August 2005

Extrapolating Analog-to-Digital Converter

Zheng Yang
University of Pennsylvania

Jan Van der Spiegel
University of Pennsylvania, jan@seas.upenn.edu

Extrapolating Analog-to-Digital Converter

Abstract
We propose a new type of oversampled analog-to-digital converter. It uses digital extrapolators to predict the analog signal before it is converted, and a coarse quantizer to convert the prediction error. Such converters are expected to have reduced complexity in their analog circuitry, thanks to the processing in the digital domain. General linear extrapolation algorithms are derived from the spline theory, and can be easily implemented using digital filters. Simulations show that the speed-resolution trade-off is 2 bits per octave with simple linear extrapolation. Noise-shaping can be added using a matched analog preemphasis filter, in which case the converter behaves similar to a delta-sigma modulator of the same order.

Keywords
oversampling, extra-polating, ADC, spline

Extrapolating Analog-to-Digital Converter

Zheng Yang and Jan Van der Spiegel
Department of Electrical and Systems Engineering
University of Pennsylvania, Philadelphia, PA 19104

***Abstract*— We propose a new type of oversampled analog-to-digital converter. It uses digital extrapolators to predict the analog signal before it is converted, and a coarse quantizer to convert the prediction error. Such converters are expected to have reduced complexity in their analog circuitry, thanks to the processing in the digital domain. General linear extrapolation algorithms are derived from the spline theory, and can be easily implemented using digital filters. Simulations show that the speed-resolution trade-off is 2 bits per octave with simple linear extrapolation. Noise-shaping can be added using a matched analog preemphasis filter, in which case the converter behaves similar to a $\Delta\Sigma$ modulator of the same order.**

I. Introduction

For high speed, medium resolution applications, most analog-to-digital converters (ADCs) employ a pipelined structure. Such converters are limited by a number of analog circuit imperfections, such as capacitor mismatch, charge injection, finite Opamp gain, gain dispersion, and comparator offsets. These errors deteriorate the sample at each stage. If left uncorrected, the errors will carry over across the pipeline stages, result in differential nonlinearity (DNL) and integral nonlinearity (INL). The problem is more significant with newer CMOS technologies, as analog performance degradates with lower supply voltages and smaller feature sizes. Current approaches to the problem include using 1.5-bit stages that are insensitive to offsets, reducing the number of stages, and various calibration schemes [1].

In this paper we propose a new class of ADC that puts less emphasis on converting the signal itself, but more on predicting the signal before it is actually converted. Only the difference between the prediction and the input signal is converted, using a coarse quantizer. The idea is similar to the differential pulse code modulation (DPCM) techniques used in signal processing and telecommunications [2]. The ADC resembles a DPCM transmitter in Fig. 1. The predictor (or extrapolator) exploits the redundancy in the input, which is removed by a difference operation, so that only the information that can't be predicted is quantized. In general, redundancy can be created by oversampling the input, implying an inherent trade-off between speed and resolution in this scheme.

The structure can be considered as a two-stage pipelined ADC with a prediction stage and an error-correcting stage. The first stage contributes to most of the ADC's resolution since the predictor is designed to closely track the input signal. The second stage only needs to convert the remaining prediction error, which would be small in amplitude, thus requring less quantizer levels. By having only two stages, the accumulative effect of errors in a multiple stage pipelined ADC is reduced.

II. Extrapolation Schemes

A. Predictor Modes

The predictor can be implemented either in the digital or analog domain. A digital predictor (Fig. 2) is similar to the one in a DPCM transmitter, which is simply a digital filter that takes input from previously converted samples. It can even be implemented separately from the analog circuitry, for example on a DSP. Besides being error-free, the predictor also has the advantage that its states are completely known and ready to output as digital signals. However, a digital-to-analog converter (DAC) is needed to convert the prediction back to analog values. This DAC needs to have at least the same resolution as the prediction stage, which could limit the speed of the ADC.

Prediction can also be done directly on the analog input signal, in a feed-forward configuration in Fig. 3. By tracking the signal before a front end sample-and-hold (S/H), the predictor has access to information that is usually ignored by conventional ADCs. One example is the continuous-time derivatives of the input, which are useful in many extrapolation algorithms. Such quantities can only be estimated from discrete-time samples, but they are available to an analog predictor with the use of differentiators. As a disadvantage of the analog domain implementation, the predictor states need to be quantized in order to generate an output. They are not explicitly known, or may not be reconstructible, from the quantized values.

