TMS320 DSP
DESIGNER'S NOTEBOOK
Number 71
TEXAS INSTRUMENTS

μ-Law Compression on the TMS320C54x

Contributed by Chuck Brokish

Design Problem How can you perform a software μ-law compression algorithm on the TMS320C54x with a minimum number of instructions, and without requiring the memory needed for a lookup table?

Solution Mu-law (μ-law) companding is a form of logarithmic data compression for audio data. Due to the fact that we hear logarithmically, sound recorded at higher levels does not require the same resolution as low-level sound. This allows us to disregard the least significant bits in high-level data. This turns out to resemble a logarithmic transformation. The resulting compression forces a 13-bit number to be represented as an 8-bit number.

Basically, the compression algorithm adds a bias to the data and preserves the five most significant bits for transmitting. The TMS320C54x implementation makes use of the **EXP** and **NORM** instruction. These instructions allows us to extract the most significant bits without requiring a look-up table, thus saving memory.

The μ-law compression algorithm defines a segment and a quantization for each value represented. By defining a segment based on the most significant bit of the data, one can use the same number of quantization bits in all cases and represent small values with tighter resolution than is used for large values.

Shown below is a table representing the translation from linear to compressed (PCM μ-255) data. Bits 6–4 represent the **segment**, which represents the logarithmic magnitude domain, while bits 3–0 represent the **quantization** within that domain.

The following code executes the μ-law conversion in 14 clock cycles and requires only three memory locations for Bias and Mask values. It also makes use of both

Biased Input Values														Compressed Code Word							
														Segment			Quantization				
Bit:	12	11	10	9	8	7	6	5	4	3	2	1	0	Bit:	6	5	4	3	2	1	0
	0	0	0	0	0	0	0	1	Q_3	Q2	Q_1	Q_0	x		0	0	0	Q_3	Q_2	Q_1	Q_0
	0	0	0	0	0	0	1	Q_3	Q2	Q_1	Q_0	x	x		0	0	1	Q_3	Q_2	Q_1	Q_0
	0	0	0	0	0	1	Q_3	Q2	Q_1	Q_0	x	x	x		0	1	0	Q_3	Q_2	Q_1	Q_0
	0	0	0	0	1	Q_3	Q2	Q_1	Q_0	x	x	x	x		0	1	1	Q_3	Q_2	Q_1	Q_0
	0	0	0	1	Q_3	Q2	Q_1	Q_0	x	x	x	x	x		1	0	0	Q_3	Q_2	Q_1	Q_0
	0	0	1	Q_3	Q2	Q_1	Q_0	x	x	x	x	x	x		1	0	1	Q_3	Q_2	Q_1	Q_0
	0	1	Q_3	Q2	Q_1	Q_0	x	x	x	x	x	x	x		1	1	0	Q_3	Q_2	Q_1	Q_0
	1	Q_3	Q_2	Q1	Q_0	x	x	x	x	x	x	x	x		1	1	1	Q_3	Q_2	Q_1	Q_0

US TMS320 HOTLINE (713) 274-2320 US TMS320 FAX (713) 274-2324 US TMS320 BBS (713) 274-2323 US email=4389750@mcimail.com

accumulators, resulting in less temporary storage of data. The required code for the compression is from `Start to Done.` The rest of the code is set up for test purposes. Note that the output is inverted within the code to conform with PCM transmission practices.

```
;***************************************************************
;       mu-Law Code for 'C54x Processor
;       Assumptions:
;               AR3 --> Q13 Linear #
;               AR2 --> Bias=33
;                       Mask=7fh (for PCM bit inversion)
;                       Sign=80h (for PCM Code)
;               DP -->  Page 0
;       Output:
;               B contains mu-law output
;***************************************************************

    stm    #q13_data, AR3   ;load AR3 at start of DMEM
Here
    stm    #Bias, AR2       ;load AR2 with count of 8
Start
    ld     *AR3,A           ;Load Q13 into Accumulator
    abs    A                ;Work with positive # only
    add    *AR2+,A          ;Add Bias (33)
    exp    A                ;Calculate leading zeros, place in Treg
    bit    *AR3,15-15       ;Check sign bit of Q13
    norm   A                ;Left justify A
    sfta   A,-16            ;Shift into low accumulator
    ld     #24,B            ;delta of segment and Treg (-1)
    sub    T,B              ;calculate segment (-1)
    sfta   B,4              ;adjust segment to proper bits
    add    A,-10,B          ;concantinate segment and quantizat'n bits
    xor    *AR2+,B          ;invert PCM result
    xc     1,NTC            ;check if Q13 input was negative
     add   *AR2,B           ;  ...if so, negate B
Done
    mar     *AR3+           ;Increment AR3 for next test Input
    b       Here            ;Loop to test additional Input Values

    .data
q13_data
    .word   0000h
    .word   0001h
    .word   001eh
    .word   001fh
    .word   005eh
    .word   005fh
    .word   00deh
    .word   00dfh
    .word   01deh
    .word   01dfh
    .word   03deh
    .word   03dfh
        .word   07deh
        .word   07dfh
        .word   0fdeh
        .word   0fdfh
        .word   1fdeh
  Bias
        .word   33
        .word   7fh
        .word   80h
```