

Using the CORDIC Algorithm for Fast Trigonometric Operations in Hardware

GIRD Systems Internal Seminar
November 25, 2009

What is the CORDIC algorithm?

- COordinate Rotation Digital Computer
- Fast way to compute:
 - Magnitude and phase of a complex value
 - Trigonometric functions such as sin and cos
- Originally developed in 1959 for real-time aircraft navigation computer
- Computes values based on a series of rotations

Why use the CORDIC algorithm?

- Major reason – only multiplications required are bit shifts (simple in hardware)
- Multiplier units are resource hogs
- Iterative process – more iterations yields better accuracy
 - Easy to trade off execution speed vs. accuracy
 - Design to the constraints of the system

When to use CORDIC?

- Any application that requires rapid calculation of trigonometric functions
 - Demodulation of successive carriers in OFDM symbols
 - Rectangular-to-polar conversion operations
- Especially useful for FPGA implementations with space constraints
 - CORDIC blocks are available from many manufacturers such as Xilinx

Other Methods of Computation

- Built-in functions for DSPs or blocks for FPGAs
- Look up table with interpolation
 - Symmetry and phase shifts can be exploited to only store values between 0 and pi/2
 - $\sin(x) = \cos(x-\pi/2)$
- Taylor series expansion

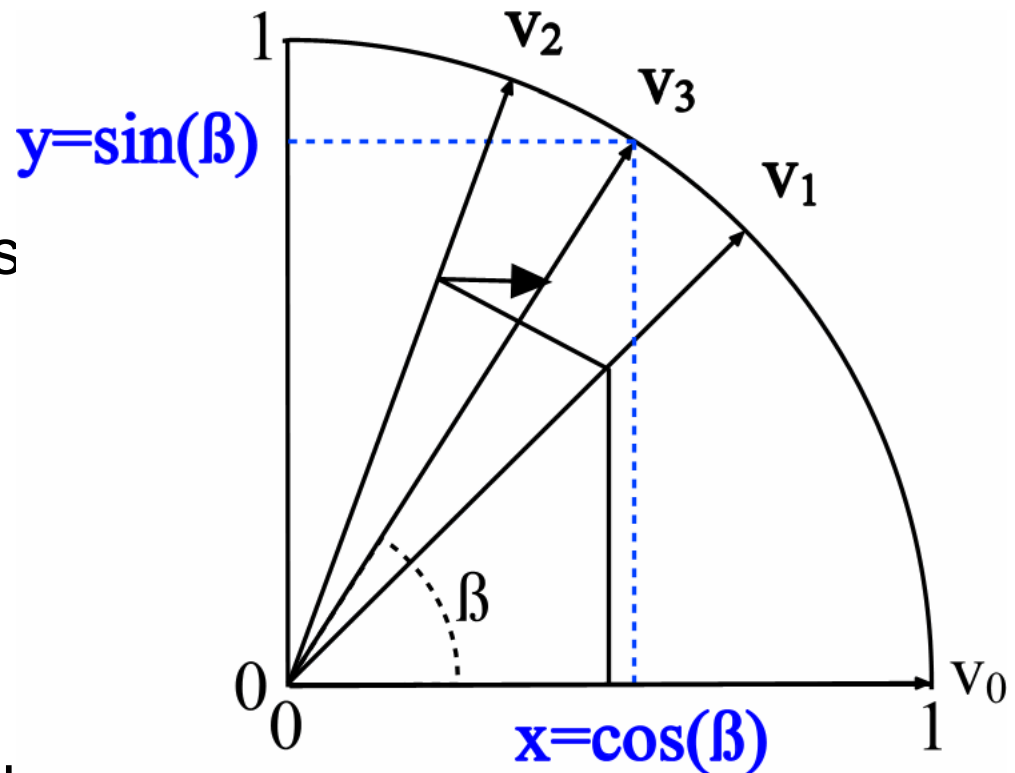
$$\sin x = \sum_{n=0}^{\infty} \frac{(-1)^n}{(2n+1)!} x^{2n+1} = x - \frac{x^3}{3!} + \frac{x^5}{5!} - \dots \text{ for all } x$$

$$\cos x = \sum_{n=0}^{\infty} \frac{(-1)^n}{(2n)!} x^{2n} = 1 - \frac{x^2}{2!} + \frac{x^4}{4!} - \dots \text{ for all } x$$

