Algorithm - Logarithmic Signal Companding - Not Just a Good Idea - It Is μ-Law

Author: David Van Ess
Associated Project: Yes
Associated Part Family: CY8C27x43, CY28x43, CY28x45, CY28x52, CY8C29x66
Software Version: PSoC® Designer™ 5.4
Related Application Notes: None

AN2095 explains how logarithmic signal compression works. Routines are developed and an application is shown to implement a μ-Law compressor that converts an analog voice band signal and produces a digitized 8-bit compressed value. An expanding DAC is also developed that restores the compressed digital value back to an analog value.

Introduction

Virtually all telephony applications are becoming digital. Be it wireless or standard line, digitization of speech for transmission has advantages over traditional analog techniques. μ-Law (pronounced mu law) is a technique of data compression and expansion that allows for a greater dynamic range given the same signal bandwidth. This Application Note explains the logarithmic nature of the human ear's response. Equations are developed to calculate the signal-to-noise ratio for variable input levels. A PSoC® implementation of a μ-Law compressor and expander (compandor) is developed. A project is included that accomplishes the following:

Digitizes and compresses incoming data to an 8-bit value.

Transmits this data via a RS232 transmitter.
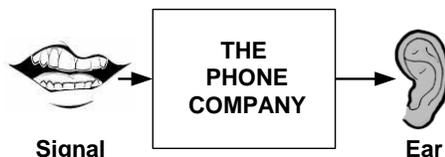
Receives this data via a RS232 receiver.

Expands this data and coverts it back to an analog value.

'C' callable companding routines are developed and presented.

The Really Big Picture

Figure 1 shows the block diagram for a typical telephony application:

Figure 1. "The Really Big Picture" Block Diagram

It doesn't get more basic than this. There are three different components to this system:

The Signal

The Phone Company

The Ear

The original analog speech must be accurately received. Accurate is such a fuzzy word. A better description is that an acceptable facsimile of the signal be transmitted to the ear. The definition of "acceptable" is as follows.

Signal

For telephony applications this is voice or speech. Human speech has a range approximately 100 Hz to 9 kHz. It has roughly 40 dB of dynamic range, however normal conversation rarely exceeds 20 dB. (Heavy equipment operators, irate customers and pointy haired managers excluded.)

The Phone Company

The phone company limits the signal bandwidth to a range of 300 Hz to 3.5 kHz. This is not a problem for voice communication applications. A voice signal limited to this range is easily understood.

Digital systems require a sample rate of 8-kilo samples per second (ksps) with 8 bits of resolution and only 8 bits of resolution. No free "evening and weekend" bits or "anytime" bits. Just 8 bits. That's all, no more. An upper limit of 3.5 kHz puts the signal bandwidth comfortably below the 8 kHz/2 Nyquist sampling limit.

The Human Ear

The human ear is an engineering marvel. It has a logarithmic response. That is, it has the ability to become more or less sensitive to sounds. The ear can hear sound pressure levels (SPL) as low as 0 dB$_{SPL}$ (a whisper) up to 120 dB$_{SPL}$ (a painfully loud rock concert). However, at any particular sound level, the dynamic range of the ear is only 40 dB. This means you can hear someone whisper and also enjoy a rock concert with same ears, just not at the same time. If you take your niece to a Britney Spears

concert she will not be able to hear you repeatedly mutter, "This is lame!"

If the noise is kept 40 db below the signal, the ear does not detect it.

Quantization Noise, Public Enemy #1

In a digitalized system the noise resulting from analog to digital conversion is one of the largest, if not the largest, noise contributor. Figure 2 is a visual example of an analog to digital (ADC) conversion.

Figure 2. Example of an Analog Digital Conversion
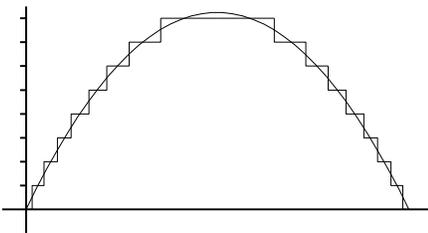
The ADC output has finite resolution. The difference between these two signals is the quantization noise. Intuitively, an ADC with finer resolution levels results in less quantization noise.

For the following mathematical models, use the following definitions:

The ADC range is normalized to +/- 1.

The input signal is a sinusoid with amplitude "a" (where $0 \le a \le 1$).

