CORDIC Background (3A)

-
-

Young Won Lim
04/09/2012

Please send corrections (or suggestions) to youngwlim@hotmail.com.

This document was produced by using OpenOffice and Octave.

CORDIC Background

1. A survey of CORDIC algorithms for FPGAs, Ray Andraka, www.andraka.com/cordic.htm

Vector Rotation (1)

$$x' = x\cos\phi - y\sin\phi$$
$$y' = y\cos\phi + x\sin\phi$$

$$x' = \cos\phi \cdot [x - y\tan\phi]$$
$$y' = \cos\phi \cdot [y + x\tan\phi]$$

$$x_{i+1} = K_i \cdot \left[x_i - y_i \cdot d_i \cdot 2^{-i}\right]$$
$$y_{i+1} = K_i \cdot \left[y_i + x_i \cdot d_i \cdot 2^{-i}\right]$$

$$K_i = \cos\phi_i = \cos(\tan^{-1}(2^{-i}))$$
$$= \frac{1}{\sqrt{1+2^{-2i}}}$$

$$d_i = \pm 1$$

Restrict rotation angle ➡ $\tan\phi_i = \pm 2^{-i}$

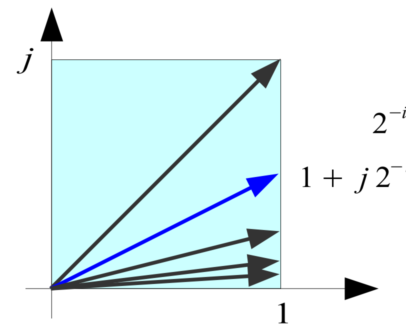
Multiplication ➡ Shift operation

$y \cdot \tan\phi_i$ ➡ $y \cdot 2^{-i}$

$x \cdot \tan\phi_i$ ➡ $x \cdot 2^{-i}$

Regardless of direction ➡ $\cos(\phi_i) = \cos(-\phi_i)$

Allowed Rotation Angles



$2^{-i} = k_i$

$1 + j2^{-i} = 1 + jk_i$

$\tan\phi_i$ ➡ 2^{-i}

$\cos\phi_i$ ➡ $\dfrac{1}{\sqrt{1+2^{-2i}}}$

$K_i \le 1$ (K_i ⬅ $\cos\phi_i$)

Vector Rotation (2)

$$x_{i+1} = K_i \cdot \left[x_i - y_i \cdot d_i \cdot 2^{-i}\right]$$
$$y_{i+1} = K_i \cdot \left[y_i + x_i \cdot d_i \cdot 2^{-i}\right]$$

$$K_i = \cos\phi_i = \cos\left(\tan^{-1}\left(2^{-i}\right)\right)$$
$$= \frac{1}{\sqrt{1 + 2^{-2i}}} \qquad K_i \le 1$$

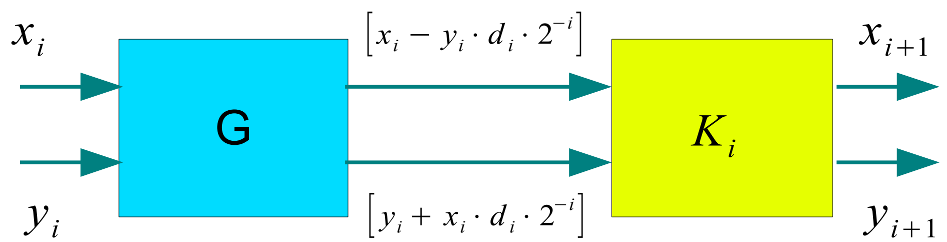
$$d_i = \pm 1$$

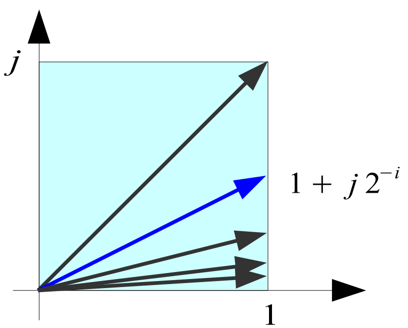
$$x_{i+1}^2 = K_i^2 \cdot \left[x_i^2 + y_i^2 \cdot 2^{-2i} - 2 x_i y_i d_i \cdot 2^{-i}\right]$$
$$y_{i+1}^2 = K_i^2 \cdot \left[y_i^2 + x_i^2 \cdot 2^{-2i} + 2 x_i y_i d_i \cdot 2^{-i}\right]$$

$$x_{i+1}^2 + y_{i+1}^2 = K_i^2 \cdot \left(1 + 2^{-2i}\right) \cdot \left(x_i^2 + y_i^2\right)$$

$$G \cdot K_i = 1$$

$K_i \le 1$ G > 1





$\tan\phi_i \Rightarrow 2^{-i}$

$\cos\phi_i \Rightarrow \dfrac{1}{\sqrt{1 + 2^{-2i}}}$

CORDIC Gain : *growing in magnitude*

$$A_n = \prod_{i=1}^{n} \frac{1}{K_i} = \prod_{i=1}^{n} \sqrt{1 + 2^{-2i}} \rightarrow 1.647$$

Vector Rotation (3)

$$x_{i+1} = K_i \cdot \left[x_i - y_i \cdot d_i \cdot 2^{-i} \right]$$
$$y_{i+1} = K_i \cdot \left[y_i + x_i \cdot d_i \cdot 2^{-i} \right]$$
$$K_i = 1 / \sqrt{1 + 2^{-2i}} \Leftarrow \cos(\phi_i)$$
$$d_i = \pm 1$$

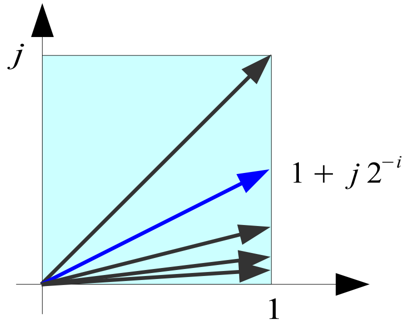
Without Scale Constants K_i

$$x_{i+1} = \left[x_i - y_i \cdot d_i \cdot 2^{-i} \right]$$
$$y_{i+1} = \left[y_i + x_i \cdot d_i \cdot 2^{-i} \right]$$
$$d_i = \pm 1$$

CORDIC Gain : *growing in magnitude*

$$A_n = \prod_{i=1}^{n} \frac{1}{K_i} = \prod_{i=1}^{n} \sqrt{1 + 2^{-2i}} \rightarrow 1.647$$

$$1 / K_i = \sqrt{1 + 2^{-2i}}$$



$$\tan\phi_i \Rightarrow 2^{-i}$$
$$\cos\phi_i \Rightarrow \frac{1}{\sqrt{1 + 2^{-2i}}}$$

For correction

Multiplying K_i*'s as a processing gain*

$$\prod_{i=1}^{n} K_i = \prod_{i=1}^{n} \frac{1}{\sqrt{1 + 2^{-2i}}} \rightarrow 0.6073$$

