Application Note AN3101-05: Compressor and Expander

by Shultz Wang

Introduction

This application note provides the algorithm for a basic compression/expansion curve.

Algorithm

When doing calculations with audio data, the amplitude scale used is decibels. Since decibels is a logarithmic scale, any plots of output versus input equations would use log/log space. In linear/linear space, a straight line is described by the equation **y=Bx+A**, where **B** is the slope and **A** the y-intercept. However, in log/log space, a straight line has to be an exponential equation to counteract the logarithmic scaling, with higher exponentials giving a greater slope, and a higher multiplier giving a larger y-intercept, resulting in the equation $\mathbf{y=Ax^B}$. By taking the log of the equation, its behavior may be examined:

$$\log(y) = \log(\mathbf{A}) + \mathbf{B}\log(x).$$

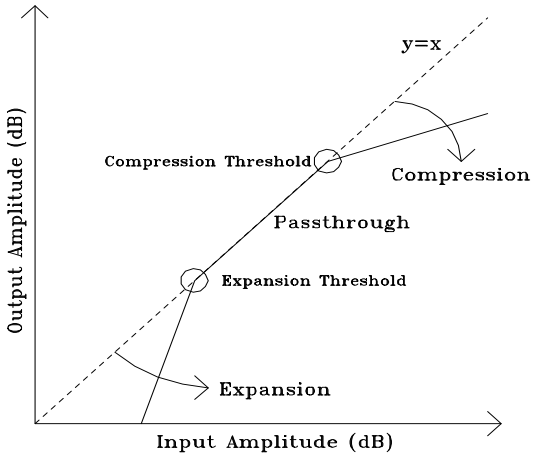
It can be seen that log(**A**) controls the y-intercept of the line, and **B** controls the slope of the line.

This equation will now be used to construct the complete compander curve. The curve is made up of three segments:

The upper compression segment $y=A_cx^{Bc}$,

the middle passthrough segment y=x, and

the lower expansion segment $y=A_ex^{Be}$.



A compressor decreases the maximum output signal levels by reducing the slope starting at the threshold upwards, from no compression (passthrough, at a ratio of 1) all the way to infinite compression (a limiter), which is a horizontal line. For compression, a decrease in slope is desired for higher ratios, so choose $B_c=1/R_c$. To solve for Ac, note that at the threshold the two lines meet, so

$$y(T_c) = T_c = A_cT_c^{(1/Rc)} \Rightarrow A_c = T_c^{(1-1/Rc)}.$$

Therefore

$$y = T_c^{(1-1/Rc)}\, x^{(1/Rc)}.$$

This is equal to the input times the compression gain G_c, therefore the gain is

$$G_c\, x = T_c^{(1-1/Rc)}\, x^{(1/Rc)},$$
$$G_c = (T_c/x)^{(1-1/Rc)}.$$

A straight audio passthrough maintains the straight line in log/log space, with a slope of 1 (output = input) and a y-intercept of 0, giving the equation y=x.

An expander increases the minimum input signal levels needed for a certain output signal level by increasing the slope starting at the threshold downwards, from no expansion (passthrough, at a ratio of 1) all the way to infinite expansion (a gate), which is a vertical line. After a derivation similar to one for the compressor, the expansion gain G_e can be derived to be

$$G_e = (T_e/x)^{(1-Re)}.$$

Smoothing

In order to avoid instantaneous compression and expansion, which can be heard as artifacts in the resulting audio, the companding is turned on and off gradually, in an exponential fashion. The smoothing equation is derived from the universal time constant formula for RC and RL circuits:

$$\text{Change} = (\text{Final-Start})[1-1/e^{(t/\tau)}].$$

To get the value at the next timestep, add the current value to both sides:

$$\text{Change} + \text{Start} = (\text{Final-Start})[1-1/e^{(t/\tau)}] + \text{Start}.$$

Rephrased,

Next = Current + k(Target-Current), where k is a fractional multiplier which determines the smoothing rate.

