A Scalar Interpolator for the Improvement of an ADC Output Linearity

Nikos Petrellis

Abstract—The architecture of an interpolator that can be used for linearity improvement and data compression at the output of an Analog Digital Converter (ADC) is presented in this paper. The developed interpolator can correct ADC linearity errors, increase its dynamic resolution or reconstruct signals from fewer samples in non-uniform distance. Its output is available both in compressed or uncompressed form. Multiple interpolators can be connected in series using the uncompressed outputs for further linearity improvement. A 3-stage interpolator with 9-bit input and 12-bit output typical resolution is implemented using 72% of the Logic Elements of a low cost Altera Cyclone II EP2CSF256C8N Field Programmable Gate Array (FPGA) and is tested using the ADC and the touch pad pressure sensor of a commercial microcontroller development board. A Signal to Noise Ratio (SNR) improvement by more than 7dB (21%) was measured using the microcontroller ADC output. A 4.54 times lower Mean Square Error (MSE) was achieved at the microcontroller touch pad sensor output and a 26.5 times lower MSE was measured when an image with degraded resolution was processed by the proposed interpolator.

Index Terms—Interpolation, Analog/Digital Conversion, Linearity, Post Processing, Compression

I. INTRODUCTION

The sampling frequency and the resolution of an ADC have to be increased in order to achieve lower error in the digitization process of a signal. The linearity errors of an ADC can dramatically change between different input signal frequencies. Although at low frequencies an ADC behaves according to its typical specifications, its behavior may be poor when operating at high input signal frequencies. Customized calibration methods can be incorporated in the same chip with the ADC, correcting the linearity errors of the ADC core.

Instead of employing ADCs with ever increasing throughput and resolution as well as customized calibration methods, a different solution can be adopted by employing generic post-processing modules that can be used for the correction of the linearity errors of different ADC modules.

The Differential or Incremental Non-Linearity (DNL/INL) errors that are corrected by post-processing modules are static and can be estimated by applying DC levels at the ADC input or more often using techniques like Histograms [1] or best fit curves [2]. The histogram approach includes the excitation of an n-bit ADC with a ramp signal in order to get the initial 2n outputs. Then, the ramp signal is shifted up and the procedure is repeated by recording another set of 2n outputs. The Least Squares method is then used at the differences of the corresponding outputs to estimate a number of parameters that characterize the input non-linearity [3]. Dynamic linearity errors are measured by Signal to Noise and Distortion Ratio (SNDR), Spurious Free Dynamic Range (SFDR) and Total Harmonic Distortion (THD) and these parameters depend on the operating frequency of the ADC. A Built-in Self Test (BIST) module is often embedded with the ADC under test providing a real time ADC characterization concerning its dynamic features [4]. It is obvious in this case that the SNR of the BIST module should be much higher than the SNR of the ADC under test.

DNL linearity errors can be corrected by adding appropriate signed error correcting factors that are stored in large look up tables. These factors require calibration based on the measurement of the INL/DNL [5]. An accurate saw tooth signal generator is used in [6] to estimate the INL and extract error compensation coefficients of pipeline ADCs. A digital correction performed in a time interleaved ADC architecture is examined in [7]. Each sub channel ADC is implemented as a dynamically configured pair of ADCs with lower resolution. The problem of error source removal is formed as a matrix problem in [7] and is implemented in hardware.

The post processing module presented in this paper for the enhancement of the linearity of an ADC output is based
on Interpolation. The conversion of the ADC output to an analog signal through an ideal Digital to Analog Converter (DAC) is actually a zero-order interpolation. First order interpolation is more efficient assuming that two successive ADC values are connected linearly. Although first-order is more accurate than zero-order interpolation, it cannot model the curvature and slope of the original ADC input signal. Cubic spline interpolation [8] can give more accurate approximations when a curve should fit to more than two points but still it is not guaranteed that the resulting cubic function will fit to all ADC input signals. Moreover, the complexity for implementing a cubic spline interpolation in hardware is much higher than first order interpolation. In [9] a barycentric interpolation is described derived either by the repeated use of Bernstein’s inequality and Taylor’s theorem, or by truncating a Lagrange-type series. The interpolator derivatives are also estimated. Its application on uniform or non-uniform and long intervals is examined. The author of [9] shows how the complexity of the calculations can be efficiently decreased although multiplications and divisions are still required.

In [10] the interpolation is also performed by a weighted summation of a number of preceding and succeeding available samples in order to estimate the current signal value. The Least Mean Lattice (LSL) interpolation algorithm is compared to a QR Decomposition LSL algorithm (QRD-LSL) developed by the authors of [10] that shows a 4dB lower squared error.