Angle Accumulator

Rotation Mode

$$z_0 \leftarrow \phi \quad (\textit{desired angle})$$

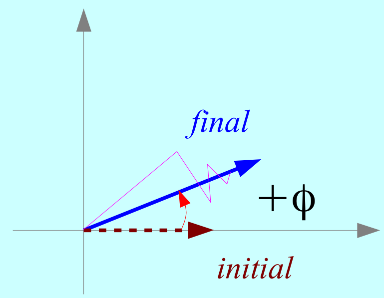
$$z_n \rightarrow 0$$

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

$$d_i = -1 \quad \textit{if} \quad z_i < 0$$

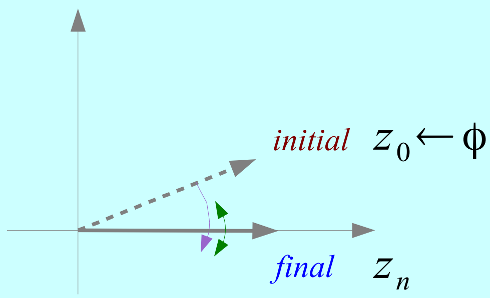
$$d_i = +1 \quad \textit{otherwise}$$

Vector View



Minimize the residual angle

Accumulator View



Subtract *angles at each step*

Vectoring Mode

$$z_0 \leftarrow 0$$

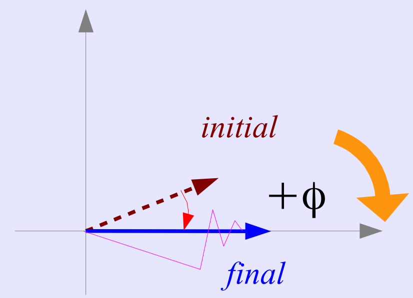
$$z_n \rightarrow z_0 + \tan^{-1}(y_0 / x_0)$$

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

$$d_i = +1 \quad \textit{if} \quad y_i < 0$$

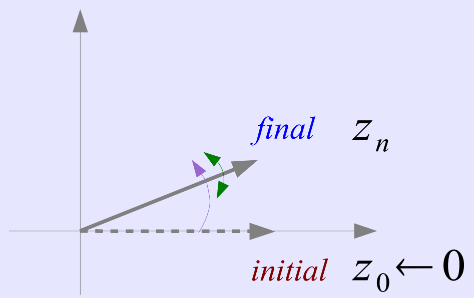
$$d_i = -1 \quad \textit{otherwise}$$

Vector View



Minimize the residual y component

Accumulator View



Add *angles at each step*

Rotation Mode

Rotation Mode

$$z_0 \leftarrow \phi \quad (desired\ angle)$$

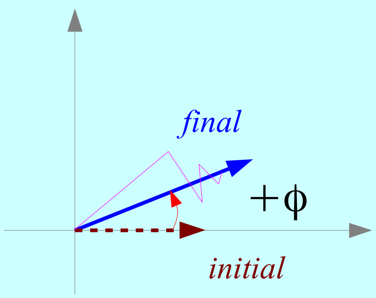
$$z_n \rightarrow 0$$

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

$$d_i = -1 \quad if \quad z_i < 0$$

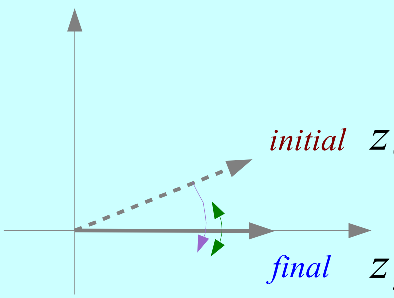
$$d_i = +1 \quad otherwise$$

Vector View



Minimize the residual angle

Accumulator View



***Subtract** angles at each step*

$$x_{i+1} = x_i - y_i \cdot d_i \cdot 2^{-i}$$

$$y_{i+1} = y_i + x_i \cdot d_i \cdot 2^{-i}$$

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

$$d_i = -1 \quad if \quad z_i < 0$$

$$d_i = +1 \quad otherwise$$

$$x_n = A_n \left[x_0 \cos z_0 - y_0 \sin z_0 \right]$$

$$y_n = A_n \left[y_0 \cos z_0 + x_0 \sin z_0 \right]$$

$$z_n = 0$$

$$A_n = \prod_{i=1}^{n} \sqrt{1 + 2^{-2i}}$$

Vectoring Mode

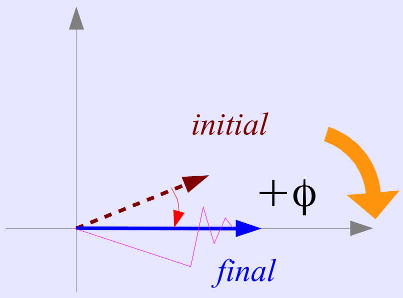
$$z_0 \leftarrow 0$$

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

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

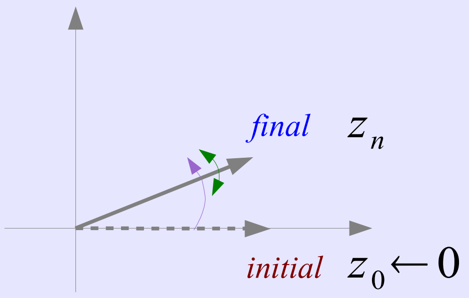
$$d_i = +1 \quad \textit{if} \quad y_i < 0$$
$$d_i = -1 \quad \textit{otherwise}$$

Vector View



Minimize the residual y component

Accumulator View



Add *angles at each step*

$$x_{i+1} = x_i - y_i \cdot d_i \cdot 2^{-i}$$
$$y_{i+1} = y_i + x_i \cdot d_i \cdot 2^{-i}$$
$$z_{i+1} = z_i - d_i \cdot \tan^{-1}(2^{-i})$$

$$d_i = +1 \quad \textit{if} \quad y_i < 0$$
$$d_i = -1 \quad \textit{otherwise}$$

$$x_n = A_n \sqrt{x_0^2 + y_0^2}$$
$$y_n = 0$$
$$z_n = z_0 + \tan^{-1}(y_0 / x_0)$$

