wavefront
SEMICONDUCTOR

We make the parts that set creative people free™

Application Note AN3101-03: Log and Exponent for 1KM

by Chris Maple

The examples shown below are for educational purposes only and are not guaranteed to work in your application.

The log and exponent instructions in the 1KM are piecewise linear approximations to log base 16 and 16-to-a-power of the number in the B register.

For log, the sign bit is ignored and the remaining bits shifted left to get a leading 1. The 4 bits after the leading 1 index a table which provides a reference point and a slope. The bits below those 4 are multiplied by the slope and the result is added to the reference point. That number is added to 0.25 times the number of bits of shifting that took place. The result is 16 segments of linear approximation for every factor of 2 used as input to the log function.

The log falls into the range 0.75>log>=-6.5. For log(0) the result is -6.75.

The reference points are as exact as can be represented in the 1KM. The errors are all negative (the result is smaller than the true log) and range from zero to -0.000165. Thus one additional bit of accuracy can be gained in a single instruction by adding 0.000082 to the result of the log. Other techniques can be used to gain additional accuracy, limited by the roundoff errors in the procedures and the amount of correction applied.

The error pattern is the same for every factor of 2 of the input to the log function. It looks like 16 parabolic sections, the biggest errors for the smallest input values. By adding a correction factor to the log, the error can be improved to 0.0000145 in 12 ticks.

Error Pattern

```
INSTRUCTION            COMMENT
CM     1.0 0x418       ;get the log field extract data. Input assumed in B
ANDC   0x03C0000       ;isolate the 16 segment numbers
CAD    4.183810 0
1AC    -1.24035894     ;create the declining magnitude function
SCA    1.0 0x400       ;save to temp location
CM     1.0 0x418       ;get log field extract data
ANDC   0x003FFFF       ;isolate the ordinate in the segment, values 0 to 1/64
CAD    2.0 -0.015625   ;2*ordinate – 1/64
AAC    0xFFFF000       ;(2*ordinate – 1/64)^2  - 1/4096  = parabola
AMC    0 0x400         ;parabola*(declining magnitude function) = correction*2
SLOGB 0x400            ;save correction*2; get log
CMA    0.5 0x400       ;correction + log
```

A better correction factor reduces the error to 0.0000035, using 14 ticks.

```
CM     1.0 0x418       ;get the log field extract data. Input assumed in B
ANDC   0x03C0000       ;isolate the 16 segment numbers
SAAC   0 0x400         ;save segment # to temp location; square it
CAD    -4.260136 0     ;multiply square by –4.260136
CMA    1.998240 0x400  ;add 1.99824*segment number
1AC    -0.3336925      ;-4.26*X^2+1.998*X-0.334 where X=segment#/64
```

The greatest log accuracy ignores the built-in log function. Field Extract is used to split the input into 2 parts (shift count and shifted part), and each part is worked on separately.

```
                                                  n

                                                   segment
                                                  here keeps coeff’s above <8
AAC    0xFFFC000       ;16*ordinate^2 -ordinate/4   (aY^2+bY+c-c)
AMC    0 0x400         ;A times temp = correction*2
SLOGB 0x400            ;save correction*2; make log
CMA    0.5 0x400       ;correction + log
```

Alesis Semiconductor
12555 Jefferson Blvd., Suite 285
Los Angeles, CA 90066
Phone (310) 301-0780 Fax (310) 306-1551 www.alesis-semi.com

```
CM     1.0 0x418    ;field extract. The input is assumed to be in the B register
SCA    1.0 0x400    ;save extracted value to temporary location
ANDC   0x07FFFFF    ;isolate shifted part. Range 0.25 to 0.5
CAD    2.666666 –1.0  ;change range to -1/3 to 1/3 for fast convergence
X1AC   0            ;A->B
C      0.09090909   ;1/11 is last term in power series expansion
BAC    -0.1         ;A*B - 0.1
BAC    0.11111111   ;
```

……… .   ;   The series is $\sum_{k=1}^{11}(-1)^{k+1}x^k/k$

(Intermediate steps not shown)

```
BAC    1.0          ;A*B + 1.0  First term of the series
BAC    -0.28768207 ;Multiply finishes the series, -ln(4/3) corrects range change
SAC    1.0 0x401    ;save power expansion to 2nd temporary location
CM     0.5 0x400    ;restore field extract and shift right by 1
ANDC   0x7C00000    ;mask what were the top 5 bits of the field extract
1AC    -6.75        ;change range.
CMA    0.360674 0x401 ;Scaling by 1/ln(16) here more exact than in expansion.
```

This takes 22 ticks.

