SIGN IDEAS

pd: 26-10-89

E p. 225-226 (2)

EDITED BY CHARLES H SMALL

DSP-μP routine computes magnitude

Amarnath Palacherla
Microchip Technology, Chandler, AZ

The 320C10 DSP μP program in **Listing 1** computes the magnitude of a complex number, X+jY. Finding the sum of the squares of the real (X) and imaginary (Y) parts of a complex number is no problem with a 320C10 because the IC performs a multiplication in one instruction cycle (122 nsec at 32.8 MHz).

But finding the square root of the sum of the squares is not trivial. If you use common iterative methods, such as the Newton-Raphson method, the 320C10 will take 350 or more clock cycles for this calculation (42 μsec).

The method embodied in the **listing** takes only about 30 clock cycles (3.7 μsec) and uses the following approximation:

$$Z = \sqrt{(X^2 + Y^2)} \approx MAX\,(X,Y) + K*MIN\,(X,Y).$$

The MAX(X,Y) function examines the real and imaginary parts of a complex number and selects the part having the largest *absolute* value. MIN(X,Y) similarly looks for the absolute minimum part of each complex number in your sample set. You can choose any value for the constant K between 0.25 and 0.31. Recommended values for K are 0.267304 for exact extimated mean, and 0.300585 for minimum variance.

Fig 1 graphs the percentage error between the actual value of the complex number's magnitude and the algorithm's estimate for phase angles in the first octant. Note that the maximum error is about 11% for K=0.267304, but the mean error is 0. The second routine in **Listing 1** modifies the algorithm slightly to lessen the estimate's error for phase angles greater than 41°. In pseudocode, the modified algorithm is

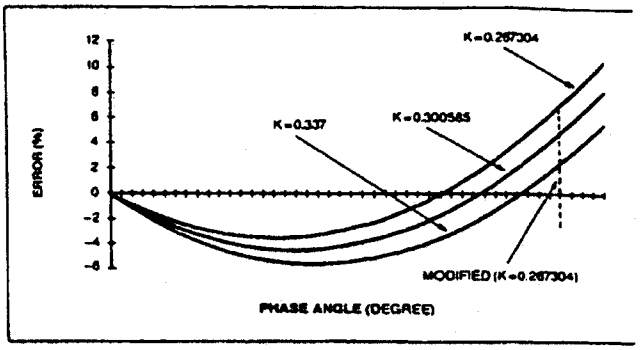

Fig 1—This plot graphs the percentage error between the actual magnitude of a complex number and the approximating algorithm's estimate of the magnitude against the phase angle of the complex number over just the first octant. The graph does not show that for K=0.2867304, the mean error equals zero.

```
x1 = MAX [ ABS(x), ABS(y) ] ;
y1 = MIN [ ABS(x), ABS(y) ] ;
IF ( x1 == y1 ) THEN Z = SQRT(2)*x1 ; RETURN ;
z = x1 + K*y1 ;
IF [ y1 > x1*TAN(41) ] THEN z = z + 1.09865; RETURN
```

This algorithm yields a maximum error of less than 6%. Of course, with this modification, the mean error is no longer zero. EOE

References

1. Onoe, Morio, "Fast Amplitude Approximation Yielding Either Exact Mean or Minimum Deviation for Quatrature Pairs," Proceedings of the IEEE, July 1972.
2. Knuth, D E, *The Art Of Computer Programming: Volume II*, Addison-Wesley Publishing Co, Reading, MA.

To Vote For This Design, Circle No 750

LISTING 1—COMPLEX-NUMBER MAGNITUDE ROUTINE

```
*****************************************************
*       Compute Magnitude , Z = SQRT( X*X + Y*Y )   *
*****************************************************
*       Assume X & Y are in Locations 0 & 1 Respectively
*
x       equ     0
y       equ     1
max     equ     2
min     equ     3
z       equ     4
```

LISTING 1—COMPLEX-NUMBER MAGNITUDE ROUTINE—continued

```
K       equ     5
theta   equ     6

const   DATA    8759            /***  0.267304*32768   ***/
angle   DATA    -28485          /*** -Tan(41)*32768     ***/
root2   DATA    13573           /***  sqrt(2)-1   **/

mag     lac     y
        abs
        sacl    min
        lac     x
        abs
        sacl    max
        sub     min
        bz      case1
        bgez    comp
        lar     AR0,max
        lac     min
        sacl    max             /***    max =  y   ***/
        sar     AR0,min         /***    min =  x   ***/
comp    lack    const
        tblr    K               /*** Load Constant K from ROM
***/
        lac     max,15
        lt      K
        mpy     min
        apac
        sach    z,1
*****************************************************
*         This Part of Code Necessary only if       *
*               Modified Method is Used             *
*****************************************************
        lack    angle
        tblr    theta
        lac     min,15
        lt      theta
        mpy     max
        apac
        bgez     case2          /** branch if min > max*Tan(41)
***/
        ret
case2   lac     z,15
        lt      z
        mpyk    3233
        apac                    /** z = z*1.0985 ***/
        sach    z,1
        ret
case1   lack    root2          /**  if ( x == y ) z = x*1.414
**/
        tblr    min
        lt      min
        mpy     max
        lac     max,15
        apac
        sach    z,1
```