

25/03/2008 (1)

CEG5010: The CORDIC Algorithm

25/03/2008 (2)

Introduction

- COordinate Rotation Digital Computer
- Efficient method to compute \sin , \cos , \tan , \sin^{-1} , \cos^{-1} , \tan^{-1} , multiplication, division, $\sqrt{\quad}$, \sinh , \cosh , \tanh
 - Only uses shifts, additions and a very small lookup table

25/03/2008 (3)

Rotations

Rotating $[x \ y]$ by ϕ

$$\begin{aligned}x' &= x \cos(\phi) - y \sin(\phi) \\y' &= y \cos(\phi) + x \sin(\phi).\end{aligned}$$

Rearranging

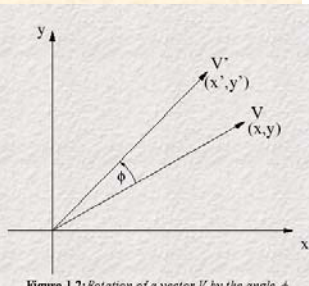


Figure 1.2: Rotation of a vector V by the angle ϕ

$$\begin{aligned}x' &= \cos(\phi)(x - y \tan(\phi)) \\y' &= \cos(\phi)(y + x \tan(\phi)).\end{aligned}$$

25/03/2008 (4)

Key idea

$$\begin{aligned}x' &= \cos(\phi)(x - y \tan(\phi)) \\y' &= \cos(\phi)(y + x \tan(\phi)).\end{aligned}$$

Can compute rotation ϕ in steps where each step is of size

$$\tan(\phi) = \pm 2^{-i}.$$

25/03/2008 (5)

Iterative rotations

$$\begin{aligned}x_{i+1} &= K_i(x_i - (y_i d_i 2^{-i})) \\ y_{i+1} &= K_i(y_i + (x_i d_i 2^{-i})).\end{aligned}$$

where $d_i = \pm 1$ and $K_i = \cos(\tan^{-1} 2^{-i})$

Choose d_i so that after n iterations the rotated angle is ϕ

25/03/2008 (6)

K_i values

$$\cos(\tan^{-1} 2^{-i}) = 1/\sqrt{(1 + 2^{-2i})}.$$

$$K = \prod_{i=1}^n \frac{1}{\sqrt{(1 + 2^{-2i})}},$$

As $n \rightarrow \infty$, $K \rightarrow 0.6073$ (constant factor which needs to be corrected for)

Actually it's easier to omit it and fix it later!

25/03/2008 (7)

d_i decision (rotation mode)

Z_i is introduced to keep track of the angle that has been rotated ($z_0 = \phi$)

$$x_{i+1} = x_i - (y_i d_i 2^{-i})$$

$$y_{i+1} = y_i + (x_i d_i 2^{-i})$$

$$z_{i+1} = z_i - d_i \tan^{-1}(2^{-i})$$

$$d_i = \begin{cases} -1 & \text{if } z_i < 0 \\ +1 & \text{otherwise} \end{cases}$$

Notice we dropped the K! Rotated value is hence (Kx_n, Ky_n)

25/03/2008 (8)

After n iterations

$$x_n = \frac{1}{K}(x_0 \cos(z_0) - y_0 \sin(z_0))$$

$$y_n = \frac{1}{K}(y_0 \cos(z_0) + x_0 \sin(z_0))$$

$$z_n \approx 0.$$

Question: What is the procedure to compute sin and cos?

25/03/2008 (9)

Computing $\sin(a)$ and $\cos(a)$

1. Initialize $(x,y,z)=(1,0,a)$
2. Iterate through cordic
3. $\cos(a)=Kx$ and $\sin(a)=Ky$

An easier way for this example is to change step 1 to $(x,y,z)=(k,0,a)$

25/03/2008 (10)

Sin(75)

```

0: xi=1.000000 yi=0.000000 zi=1.308997 k=1.000000 kx=1.000000 ky=0.000000
1: xi=1.000000 yi=1.000000 zi=0.523599 k=0.707107 kx=0.707107 ky=0.707107
2: xi=0.500000 yi=1.500000 zi=0.059951 k=0.632456 kx=0.316228 ky=0.948683
3: xi=0.125000 yi=1.625000 zi=-0.185027 k=0.613572 kx=0.076696 ky=0.997054
4: xi=0.328125 yi=1.609375 zi=-0.060673 k=0.608834 kx=0.199774 ky=0.979842
5: xi=0.428711 yi=1.588867 zi=0.001746 k=0.607648 kx=0.260505 ky=0.965472
6: xi=0.379059 yi=1.602264 zi=-0.029494 k=0.607352 kx=0.230222 ky=0.973138
7: xi=0.404094 yi=1.596342 zi=-0.013870 k=0.607278 kx=0.245397 ky=0.969423