The time constant τ gives the amount of time it takes for the curve to rise 63% closer from its current value to its final value. Solving k for τ:

$k = 1 - 1/e^{(t/\tau)}$

$e^{(t/\tau)} = 1/(1-k)$

$t/\tau = \ln[1/(1-k)] = -\ln(1-k)$

$\tau = -t/\ln(1-k)$, where t is one sample period.

The basic smoothing equation, applied to the calculated gain coefficients, generates the attack and release times.

The following table can serve as a guide for selecting the proper fractional value for good attack and release time constants (sampling period = 1/48kHz). Common attack times are 1ms to 100ms. Common release times are 0.5s to 3s.

Shift	Fractional	Hex	τ = -(Sample Pd)/ln(1-Fractional)
1 bit	1/2	$020000	0.030ms
2 bits	1/4	$010000	0.072ms
3 bits	1/8	$008000	0.156ms
4 bits	1/16	$004000	0.323ms
5 bits	1/32	$002000	0.656ms
6 bits	1/64	$001000	1.323ms
7 bits	1/128	$000800	2.656ms
8 bits	1/256	$000400	5.323ms
9 bits	1/512	$000200	10.656ms
10 bits	1/1024	$000100	21.323ms
11 bits	1/2048	$000080	42.656ms
12 bits	1/4096	$000040	85.323ms
13 bits	1/8192	$000020	0.170s
14 bits	1/16384	$000010	0.341s
15 bits	1/32768	$000008	0.682s
16 bits	1/65536	$000004	1.365s
17 bits	1/131072	$000002	2.731s
18 bits	1/262144	$000001	5.461s

Source Code

First calculate or select the following values for each channel (the x subscript in the names means it refers to both compression (subscript c) and expansion (subscript e)).

T_x (Threshold)
G_x (Gain) Address (non-rotating)
R_x (Ratio)
k_{xa} (Attack fractional)
k_{xr} (Release fractional)
G (Final gain)

Note that since compression and expansion each need a register to store the current gain, to have both on every channel will require 16 registers. In this implementation only the threshold parameter requires two instruction alterations, all others require only one.

The compression algorithm is:

```
  ; Threshold detection

CM    $080000     $41#  ; Read Input channel # into A, scaled to +1 → -1
SKIP  !N          $1    ; Skip multiply by -1 if data positive
CM    $380000     $41#  ; Read -Input into A, scaled to +1 → -1
X1AC  $(-Tc)            ; Store |Input| into B, subtract Tc from |Input|
SKIP  N           $9    ; Skip Gc attack calculations if |Input| < Tc

  ; Attack conditional

LOGB                    ; LOG16(|Input|)
DAC   $f00        $(LOG16(Tc)); LOG16(Tc)-LOG16(|Input|)
CAD   $(1-1/Rc)   $0    ; (1-1/Rc)(LOG16(Tc)-LOG16(|Input|))
X1AC  $0                ; A->B
EXPB                    ; EXP16((1-1/Rc)(LOG16(Tc)-LOG16(|Input|)))
CMA   $3c0000     $GcA  ; (-1)Gc+EXP16((1-1/Rc)(LOG16(Tc)-LOG16(|Input|)))
CAM   $(kca)      $GcA  ; Gc+kca(EXP16((1-1/Rc)(LOG16(Tc)-LOG16(|Input|)))-Gc)
SXCA  $0          $GcA  ; Write new attack Gc, store Gc in B.
SKIP              $3    ; Skip Gc release calculations since |Input| ≥ Tc

  ; Release conditional

1MC   $3c0000     $GcA  ; (Gc-1)
CAM   $(-kcr)     $GcA  ; Gc-kcr(Gc-1) = Gc+kcr(1-Gc)
SXCA  $0          $GcA  ; Write new release Gc, store Gc in B.

  ; Gc in A and B
```