$$A_n = \prod_{i=1}^{n} \sqrt{1 + 2^{-2i}}$$

Angle Accumulator – Rotation Mode

Rotation Mode

$$z_0 \leftarrow \phi \quad (input)$$

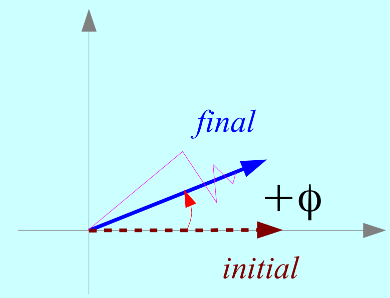
$$z_n \rightarrow 0$$

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

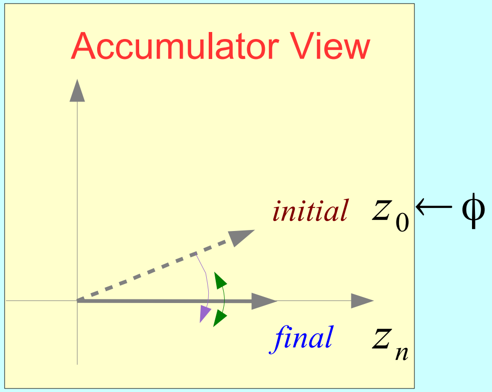
$$d_i = -1 \quad if \quad z_i < 0$$

$$d_i = +1 \quad otherwise$$

Vector View



Minimize the residual angle



***Subtract** angles at each step*

$z_i < 0$

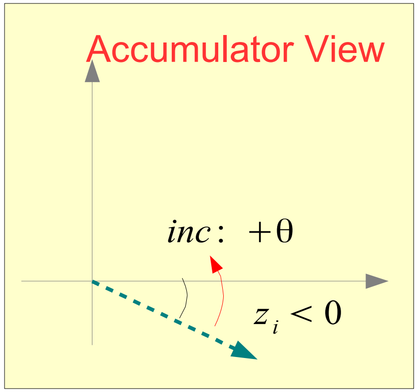
Increase Angle $\quad d_i = -1$

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

$z_i \geq 0$

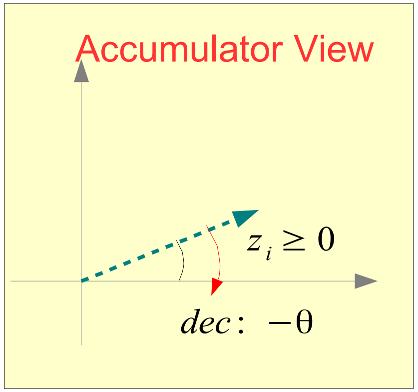
Decrease Angle $\quad d_i = +1$

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



$z_i < 0$

Increase Angle $+\theta$



$z_i \geq 0$

Decreases Angle $-\theta$

Angle Accumulator – Vectoring Mode

Vectoring Mode

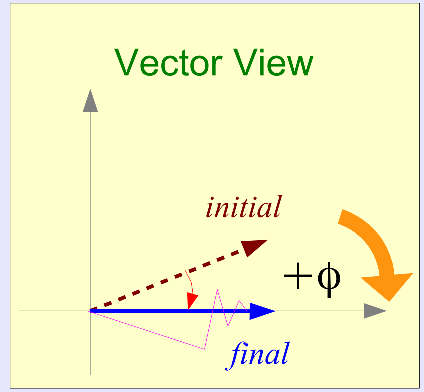
$$z_0 \leftarrow 0$$

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

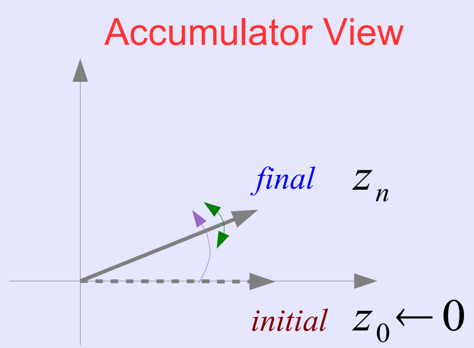
$$z_{i+1} = z_i - d_i \cdot \tan^{-1}(2^{-i})$$

$$d_i = +1 \quad \textit{if} \quad y_i < 0$$

$$d_i = -1 \quad \textit{otherwise}$$



Minimize the residual y component



Add *angles at each step*

$y_i < 0$

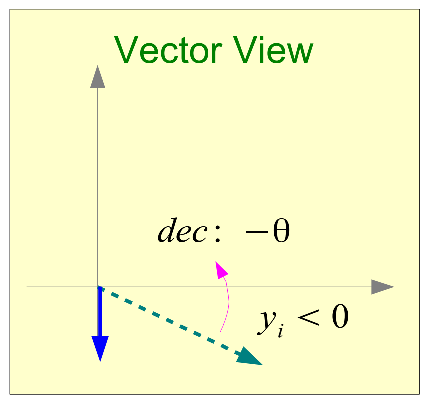
Decrease Angle $d_i = +1$

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

$y_i > 0$

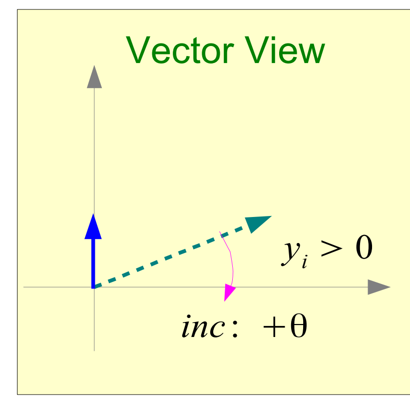
Increase Angle $d_i = -1$

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



$y_i < 0$

Decrease Angle $-\theta$

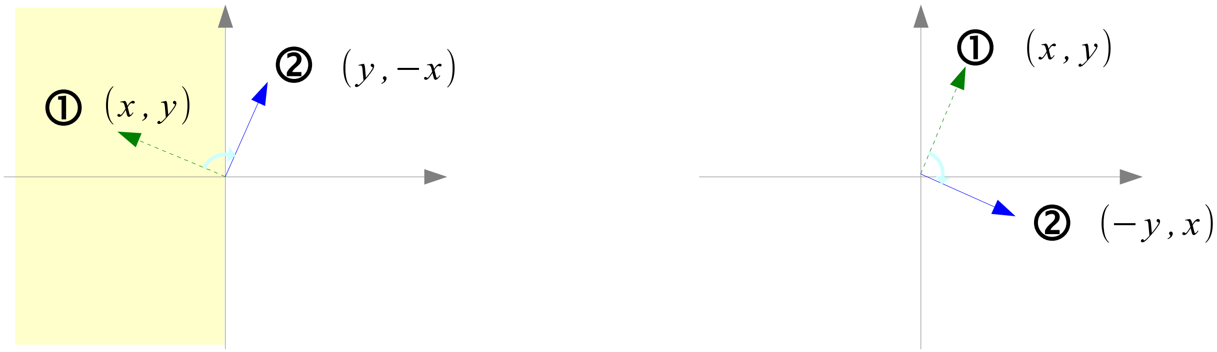


$y_i > 0$

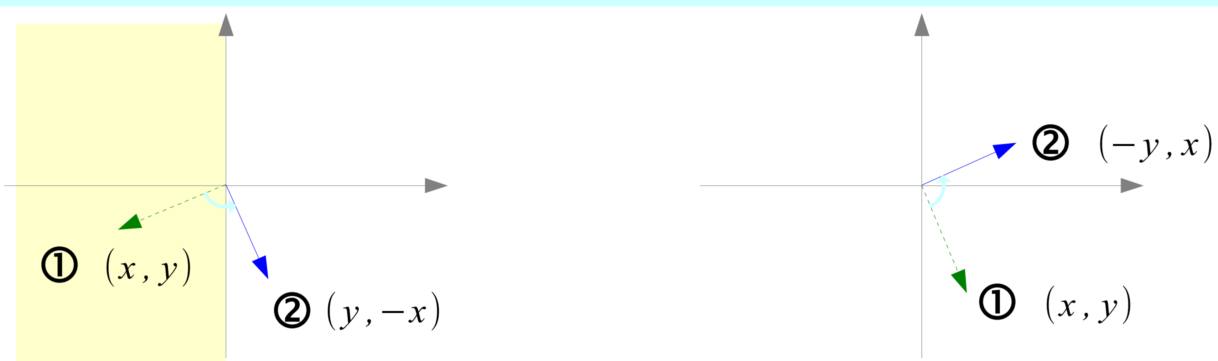
Increases Angle $+\theta$

Initial Rotation $\pm\pi/2$

Positive Phase (y > 0) ➡ *Rotate by –90 degrees*



Negative Phase (y < 0) ➡ *Rotate by +90 degrees*



Resulting Phase ➡ [-90, +90]

