# INSTANTANEOUSLY COMPANDING DIGITAL SIGNAL PROCESSORS

*Ari Klein and Yannis Tsividis*

Department of Electrical Engineering, Columbia University, New York, NY, USA

## ABSTRACT

We extend the technique of instantaneous companding (compressing and expanding) to digital signal processors (DSPs). Companding has been used for many years in non-dynamical channels, but the dynamical nature of DSPs causes output distortion in standard implementations of companding, where a compressor and expander are used at the input and output, respectively, without additionally modifying the DSP. In the proposed technique, we specify a method for combining input compression, output expansion, and application of nonlinear functions internal to the DSP, all in a manner transparent to the input-output characteristics of the DSP and its A/D interfaces, thus eliminating distortion in the final analog output. As a result, all the signals involved span most of the available bits, resulting in significant improvement in quantization errors and signal-to-noise-plus-distortion ratios over a large input range. The theory is supported by simulation and subjective listening tests.

***Index Terms***— Companding, digital signal processors, discrete time systems, nonlinear systems

## 1. INTRODUCTION

For many years, companding (compressing/expanding) techniques have been successfully implemented in such applications as transmission and sound recording [1, 2]. By compressing the dynamic range of input signals before transmission or recording, companding yields high signal-to-noise-plus-distortion ratios (SNDR) over a large input range, as the transmitted or recorded signal level remains well above the noise in the channel or the storage medium even at low input levels. The expander at the output restores the original dynamic range. Recently, companding has been extended to dynamical systems [3] of the type shown in Fig. 1(a), composed of analog-to-digital converters (ADCs), digital signal processors (DSPs), and digital-to-analog converters (DACs); the input to the ADC is, for simplicity, assumed to be an already sampled analog value. The technique developed in [3] uses the envelopes of signals in the prototype system of Fig. 1(a) to control the state variables of a companding system; it was demonstrated that in this way, syllabic companding may be achieved in a DSP without disturbing the final output. Unfortunately, that technique is somewhat impractical, because it essentially requires two implementations of the system: the prototype system which produces the envelopes, and the companding system which uses the envelopes to produce the desired output with low quantization distortion. Also, the technique requires more than one ADC and DAC, and significant computation to produce the envelopes and integrate them into the companding system. In this paper, we explore an alternative technique, in which companding is achieved through the use of nonlinear time-invariant compressing and expanding functions. These functions are used to create a modified, internally nonlinear DSP, which, when combined with the corresponding compressing ADC and expanding DAC, yield a system whose input-output behavior is linear and time-invariant (LTI), and, in the absence of quantization error, identical to that of the original system of Fig. 1(a); the quantization error at the output of the modified system is significantly lower than that of the original. The proposed system is thus externally linear and internally nonlinear (ELIN); analog versions of ELIN systems have been discussed in [4] and [5]. In essence, the proposed technique is an extension of *instantaneous* companding to DSPs, whereas the technique in [3] is an extension of *syllabic* companding. It will be shown that the proposed technique avoids many of the problems that plague [3], while still providing a large (for a given number of bits) SNDR over a large input range. The ADC and DSP may thus be implemented with relatively few bits, yet still have relatively low quantization distortion, which is desirable for applications where large amounts of computation are required, such as multimedia.

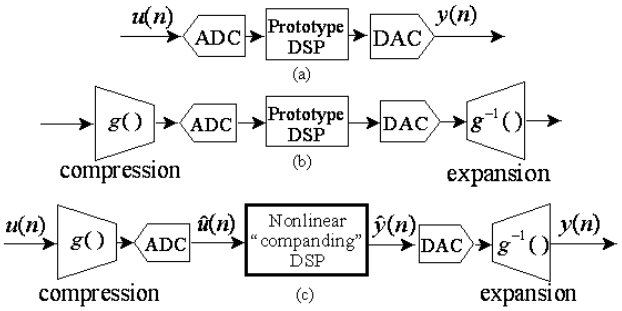


**Fig. 1**. (a) A system composed of an LTI DSP and its A/D interfaces. (b) Instantaneous companding added to the system in (a) without internal correction in the DSP, which will cause output distortion. (c) Properly companding system using internally nonlinear DSP.