CORDIC Algorithm

- Apply a series of rotations in the positive or negative direction
- Rotation is in increments of $\arctan(2^{-L})$:

$L = 0$	45°
$L = 1$	26.565°
$L = 2$	14.036°
$L = 3$	7.125°
- Values can be precomputed and stored



CORDIC Algorithm

- Mathematical view of the approach:

Start with a complex value: $C = I_c + jQ_c$

And a rotational value: $R = I_r + jQ_r$

To *add* a rotation to C: $C' = C \cdot R$

To *subtract* a rotation: $C' = C \cdot R^*$

Adding a Rotation

- For the CORDIC algorithm $R = 1 + jK$ where $K = 2^{-L}$
- Breaking out the equation for adding an equation:

$$C' = C \cdot R$$

$$I_c' = I_c I_r - Q_c Q_r = I_c - (Q_c \gg L)$$

$$Q_c' = Q_c I_r + I_c Q_r = Q_c + (I_c \gg L)$$

- Special case: Adding 90°

$$R = 0 + j1$$

$$\begin{aligned} I_c' &= -Q_c \\ Q_c' &= I_c \end{aligned}$$

(negate Q_c and swap)

Subtracting a Rotation

- For the CORDIC algorithm $R^* = 1 - jK$ where $K = 2^{-L}$
- Breaking out the equation for adding an equation:

$$C' = C \cdot R^*$$

$$I_c' = I_c I_r + Q_c Q_r = I_c + (Q_c \gg L)$$

$$Q_c' = Q_c I_r - I_c Q_r = Q_c - (I_c \gg L)$$

- Special case: Subtracting 90°

$$R = 0 - j1$$

$$\begin{aligned} I_c' &= Q_c \\ Q_c' &= -I_c \end{aligned}$$

(negate I_c and swap)

Computing sin and cos

- Step 1: Start at $I_c = 1$ and $Q_c = 0$
- Step 2: Rotate in increments of 90° to move into the correct quadrant
- Step 3: Iterate over L , starting at $L=0$, until the desired accuracy is achieved. Add or subtract rotations so that the sum converges on the desired angle
- $I_c = \cos(\beta)$ $Q_c = \sin(\beta)$

Important Caveat

- Since the magnitude of the rotation vector is not unity magnitude, we need to divide our results by a term called the CORDIC gain.
 - CORDIC gain = $\text{prod}(\text{abs}(1+j/2^L), 0, L)$
 - This term can be precomputed and stored
- | | | |
|---------|----------------|------------------|
| $L = 0$ | $C_g = 1.4142$ | $1/C_g = 0.7071$ |
| $L = 1$ | $C_g = 1.5811$ | $1/C_g = 0.6325$ |
| $L = 2$ | $C_g = 1.6298$ | $1/C_g = 0.6136$ |
- C_g converges to approximately 1.647

Computing sin and cos – A Simple Example

- Find the sin and cos of 108.435°
- Start at 0° $C = 1 + j0$
- Add 90° $C = 0 + j1$
- Add 45° ($L=0, \beta=135^\circ$) $C = (0-1) + j(1+0) = -1+j$
- Subtract 26.565° ($L=1, \beta=108.435^\circ$)
 $C = (-1 + 1/2) + j(1-(-1)/2) = -0.5 + j1.5$
- Divide by the CORDIC gain for $L = 1$, $1/C_g = 0.6325$
- $\sin(108.435^\circ) = 1.5(0.6325) = 0.9487$
 $\cos(108.435^\circ) = -0.5(0.6325) = -0.3162$

Computing sin and cos – Another Example

- Find the sin and cos of 292.2°
- Start at 270° $C = 0-j$
- Add 45° ($L=0, \beta=315^\circ$) $C = 1-j$
- Subtract 26.565° ($L=1, \beta=288.435^\circ$) $C = 0.5 - j1.5$
- Add 14.036° ($L=2, \beta=302.471^\circ$) $C = 0.875 - j1.375$
- Subtract 7.125° ($L=3, \beta=295.346^\circ$) $C = 0.703 - j1.484$
- Subtract 3.576° ($L=4, \beta=291.767^\circ$) $C = 0.610 - j1.528$
- Add 1.790° ($L=5, \beta=293.560^\circ$) $C = 0.658 - j1.509$
- Subtract 0.895° ($L=6, \beta=292.665^\circ$) $C = 0.635 - j1.520$
- Add 0.448° ($L=7, \beta=292.217^\circ$) $C = 0.623 - j1.524$

Computing sin and cos (cont.)