$$x' = -d \cdot y$$
$$y' = +d \cdot x$$
$$z' = z + d \cdot \frac{\pi}{2}$$

$$d = +1 \quad if \quad y < 0$$
$$d = -1 \quad otherwise$$

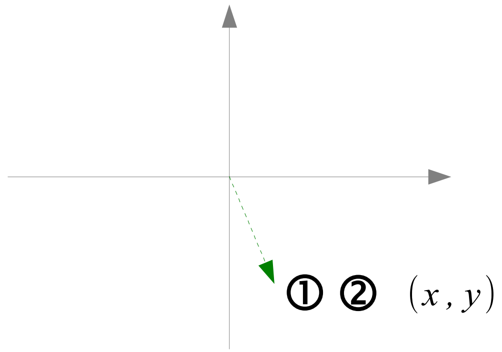
No magnitude change

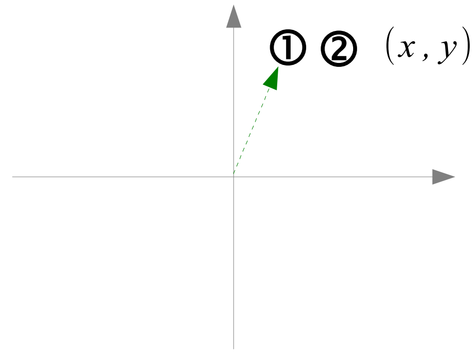
$$x' \leftarrow y$$
$$y' \leftarrow x$$

Consistent

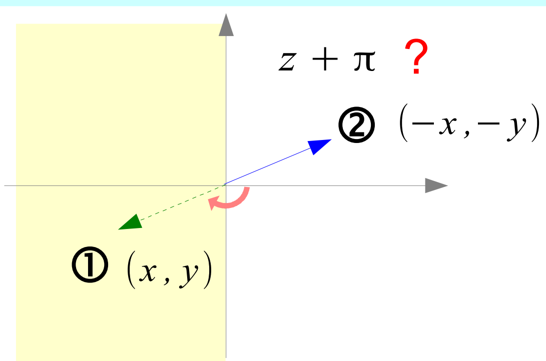
Initial Rotation $0, \ +\pi$

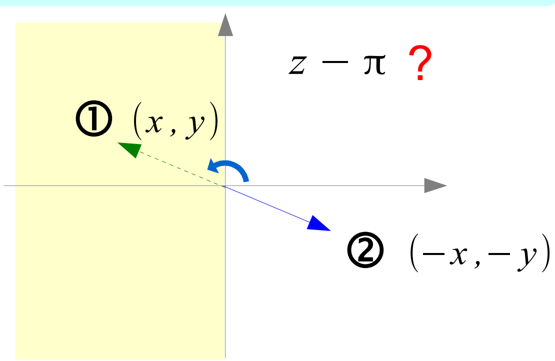
Positive x (x > 0) ➡ *Rotate by 0 degrees*





Negative x (x < 0) ➡ *Rotate by +180 degrees*





Resulting Phase ➡ [-90, +90]

$$x' = +d \cdot x$$
$$y' = +d \cdot y$$
$$z' = z \quad if \;\; d = 1$$
$$z' = \pi - z \quad if \;\; d = -1$$

$$d = -1 \quad if \;\; x < 0$$
$$d = +1 \quad otherwise$$

No magnitude change

$$x' \leftarrow y$$
$$y' \leftarrow x$$

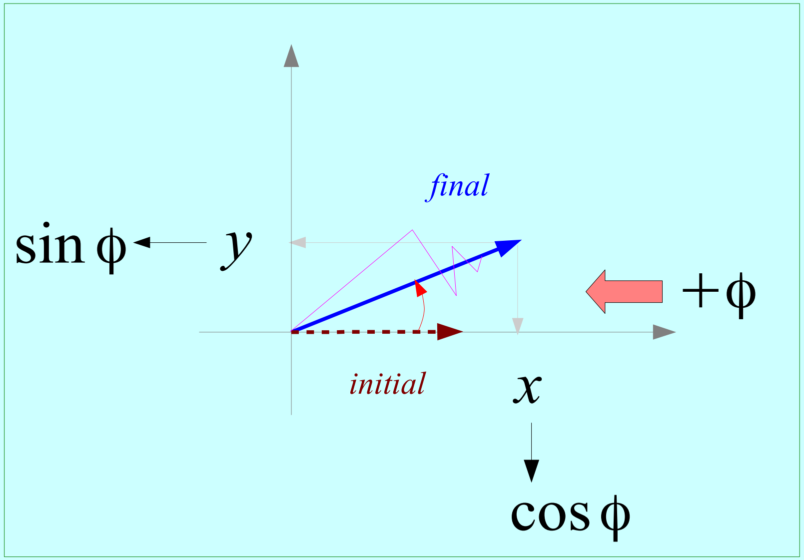
Convenient wiring in FPGA

Application Modes (1)

Rotation Mode

Input angle is given

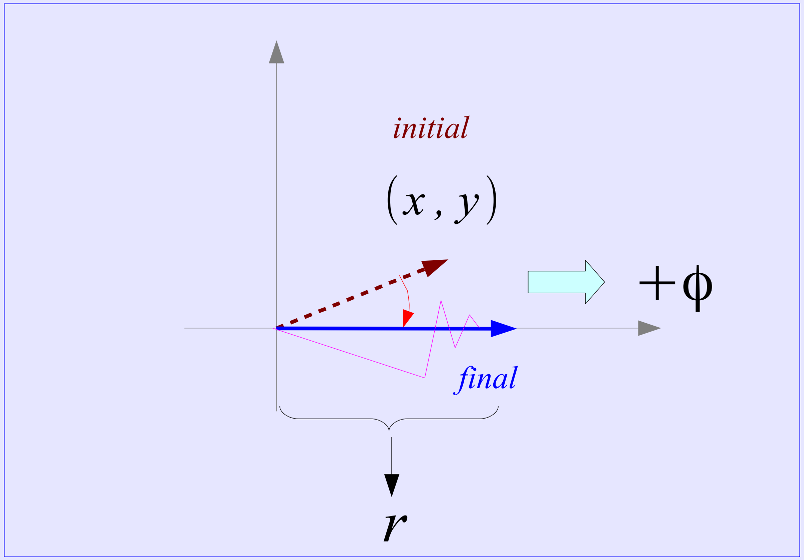
- **sin** and **cos**
- $(r, \theta) \rightarrow (x, y)$
- General vector rotation



Vectoring Mode

Finding the resulting angle

- **tan^{-1}**
- Vector Magnitude
- $(x, y) \rightarrow (r, \theta)$



Application Modes (2)

A. **sin & cos**

B. **(r, θ) → (x, y)**

C. General Vector Rotation

D. **$\tan^{-1}$**

E. Vector Magnitude

F. **(x, y) → (r, θ)**

G. **$\sin^{-1}$**

H. **$\cos^{-1}$**

I. Linear Functions

J. Hyperbolic Functions

A. Sine and Cosine

Rotation Mode

$$z_0 \leftarrow \phi \quad (desired\ angle)$$

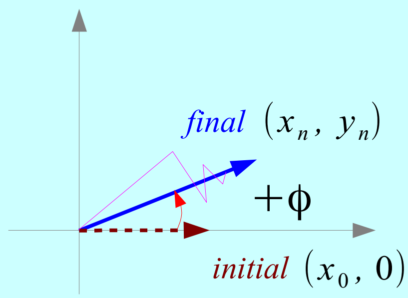
$$z_n \rightarrow 0$$

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

$$d_i = -1 \quad if \quad z_i < 0$$

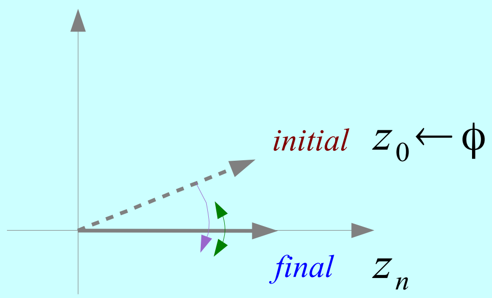
$$d_i = +1 \quad otherwise$$

Vector View



Minimize the residual angle

Accumulator View



***Subtract** angles at each step*

Finding Sine and Cosine

$$(x_0, 0) \rightarrow (x_n, y_n)$$

$$x_n = A_n \cdot x_0 \cos z_0$$

$$y_n = A_n \cdot x_0 \sin z_0$$

CORDIC Gain : *growing in magnitude*

$$A_n = \prod_{i=1}^{n} \frac{1}{K_i} = \prod_{i=1}^{n} \sqrt{1 + 2^{-2i}} \rightarrow 1.647$$