## 2. INSTANTANEOUS COMPANDING APPLIED TO DSPS

To achieve a large SNDR over a large input range, the signal at the ADC must span most of the available bits, even at low input levels. It is therefore necessary to compress the input dynamic range before the ADC, which will be done here using a nonlinear compressing function $g(v)$, assumed to be odd and invertible. Consequently, the output of the DAC will also be compressed, so to restore the desired output, it is necessary to re-expand the dynamic range after the DAC. In this paper, the expanding function considered will be the inverse of the compressing function, and will be denoted by $g^{-1}(v)$. When there is a dynamical system, such as a DSP, between the ADC and the DAC, simply applying $g(v)$ at the input and $g^{-1}(v)$ at the output, as shown in Fig. 1(b) , will in general cause unacceptable nonlinear

This work was supported in part by National Science Foundation Grant CCR-02-09109.

distortion in the output. As an example, suppose that the DSP of Fig. 1(a) has input-output behavior given by $y(n) = \alpha u(n) + \beta u(n-k)$, and that the desired compression law is $g(v) = \sqrt[3]{(v)}$, meaning that $g^{-1}(v) = v^3$. Applying the technique of Fig. 1(b) gives $y(n) = (\alpha \sqrt[3]{u(n)} + \beta \sqrt[3]{u(n-k)})^3$, which can be rewritten as $y(n) = \alpha^3 u(n) + \beta^3 u(n-k) + 3[\alpha^2 \beta u(n)^{\frac{2}{3}} u(n-k)^{\frac{1}{3}} + \alpha \beta^2 u(n)^{\frac{1}{3}} u(n-k)^{\frac{2}{3}}]$. The first two terms resemble the desired output, but the gains are $\alpha^3$ and $\beta^3$ instead of $\alpha$ and $\beta$. The rest of the terms represent nonlinear distortion. In general, to preserve the original linear input-output behavior, the DSP must be made internally nonlinear, as is the case with analog systems [4, 5]. An exact and systematic method for properly modifying the DSP will be derived below.

To derive a mathematical formulation for introducing the necessary internal changes to the DSP, we need access to the internal states of the DSP; the input-output description of the DSP's behavior is therefore insufficient, and we will instead make use of the state-space description. Consider an LTI discrete-time $m^{th}$ order system with single input $u(n)$, single output $y(n)$, and state vector $x(n) = (x_i(n))$. The state equations of this system are of the form:

$$\begin{array}{rcl} x(n+1) & = & Ax(n) + Bu(n) \\ y(n) & = & Cx(n) + Du(n) \end{array} \quad (1)$$

where $A = (a_{ij})$ is a $m \times m$ matrix, $B = (b_{ij})$ is a $m \times 1$ vector, $C = (c_{ij})$ is a $1 \times m$ vector, and $D = (d_{ij})$ is a scalar. Expanding (1) gives equations (for $1 \le i \le m$) of the form:

$$\begin{array}{rcl} x_i(n+1) & = & \sum_{j=1}^{m} a_{ij} x_j(n) + b_i u(n) \\ y(n) & = & \sum_{j=1}^{m} c_j x_j(n) + du(n) \end{array} \quad (2)$$

This system will be referred to as the "prototype system", and corresponds to the system in Fig. 1(a), assuming that the ADC and DAC are of infinite resolution.

To achieve low quantization distortion, and thus large SNDR, we require nonlinear instantaneous compression of any digital signal which is represented by only a few bits. To represent the input, output, and states with only $N$ bits, where $N$ is assumed to be a small integer (such as 8), we will create a modified DSP, which we refer to as the "companding DSP", with input $\hat{u}(n) = g(u(n))$, output $\hat{y}(n) = g(y(n))$, and state vector $\hat{x}(n) = (\hat{x}_i(n)) = (g(x_i(n)))$. Substituting for $x$, $u$, and $y$ in (2), after some algebra, gives:

$$\begin{array}{rcl} \hat{x}_i(n+1) & = & g(\sum_{j=1}^{m} a_{ij} g^{-1}(\hat{x}_j(n)) + b_i g^{-1}(\hat{u}(n))) \\ \hat{y}(n) & = & g(\sum_{j=1}^{m} c_j g^{-1}(\hat{x}_j(n)) + d g^{-1}(\hat{u}(n))) \end{array} \quad (3)$$

This companding DSP has three important features. First, its input, output and states are all compressed, so representing them with only $N$ bits will not result in significant quantization errors. Second, its input is $g(u(n))$, so it may be gotten by simply applying $g()$ to the system input $u(n)$, which should be done before the ADC, allowing the signal at the ADC to take advantage of all the available bits. Third, the output of the companding DSP is $g(y(n))$, so the system output $y(n)$ may be recovered exactly, with no distortion, by simply applying the expanding operation $g^{-1}()$ to the companding DSP output $\hat{y}(n)$, which should be done after the DAC, allowing the signal at the DAC to take advantage of all the available bits. The complete, properly companding system is thus as shown in Fig. 1(c), where the modified DSP shown in the figure is given by the internally nonlinear, companding DSP specified by (3).