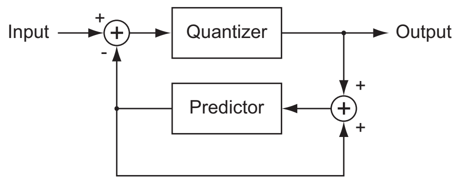

Fig. 1. Block diagram of a DPCM transmitter.

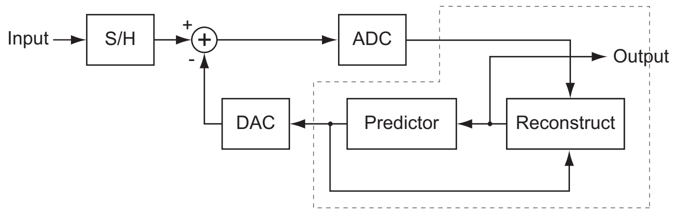

Fig. 2. Predictor in the digital domain. Dashed line indicates the digital components that can be implemented outside the converter.

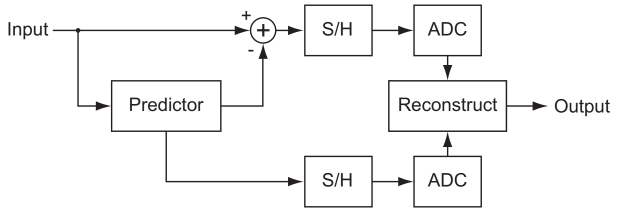

Fig. 3. Predictor in the analog domain.

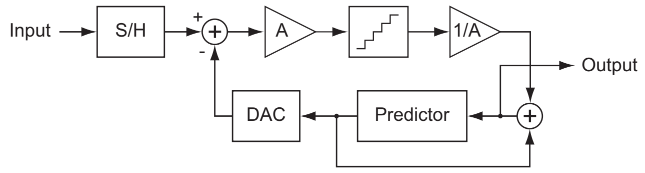

Fig. 4. Block diagram of the proposed extrapolation scheme.

B. Signal Representing Spaces

Both analog and digital predictors may be designed to exploit the redundancy of the input signal in the most suitable representing space. In general, oversampling a signal creates redundancy which shows up as a "smoothness" in the time-domain waveform. Thus the simplest time-domain predictor can be one that uses the current signal value as the prediction. More complex linear predictors have been studied in signal processing and adaptive filter theories.

Similarly, if the signal is known to have certain characteristics in a specifc signal representing space, the predictor should be designed to work in that space. As an example, for an amplitude-modulated (AM) signal, a frequency-domain predictor may have a phase-locked loop that tracks the carrier frequency, and an amplitude variable which it must predict. The simplest predictor in this case would be one that uses the current amplitude value as the prediction. Here the redundancy lies in both frequency (constant over time) and amplitude (varies slowly over time) components of the input.

Prediction is not limited to time-domain, even for signals that do not conform to a known model. Besides the $sinc$-based approach as with Shannon's sampling theory, other methods can also be used to represent the prediction. Candidates may include wavelets or splines [3].

C. Proposed Scheme

1) Predictor Configuration: Our goal is to reduce analog circuit imperfections by doing the processing in the digital domain as much as possible. Therefore, the proposed scheme is based on a digital predictor shown in Fig. 4. The extrapolation algorithm used by the predictor is derived from the spline theory, which is commonly used in data analysis [4]. Before sampling the input data, the predictor constructs a *piecewise cubic spline* from previously converted samples. It then extends the last segment of the spline one step further, and output that value as the prediction. This output is converted to an analog value by the DAC. Right after sampling the input signal, the difference between the sample and the prediction is quantized using a coarse quantizer. The result is sent back to the predictor, which uses it to correct the previous prediction, and subsequently update the spline to predict the next sample. Note that a linear amplifier is needed for the small prediction error in order to match the range of the quantizer. Compared to Fig. 2, the reconstruction step is replaced by a constant multiplication and an addition in the digital domain.

In the algorithm, a piecewise cubic spline for the n data points $(t_1, x_1), (t_2, x_2)...(t_n, x_n)$ is a smooth curve that goes through all the points and is twice continuously differentiable. It consists of $n-1$ segments connected end-to-end at the $n-2$ *interior knots* on $t_2, t_3...t_{n-1}$. Each segment is a cubic polynomial function of the form

$$x(t) = s_{i,0} + s_{i,1}(t-t_i) + s_{i,2}(t-t_i)^2 + s_{i,3}(t-t_i)^3 \quad (1)$$

for the segment defined between t_i and t_{i+1}. $s_{i,0}$ through $s_{i,3}$ are coefficients that must be determined by the predictor. Since samples are uniformly spaced, for simplicity we let $t_i = i$; i.e. the current time is $t = n$, while prediction is to be made for $t = n+1$.