An "n" bit ADC has 2^n quantization levels. Equation (1) defines the ADC resolution.

$$\Delta = \frac{ADCRange}{\#QuantizationLevels} = \frac{2}{2^n}$$
Equation 1

The quantization error ε is the difference between the actual signal and the quantized value. For a dynamic signal, this error averages to zero and is limited to +/- ½Δ. It is evenly distributed between the limits of +/-½Δ. Figure 2 defines the noise as the RMS value of this error.

$$Noise = \varepsilon_{RMS} = \sqrt{\frac{1}{\Delta}\int_{-\frac{\Delta}{2}}^{\frac{\Delta}{2}}\varepsilon^2 d\varepsilon} = \sqrt{\frac{1}{\Delta}\frac{\varepsilon^3}{3}\Big|_{-\frac{\Delta}{2}}^{\frac{\Delta}{2}}} = \frac{\Delta}{\sqrt{12}}$$
Equation 2

The quantization noise is directly proportional to the ADC resolution.

The signal is a sinusoid with amplitude of "a." Equation (3) defines its RMS value:

$$Signal = \sqrt{\frac{1}{\pi}\int_0^\pi (a\sin(t))^2 dt} = \frac{a}{\sqrt{2}}$$
Equation 3

The ratio of these two values is defined as the Signal-to-Noise Ratio (SNR). It is normally expressed in dB as shown in Equation (4):
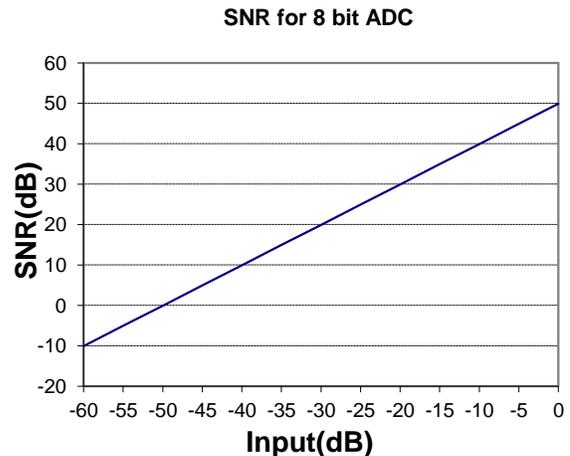
$$SNR_{dB} = 20\log\left(\frac{Signal}{Noise}\right) = 20\log\left(\frac{a}{\sqrt{2}}\frac{\sqrt{12}}{\Delta}\right)$$
Equation 4

Combining equations (1) and (4) results in simplified Equation (5):

$$SNR_{dB} = 20\log\left(a \cdot 2^n \sqrt{\frac{3}{2}}\right) = 6.02n + 1.76 + 20\log(a)$$
Equation 5

Figure 3 is a plot of the SNR for an 8-bit ADC:

Figure 3. SNR vs. Input for 8-Bit ADC

The SNR is as high as 50 dB for a full-scale input but falls to 40 dB for a -10 dB input. For a -20 dB input, the SNR is down to 30 dB. This SNR calculation only accounts for quantization noise. Other noise sources in the system reduce the actual SNR value even further.

Equation (6) shows the dynamic range to be the ratio of the range to the resolution:

$$DR_{dB} = 20\log\left(\frac{ADCRange}{\Delta}\right) = 6.02 \cdot n$$
Equation 6

Non-Linear Quantization

Figure 3 shows the SNR as a function of the relative amplitude of the input. It is 50 dB for a full-scale input but falls rapidly. More bits of resolution would allow a wider range of input signals but the phone company makes it pretty clear that this is not going to happen.

An alternative is to make Δ a function of input signal amplitude. If Δ is made larger for large input signals and smaller for small signals, some of the excess SNR at the

top of the input range can be used to boost the SNR for low-level inputs. This is called non-linear quantization. Most non-linear quantization techniques are based on some logarithmic transfer function.

μ-Law Compression

North American and Japanese telephony applications use μ-Law compression. Equation (7) is the compression function. Where:

The input in normalized (-1 ≤ x ≤ 1).

y is the compressed output.

μ is the compression factor.

$$y = sign(x) \cdot \frac{\ln(1 + \mu \, | \, x |)}{\ln(1 + \mu)}$$

Equation 7

μ can be any positive value. The larger μ becomes, the greater the compression. For North American and Japanese applications, μ is set to 255. Figure 4 is the plot of Equation (7) with μ set to 255 and x limited to its positive range:
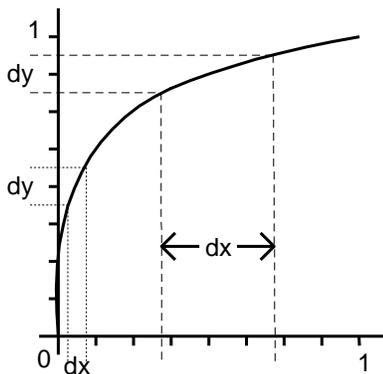
Figure 4. μ-Law Plot μ = 255

Figure 4 graphically shows that for uniform quantization levels on the y-axis, the quantization levels on x-axis increase with amplitude.