## 3. IMPLEMENTATION

### 3.1. Implementation of Companding Functions

For companding in non-dynamical systems, it has been shown [1] that to obtain an SNDR which is independent of input power, and thus results in very low noise at low and moderate input levels, the compression characteristic should be approximately logarithmic. One standard compression characteristic is the "$\mu$ law," given by [1]:

$$\begin{array}{l} \gamma(v) = \frac{\log(1+\mu v)}{\log(1+\mu)}, \quad v \ge 0 \\ \gamma(v) = -\gamma(-v), \quad v < 0 \end{array} \quad (4)$$

where $\mu$ corresponds to the amount of compression. Roughly speaking, increasing $\mu$ improves the SNDR for small inputs (thus increasing dynamic range), at the expense of the SNDR for large inputs. For the rest of this report, all logarithms are assumed to be base 2, and will thus be written as lg. Also, in equation (4), $v$, and therefore also $\gamma(v)$, are assumed normalized to be between $-1$ and 1.

We will begin by considering the implementation of (3) in an $N$-bit fixed point DSP. To allow for $N$-bit multiplication, an $N$-bit DSP typically supports $2N$-bit addition. It is therefore assumed that $2N$-bit numbers may be stored temporarily and added (or subtracted), but they may not be stored in registers (they may only be used in the same timestep during which they are generated) and they may not be multiplied (or shifted). Since $2N$ bit numbers may be temporarily used, it will be assumed that inside the DSP, the compressing operator $g()$ takes a $2N$-bit number and returns an $N$-bit number, while the expanding operator $g^{-1}()$ takes an $N$ bit number and returns a $2N$ bit number. Modifying (4) accordingly gives the DSP compression characteristic which we will use for the remainder of the paper:

$$\begin{array}{c} g(v) = \frac{2^{N-1} \lg(1+\mu \frac{v}{2^{2N-1}})}{\lg(1+\mu)}, \quad v \ge 0 \\ g(v) = -g(-v), \quad v < 0 \end{array} \quad (5)$$

Inverting (5) gives:

$$\begin{array}{c} g^{-1}(v) = \frac{2^{2N-1}}{\mu}[(1+\mu)^{\frac{v}{2^{N-1}}} - 1], \quad v \ge 0 \\ g^{-1}(v) = -g^{-1}(-v), \quad v < 0 \end{array} \quad (6)$$

Referring back to Fig. 1(c), the analog versions of $g()$ and $g^{-1}()$ may either be implemented with analog circuitry at the input of the ADC and output of the DAC, as shown in the figure, or they may be "absorbed" into the ADC and DAC by using a nonuniform ADC and DAC. If it is assumed that the original $u(n)$ and $y(n)$ are normalized between $-1$ and 1, then $u(n)$ should be multiplied by $2^{2N-1}$ before applying $g()$, and $y(n)$ should be multiplied by $2^{-(2N-1)}$ after applying $g^{-1}()$. Clearly, these multiplications cancel, and thus don't change the system's input-output behavior. The multiplications should be absorbed into the analog circuitry, giving $g(2^{2N-1}v) = 2^{N-1}\gamma(v)$ at the input, and $2^{-(2N-1)}g^{-1}(v)$ at the output.

In the DSP, the nonlinear functions can be implemented using lookup tables generated before runtime, for example on a computer. The compressive nature of $g()$ ensures that $\hat{x}(n)$, $\hat{u}(n)$, and $\hat{y}(n)$ may be stored in $N$-bit registers without causing too much quantization error. However, $g^{-1}(\hat{x}_j(n))$, $g^{-1}(\hat{u}(n))$, and $g^{-1}(\hat{y}(n))$ are simply $x_j(n)$, $u(n)$ and $y(n)$, respectively, so storing them in $N$-bit registers would lead to significant quantization distortion; $2N$ bits should therefore be used to temporarily store these numbers. Unfortunately, it is seen in (3) that $g^{-1}(\hat{x}_j(n))$ and $g^{-1}(\hat{u}(n))$ in general need to be multiplied by the coefficients $a_{ij}$, $b_i$, $c_j$, and $d$. To avoid this, these gains are "absorbed" into the lookup tables for

$g^{-1}(v)$. Specifically, if the desired signal is $\alpha g^{-1}(v)$, the corresponding lookup table will actually output a $2N$-bit number which represents $\alpha g^{-1}(v)$, instead of simply giving $g^{-1}(v)$, which would then need to be multiplied by $\alpha$.