Consider the last segment ($i = n-1$) which will be extended to calculate the prediction. It must go through $(n-1, x_{n-1})$ and (n, x_n), reducing (1) to:

$$x_{n-1} = s_{n-1,0} \quad (2)$$

and

$$x_n = s_{n-1,0} + s_{n-1,1} + s_{n-1,2} + s_{n-1,3} \quad (3)$$

Two extra equations are needed to solve for the 4 unknowns. They must come from the conditions on the first or second derivatives at the two end points. Since this information is not directly available to the predictor (which might not be the case for an analog predictor), the derivatives need to be set arbitrarily, or estimated from the previous segments. For a *natural spline*, the second derivatives at the two ends are set to zero. This gives the following two equations:

$$2s_{n-1,2} = 0 \quad (4)$$

$$2s_{n-1,2} + 6s_{n-1,3} = 0 \quad (5)$$

Solving (2) through (5), we get $s_{n-1,0} = x_{n-1}$, $s_{n-1,1} = x_n - x_{n-1}$, and $s_{n-1,2} = s_{n-1,3} = 0$. Therefore the polynomial (1) becomes:

$$x = x_{n-1} + (x_n - x_{n-1})(t - (n-1)) \quad (6)$$

and the prediction at $t = n+1$ is

$$x_{n+1} = 2x_n - x_{n-1} \quad (7)$$

which has reduced to a simple linear extrapolation of the last two data points. It is not surprising, since we have set the second derivatives to zero without making use of the smoothness property between spline segments.

To estimate the derivatives from adjacent segments, an *extrapolated spline* can be used. It assumes that the two ending segments are part of, or extrapolated from, their neighboring segments; otherwise known as the *not-a-knot (NAK)* condition.

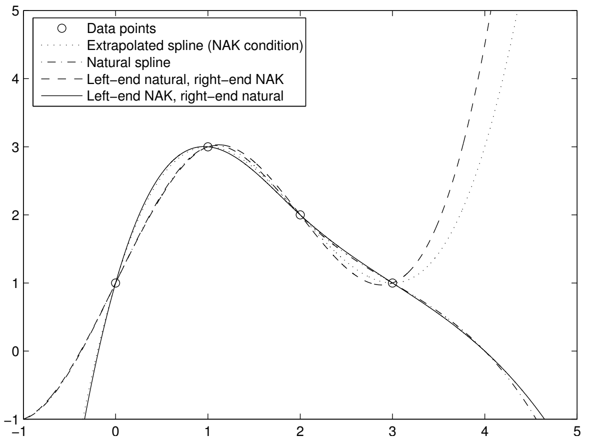

Fig. 5. Piecewise cubic spline functions constructed from 4 data points.

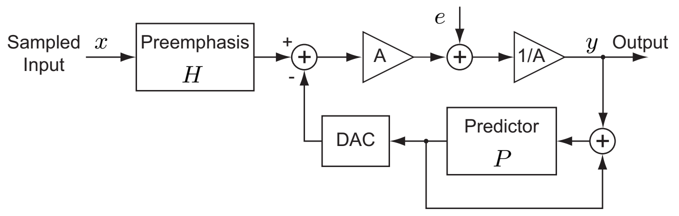

Fig. 6. Block diagram of the proposed scheme with noise shaping.

The shortest spline of this kind involves 4 data points that belong to the same segment. Therefore two more equations are created using the values at the two extra points. This gives the following prediction on x_{n+1}:

$$x_{n+1} = -x_{n-3} + 4x_{n-2} - 6x_{n-1} + 4x_n \tag{8}$$

which is the same as a polynomial extrapolation on the 4 data points.

Alternatively, one may use the natural condition on the left-end and the NAK condition on the right-end. The shortest spline of this kind involves 3 data points and gives the following prediction:

$$x_{n+1} = 2x_{n-2} - 5x_{n-1} + 4x_n \tag{9}$$

which is nontrivial and behaves according to intuition, when testing with data points such as $(1, 1, 1)$ or $(1, 2, 3)$.

Fig. 5 shows the different spline functions constructed from the same 4 data points. It can be seen that the splines with NAK conditions behave less stable than those with natural conditions outside the range of data points. This problem is expected with polynomial extrapolation. On the other hand, the splines with natural endpoint condition give the same value of prediction at $t = 4$. For ease of implementation, we chose the natural spline eq. (7) among the others, which can be done in merely 2 simple operations.