For expansion, replace all compression constants with expansion constants, replace $(1-1/R_c)$ with $(1-R_e)$, replace the G_c address with the G_e address, and replace the second skip instruction's condition with !N.

```
SKIP  !N          $8    ; Skip Ge attack calculations if |Input| > Te
```

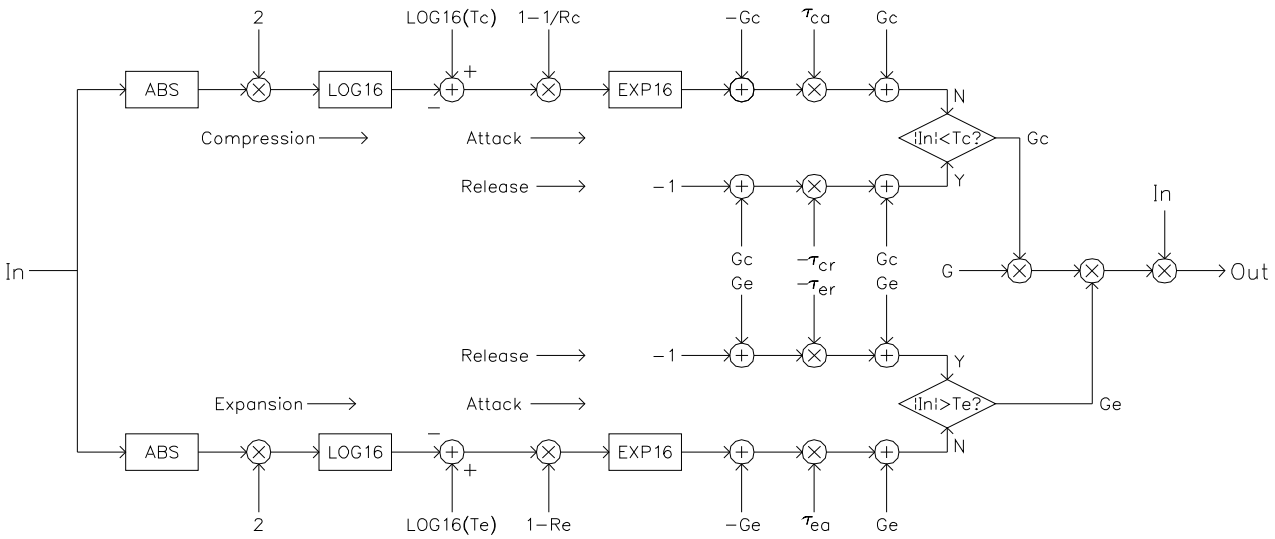
Multiply the input by the gains and write to output.

```
CM    $(G)        $GcA  ; Read Gc, multiplied by final gain
AMC   $0          $GeA  ; Gc*Ge*G
AMC   $0          $41#  ; Input=Input*Gc*Ge*G
SCA   $0          $41#  ; Write companded input to chan #
```

Note that gain reduction metering may be done by reading the gain value and subtracting it from unity. To output the metering to a channel, use the following code:

```
CM    $3c0000     $GxA  ; Read -Gx
1AC   $040000           ; 1-Gx
SCA   $0          $4##  ; Write gain reduction meter value to address ##
```