Unscaled Sine and Cosine

$$x_0 \leftarrow \frac{1}{A_n} = 0.6073$$

$$x_n = \cos z_0$$

$$y_n = \sin z_0$$

Modulated Sine and Cosine

$$x_0 \leftarrow \left\{\prod_{i=1}^{n} K_i\right\} \cdot x_0 = 0.6073 \cdot x_0$$

$$x_n = x_0 \cdot \cos z_0$$

$$y_n = x_0 \cdot \sin z_0$$

LUT → a pair of MULT

CORDIC → rotation operations

Single MULT

B. Polar to Rectangular

Rotation Mode

$$z_0 \leftarrow \phi \quad (desired\ angle)$$

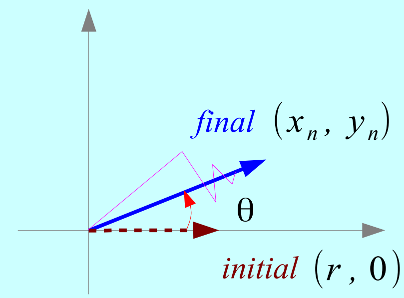
$$z_n \rightarrow 0$$

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

$$d_i = -1 \quad if \quad z_i < 0$$

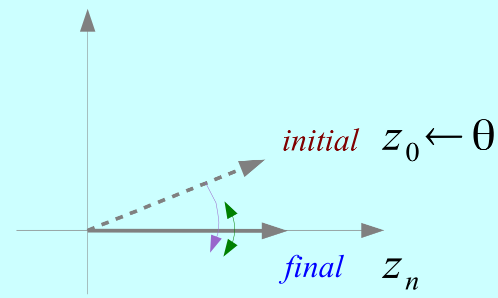
$$d_i = +1 \quad otherwise$$

Vector View



Minimize the residual angle

Accumulator View



***Subtract** angles at each step*

Finding Sine and Cosine

$$(x_0, 0) \rightarrow (x_n, y_n)$$

$$x_n = A_n \cdot x_0 \cos z_0$$

$$y_n = A_n \cdot x_0 \sin z_0$$

$$x_0 \leftarrow r, \quad z_0 \leftarrow \theta$$

$$(r, 0) \rightarrow (x_n, y_n)$$

$$x_n = A_n\, r \cos\theta$$

$$y_n = A_n\, r \sin\theta$$

$$x_0 \leftarrow r \cdot \frac{1}{A_n}, \quad z_0 \leftarrow \theta$$

$$\left(\frac{r}{A_n}, 0\right) \rightarrow (x_n, y_n)$$

$$x_n = r \cos\theta$$

$$y_n = r \sin\theta$$

CORDIC Gain : *growing in magnitude*

$$A_n = \prod_{i=1}^{n} \frac{1}{K_i} = \prod_{i=1}^{n} \sqrt{1 + 2^{-2i}} \rightarrow 1.647$$

Young Won Lim
04/09/2012

Rotation Mode

$$z_0 \leftarrow \phi \quad (desired\ angle)$$

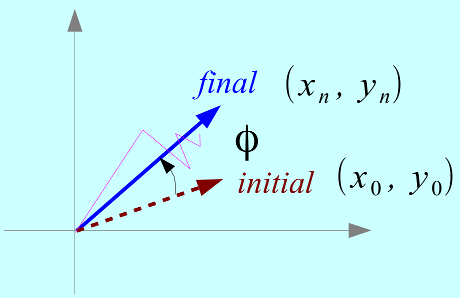
$$z_n \rightarrow 0$$

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

$$d_i = -1 \quad if \quad z_i < 0$$

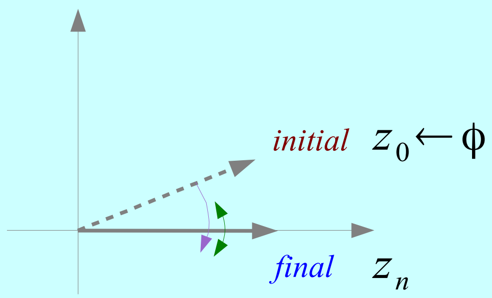
$$d_i = +1 \quad otherwise$$

Vector View



Minimize the residual angle

Accumulator View



***Subtract** angles at each step*

Motion Correction and Control System

$$\begin{bmatrix} x_n \\ y_n \end{bmatrix} = A_n \cdot \begin{bmatrix} \cos z_0 & -\sin z_0 \\ \sin z_0 & \cos z_0 \end{bmatrix} \begin{bmatrix} x_0 \\ y_0 \end{bmatrix}$$

* n-dim rotation

$\rightarrow$ tree architecture

Unscaled Rotation

$$\begin{bmatrix} x_n \\ y_n \end{bmatrix} = A_n \cdot \begin{bmatrix} \cos z_0 & -\sin z_0 \\ \sin z_0 & \cos z_0 \end{bmatrix} \begin{bmatrix} \frac{x_0}{A_n} \\ \frac{y_0}{A_n} \end{bmatrix}$$

A pair of MULT

$$\begin{bmatrix} x_n \\ y_n \end{bmatrix} = \begin{bmatrix} \cos z_0 & -\sin z_0 \\ \sin z_0 & \cos z_0 \end{bmatrix} \begin{bmatrix} x_0 \\ y_0 \end{bmatrix}$$

D. Arctangent

Vectoring Mode

$$z_0 \leftarrow 0$$

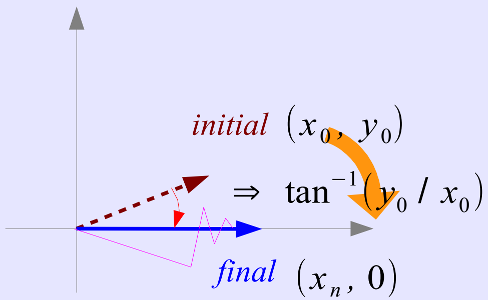
$$z_n \rightarrow z_0 + \tan^{-1}(y_0 / x_0)$$

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

$$d_i = +1 \quad \text{if} \quad y_i < 0$$

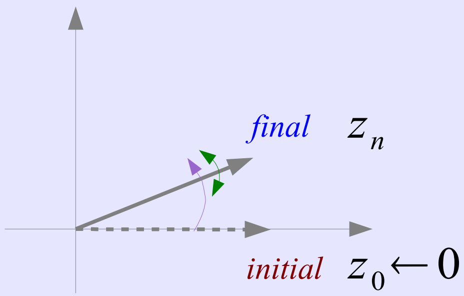
$$d_i = -1 \quad \text{otherwise}$$

Vector View



Minimize the residual y component

Accumulator View



Add *angles at each step*

Input

$$(x_0, y_0) \rightarrow \quad \textit{ratio} \quad \frac{y_0}{x_0}$$

$$(0, y_0) \rightarrow \quad \textit{ratio} \quad \pm\infty$$

Output

Angle Accumulator Value

→ CORDIC gain does not affect

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

E. Vector Magnitude

Vectoring Mode

$$z_0 \leftarrow 0$$

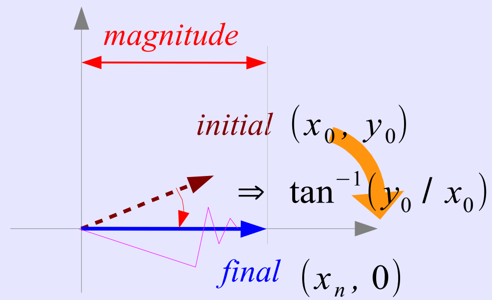
$$z_n \rightarrow z_0 + \tan^{-1}(y_0 / x_0)$$

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

$$d_i = +1 \quad \text{if} \quad y_i < 0$$

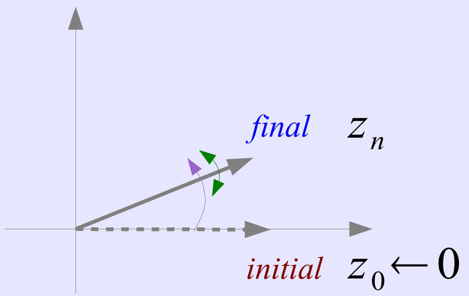
$$d_i = -1 \quad \text{otherwise}$$

Vector View



Minimize the residual y component

Accumulator View



Add *angles at each step*

The magnitude:

- *byproduct of computing arctangent*
- *the result vector is aligned with x-axis*
- *the x component of the result vector*
- *increased by CORDIC gain*
- *can be scaled by the processor gain*
- *one MULT hardware cost*

$$x_n = A_n \sqrt{x_0^2 + y_0^2}$$

The accuracy of the magnitude result

- *Improves by 2 bits for each iteration performed*

F. Cartesian to Polar Transformation

Vectoring Mode

$$z_0 \leftarrow 0$$

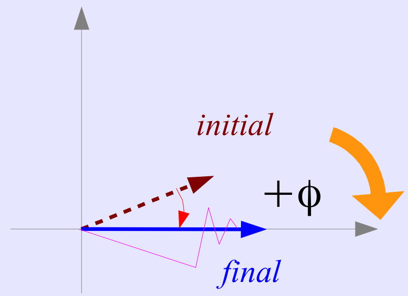
$$z_n \rightarrow z_0 + \tan^{-1}(y_0 / x_0)$$

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

$$d_i = +1 \quad \textit{if} \quad y_i < 0$$

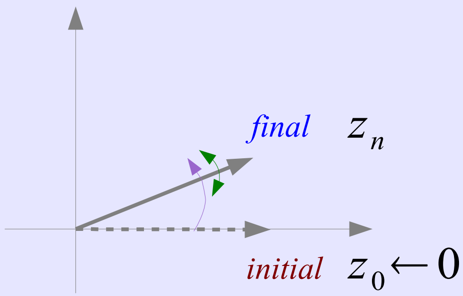
$$d_i = -1 \quad \textit{otherwise}$$

Vector View



Minimize the residual y component

Accumulator View



Add *angles at each step*

input vector (x, y)

magnitude $r = \sqrt{x^2 + y^2}$ ➡ $x_n = A_n \sqrt{x_0^2 + y_0^2}$

phase angle $\phi = \tan^{-1}(y / x)$ ➡ $z_n = z_0 + \tan^{-1}(y_0 / x_0)$

G. ArcSine (1)

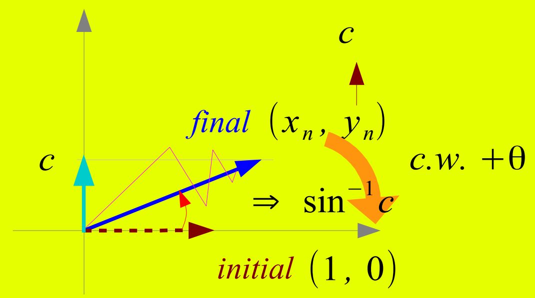
Exploit Vector Mode HW

$$z_n \rightarrow z_0 + \sin^{-1}\left(\frac{c}{A_n \cdot x_0}\right)$$

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

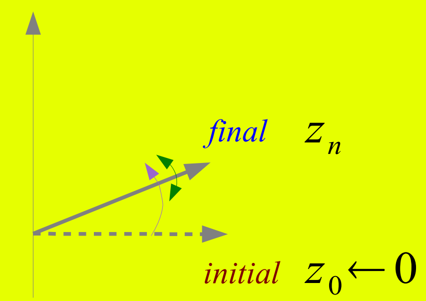
$$d_i = +1 \quad if \quad y_i < c$$
$$d_i = -1 \quad otherwise$$

Vector View



Minimize the residual y component

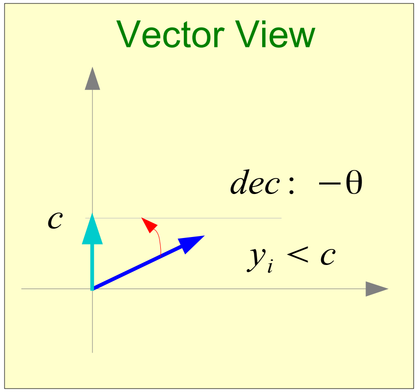
Accumulator View



Add *angles at each step*

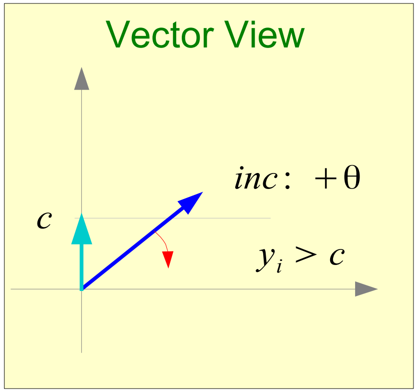
$$x_{i+1} = x_i - y_i \cdot d_i \cdot 2^{-i}$$
$$y_{i+1} = y_i + x_i \cdot d_i \cdot 2^{-i}$$
$$z_{i+1} = z_i - d_i \cdot \tan^{-1}(2^{-i})$$

$$d_i = +1 \quad if \quad y_i < c$$
$$d_i = -1 \quad otherwise$$



$y_i < c$ Dec Angle

Add (–) Angle



$y_i > c$ Inc Angle

Add (+) Angle

Young Won Lim
04/09/2012

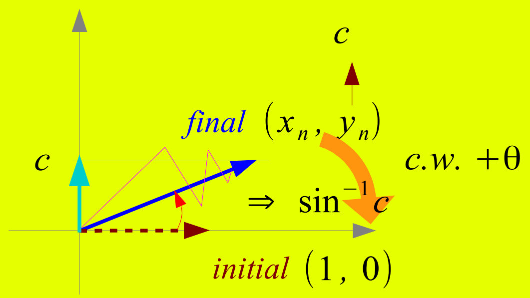
G. ArcSine (2)

Exploit Vector Mode HW

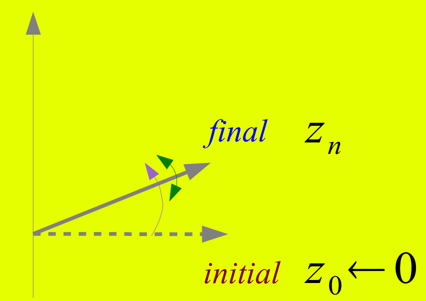
$$z_n \rightarrow z_0 + \sin^{-1}\left(\frac{c}{A_n \cdot x_0}\right)$$

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

$$d_i = +1 \quad \text{if} \quad y_i < c$$
$$d_i = -1 \quad \text{otherwise}$$



Minimize the residual y component



Add *angles at each step*

$$x_{i+1} = x_i - y_i \cdot d_i \cdot 2^{-i}$$
$$y_{i+1} = y_i + x_i \cdot d_i \cdot 2^{-i}$$
$$z_{i+1} = z_i - d_i \cdot \tan^{-1}(2^{-i})$$

$$d_i = +1 \quad \text{if} \quad y_i < c$$
$$d_i = -1 \quad \text{otherwise}$$

$$x_n = \sqrt{(A_n \cdot x_0)^2 - c^2}$$
$$y_n = c$$
$$z_n = z_0 + \sin^{-1}\left(\frac{c}{A_n \cdot x_0}\right)$$
$$A_n = \prod_{i=1}^{n} \sqrt{1 + 2^{-2i}}$$

H. Arccosine (1)

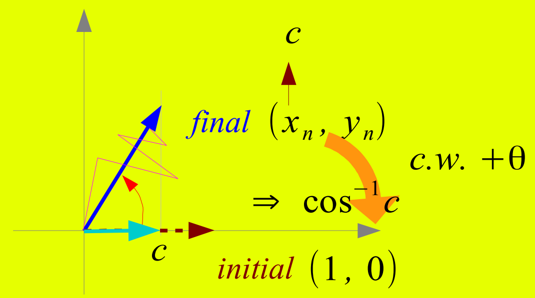
Exploit Vector Mode HW

$$z_n \rightarrow z_0 + \sin^{-1}\left(\frac{c}{A_n \cdot x_0}\right)$$

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

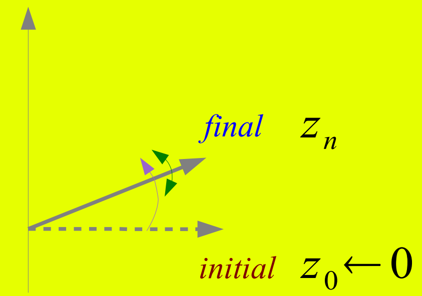
$$d_i = +1 \quad if \quad y_i < c$$
$$d_i = -1 \quad otherwise$$