Since the $g()$ function takes a $2N$-bit integer and returns an $N$-bit integer, there are $2^{2N}$ possible inputs, but only $2^N$ possible outputs, and therefore $2^N$ distinct table entries. The table is constructed by choosing $2^N$ of the $2N$-bit integers such that each integer $v$ gives a distinct $N$-bit $g(v)$. Each entry in the table has a $2N$-bit number $v$, and a corresponding $N$-bit $g(v)$; given a particular $2N$-bit input argument $a$, the table finds the entry whose $2N$-bit value $v$ is closest to $a$, and then returns the corresponding $N$-bit output value $g(v)$. Since every application of $g()$ can use the same lookup table, only one instance of it needs to be stored in memory. To obtain a lookup table for $g^{-1}()$, we flip the input and output columns of the $g()$ lookup table. To obtain the $\alpha g^{-1}()$ lookup table, we multiply each entry of the output column by $\alpha$, and we round the result to the nearest $2N$-bit number. As a result, all the coefficient multiplications are done during the process of generating the lookup tables, which can be done in any desired precision, such as double-precision floating point. The resulting lookup table entries are then quantized to a $2N$-bit fixed point representation and transferred to the DSP.

In some cases in a companding DSP, limit cycles can be observed, similar to the ones in conventional DSPs [6]. Indeed, for the companding system we describe in Section 3.2, we observed low-amplitude limit cycles, but their effect was inaudible in our audio tests. We plan to try classical techniques to eliminate these limit cycles, but we have not done so at this point.

### 3.2. Example

We will consider the implementation of instantaneous companding on a reverberator similar to the one which was used as a case study in [3], which allows us to compare the relative ease of implementation for the two techniques. The reverberator prototype system consists of a cascade of two stages, each of which has state equations given by:

$$\begin{array}{rcl} x_1(n+1) & = & -0.8x_k(n) + 0.2u(n) \\ x_i(n+1) & = & x_{i-1}(n), \quad 2{\leq}i{\leq}k \\ y(n) & = & 1.8x_k(n) + 0.8u(n) \end{array} \tag{7}$$

where $k = 1999$ for the first stage and $k = 4411$ for the second, with an assumed sampling rate of 44.1 kHz.

Applying (3) to the state equations of either stage gives:

$$\begin{array}{c} \hat{x}_1(n+1) = g[-0.8g^{-1}(\hat{x}_k(n)) + 0.2g^{-1}(\hat{u}(n))] \\ \hat{x}_i(n+1) = \hat{x}_{i-1}(n), \quad 2{\leq}i{\leq}k \\ \hat{y}(n) = g[1.8g^{-1}(\hat{x}_k(n)) + 0.8g^{-1}(\hat{u}(n))] \end{array} \tag{8}$$

Note the simplicity of the second equation of (8), which comes from a $k$-delay block; for $k-1$ of the states in any $k$-delay block, no applications of $g()$ are necessary!

The input to the first stage is obtained by applying $g()$ before the ADC, thus producing $\hat{u}(n)$, and $g^{-1}()$ is applied after the DAC (after the second stage) to retrieve $y(n)$ from $\hat{y}(n)$. The system is thus as shown in Fig. 1(c), where the modified DSP is given by two cascaded stages, each given by (8), with $k = 1999$ for the first stage and $k = 4411$ for the second and with $g()$ and $g^{-1}()$ given by (5) and (6). Note that in contrast to the technique in [3], no copy of the prototype system is required, no envelope detection is necessary, and only one ADC and one DAC are required.