A first order interpolation in time is used in this paper attempting to achieve a very low cost hardware implementation that does not require complicated operations like multiplications, divisions, Fourier transforms, etc. Although, this type of interpolation does not take into consideration the slope and curvature of the initial signal it can significantly improve low resolution signals highly distorted by the digitization process. More specifically, the SNR of a sinusoidal signal is improved by 7dB (21%) using the proposed post processing module. Its use with different type of signals like sensor outputs or signals representing images is also tested.

Except from the SNR improvement, the described architecture can also perform real time compression in the case of sparse or slowly changing signals. The scalability of the proposed architecture is another critical feature allowing the achievement of higher linearity if multiple interpolators are connected in series or if an interpolator is used recursively.

The low complexity of the proposed interpolator is proven by the fact that three interpolators with 9, 10 and 11 bit input resolution and their corresponding decompression units at their outputs fit in a small Altera Cyclone II EP2C5F256C8N FPGA using only 72% of its Logic Elements (LEs).

The modeling of the interpolation method used is described in Section 2. The architecture and implementation issues of the interpolator are discussed in Section 3. The simulation and experimental results are presented in Section 4.

II. THE PROPOSED INTERPOLATION METHOD

In the zero-order interpolation (or zero-order hold) the intermediate points of a signal between two successive known samples take the value of the first sample. This is actually the output of a DAC if its input is connected to the ADC and no filtering is used. The first-order interpolation (or first-order hold) approximates the intermediate points using a piece of line connecting them and it is more accurate than zero-order hold if the distance between the two successive samples is small enough. In practical applications a first-order or a different type of interpolator is implemented through filtering.

The estimation of the intermediate points between two successive ADC outputs (not necessarily in uniform intervals) with first order hold is shown in Fig. 1. The employed interpolation method does not use the ADC sampling rate since the digital circuit that implements this interpolator uses a local higher frequency clock. The original ADC output consists of the values $V_1$, $V_2$, etc. The solid line in Fig. 1 is the zero-order interpolation between these values. The value $V_1$ appears for $N_1$ interpolator clock periods. Then the value $V_2$ appears for $N_2$, etc.
The proposed interpolation scheme is based on replacing half of the occurrences of the last sample value by the average value of the two successive samples. For example, the $V_1$ values in the last $N/2$ interpolator clock periods are replaced by $(V_1+V_2)/2$. The dashed line in Fig. 1 shows the resulting interpolated signal. Additional intermediate points could have been estimated if the aforementioned procedure is applied recursively.

This interpolation procedure is successful if the original signal is monotonically increasing or decreasing between each pair of successive known sample values. This can be viewed graphically in Fig. 2a where the dashed line (interpolated signal) is obviously closer to the original curve. In Fig. 2b instead, where the input signal does not change monotonically between successive sample values, the proposed interpolation scheme does not guarantee a lower approximation error. The case of Fig. 2b may appear when the successive ADC outputs are retrieved using too long sampling intervals, compared to the input signal frequency.

More formally, if for example the input signal $s(t)$ is monotonically decreasing between the successive samples $s(t_0)=V_1$ and $s(t_{N_1+1})=V_2$ ($V_1>V_2>0$) and the interpolator clock samples $N_1$ times the value $V_1$ before the $V_2$ appears, then the error $\epsilon_1$ between the original signal and the initial digitized one can be expressed as:

$$\epsilon_1 = \frac{N_1}{2} - s(t_k) = \sum_{k=0}^{N_1/2-1} (V_1 - s(t_k)) + \sum_{k=N_1/2}^{N_1-1} (V_1 - s(t_k))$$  \hspace{1cm} (1)$$

The interpolated signal will replace the last $N/2$ samples with the value $(V_1+V_2)/2$ and the corresponding error $\epsilon_2$ will be:

$$\epsilon_2 = \sum_{k=0}^{N_1/2-1} (V_1 - s(t_k)) = \frac{N_1}{2} - s(t_k) = \sum_{k=0}^{N_1/2-1} (V_1 + \frac{V_2}{2} - s(t_k))$$  \hspace{1cm} (2)$$

From (1) and (2) above it is obvious that $\epsilon_2<\epsilon_1$ since:

$$V_1 > \frac{V_1 + V_2}{2}$$  \hspace{1cm} for $\frac{N_1}{2} \leq k < N_1$$

In a similar way, it can be proved that the error of the proposed interpolation is lower if the input signal is monotonically increasing instead of decreasing. If a second interpolation is used at the output of the first interpolator, a second pair of values will be produced. The first is $(3V_1+V_2)/4$ and appears between $V_1$ and $(V_1+V_2)/2$ (after the $N/4$-th sample) and the second is $(V_1+3V_2)/4$ and appears between $(V_1+V_2)/2$ and $V_2$ (after the $3N/4$-th sample). The approximation error is further decreased in this way since the error $\epsilon_3$ between the output
of the second interpolator and the original signal is:

\[ f(t) = \frac{V_1 + V_2}{2} \]

Fig. 2. The interpolation method used in monotonic (a) and non-monotonic (b) intervals.