2) Noise Shaping: As with DPCM coders, noise shaping can be added to the ADC with the use of an analog preemphasis filter [5]. Fig. 6 shows this configuration, where the quantizer has been replaced by an addition with noise e. The z-domain expression for the output y can be shown to be

$$Y(z) = H(z)(1 - P(z))X(z) + \frac{1 - P(z)}{A}E(z) \tag{10}$$

The signal transfer function (STF) for the input x is equal to unity when H and P are matched, such that

$$H(z) = \frac{1}{1 - P(z)} \tag{11}$$

The noise transfer function (NTF) for the quantization noise e consists of a constant factor $1/A$ and a shaping function $1-P(z)$. If the prediction error is small, A can be made large, implying a large uniform noise suppression. The predictor function derived from natural spline (eq. 7) is $P_{natural}(z) = 2z^{-1} - z^{-2}$, which gives

$$1 - P_{natural}(z) = (1 - z^{-1})^2 \tag{12}$$

Similarly, from eq. (8), we get

$$1 - P_{extrap}(z) = (1 - z^{-1})^4 \tag{13}$$

Interestingly, these shaping functions are identical to the loop filters implemented in a delta-sigma ($\Delta\Sigma$) modulator of order 2 and 4 respectively. Although originally designed from a extrapolation perspective, the spline-based algorithms also perform ideal high-pass noise shaping. In the case that H and P are ideally matched according to eq. (11), the final converter output can be obtained by a digital decimation filter connected to the y signal. The system could perform better than its $\Delta\Sigma$ counterpart because of the additional noise suppression factor $1/A$. However, in practice these matched filters are difficult to implement, because the Hs need to have poles at $z = 1$, which make them unstable.

We have ignored the DAC noise in the above derivation. In fact, it can be mapped to the port of the quantization noise, and would still be shaped by $1 - P(z)$, albeit without the $1/A$ factor. Another source of nonlinearity may come from the amplifier itself. This effect can be minimized using a 1-bit quantizer, which is possible as was demonstrated in simulations.

III. Simulation Results

We performed Matlab simulations on the proposed ADC scheme. Without noise shaping, we used a band-limited white Gaussian noise as the analog input signal. The noise source outputs at the sampling frequency, but is low-pass filtered to the desired signal bandwidth. The input is compared with the output of the ADC to obtain the signal-to-noise ratio (SNR).

In all our simulations, natural spline prediction (eq. 7) resulted in the smallest amplitude error among the other splines. The extrapolated spline (eq. 9) has the largest prediction error, which must be correct using a smaller amplification factor A, or equivalently by using more bits in the quantizer. Otherwise, if the prediction error exceeds the range of the error-correcting stage, the predictor would lose track of the input, and the loop could become unstable. In most cases, the extrapolated spline needs 2 more quantizer bits in order to achieve the same overall resolution of the natural spline.

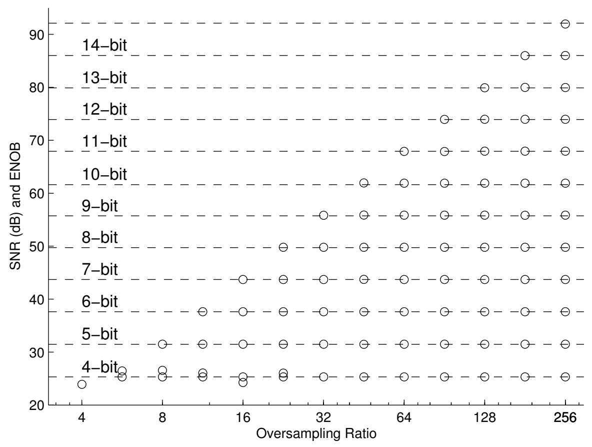

Fig. 7. SNR and ENOB of the proposed ADC with natural spline predictor and 3-bit quantizer.

With fixed sampling rate and quantizer bits, the optimal value of A is found by increasing it until the system becomes unstable. Each octave increase of A represents a 1-bit increase in overall resolution, since the second stage would be twice as accurate. This resolution is taken as the effective number of bits (ENOB) of the ADC. For a natural spline predictor and a 3-bit quantizer, the results is summarized in Fig. 7, where the data point on the top of each column represents the optimal operation condition.

It can be seen that the ENOB increases by 2 bits for each doubling of sampling rate. Therefore the speed-resolution trade-off is 2 bits per octave. We have observed that this trade-off is independent of the number of quantizer bits. In other words, a 1-bit quantizer can be used, whose ENOB simply shifts down by 2 bits in Fig. 7.