Taking the derivative of Equation (7) results in Equation (8):

$$\frac{dy}{dx} = \frac{\mu}{\ln(1 + \mu)} \cdot \frac{1}{1 + \mu \cdot x}$$

Equation 8

Solving Equation (8) for dx results in Equation (9):

$$dx = (1 + \mu \cdot x) \cdot \frac{\ln(1 + \mu)}{\mu} dy$$

Equation 9

Equation (10) defines a uniform resolution of the y-axis for an n bit ADC:

$$\Delta_y = \frac{2}{2^n}$$

Equation 10

Substituting Δ_y for dy in Equation (9) results in Equation (11):

$$\Delta_x(x) = dx = (1 + \mu \cdot x) \cdot \frac{\ln(1 + \mu)}{\mu} \cdot \frac{2}{2^n}$$

Equation 11

The resolution on the x-axis is shown to be a function of x. Equation (12) shows that the noise is also a function of x:

$$Noise(x) = \frac{\Delta_x(x)}{\sqrt{12}} = \frac{(1 + \mu \cdot x) \cdot \dfrac{\ln(1 + \mu)}{\mu} \cdot \dfrac{2}{2^n}}{\sqrt{12}}$$

Equation 12

The total noise is the RMS of all the quantization noise of the input signal. Equation (13) defines the noise for a sinusoidal input with an amplitude of "a."

$$Noise_{RMS} = \sqrt{\frac{1}{\pi} \int_0^\pi Noise(a \cdot \sin(x))^2 \, dx}$$

Equation 13

Combining equations (12) and (13) results in Equation (14):

$$Noise_{RMS} = \sqrt{\frac{1}{\pi} \int_0^\pi \left(\frac{(1 + \mu a \sin(x)) \dfrac{\ln(1 + \mu)}{\mu}}{\dfrac{2^n}{2} \sqrt{12}} \right)^2 dx}$$

Equation 14

Solving the integral in Equation (14) results in Equation (15):

$$Noise_{RMS} = \frac{\dfrac{2}{2^n}}{\sqrt{12}} \cdot \frac{\ln(1 + \mu)}{\mu} \cdot \sqrt{1 + \frac{a^2 \mu^2}{2} + \frac{4 \mu a}{\pi}}$$
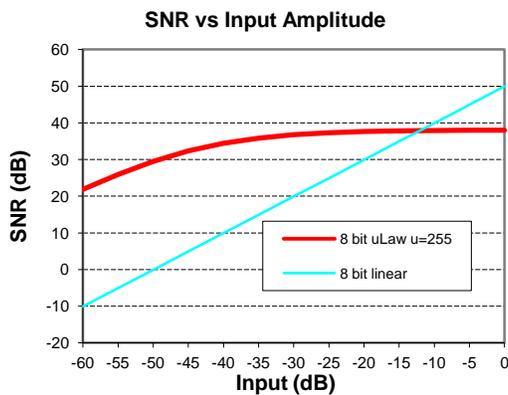
Equation 15

Equation (16) defines the RMS value for the same sinusoidal input:

$$Signal = \sqrt{\frac{1}{\pi} \int_0^\pi (a \sin(x))^2 \, dx} = \frac{a}{\sqrt{2}}$$

Equation 16

Equation (17) takes equations (15) and (16) to calculate the SNR:

$$SNR_{u(dB)} = 20 \log \left(a \cdot 2^n \cdot \sqrt{\frac{3}{2}} \cdot \frac{\dfrac{\mu}{\ln(1 + \mu)}}{\sqrt{1 + \dfrac{a^2 \mu^2}{2} + \dfrac{4 \mu \cdot a}{\pi}}} \right)$$

Equation 17

Figure 5 is a plot of the SNR for a μ-Law compressed signal with n set to 8 and μ set to 255:

Figure 5. Linear and μ-Law SNR Plot (n=8 μ=255)

The math just described can be a bit overwhelming. The spreadsheet used to generate Figure 5 *uLawSNRCalculation.xls,* is located in the project file associated with this Application Note. It allows you to change the level of compression or digitization and view the results. The reader is encouraged to manipulate the values of **n** and **μ** to develop an intuitive feel for their effect on signal compression and SNR.

Figure 5 also includes a plot of the SNR for an 8 bit. It is apparent that the compression allows for an acceptable SNR for much smaller inputs. It does so at the expense of peak SNR. But anything over 40 dB is a waste anyway.

Equation (18) shows the dynamic range to be the ratio of the signal range to the smallest resolution:

$$DR_{u(dB)} = 20\log\left(\frac{ADCRange}{\min \Delta_x(x)}\right) = 20\log\left(\frac{2}{\Delta_x(0)}\right) \quad \text{Equation 18}$$

The resolution is smallest when x = 0. Combining equations (11) and (18) result in Equation (19):

$$DR_{u(dB)} = 20\log\left(\frac{2}{\frac{\ln(1+\mu)}{\mu}\frac{2}{2^n}}\right) = 20\log\left(2^n\frac{\mu}{\ln(1+\mu)}\right) \quad \text{Equation 19}$$

With μ set to 255, Equation (19) reduces to a simplified Equation (20):

$$DR_{u(dB)} = 6.02 \cdot n + 33.25 = 6.02 \cdot (n + 5.52) \quad \text{Equation 20}$$

The dynamic range for a μ-Law compressed signal is 33 dB or 5½ bits greater than the dynamic range for a linear signal, given the same sampling bandwidth.