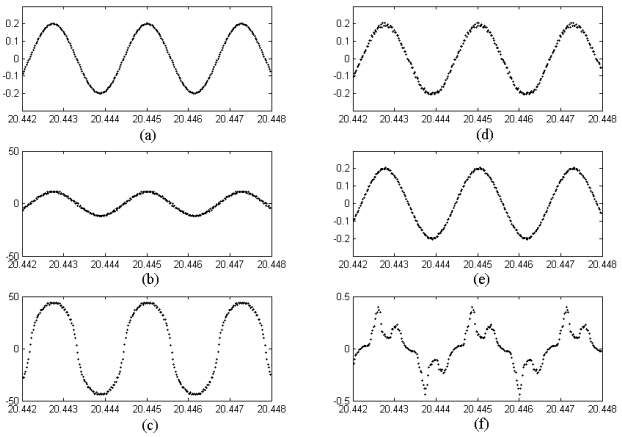


**Fig. 2**. Simultaneous waveforms for a sinusoidal input signal, all versus time in seconds: (a) Input. (b) State in system of Fig. 1(a). (c) Corresponding state in companded system of Fig. 1(c). (d) Output of system in Fig. 1(a). (e) Output of companded system in Fig. 1(c). (f) Output of system in Fig. 1(b).

### 3.3. Simulation Results

Matlab/Simulink was used to implement and simulate the prototype and companding reverberator systems described above. The internal signals of the non-companding system were optimally scaled for maximum amplitude inputs. Setting $\mu = 255$, we used 8-bit ADCs and DACs, implemented the DSPs with 8-bit fixed-point arithmetic, and ran all three systems in Fig. 1 with the same sinusoidal input. The systems were observed in steady state. Two cycles of the input are shown in Fig. 2(a). Fig. 2(b) shows a state in the prototype DSP of Fig. 1(a). The corresponding "hat" state in the internally nonlinear DSP of Fig. 1(c), shown in Fig. 2(c), is clearly a very different waveform; both the signal's level and shape have been modified. Despite this, the outputs of the systems of Fig. 1(a) and Fig. 1(c), shown in Fig. 2(d) and Fig. 2(e), are very similar; the difference between the two is on the order of quantization errors, as expected. In contrast, the output of the system of Fig. 1(b), shown in Fig. 2(f), is extremely distorted, looking nothing like the desired output. Thus, our simulation results have confirmed that the companding technique shown in Fig. 1(b) is inadequate, and that the significant output distortion caused by such incomplete companding may be eliminated by using equation (3) to modify the DSP, and then using the resulting internally nonlinear DSP as shown in Fig. 1(c). We also see that for the input of moderate level shown in Fig. 2(a), the state of the prototype system, shown in Fig. 2(b), does not span most of the available bits, and thus is significantly affected by quantization and by the round-off errors resulting from fixed-point arithmetic operations. In contrast, the corresponding state in the companding system, shown in Fig. 2(c), spans most of the available bits, and thus has quantization and round-off errors which are significantly reduced relative to the signal.

The steady-state outputs for the cases of no companding, and companding with $\mu = 63$ and $\mu = 255$ were observed for sinusoidal inputs of various amplitudes; the SNDRs are compared in Fig. 3. The SNDR for the non-companding system varies approximately in proportion to the input signal. In contrast, for the companding systems, the SNDR is relatively constant, and at a relatively high value, over a large range of input levels. Roughly, increasing $\mu$ increases

the dynamic range over which the SNDR is roughly constant, but at the expense of reducing the peak SNDR. This is as expected since the number of quantization levels is fixed at $2^N$, and essentially, larger $\mu$ means that more of these levels are used for small signals, implying less levels available for large signals. Thus, larger $\mu$ does not always imply "better" output quality; a reasonable balance must be found between having high dynamic range, for which larger $\mu$ is preferred, and having high peak SNDR, for which lower $\mu$ is preferred.

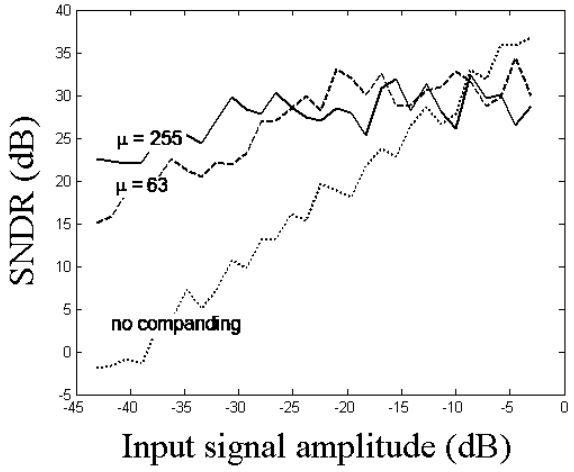


**Fig. 3**. SNDR for 8-bit implementation, for several values of $\mu$.

We ran all three systems in Fig. 1 on audio inputs, and posted a few of the resulting audio clips on a website[7]. To test the principles, the three systems were first simulated using double-precision floating-point arithmetic; the ADCs and DACs were assumed to have infinite precision, and Matlab functions were used to implement all the $g()$ and $g^{-1}()$ functions, even in the modified DSP, instead of lookup tables. The output of the system in Fig. 1(c) sounded identical to that of the prototype in Fig. 1(a), as expected, whereas the output of the system in Fig. 1(b) sounded grossly distorted.