The error \( \varepsilon_3 \) is lower than \( \varepsilon_2 \) since the relation between the second and fourth terms of equation (4) and the corresponding terms of equation (2) is:

\[ V_1 > V_2 \Rightarrow 4V_1 > 3V_1 + V_2 \Rightarrow V_1 > \frac{3V_1 + V_2}{2} \]

\[ \Rightarrow V_1 - s(t_k) > \frac{3V_1 + V_2}{2} - s(t_k) \]  

and

\[ V_1 > V_2 \Rightarrow 2V_1 > 2V_2 \Rightarrow 2V_1 + 4V_2 > 6V_2 \Rightarrow \]

\[ 4V_1 + 4V_2 > 2V_1 + 6V_2 \Rightarrow \frac{V_1 + V_2}{2} > \frac{V_1 + 3V_2}{4} \]

\[ \Rightarrow \frac{V_1 + V_2}{2} - s(t_k) > \frac{V_1 + 3V_2}{4} - s(t_k) \]

Nevertheless, it has to be noted that the sample \( s(t_{N/2}) \) is not necessarily equal to \( (V_1 + V_2)/2 \), thus the second interpolation is performed between a real and an estimated signal value. If additional interpolations are recursively performed, the improvement will gradually decrease since they will be applied on estimated intermediate values and not real ones. A trade off has to be made between the desired improvement and the use of additional interpolators connected in series.

### III. THE ARCHITECTURE OF THE INTERPOLATOR

The architecture of the digital interpolator that implements the method described in the previous section is shown in Fig. 3. The M-bits ADC output which is actually the interpolator input is connected to a pair of latches. One of the latches is connected to the interpolator clock CLK enabling each ADC output value to appear at the latch output with one CLK period delay. The input and the output of this latch are digitally compared by CMP that generates a pulse at its output (NEQ) when it detects that the ADC output has changed. The pulse at the NEQ signal is appropriately delayed (signal NEQ_DEL) and combined with the interpolator clock CLK in order to generate a pair of pulses at the signal BUF_EN when the ADC output changes. Moreover, the signal SEL is generated by
combining the outputs of the delay elements at the CMP output so that it is high at the first and low at the second CLK pulse during the time interval where the signal BUF_EN is active.

The interpolator clock CLK has a much higher frequency than the frequency of the ADC input signal, thus the same ADC output value may last several interpolator clock periods. The clock (or load input) used at the second latch is the signal NEQ_DEL that allows a new ADC output value to appear at its output with two CLK periods delay. During these two clock periods, the ADDER generates the sum of the two successive values of the ADC Output or in other words their average multiplied by 2. Since the ADDER inputs are represented with M-bits, its output has a resolution of M+1 bits.

When the NEQ_DEL signal gets active, the K-bit counter at the bottom-right of Fig. 3 is reset and starts counting the number of the interpolator clock periods that the new ADC output value will last. During the two interpolator clock periods between the activation of NEQ and NEQ_DEL the alternating value of the SEL signal selects either the previous value V1 of the ADC output (from the bottom latch) or the ADDER output (V1+V2) to pass through the multiplexor MUX. The ADDER output is M+1 bits while V1 is represented with M bits. This ADC output (V1) is actually shifted left (multiplied by 2) at the input of the MUX by padding it with a ‘0’ as its least significant bit.

The two MUX outputs described in the previous paragraph (2V1 and V1+V2) are stored at the end of a FIFO buffer since the BUF_EN signal allows these values to be stored in two successive interpolator clock periods. During these periods, the K-bit Counter output denotes the number of interpolator clock periods N1 that the value V1 appeared at the ADC output. The K-1 most significant bits of the K-bit Counter are stored in a second FIFO buffer along with the values 2V1 and V1+V2 forming pairs of type: (VAL, CNT). The value VAL appears for CNT interpolator clock periods.