Again, an increase in dynamic range comes at the expense of peak SNR.

Building a μ-Law Compressor

Figure 6 shows two possible ways to implement a μ-Law compressor:

Figure 6. Block Diagrams for μ-Law Compression
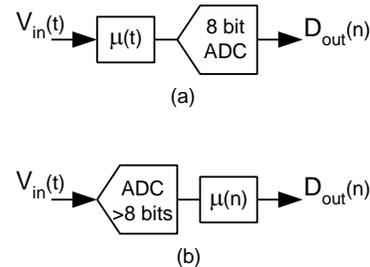
Two different techniques are shown:

Compress the signal and digitize with an 8-bit ADC.

Digitize with a higher resolution ADC and compress the digitized signal down to 8 bits.

Each technique has its own particular advantages and disadvantages. The first requires an ADC with only 8 bits of resolution but requires a logarithmic amplifier to implement the compression function. This technique is widely used in highly integrated single-chip designs for high volume consumer applications.

The second technique requires an ADC with a higher resolution, but no logarithmic amplifier is required. Equation (19) shows that 13.5 bits is the most resolution required.

As the cost of ADCs continues to decrease faster than the cost of logarithmic amplifiers, more μ-Law compression designs can be implemented using the second technique.

The PSoC philosophy is to offer reconfigurable general-purpose components. A logarithmic amplifier does not meet these guidelines. It is hard to make a logarithmic amplifier anything other than a logarithmic amplifier.

The PSoC solution uses the second technique. It digitizes, then compresses.

For an 8-ksps sample rate, the compression in Equation (21) is calculated every 125 μsec:

$$y(n) = sign(x(n)) \cdot \frac{\ln(1 + 255 | x(n) |)}{\ln(256)} \quad \text{Equation 21}$$

Three methods of calculating Equation (21) are:

Complete Mathematical Operation

Linear Approximation

Lookup Table

Complete Mathematical Solution requires:

Normalizing the Input Data

Calculating a Logarithm

At Least One Multiplication

Several Additions

Doing this all in just 125 μsec requires a DSP or a processor with a fast mathematical engine. It is just not an option for a microcontroller.

The **Linear Approximation** requires that the data be normalized to 14 bits. A bias value of 33 is added to this linear data and the five most significant bits and their position in the data word is used to calculate the compressed value. Figure 7 shows a table for a Linear Approximation to Equation (21):

Figure 7. Linear to Compressed Format Chart

Biased Linear Input Data														Compressed Data							
S = 0 for positive value / S = 1 for negative value														Exponent				Mantissa			
S	12	11	10	9	8	7	6	5	4	3	2	1	0	S	6	5	4	3	2	1	0
+/-	0	0	0	0	0	0	0	1	d	c	b	a	x	+/-	0	0	0	d	c	b	a
+/-	0	0	0	0	0	0	1	d	c	b	a	x	x	+/-	0	0	1	d	c	b	a
+/-	0	0	0	0	0	1	d	c	b	a	x	x	x	+/-	0	1	0	d	c	b	a
+/-	0	0	0	0	1	d	c	b	a	x	x	x	x	+/-	0	1	1	d	c	b	a
+/-	0	0	0	1	d	c	b	a	x	x	x	x	x	+/-	1	0	0	d	c	b	a
+/-	0	0	1	d	c	b	a	x	x	x	x	x	x	+/-	1	0	1	d	c	b	a
+/-	0	1	d	c	b	a	x	x	x	x	x	x	x	+/-	1	1	0	d	c	b	a
+/-	1	d	c	b	a	x	x	x	x	x	x	x	x	+/-	1	1	1	d	c	b	a

This is the most popular μ-Law compression technique. It requires hardly any code to implement. So simple, in fact, that it has become the de-facto standard for μ-Law compression. The International Telecommunication Union standard, ITU-T G.711, defines this approximation as actual μ-Law compression.

It somehow seems wrong that an approximation be elevated to a higher level than the real function (sort of like picking your target after shooting). Try reading ITU-T G.711. (Just try finding it) and you happily just accept it.

Please note that this compression algorithm is done with 14-bit data. The dynamic range of a μ-Law compressed signal is no more than 13½ bits. Several software functions claim to compress 16-bit linear data to an 8-bit compressed value. They do so by discarding the least significant 2 bits before compressing. And they do so with no shame! How convenient, just throw them away. Do not fall into the trap of assuming a 16-bit compression routine requires a 16-bit ADC. Following this logic, one could easily build a 24-bit linear data to an 8-bit compressor merely by discarding the least significant 10 bits.