For Exponent, 16 is raised to the value in the B register. The most significant 6 bits (sign, integer, 2 fraction bits) determine how much the lookup table output will be shifted. The next 4 bits index the lookup table, providing reference point and slope for 16 segments. The remaining bits are multiplied by the (shifted) slope and added to the (shifted) reference point. The reference points are as exact as can be represented in the 1k. The errors are all positive (the result is larger than the true exponential) and range from 0 to 0.024% of the result. Thus, accuracy may be improved to 0.012% in a single tick by multiplying the result by 0.99988.

The accuracy may be improved to 0.00007% (0.7ppm) by adding a quadratic correction -3.927942*x*(x-0.015625) where x is the value of the "remaining bits" (segment). This takes 7 ticks, or 6 if more roundoff error is tolerable.

```
X1AC    0               ;A->B   the input is assumed to be in the accumulator
ANDC    0x003FFFF       ;isolate the segment
SCA     1.0 0x400       ;save segment to temporary location
CAD     4.0 –0.0625     ;(A-1/64)*4.  This scaling improves accuracy.
AMC     0 0x400         ;make a positive parabola, zero at the segment endpoints.
SEXPB   0x00            ;save parabola to temp and do the exponent
CMA     -0.981991 0x400 ;A-0.981991*temp.   Add parabolic correction
```

One more tick would allow a cubic correction to be added to the preceding. Its value is -3.629247365*x*(x-0.015625)*(x-0.0078125) and the result is limited by roundoff.

Another method to achieve exponentiation accuracy limited by roundoff is to use $16^{x+y}=16^x16^y$, with 16^x exact and 16^y a power series expansion which converges rapidly because y is small. This takes 13 ticks.

```
SCA    1.0 0x400    ;A->temp   the input is assumed to be in the accumulator
ANDC   0xFFC0000    ;isolate the bits that can produce an exact result
X1AC   0            ;A->B
EXPB                ;16^B,  exact result for leading 10 bits
SCA    1.0 0x401    ;save exact portion to 2nd temporary location
CMB    -1.0 0x401   ;temp1-B          this is the 18 LSBs
X1AC   0            ;A->B
C      2.46224105   ;(ln(16))^4/24 -> A
BAC    3.55226296   ;A*B+(ln(16))^3/6
BAC    3.84362411   ;A*B+(ln(16))^2/2
BAC    2.77258872   ;A*B+ln(16)
BAC    1.0          ;A*B+1            1+z+z^2/2!+z^3/3!+z^4/4! Where z=y*ln(16)
AMC    0 0x401      ;A*temp2
```

To get the highest possible accuracy (without going to extreme efforts such as representing a number by more than one storage location) the programmer must use the fact that the multiplicand is 28 bits but the multiplier is only 22. Thus small numbers whose accuracy must be preserved should not be used as a multiplier. Little attention has been paid to this consideration in this paper; improvements in roundoff performance could be made at the expense of additional execution time.

The Log Field Extract address ($418)

Log field extract (FX[27:0]) is created from the B register (B[27:0]).

B[27]   discard

C[4:0] = count of leading zeroes in B[26:0]

FX[22:0] = B[26:0] << C[4:0]-4
    in other words, shift B left to get a leading 1 if possible.

FX[27:23] = C[4:0] xor 11111b
    in other words, invert each bit of C.

This is called log field extract because extracting these fields from the B register yields the numbers from which the log is approximated. Every factor of two (FX[27:23]) is divided into 16 segments (FX[21:18]). FX[21:18] indexes a 16-entry table, the entries in which are pairs of endpoints and slopes. The end of each segment is as exact as possible; the slopes provide a linear approximation between endpoints.

Exponent from the B register is created by:
B[27:22] shift select
B[21:18] segment select
B[17:0] ordinate on segment.
Here again, segment ends are "exact".