Vector View



Minimize the residual x component

Accumulator View

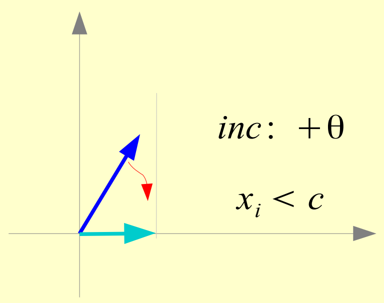


***Add** angles at each step*

$$x_{i+1} = x_i - y_i \cdot d_i \cdot 2^{-i}$$
$$y_{i+1} = y_i + x_i \cdot d_i \cdot 2^{-i}$$
$$z_{i+1} = z_i - d_i \cdot \tan^{-1}(2^{-i})$$

$$d_i = +1 \quad if \quad x_i > c$$
$$d_i = -1 \quad otherwise$$

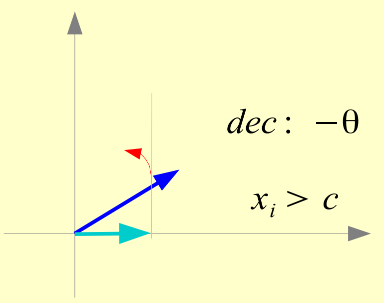
Vector View



$x_i < c$ Inc Angle

Add (+) Angle

Vector View



$x_i > c$ Dec Angle

Add (–) Angle

H. Arccosine (1)

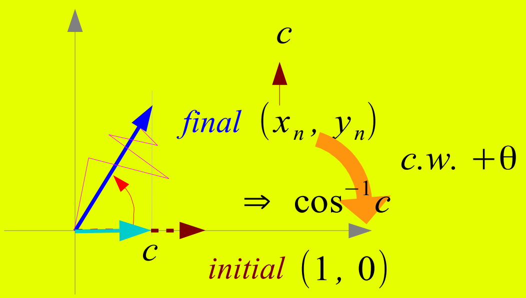
Exploit Vector Mode HW

$$z_n \rightarrow z_0 + \sin^{-1}\left(\frac{c}{A_n \cdot x_0}\right)$$

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

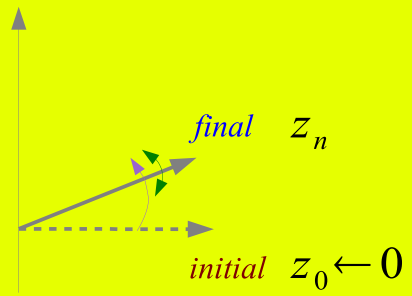
$$d_i = +1 \quad \text{if} \quad y_i < c$$
$$d_i = -1 \quad \text{otherwise}$$

Vector View



Minimize the residual x component

Accumulator View



Add *angles at each step*

$$x_{i+1} = x_i - y_i \cdot d_i \cdot 2^{-i}$$
$$y_{i+1} = y_i + x_i \cdot d_i \cdot 2^{-i}$$
$$z_{i+1} = z_i - d_i \cdot \tan^{-1}(2^{-i})$$

$$d_i = +1 \quad \text{if} \quad x_i > c$$
$$d_i = -1 \quad \text{otherwise}$$

$$y_n = \sqrt{(A_n \cdot y_0)^2 - c^2}$$
$$x_n = c$$
$$z_n = z_0 + \cos^{-1}\left(\frac{c}{A_n \cdot y_0}\right)$$
$$A_n = \prod_{i=1}^{n} \sqrt{1 + 2^{-2i}}$$

I. Linear Functions (1)

Rotation Mode

$z_0 \leftarrow \phi \quad (desired\ angle)$

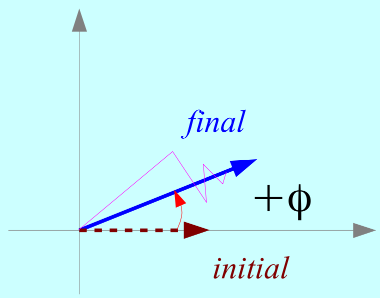
$z_n \rightarrow 0$

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

$d_i = -1 \quad if \quad z_i < 0$

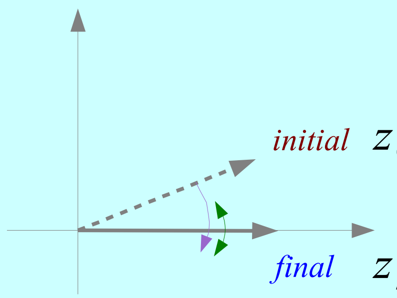
$d_i = +1 \quad otherwise$

Vector View



Minimize the residual angle

Accumulator View



***Subtract** angles at each step*

$$
\begin{aligned}
x_{i+1} &= x_i - y_i \cdot d_i \cdot 2^{-i} \\
y_{i+1} &= y_i + x_i \cdot d_i \cdot 2^{-i} \\
z_{i+1} &= z_i - d_i \cdot \tan^{-1}(2^{-i})
\end{aligned}
$$

$$
\begin{aligned}
x_{i+1} &= x_i - \mathbf{0} \cdot y_i \cdot d_i \cdot 2^{-i} = x_i \\
y_{i+1} &= y_i + x_i \cdot d_i \cdot 2^{-i} \\
z_{i+1} &= z_i - d_i \cdot (2^{-i})
\end{aligned}
$$

$$
\begin{aligned}
x_n &= A_n \left[x_0 \cos z_0 - y_0 \sin z_0 \right] \\
y_n &= A_n \left[y_0 \cos z_0 + x_0 \sin z_0 \right] \\
z_n &= 0
\end{aligned}
$$

$$
\begin{aligned}
x_n &= x_0 \\
y_n &= y_0 + x_0 z_0 \\
z_n &= 0
\end{aligned}
$$

Vectoring Mode

$$z_0 \leftarrow 0$$

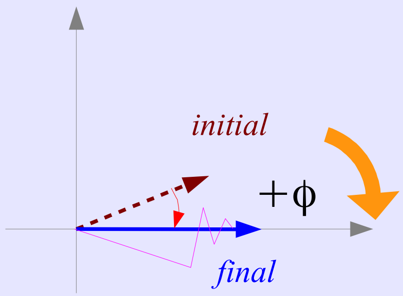
$$z_n \rightarrow z_0 + \tan^{-1}(y_0 / x_0)$$