- We are pretty close to 292.2° , so we will stop here
- $C = 0.623 - j1.524$
- Divide by CORDIC gain
 $I_c = \cos(292.217^\circ) = 0.623/1.64674 = 0.3781$
 $Q_c = \sin(292.217^\circ) = -1.524/1.64674 = -0.9258$
- Compare to the actual values:
 $\cos(292.2^\circ) = 0.3778$
 $\sin(292.2^\circ) = -0.9259$
- Result will improve as more iterations are used

Matlab Code for Previous Example

```
beta = 270; phi = 292.2; C = 0-j; %initialize terms
num_iter = 7; angles = 180/pi.*atan(2.0.^[0:-1:-1*num_iter]); %precompute CORDIC angles

for N = 1:length(angles)
    L = N-1 % print out L
    if (beta < phi) % add a rotation if beta < phi
        beta = beta + angles(N)
        C = C*(1+(j/(2^(N-1))))
    elseif (beta > phi) % subtract rotation if beta > phi
        beta = beta - angles(N)
        C = C*(1-(j/(2^(N-1))))
    end
end

Cg = prod(abs(1+j./2.^[0:length(angles)-1])) %compute the CORDIC gain
C = C/Cg %cos(beta) = real portion, sin(beta) = imaginary portion
```

Computing Magnitude and Phase

- To compute magnitude and phase, we work backwards – start with a complex value and rotate until the Q component is 0 and the I component is positive (i.e. 0° angle)
- Like sin and cos, magnitude and phase are computed simultaneously
- Again, we have to factor in the CORDIC gain when computing the magnitude

Computing Magnitude and Phase – A Simple Example

- Use CORDIC to estimate the mag. and phase of $-1+j3$
- Since this number is in Quadrant II, we need to subtract 90° to get back to Quadrant I: $C = 3+j1$
- Now since I and Q are positive, we subtract the 45° term: $C = (3+1 \gg 0) + j(1-3 \gg 0) = 4-j2$
- Q is now negative, so we add the next phase term (26.57°): $C = (4-(-2) \gg 1) + j(-2+4 \gg 1) = 5+j0$
- Now we are located on the real axis. The phase is the inverse of the sum of the rotations we applied:
 $-1*(-90-45+26.57) = 108.43^\circ$

Computing Magnitude and Phase – A Simple Example (cont.)

- To compute the magnitude, we take the real result and divide by the CORDIC gain
- Real result = 5
- CORDIC gain for $L = 1$ is 1.5811
- $5/1.5811 = 3.162$
- Compare to the actual result

$$\text{abs}(1+j*3) = \text{sqrt}(10) = 3.162$$

Computing Magnitude and Phase – Another Example

- Find the magnitude and phase for $-3.84-j2.51$
- Add 180° to move to Quadrant I : $C = 3.84+j2.51$
- Subtract 45° : $C = 3.84+(2.51 \gg 0) + j(2.51-3.84 \gg 0)$
 $C = 6.35 - j1.33$
- Add 26.565° : $C=6.35-1.33 \gg 1+j(1.33+6.35 \gg 1)$
 $C=7.02+j1.85$
- Subtract 14.036° : $C=7.02+1.85 \gg 2+j(1.85-7.02 \gg 2)$
 $C=7.48+j0.09$
- Q term is almost zero, so we will use this approximation
- Phase = $-1*(180-45+26.565-14.036) = -147.529^\circ$
- Magnitude = $7.476/1.630 = 4.5872$
- Compare to actual answer : $4.5876 @ -146.83^\circ$

Summary

- CORDIC is a fast way to compute trig functions and phase/magnitude in hardware
- Almost no multiplication or division required other than bit shifts
- Iterative process – more iterations helps zero in on a more correct result
- CORDIC functions are available for FPGAs and the algorithm can be incorporated into other devices such as fixed point DSPs
- Good for resource constrained platforms, but computing values directly is more accurate if hardware is able to meet speed requirements