The (VAL, CNT) pairs are the main interpolator output and implement the method described in the previous section. Moreover, the information provided by the (VAL, CNT) pairs is compressed following the technique used in lossless compression standards like LZ77/78 employed by PNG, GIF etc. The higher the sparse level is, or the lower the frequency of the ADC input signal is, the higher compression ratio can be achieved.

The decompression of the interpolator output pairs (VAL, CNT) may be useful either at the side of the module that receives the interpolator outputs or if multiple interpolators need to be connected in series in order to achieve higher linearity (an interpolator requires uncompressed input values). It can be merely implemented by a new pair of FIFO buffers, a down counter with parallel input and a decompression latch at the output of the interpolator FIFO buffers. Each time a new pair (VAL, CNT) appears at the beginning of the interpolator FIFO buffers, it is transferred at the FIFO buffers of the decompression unit. The value VAL from the first pair of the decompression unit FIFO buffers is transferred to the decompression latch and the corresponding CNT value is loaded to the down counter. When the value of the counter is equal to zero, the next pair of (VAL, CNT) values is popped from the decompression unit FIFOs. In this way, the output of the decompression latch represents the uncompressed values of the ADC output that are corrected by the interpolator. The FIFOs are used for the compensation of the rate that the values change at the ADC output and the processing speed of the interpolator and the decompression unit.

Both the interpolator and the decompression unit architecture described in this section, where described in VHDL and implemented on an Altera Cyclone II EP2C5F256C8N FPGA using the low cost FTDI Morph-IC-II evaluation board. The size of the counters is selected to be K=16 while the total FIFO depth is 16. An interpolator with 9-bit input and 10-bit output required 591 LEs (13% of the FPGA LEs) while a decompression unit for the output of this interpolator required 748 LEs (16% of the FPGA LEs). The three stages of interpolation used in the test setup of the next section and their corresponding decompression units occupied 72% (3,334 LEs) of the FPGA LEs.
Fig. 3. The architecture of the interpolator.

IV. EXPERIMENTAL RESULTS

Fig. 4. The experimental setup.
The efficiency of the interpolator described in the previous paragraphs is evaluated using three types of input: a) the values at the ADC output of the MK70FN1M0VMJ12 microcontroller from a Freescale Kinetis TWR K70F120M development board, b) the indications of a touch pad of a TWR K70F120M board which is actually a pressure sensor and c) a 250×500 pixels grey image. The experimental setup used in this paper is shown in Fig. 4. An 8-bit resolution was selected for the Segmentation and Reassembly ADC of the Freescale TWR K70F120M development board. Its input is connected to a function generator that is used to produce a sinusoidal signal. Two periods of the sinusoidal signal are isolated and the corresponding ADC output values are initially stored at the RAM of the MK70FN1M0VMJ12 microcontroller and then transferred to a Host computer. In a similar manner, the Touch Sense Input (TSI) module of the TWR K70F120M board microcontroller is used to sample a touch pad pressure sensor. The host computer controlling the TWR K70F120M board is also connected to an FTDI Morph IC II (with 50 MHz oscillator) evaluation board with an Altera Cyclone II EP2C5F256C8N (maximum operating frequency 320MHz) FPGA where the interpolation method described in Section 3 is implemented. More specifically, 3 interpolators with 9, 10 and 11 bits input resolution are connected in series through decompression units. The results are sent back to the Host for processing and measurement using MATLAB. Except from the ADC and TSI indications, the host may send to the FPGA other type of data like images or virtual sensor indications for the evaluation of the interpolation algorithm.
As far as the ADC data are concerned, the function generator produces a sinusoidal signal of 10 kHz. The period of the sinusoidal signal is high enough compared to the sampling period of the ADC which is 3.125 usec, in order to have short enough sampling intervals where the signal changes monotonically (Fig. 5a). The reconstructed sinusoidal signal from the 1st and the 3rd Interpolator output is shown in Fig. 5b and Fig. 5c respectively. The SNR of the ADC output is 34.7 dB while the SNR of the 1st, the 2nd and the 3rd stage interpolators’ output are 37.9 dB, 40.9 dB and 42.1 dB. Thus, an improvement of 7.4 dB (or 21.3%) is achieved in the SNR.