The proposed ADC with noise shaping is simulated with a sinusoidal input. Natural spline prediction is again used, while a sign function replaces the amplifier and 1-bit quantizer. The preemphasis filter function is $H(z) = (1 - 0.999z^{-1})^{-2}$, where the poles are moved to keep the filter stable. The output spectrum is shown in Fig. 8, which agrees well with the output of an ideal second-order $\Delta\Sigma$ modulator in Fig. 9. The SNR in both simulations is around 70dB assuming an OSR of 64. Since no amplification is needed for a 1-bit quantizer, we do not see the effect of additional noise suppression.

IV. Discussion

Our proposed ADC scheme requires only minimum analog circuitry, and demonstrated a gain in resolution using very simple digital extrapolation algorithms. With noise-shaping, it has the potential to outperform $\Delta\Sigma$ converters of the same order. However, a number of factors could limit its performance. First, the DAC puts a cap on the accuracy of the prediction. It also operates at the full sampling rate, which may limit the OSR. Also, the current algorithm requires that input samples must not change significantly over time. Therefore a slow start-up from a known initial state is needed. Moreover, the analog preemphasis filter for noise-shaping is difficult to

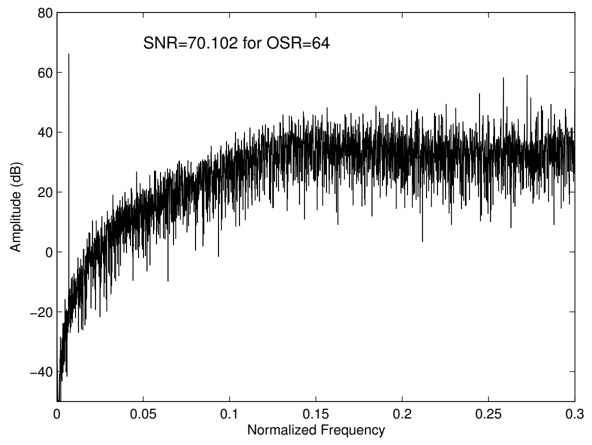

Fig. 8. Output spectrum of the proposed ADC with noise shaping.

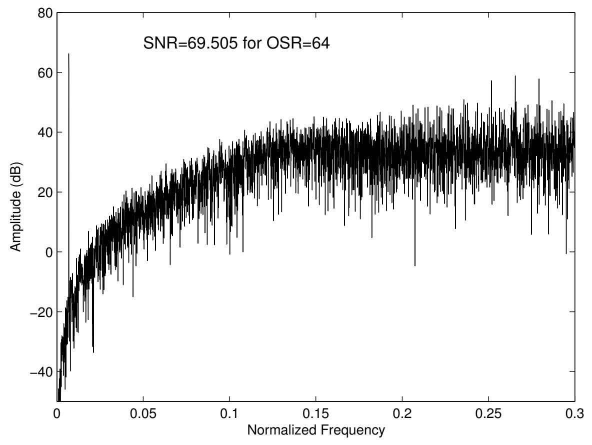

Fig. 9. Output spectrum of a second-order $\Delta\Sigma$ modulator.

implement. In practice a compromise must be made between the shape of NTF and the level of mismatch.

Extrapolating ADCs provide a new direction where various signal processing techniques can be incorporated. It may prove to be a viable approach for signals that can be modeled prior to arrival. It also opens up the possibility to new sampling methods and calibration schemes.

Acknowledgment

The authors would like to thank Dr Viktor Gruev for his insightful suggestions and discussions.

References

[1] S. Sonkusale, J. van der Spiegel, and K. Nagaraj, "True background calibration technique for pipelined ADC," *Electronics Letters*, vol. 36, no. 9, pp. 786–788, 2000.
[2] J. G. Proakis, *Digital Communications*. Boston: McGraw-Hill, 2001.
[3] F. Muller, P. Brigger, K. Illgner, and M. Unser, "Multiresolution approximation using shifted splines," *IEEE Trans. Signal Processing*, vol. 46, no. 9, pp. 2555–2558, 1998.
[4] J. H. Mathews and K. D. Fink, *Numerical methods using MATLAB*. Upper Saddle River, NJ: Pearson, 2003.
[5] S. Tewksbury and R. Hallock, "Oversampled, linear predictive and noise-shaping coders of order $N > 1$," *IEEE Trans. Circuits Syst.*, vol. 25, no. 7, pp. 436–447, 1978.