A **Lookup Table** allows Equation (22) to be implemented:

$$y = f(x)$$

Equation 22

f(x) can be any function you desire. It can be a table of mathematically calculated μ-Law compression values. Or it can be a table of values generated using the ITU-T G.711 algorithm. The table can compensate for any known ADC non-linearity or gain errors. Merely changing the tables easily allows A-Law compression. A-Law is the compression standard used for European telephony applications. It is similar to μ-Law with its key feature being that it is not American. Think of it as "Metric μ-Law." Several tables can be stored to allow for different compression schemes within a single product. The only down side is storage space.

Equation (23) shows that only positive values of x need be in the table:

$$-f(x) = f(-x)$$

Equation 23

So with only of half the data needed to construct a table, Table 1 shows the table size verses the input data resolution.

**Table 1. ADC Resolution vs. Table Size**

ADC Resolution	Table Size
9 Bits	256 Bytes
10 Bits	512 Bytes
11 Bits	1024 Bytes
12 Bits	2046 Bytes
13 Bits	4096 Bytes
14 Bits	8096 Bytes
15 Bits	8096 Bytes
16 Bits	8096 Bytes

Notice that the table allows for data with less than 14 bits of resolution. This is quite acceptable. The compression data has 13½ bits of dynamic range. Inputs with less resolution can be compressed. The over all dynamic range is either the resolution of the ADC or 13½ bits, whichever is smaller.

Building a μLaw Expander

Figure 8 shows a block diagram for implementing a μ-Law expander.

Figure 8. Block Diagram for μ-Law Expander

$$D_{in}(n) \rightarrow \boxed{\mu^{-1}(n)} \rightarrow \boxed{DAC} \rightarrow V_{out}(n)$$
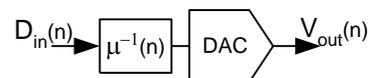
The digital signal is expander fed to a DAC. Like the compressor, there are three ways to implement an expander:

Complete Mathematical Operation

Linear Approximation

Lookup Table

The **Complete Mathematical Solution** requires performing the calculation in Equation (24):

$$x(n) = sign(x(n) \cdot \frac{(1+\mu)^{|y(n)|} - 1}{\mu} f(x)$$

<div align="right">Equation 24</div>

As with the compressor, this expansion requires a lot of mathematical operation in the 125-µsec data sample time.

The **Linear Approximation** has a normalized output of 14 bits. It is shown in Figure 9:

Figure 9. Compressed to Linear Format Chart

Compressed Data								Biased Linear Input Data													
Exponent			Mantissa																		
S	6	5	4	3	2	1	0	S	12	11	10	9	8	7	6	5	4	3	2	1	0
+/-	0	0	0	d	c	b	a	+/-	0	0	0	0	0	0	1	d	c	b	a	1	
+/-	0	0	1	d	c	b	a	+/-	0	0	0	0	0	0	1	d	c	b	a	1	x
+/-	0	1	0	d	c	b	a	+/-	0	0	0	0	0	1	d	c	b	a	1	x	x
+/-	0	1	1	d	c	b	a	+/-	0	0	0	0	1	d	c	b	a	1	x	x	x
+/-	1	0	0	d	c	b	a	+/-	0	0	0	1	d	c	b	a	1	x	x	x	x
+/-	1	0	1	d	c	b	a	+/-	0	0	1	d	c	b	a	1	x	x	x	x	x
+/-	1	1	0	d	c	b	a	+/-	0	1	d	c	b	a	1	x	x	x	x	x	x
+/-	1	1	1	d	c	b	a	+/-	1	d	c	b	a	1	x	x	x	x	x	x	x

Thirty-three is subtracted from the bias output value to produce the linear output. As with compression, this is the ITU-T G.711 standard for µ-Law data expansion.

A **Lookup Table** allows Equation (25) to be implemented:

$$x = f^{-1}(y)$$

<div align="right">Equation 25</div>

As with the compressor, this equation can implement any function you desire. It can be a table of mathematically calculated µ-Law compression values. Or it can be a table of values generated using the ITU-T G.711 algorithm. It can compensate for any system non-linearity. The actual output can be a 14-bit linear output or it can be the data required to control a DAC. The symmetry of the positive and negative values allows for only 128 words of table storage. With most likely two bytes per word, the storage requirement is 256 bytes.