The efficiency of the interpolator when the touch pad pressure sensor is used is shown in Fig. 6. The TSI interface has a resolution of 16-bits but the examined part of the curve can be considered to have a 9-bit resolution which is further degraded by 3-bits to avoid fluctuations with monotonic errors. The values of the degraded signal are the input of the 1st interpolator. The degraded signal and the outputs of the 1st and 2nd interpolator are best fitted in the original sensor signal and the Mean Square Error (MSE) as well as the Normalized MSE (NMSE) is measured. The MSE/NMSE of the degraded signal is 42.59/0.000336 while the MSE/NMSE of the 1st and the 2nd interpolator outputs are 9.56/7.4×10⁻⁵ and 8.8/6.6×10⁻⁵ respectively. The 3rd interpolator output was not used since it was difficult to achieve a best fitting with the rest of the curves.

The proposed interpolation scheme is not targeted for image processing applications since the values of adjacent pixels do not always change smoothly. Nevertheless, the proposed interpolator may be useful for processing infrared images, night vision cameras signals, medical images like MRI, PET etc, where the color or grey level changes smoothly. The grey image of Fig. 7a was used as an input to the 1st stage of the interpolator output after degrading to 8 grey levels only (3-bit resolution) as shown in Fig. 7b. The output of the 1st stage of the interpolator is shown in Fig. 7c. The MSE/NMSE of the degraded image (Fig. 7b) compared to the original one of Fig. 7a is 725/0.243 while the MSE/NMSE of the 1st stage interpolator output is reduced to 30.4/0.0093.

Although the error in the corrected pixels is significantly reduced compared to the 3-bit grey level image, the quality is not improved since the picture appears to be dragged from left to right. An optical improvement can also be achieved by using additional rules to the interpolator that are customized for image processing applications. For example, the interpolation can only be applied to grey level values that do not differ much, if these levels appear at a comparable number of adjacent pixels. This rule was incorporated in the interpolator that used as input the image that was degraded to a 5-bit resolution. The resulting output is shown in Fig. 7d and its MSE/NMSE compared to the initial image is 9.88/0.000291. The further study of customized interpolator versions for image processing applications is out of the scope of this paper.

Table I summarizes the features of the referenced approaches. Although some authors may achieve a higher improvement either in the INL error of an ADC or its dynamic behavior, either no information is provided on their
hardware implementation or they require complicated hardware (large look up tables, multiplications/divisions, etc). The proposed interpolator instead, is based only on adders, counters and small buffers.

Fig. 7. Original image (a), the image degraded to 3-bit resolution (b), the image corrected by the interpolator (c) and the corrected by the customized interpolator output image (d).

<table>
<thead>
<tr>
<th>Reference</th>
<th>Improvement</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>[3]</td>
<td>Upgrade to 14 bits resolution from 10 bits</td>
<td>INL improvement by using 256 correction codes extracted from the 7 MSBs of the ADC. Tested in simulation level only</td>
</tr>
<tr>
<td>[5]</td>
<td>SFDR improved from 12 to 37dB, SNDR improved from 12 to 26dB</td>
<td>The 14-bit AD6645 is used, Bayesian calibration of the look up table was performed at a simulation level in MATLAB</td>
</tr>
<tr>
<td>[7]</td>
<td>34.2dB (uncorrected output), 92.9dB (with error correction)</td>
<td>Pairs of subchannel ADCs with: 16b resolution, ±1LSB INL, ±0.5LSB DNL error. The error correction circuit (excluding the additional area of the ADC pairs) occupies 0.5m² in 0.25um process.</td>
</tr>
<tr>
<td>[10]</td>
<td>4dB improvement in square error</td>
<td>Achieved with 8-bit precision. QRD-LSL requires fewer iterations than LSL interpolator</td>
</tr>
<tr>
<td>This work</td>
<td>SNR increased from 34.7dB to 42.1dB</td>
<td>72% of LEs of an Altera Cyclone II EP2C5F256 FPGA used, LZ77 compression performed</td>
</tr>
</tbody>
</table>

ACKNOWLEDGMENT

The work presented in this paper is protected by the provisional patent with appl. no. 20130100627 (Greek patent office)

REFERENCES


AUTHOR BIOGRAPHY

Nikos Petrellis is an assistant professor at the Computer Science and Engineering Dept., TEI of Thessaly, Greece from 2010. He received his diploma from Computer Engineering and Informatics Dept. and his PhD in Electrical and Computer Engineering Dept. in 1994 and 1999 respectively both from University of Patras, Greece. He has worked with several companies like Giga Hellas SA (Intel Corp.) and Atmel Hellas SA. He has been an adjunct professor for 8 years at University of Patras teaching Microcomputer and Microprocessor courses. He has published more than 60 papers in international journals, conference proceedings and book chapters. He is a member of IEEE and Technical Chamber of Greece. His research interests include Embedded System design, VLSI, mixed analog/digital systems design.