```

25/03/2008 (11)

d_i decision (vectoring mode)

$$d_i = \begin{cases} +1 & \text{if } y_i < 0 \\ -1 & \text{otherwise.} \end{cases}$$

$$x_n = \frac{1}{K} \sqrt{x_0^2 + y_0^2}$$

$$y_n \approx 0$$

$$z_n = z_0 + \tan^{-1}(y_0/x_0).$$

- y_n minimized use to compute \tan^{-1} and magnitude

25/03/2008 (12)

Linear functions instead of trig

$$x_{i+1} = x_i - 0(y_i d_i 2^{-i}) = x_i$$

$$y_{i+1} = y_i + (x_i d_i 2^{-i})$$

$$z_{i+1} = z_i - d_i 2^{-i}$$

$$d_i = \begin{cases} -1 & \text{if } z_i < 0 \\ +1 & \text{otherwise.} \end{cases}$$

25/03/2008 (13)

After n iterations

$$x_n = x_0$$

$$y_n = y_0 + x_0 z_0$$

$$z_n = 0,$$

No need for K_n correction.

25/03/2008 (14)

Division

$$d_i = \begin{cases} +1 & \text{if } y_i < 0 \\ -1 & \text{otherwise.} \end{cases}$$

$$x_n = x_0$$

$$y_n = y_0$$

$$z_n = z_0 - y_0/x_0.$$

No need for K_n correction.

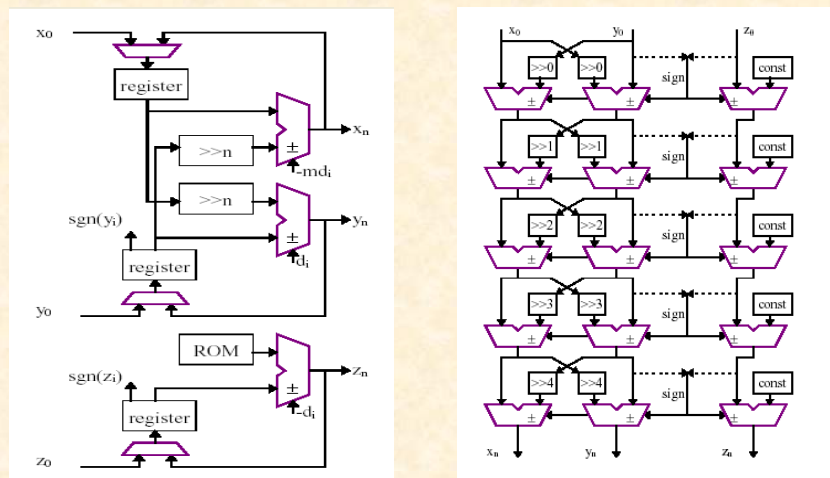
25/03/2008 (15)

Hyperbolic functions

- Similarly, can get cosh and sinh using \tanh^{-1} instead of \tan^{-1}
- Can also get ln and exp easily

25/03/2008 (16)

Andraka's iterative and unrolled cordic structure



25/03/2008 (17)

Implementation

- Can develop generalized cordic processors which can compute many different functions using similar hardware
- Implementations can be bit serial and/or pipelined as well

25/03/2008 (18)

Precision

- Need n iterations for n bits
- Converges for $-99.7 \leq z \leq 99.7$ (sum of all the angles $\tan^{-1}(2^{-i})$, $i = 0 \dots n$)
 - must convert to this range first

25/03/2008 (19)

Conclusion

- CORDIC algorithms are an efficient method to compute many different functions
- Low area, high speed
- Used in calculators, DSPs, math coprocessors and supercomputers.

25/03/2008 (20)

References

- Ray Andraka, "A survey of CORDIC algorithms for FPGAs", FPGA '98. Proceedings of the 1998 ACM/SIGDA sixth international symposium on Field programmable gate arrays, Feb. 22-24, 1998, Monterey, CA. pp191-200 (<http://www.andraka.com/cordic.htm>)

25/03/2008 (21)

Review Question

- Calculate $\sqrt{2/K}$ using the CORDIC algorithm (4 iterations)
- Hint: use vectoring mode