A PSoC µ-Law Compressor

Figure 10 shows a block diagram for a µ-Law compressor:

Figure 10. PSoC µ-Law Compressor Block Diagram
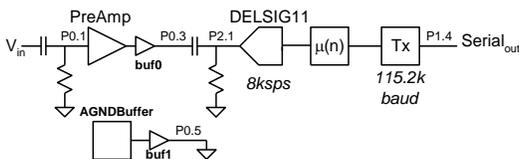
It consists of the following sections:

 PreAmp

11-Bit Delta Sigma ADC

µ-Law Compressor

Serial Transmitter

PreAmp

The input signal is AC coupled to the input of PreAmp. It is a PGA User Module set for unity gain. Depending on the application, the gain can vary as much as +/- 24 dB. The output is brought outside the chip via an analog buffer.

ADC

The signal is, again, AC coupled and brought to DELSIG11. Equation (26) defines the sample rate:

$$SampleRate = \frac{DataClock}{1024} \quad f(x) = f(-x)$$

<div align="right">Equation 26</div>

For a maximum data clock of 8 MHz, the sample rate is only 7.81 ksps. This falls well below the required 8 ksps. The DELSIG11 uses a timer with a period of 256 to control its decimator. If its period is changed to 250 then Equation (27) defines the sample rate:

$$SampleRate = \frac{DataClock}{1000} \quad f(x) = f(-x)$$

<div align="right">Equation 27</div>

For a data clock of 8 MHz, the sample rate is now 8 ksps. Without going into details about delta-signal operation, the gain is also reduced by $(250/256)^2$.

The following code segment shows how to alter the ADC for 8 ksps operation:

```
    DELSIG11_StartAD();
    DELSIG11_TimerDR1 = 0xF9;//set period to
250
```

µ-Law Compressor

The compressor uses a lookup table. The table has a gain adjust factored into it to compensate for the ADC loss. It implements the ITU-T G.711 compression algorithm. It takes 11-bit signed output data from the DELSIG11 and converts it to an 8-bit compressed value. Example Code 1 is the actual routine:

Code 1

```
;----------------------------------------
;;  cDS11_to_uLaw:
;;  Takes an 11  linear value and
;;  converts it to a uLaw Value
;;  and mantissa.  Updates DACs.
;;  INPUTS:  X, A ia an 11 bit integer
;;  OUTPUTS: A is the uLaw Value
;;----------------------------------------
 cDS11_to_uLaw:
_cDS11_to_uLaw:
   swap A,X
   and A,07h        ;keep lower 3 bits
   add  A,(>uLawData);point   to   uLawData
Table
   romx             ;uLaw Value returned in
A
ret
```

This function can be found in *uLawStuff.asm*, located in the project file associated with this Application Note. Also included is *uLawStuff.h* that makes it a fastcall 'C' function.
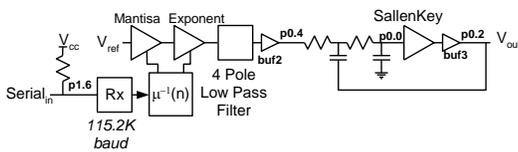
Serial Transmitter

It is just a 115.2 k-baud serial UART transmitter. For data having a bandwidth of 64 kbps, this transmission rate is adequate.

A PSoC μ-Law Expander

Figure 11 shows a block diagram for a μLaw expander:

Figure 11. PSoC μ-Law Expander Block Diagram

It consists of the following sections:

Serial Receiver

μ-Law Expander

Four-Pole Switched Cap Filter

Two-Pole Sallen Key Filter

Serial Receiver

It is just a 115.2 k-baud serial UART receiver. For data being received having a bandwidth of 64 kbps, this is adequate.

μ-Law Expander

The expander uses lookup tables to control the Matissa and Exponent multiplying DACs.

The ITU-T G.711 expansion is implemented in these tables to provide an 11-bit signed linear output. Example Code 2 is the actual routine:

Code 2

```
;-------------------------------------
;;   WriteuLawDAC:
;;   Takes an 8 bit signed value and
;;   separates into sign, exponent
;;   and mantissa.  Updates DACs.
;;   INPUTS:  A contains the uLaw value
;;   OUTPUTS: None.
;;-------------------------------------
 WriteuLawDAC:
_WriteuLawDAC:
  mov   X,A
  add   A,80h
  if1: jnc else1 ;(is a neg value)
      index MantissaData
      jmp endif1
  else1:;(A is a positive value)
     index (MantissaData - 128)
```

```
    add   A,32  ;pos is neg * neg
endif1:
swap  A,X
and   A,7fh ;mask out sign
index ExponentData
M8C_Stall
mov   reg[Exponent_cr0],A
M8C_Unstall
mov   A,X
mov   reg[Mantissa_cr0],A
ret
```

As with the compression function, this expander function can be found in *uLawStuff.asm*, located in the project file associated with this Application Note. It also is defined as a fastcall 'C' function.

The DAC output updates at 8 ksps. For a 300 Hz sinusoid this works out to 26.67 samples. Twenty-six points do make up something that resembles a sinusoid. But a 3 kHz sinusoid only has only 2.67 samples. Not much of a sinusoid. A reconstruction filter is required to produce a smooth output over 300 Hz - 3500 Hz bandwidth. This application uses a six-pole, low pass filter.