To judge the technique's performance on the audio clips for the case of limited resolution, we ran the systems of Fig. 1(a) and Fig. 1(c) using 8-bit ADCs and DACs; the DSPs were implemented with 8-bit fixed-point arithmetic. Listening tests were performed with speech and music inputs, and it was found empirically that for the 8-bit case, $\mu = 63$ results in the (subjectively) optimal output quality. For both systems, quantization noise resulted in an audible "hiss" in the output. However, whereas the hiss of the prototype's output was relatively constant, the hiss in the companding system's output became much less audible as the music got softer, and was often completely inaudible during very soft music passages. These results are similar to those in [3], even though the instantaneous companding technique proposed in this paper is much simpler to implement than the syllabic companding technique in [3].

## 4. CONCLUSIONS AND DISCUSSION

In this paper, instantaneous companding has been extended to DSPs through the application of nonlinear functions to the signals of a DSP. It has been shown that combining input compression, output expansion, and internal nonlinear modification of a DSP can be done in a manner transparent to the original input-output behavior of the DSP and its A/D interfaces. The resulting companding DSP is internally nonlinear; despite this, assuming no quantization, the input-output behavior of the entire companding system, consisting of the companding DSP, the input compressor, and the output expander, remains identical to that of the original LTI system. In the presence of quantization, it was shown that the companding system has large SNDR over a large input range, in contrast to the original non-companded system, which suffers from significantly reduced SNDR at low and moderate input levels. By significantly reducing the quantization distortion at low and moderate input levels, the proposed technique can reduce the number of bits required in achieving a given minimum SNDR over a large input range, which is attractive for applications such as multimedia.

The proposed instantaneous companding technique has many advantages over the syllabic companding technique in [3]. In the proposed technique, the uncompressed signals may be obtained by simply applying $g^{-1}()$ to the corresponding compressed signals, so no copy of the prototype system is necessary. Additionally, no envelope detection is required and no control inputs are needed in the DSP. Also, the input and output of the DSP, $\hat{u}(n)$ and $\hat{y}(n)$, are both compressed, so they will both be $N$-bit numbers, and thus only one $N$-bit ADC and one $N$-bit DAC are required. Thus, the presented instantaneous companding technique yields benefits similar to those in [3], without requiring a copy of the prototype, envelope detection, extra control inputs, or extra ADCs or DACs.

The main disadvantage of the proposed technique is the necessity of lookup tables for implementing the nonlinear functions. Fortunately, the maximum number of rows in each lookup table is $2^N$, which is relatively small, as the necessity of companding implies small $N$. It may be possible to get good results with even smaller lookup tables, either by accepting some additional quantization or by using interpolation. Adjusting the lookup table size is likely to be an effective way to trade off memory requirements, speed and precision. If interpolation is used, the nonlinear functions are actually piecewise linear; the authors are currently looking into such implementations, as well as hardware realizations.

## 5. REFERENCES


[1] Member of the Technical Staff of Bell Telephone Laboratories, *Transmission Systems for Communications*, Western Electric Company, Winston-Salem, NC, 1970.

[2] R. M. Dolby, "Signal compressors and expanders," US Patent 3,345,416, Oct. 1974.

[3] A. Klein and Y. Tsividis, "Companding digital signal processors," in *Proc. 2006 IEEE ICASSP*, May 2006, vol. 3, pp. III–700 – III–703.

[4] Y. Tsividis, "Externally linear time-invariant systems and their application to companding signal processors," *IEEE Trans. Circuits and Systems II*, vol. 44, pp. 65–85, Feb. 1997.

[5] G. Efthivoulidis and Y. Tsividis, "Signal analysis of externally linear filters," in *Proc. 1999 IEEE ISCAS*, June 1999, pp. 65–68.

[6] A. Oppenheim and R. Schafer, *Discrete-Time Signal Processing*, pp. 373–377, Prentice Hall, New Jersey, 1989.

[7] "http://www.columbia.edu/∼aek84/spconf2007.htm," .