Compression/Expansion Algorithm Block Diagram



Sample compander:

```
; Compressor: Threshold = -6dB; Ratio = 4
; Attack τ = 0.323ms; Release τ = 0.682s

CM    $080000     $410  ; Read Input channel 0 into A, scaled to +1 → -1
SKIP  !N          $1    ; Skip multiply by -1 if data positive
CM    $380000     $410  ; Read -Input into A, scaled to +1 → -1
X1AC  $f800000          ; Store |Input| into B, subtract Tc from |Input|
SKIP  N           $9    ; Skip Gc attack calculations if |Input| < Tc
LOGB                    ; LOG16(|Input|)
DAC   $f00        $3f0000     ; LOG16(Tc)-LOG16(|Input|)
CAD   $030000     $0    ; (1-1/Rc)(LOG16(Tc)-LOG16(|Input|))
X1AC  $0                ; A->B
EXPB                    ; EXP16((1-1/Rc)(LOG16(Tc)-LOG16(|Input|)))
CMA   $3c0000     $40f  ; (-1)Gc+EXP16((1-1/Rc)(LOG16(Tc)-LOG16(|Input|)))
CAM   $004000     $40f  ; Gc+kea(EXP16((1-1/Rc)(LOG16(Tc)-LOG16(|Input|)))-Gc)
SCA   $0          $40f  ; Write new attack Gc
SKIP              $3    ; Skip Gc release calculations since |Input| ≥ Tc
1MC   $3c0000     $40f  ; (Gc-1)
CAM   $3ffff8     $40f  ; Gc-kcr(Gc-1) = Gc+kcr(1-Gc)
SCA   $040000     $40f  ; Write new release Gc.
```

```
; Expander: Threshold = -72dB; Ratio = 4
; Attack τ = 0.323ms; Release τ = 0.682s

CM    $080000     $410  ; Read Input channel 0 into A, scaled to +1 → -1
SKIP  !N          $1    ; Skip multiply by -1 if data positive
CM    $380000     $410  ; Read -Input into A, scaled to +1 → -1
X1AC  $ffff000          ; Store |Input| into B, subtract Te from |Input|
SKIP  !N          $9    ; Skip Ge attack calculations if |Input| > Te
LOGB                    ; LOG16(|Input|)
DAC   $f00        $340000     ; LOG16(Te)-LOG16(|Input|)
CAD   $340000     $0    ; (1-Re)(LOG16(Te)-LOG16(|Input|))
X1AC  $0                ; A->B
EXPB                    ; EXP16((1-Re)(LOG16(Te)-LOG16(|Input|)))
CMA   $3c0000     $40e  ; (-1)Ge+EXP16((1-Re)(LOG16(Te)-LOG16(|Input|)))
CAM   $004000     $40e  ; Ge+kea(EXP16((1-Re)(LOG16(Te)-LOG16(|Input|)))-Ge)
SCA   $0          $40e  ; Write new attack Ge
SKIP              $3    ; Skip Ge release calculations since |Input| ≤ Te
1MC   $3c0000     $40e  ; (Ge-1)
CAM   $3ffff8     $40e  ; Ge-ker(Ge-1) = Ge+ker(1-Ge)
SCA   $040000     $40e  ; Write new release Ge.

; Multiply input by gains; Final gain = 1

CM    $040000     $40f  ; Read Gc, multiplied by final gain
AMC   $0          $40e  ; Ge*Gc*G
AMC   $0          $410  ; Input = Input*Ge*Gc*G
SCA   $0          $410  ; Write chan 0
```

NOTICE

Wavefront Semiconductor reserves the right to make changes to their products or to discontinue any product or service without notice. All products are sold subject to terms and conditions of sale supplied at the time of order acknowledgement. Wavefront Semiconductor assumes no responsibility for the use of any circuits described herein, conveys no license under any patent or other right, and makes no representation that the circuits are free of patent infringement. Information contained herein is only for illustration purposes and may vary depending upon a user's specific application. While the information in this publication has been carefully checked, no responsibility is assumed for inaccuracies.

Wavefront Semiconductor products are not designed for use in applications which involve potential risks of death, personal injury, or severe property or environmental damage or life support applications where the failure or malfunction of the product can reasonably be expected to cause failure of the life support system or to significantly affect its safety or effectiveness.

All trademarks and registered trademarks are property of their respective owners.

Contact Information:

Wavefront Semiconductor
200 Scenic View Drive
Cumberland, RI 02864 U.S.A.
Tel: +1 401 658-3670
Fax: +1 401 658-3680
On the web at www.wavefrontsemi.com
Email: info@wavefrontsemi.com

Application note revised March, 2005