Four-Pole Switched Cap Filter

Four of these poles are implemented with two LPF2 User Modules in series. They are switched capacitor bi-quad filters. Their component values can be viewed in the parameter section of each filter in the project associated with this Application Note. The output of these filters is brought outside the chip via an analog buffer.

Two-Pole Sallen Key Filter

This filter is built with a PGA User Module configured as a buffer, two external resistors, and two external capacitors. It completes the rest of six-pole response. It also removes switch noise generated by the previous filter. Appendix A has the values of the discrete components. It also shows User Module placement for the whole project.

A Complete System Test

Figure 12 is a block diagram of the compressor and expander figured as a compandor.

Figure 12. Compressor and Expander
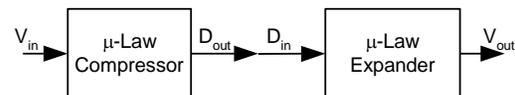
Of course, for real applications, the compressor and expander would be separate applications in different chips, if not different products. Combining them allows the SNR of a complete compression, expansion to be measured.

The code to do this is added into the DELSIG11 interrupt at the point marked for user code addition. It is shown in following example Code 3:

Code 3

```
;;----------------------------------------
-------

    ;;  data is now in X,A
    ;;   The user's handler should be placed
here

    call cDS11_to_uLaw
    call TX8_1_SendData
    include "RX8_1.inc"
    mov A,reg[RX8_1_CONTROL_REG]
    and A,RX8_RX_COMPLETE
    if_100:   jz   endif_100   ;(data   is
available)
        mov A, reg[RX8_1_RX_BUFFER_REG]
        call WriteuLawDAC
    endif_100:

;;----------------------------------------
--------
```

The 11-bit data is compressed and sent out the serial port. The serial receiver is polled for an available byte. If an available byte is found, it is retrieved, expanded, and converted back to an analog signal.

This works only if the data rates are exact. It only happens when they are generated with the same clock. For real applications, incoming data would be handled with its own interrupt or collected in a circular FIFO.

The performance measures of a signal compression system are harmonic distortion and noise over the range of signal frequency and level. Figure 13 shows a 1 kHz mid-band signal at an input level of $2V_{pp}$. The highest harmonic is 45 dB below the fundamental. Noise contributions are from broadband noise and sampling aliases.

The first sampling alias (f_{sample}-f_{signal}) is at 7.0 kHz (8 kHz-1 kHz) is 65 dB below signal level:

Figure 13. Spectral Plot for 1 kHz 2 V_{pp} Input
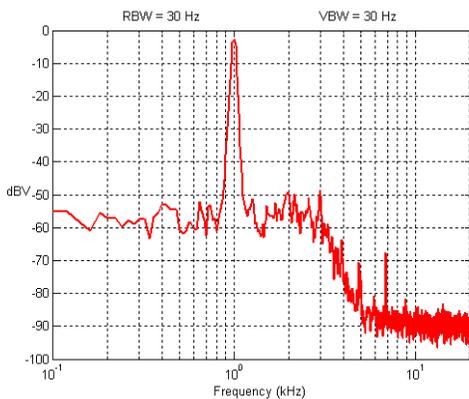
The nature of μ-Law compression is to generate essentially fixed SNR over a wide range of signal levels.

Figure 14 shows the same 1.0 kHz signal but at an input level of 100 mVp-p, 26 dB down from the input level of Figure 13. The harmonic levels and noise level drop along with the signal level to maintain essentially constant signal-to-noise ratio. The first sampling alias at 7.0 kHz remains at 65 dB below signal level:

Figure 14. Spectral Plot for 1 kHz 100 mV$_{pp}$ Input
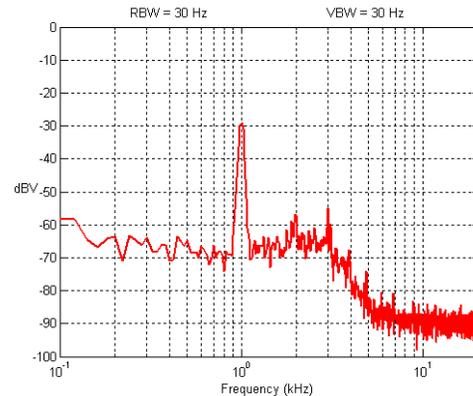
Signals at 300 and 3.0 kHz and at 2.0 Vp-p and 100 mVp-p are shown in figures 15 through 18.