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

$$d_i = +1 \quad \textit{if} \quad y_i < 0$$

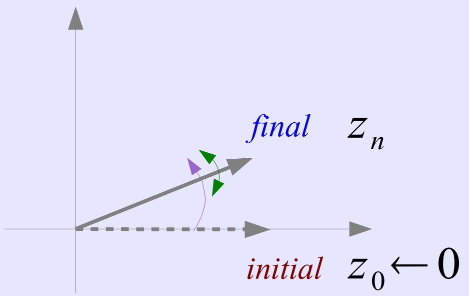
$$d_i = -1 \quad \textit{otherwise}$$

Vector View



Minimize the residual y component

Accumulator View



Add *angles at each step*

$$x_{i+1} = x_i - y_i \cdot d_i \cdot 2^{-i}$$

$$y_{i+1} = y_i + x_i \cdot d_i \cdot 2^{-i}$$

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

$$x_{i+1} = x_i - \mathbf{0} \cdot y_i \cdot d_i \cdot 2^{-i} = x_i$$

$$y_{i+1} = y_i + x_i \cdot d_i \cdot 2^{-i}$$

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

$$x_n = A_n \sqrt{x_0^2 + y_0^2}$$

$$y_n = 0$$

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

$$x_n = x_0$$

$$y_n = 0$$

$$z_n = z_0 - (y_0 / x_0)$$

Rotation Mode

$$z_0 \leftarrow \phi \quad (desired\ angle)$$

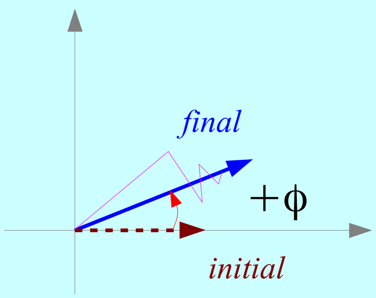
$$z_n \rightarrow 0$$

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

$$d_i = -1 \quad if \quad z_i < 0$$

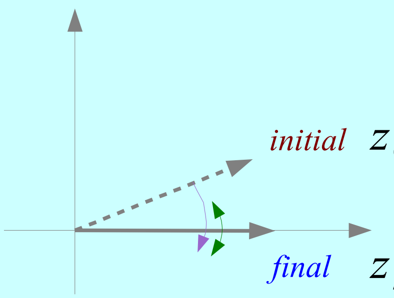
$$d_i = +1 \quad otherwise$$

Vector View



Minimize the residual angle

Accumulator View



***Subtract** angles at each step*

$$x_{i+1} = x_i - y_i \cdot d_i \cdot 2^{-i}$$

$$y_{i+1} = y_i + x_i \cdot d_i \cdot 2^{-i}$$

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

$$x_{i+1} = x_i + y_i \cdot d_i \cdot 2^{-i}$$

$$y_{i+1} = y_i + x_i \cdot d_i \cdot 2^{-i}$$

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

$$x_n = A_n \left[x_0 \cos z_0 - y_0 \sin z_0 \right]$$

$$y_n = A_n \left[y_0 \cos z_0 + x_0 \sin z_0 \right]$$

$$z_n = 0$$

$$x_n = A_n \left[x_0 \,\mathbf{cosh}\, z_0 - y_0 \,\mathbf{sinh}\, z_0 \right]$$

$$y_n = A_n \left[y_0 \,\mathbf{cosh}\, z_0 + x_0 \,\mathbf{sinh}\, z_0 \right]$$

$$z_n = 0$$

J. Hyperbolic Functions (1)

Vectoring Mode

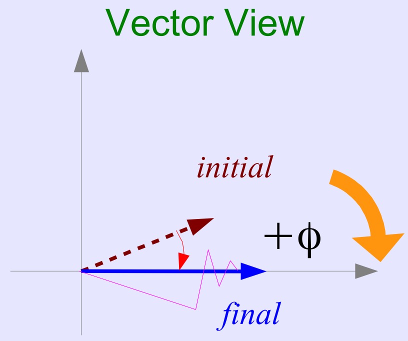
$$z_0 \leftarrow 0$$

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

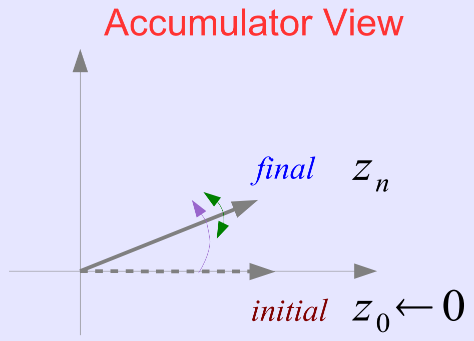
$$z_{i+1} = z_i - d_i \cdot \tan^{-1}(2^{-i})$$

$$d_i = +1 \quad \textit{if} \quad y_i < 0$$

$$d_i = -1 \quad \textit{otherwise}$$



Minimize the residual y component



Add *angles at each step*

$$x_{i+1} = x_i - y_i \cdot d_i \cdot 2^{-i}$$
$$y_{i+1} = y_i + x_i \cdot d_i \cdot 2^{-i}$$
$$z_{i+1} = z_i - d_i \cdot \tan^{-1}(2^{-i})$$

$$x_{i+1} = x_i + y_i \cdot d_i \cdot 2^{-i}$$
$$y_{i+1} = y_i + x_i \cdot d_i \cdot 2^{-i}$$
$$z_{i+1} = z_i - d_i \cdot \mathbf{tanh}^{-1}(2^{-i})$$

$$x_n = A_n \sqrt{{x_0}^2 + {y_0}^2}$$
$$y_n = 0$$
$$z_n = z_0 + \tan^{-1}(y_0 / x_0)$$

$$x_n = A_n \sqrt{{x_0}^2 - {y_0}^2}$$
$$y_n = 0$$
$$z_n = z_0 + \mathbf{tanh}^{-1}(y_0 / x_0)$$

Unified CORDIC Iteration Eq

$$x_{i+1} = x_i - \boldsymbol{m} \cdot y_i \cdot d_i \cdot 2^{-i}$$
$$y_{i+1} = y_i + x_i \cdot d_i \cdot 2^{-i}$$
$$z_{i+1} = z_i - d_i \cdot \boldsymbol{e}_i$$

$$\boldsymbol{m} = 1 \Rightarrow \boldsymbol{e}_i = \tan^{-1}(2^{-i})$$
$$\boldsymbol{m} = 0 \Rightarrow \boldsymbol{e}_i = (2^{-i})$$
$$\boldsymbol{m} = -1 \Rightarrow \boldsymbol{e}_i = \mathbf{tanh}^{-1}(2^{-i})$$

$$\mathbf{tan}\,\alpha = \frac{\mathbf{sin}\,\alpha}{\mathbf{cos}\,\alpha}$$

$$\mathbf{tanh}\,\alpha = \frac{\mathbf{sinh}\,\alpha}{\mathbf{cosh}\,\alpha}$$

$$\mathbf{exp}\,\alpha = \mathbf{sinh}\,\alpha + \mathbf{cosh}\,\alpha$$

$$\mathbf{ln}\,\alpha = 2\,\mathbf{tanh}^{-1}(y/x)$$
$$x = \alpha + 1$$
$$y = \alpha - 1$$

$$(\alpha)^{1/2} = (x^2 - y^2)^{1/2}$$
$$x = \alpha + 1/4$$
$$y = \alpha - 1/4$$

Unified CORDIC Iteration Eq

$$x_{i+1} = x_i - \boldsymbol{m} \cdot y_i \cdot d_i \cdot 2^{-i}$$
$$y_{i+1} = y_i + x_i \cdot d_i \cdot 2^{-i}$$
$$z_{i+1} = z_i - d_i \cdot \boldsymbol{e}_i$$

$$\boldsymbol{m} = 1 \quad \Rightarrow \quad \boldsymbol{e}_i = \tan^{-1}(2^{-i})$$
$$\boldsymbol{m} = 0 \quad \Rightarrow \quad \boldsymbol{e}_i = (2^{-i})$$
$$\boldsymbol{m} = -1 \quad \Rightarrow \quad \boldsymbol{e}_i = \mathbf{tanh}^{-1}(2^{-i})$$

$$\mathbf{cosh}\, i\, x = \frac{1}{2}(e^{ix} + e^{-ix}) = \cos x \qquad \mathbf{cosh}\, x = \frac{1}{2}(e^{x} + e^{-x}) = \cos i\, x$$
$$\mathbf{sinh}\, i\, x = \frac{1}{2}(e^{ix} - e^{-ix}) = i\, \sin x \qquad \mathbf{sinh}\, x = \frac{1}{2}(e^{x} - e^{-x}) = -i\, \sin ix$$
$$\mathbf{tanh}\, i\, x = \frac{(e^{ix} + e^{-ix})}{(e^{ix} - e^{-ix})} = i\, \tan x \qquad \mathbf{tanh}\, x = \frac{(e^{x} + e^{-x})}{(e^{x} - e^{-x})} = -i\, \tan i\, x$$

Unified CORDIC Iteration Eq

References

[1] http://en.wikipedia.org/
[2] CORDIC FAQ, www.dspguru.com
[3] R. Andraka, A survey of CORDIC algorithms for FPGA based computers
[4] J. S. Walther, A Unified Algorithm for Elementary Functions