper sampling and the digital correction logic. The random chopper sampling is analog and therefore critical. It is thus implemented in a test circuit. The random chopper works as follows:

The input signal is differential $v(t) = va(t) - vb(t)$. The input signal is connected through sw_1 and sw_2 and sampled using the bottom plate sampling switches, sa_1 and sa_2, Fig. 2. The sampled signal is then represented by the charge

$$Q(t) = C_{S1} \cdot va(t) - C_{S2} \cdot vb(t). \quad (7)$$

By closing the switches ch_1 and ch_2 instead of sw_1 and sw_2 when sampling the signal the polarity is changed. The sampled charge is

$$Q_{ch}(t) = C_{S1} \cdot vb(t) - C_{S2} \cdot va(t). \quad (8)$$

Assume $vb(t) = -va(t)$, and $C_{S1} = C_{S1}$. The sampled charges are then $Q(t) = 2 \cdot C_{S1} \cdot va(t)$ and $Q_{ch}(t) = -2 \cdot C_{S1} \cdot va(t)$ respectively. With a proper layout, the effect on the signal path is minimized to two pass-transistor switches.

The digital correction logic is verified in Matlab. This gives the possibility to test different algorithms. The chopper control sequence is implemented as a PRBS-Matlab model.

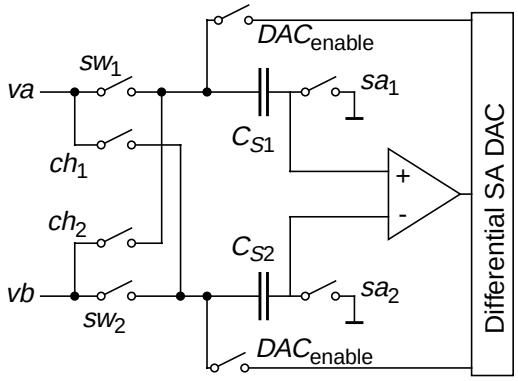

Fig. 2. Sampling circuit including chopping function. The principle implementation of the SA-DAC

4. MEASUREMENTS

The output spectrum of the ADC without offset compensation is shown in Fig. 3. The SFDR = 53dB and SNDR = 49dB due to the offset. The measurements are done at low speed, f_S = 22MS/s, in order to avoid non-idealities in the ADC to limit performance. The offset values in the ADC range from -10 to 13 with an RMS value of 5.4.

The measurements agree well with theory. For $N = 3 \cdot 10^5$, the measured SNDR = 55dB, which very close to the expected 54.8dB, (4). For long sequences, SFDR = 72dB and SNDR = 59.0dB Fig. 4, which is an improvement of 19dB and 10dB respectively, compared to the uncorrected signal in Fig. 3.

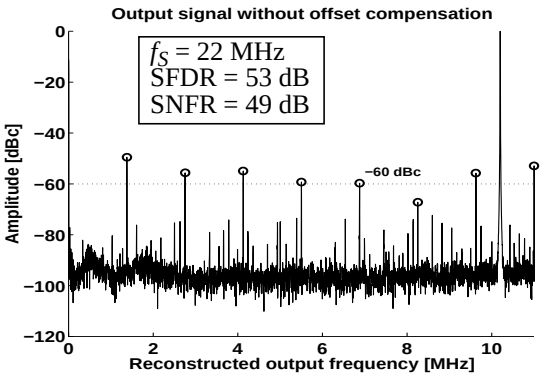

Fig. 3. FFT of output signal with offset error marked with circles.

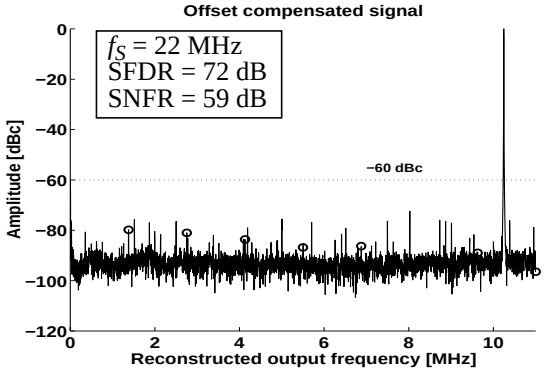

Fig. 4. FFT of offset compensated signal. f_S = 22MHz.