Figure 15. Spectral Plot for 300 Hz 2 V_{pp} Input
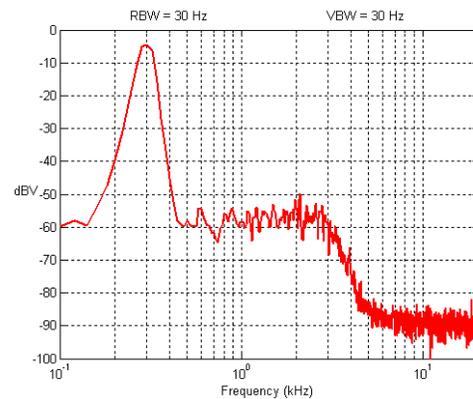
Figure 16. Spectral Plot for 300 Hz 100 mV$_{pp}$ Input
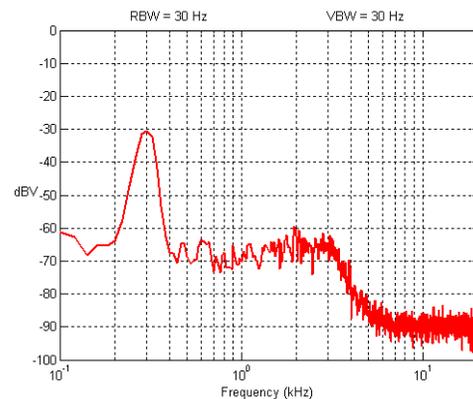
Figure 17. Spectral Plot for 3 kHz 2 V$_{pp}$ Input

Figure 18. Spectral Plot for 3 kHz 100 mV$_{pp}$ Input
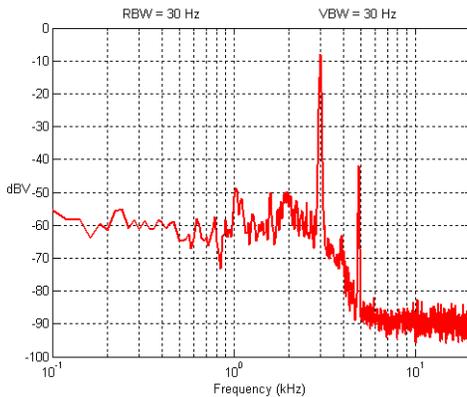
Note The low harmonic distortion level is maintained.

Higher frequencies have harmonics out of the nominal voice band, but alias harmonics can wind up in the voice band. The largest alias, shown in figures 17 and 18, is the 6 kHz harmonic of the 3 kHz signal below the 8 kHz sample frequency or 2.0 kHz for a 3.0 kHz input. This represents the highest noise case, but in a practical system these alias levels almost certainly would not impact usable signal fidelity.

Summary

Logarithmic Data Compression makes acceptable quality voice communication with a greater reduced signal bandwidth. The PSoC architecture allows easy implementation of the ITU-T G.711 µ-Law compression/expansion format. The large amount of chip analog resources allows complete signal conditioning, digitization, filtering, and signal reconstruction with only a handful of additional passive components.

Appendix A. PSoC User Module Placement and Pin Interface Schematic

Document History

Document Title: Algorithm - Logarithmic Signal Companding - Not Just a Good Idea - It Is μ-Law – AN2095

Document Number: 001-38006

Revision	ECN	Orig. of Change	Submission Date	Description of Change
**	1520284	PFZ	10/09/2007	New application note
*A	3110002	PFZ	12/14/2010	No Technical updates. Correct few typo errors. Added Document History details.
*B	3658917	SAMP	6/28/2012	Updated Appendix A. PSoC User Module Placement and Pin Interface Schematic: Updated the Analog and Digital Block Diagrams to PD 5.2. Included all PSoC1 part numbers with 4 analog columns. Put in new application note template.
*C	4224527	DCHE	12/18/2013	Updated in new template. Completing Sunset Review.
*D	4609931	DCHE	12/29/2014	Updated example project to PSoC Designer 5.4 Removed references of AN2036 Rephrased application note abstract

Worldwide Sales and Design Support

Cypress maintains a worldwide network of offices, solution centers, manufacturer's representatives, and distributors. To find the office closest to you, visit us at Cypress Locations.

Products

Automotive	cypress.com/go/automotive
Clocks & Buffers	cypress.com/go/clocks
Interface	cypress.com/go/interface
Lighting & Power Control	cypress.com/go/powerpsoc
Memory	cypress.com/go/memory
PSoC	cypress.com/go/psoc
Touch Sensing	cypress.com/go/touch
USB Controllers	cypress.com/go/usb
Wireless/RF	cypress.com/go/wireless

PSoC® Solutions

psoc.cypress.com/solutions

PSoC 1 | PSoC 3 | PSoC 4 | PSoC 5LP

Cypress Developer Community

Community | Forums | Blogs | Video | Training

Technical Support

cypress.com/go/support

Cypress Semiconductor
198 Champion Court
San Jose, CA 95134-1709

Phone : 408-943-2600
Fax : 408-943-4730
Website : www.cypress.com