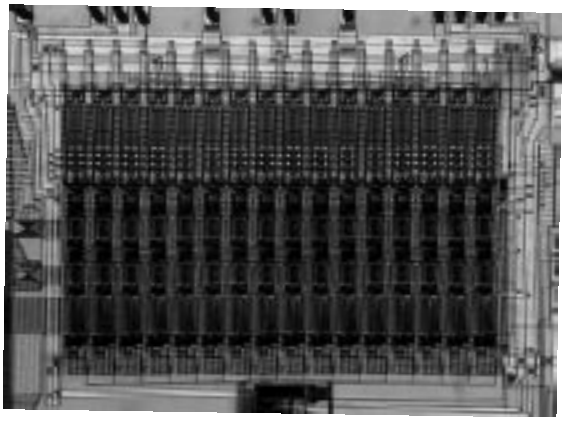

Fig. 5. Chip photo, 12 bits SA-ADC 16 cells.

The chopper at the input increases the background noise floor 1.6dB compared to the measurement without chopper. The test signal is generated by a freqyency synthesizer, followed by a filter. These components are not designed for the varying load which is a result of the chopper. This problem can be solved by either a better signal source or a reset of the sampling capacitor.

5. CONCLUSIONS AND DISCUSSION

5.1. General

We have demonstrated a method of handling the offset in a parallel ADC in the digital domain. We have verified the theoretical description with measurements. The method is independent of the input signal and thus no calibration signal is needed. The algorithm is adaptive since it constantly measures the offset and improves the offset correction values.

5.2. Impact on the analog domain

This work is relaxing analog requirements in the ADC. No analog offset compensation is needed. The extra analog components are minimized to two switches.

5.3. Impact on the digital domain

The digital part can easily be implemented since no multipliers, dividers nor large memories are needed. One 32 bits accumulator one 12 bit register and one 12 bits adder is required for each ADC cell. The increased latency due to the compensation is only the time for the 12 bits subtraction in the signal path, Fig. 1, approximately one clock period.

6. REFERENCES

[1] Kim K. Y., Kusayanagi N. and Abidi A., “A 10-b, 100MS/s CMOS A/D converter”, IEEE Journal of Solid State Circuits, vol. 32, March 1997, pp. 302-311

[2] Yuan J. and Svensson C., “A 10-bit 5MS/s successive approximation ADC cell used in a 70MS/s ADC array in 1.2μm CMOS”, IEEE Journal of Solid State Circuits, vol. 29, August 1994, pp. 866-872,

[3] Kusayanagi N., Choi T., Hiwatashi M.,Segami M., Akasaka Y. and Wakabayashi T., “A 25 Ms/s 8-b-1-Ms/s 10-b CMOS Data Acquisition IC for Digital Storage Oscilloscopes”, IEEE Journal of Solid State Circuits, vol. 33, March 1998, pp. 492-496

[4] H. Jin, E. Lee, M. Hassoun, “Time-Interleaved A/D Converter with Channel Randomization” ISCAS’97, June 9-12, 1997, pp. 425-428.

[5] K. Dyer, D. Fu, P. Hurst and S. Lewis, “A comparision of monolithic background calibration in two time-interleaved analog-to-digital converters” ISCAS’98, pp 13-16.

[6] J.-E. Eklund, “A 200 MHz cell for a parallel-successive-approximation ADC in 0.8 μm CMOS, using a reference pre-select scheme.” Proceedings of ESSCIRC, vol. 23, Southampton, UK, 16-18 September 1997, pp. 388-391.