Cordic Algorithm

Come realizzare funzioni trigonometriche in Hardware

Introduzione

- CORDIC: **CO**rdinate **R**otation **DI**gital **C**omputing
  - E' un metodo iterativo per calcolare funzioni trigonometirche
- Trova impiego in diversi campi:
  - Rotazione di vettori
  - Generazione di segnali sinusoidali, chirp.
  - Trasformate DFT, DCT
  - Filtri
  - Modulatori
  - …

Concetti base

Una rotazione può venir espressa come:

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

Oppure come

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

Inoltre una rotazione può essere ottenuta attraverso una serie di rotazioni di entità diverse

Potenze di 2

Se le rotazioni base sono limitate a quelle per cui

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

La rotazione può venir espressa in forma ricorsiva come

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

Ovvero attraverso semplici operazioni si somma e shift

Dove

$$K_i = \cos\left(\tan^{-1} 2^{-i}\right) = 1\Big/\sqrt{1 + 2^{-2i}}$$
$$d_i = \pm 1$$

ALTERA

Scaling

0.7071
0.8944
0.9701
0.9923
0.9981
0.9995
0.9999
1.0000
1.0000
1.0000

- Per quanto riguarda i termini Ki
  - Si possono “accumulare” i prodotti
  - Il valore asintottico tende a 0.6073
  - Trascurando i temini K si ottengono valori maggiorati di 1.647
  - Nel caso di poche iterazioni il valore esatto è

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

Terza equazione

In un sistema CORDIC si aggiunga anche una terza equazione per aggiornare anche il valore dell'angolo

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

I Valori di $\tan^{-1}(2^{-i})$ possono essere precalcolati e salvati in un piccola LUT

Impieghi

- "*Rotation mode" :* ruota il vettore d'ingresso di un determinato angolo
  - Data la posizione iniziale e l'angolo trova la posizione finale

- *"Vectoring" mode":* Ruota il vettore d'ingresso fino ad adagiarsi sull'asse x e restituisce l'angolo di rotazione
  - *Data la posizione iniziale restituisce l'angolo*

Rotation Mode

Il valore dell'angolo viene inizializzato col valore desiderato poi ad ogni passo si calcola d_i

$$d_i = -1 \text{ if } z_i < 0, \ +1 \text{ otherwise}$$

e si aggiornano i valori fintanto che z=0;

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

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

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

Alla fine

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

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

$$z_n = 0$$

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

Vectoring Mode

Il valore dell'angolo viene inizializzato col valore z_0 poi ad ogni passo si calcola d_i

$$d_i = +1 \text{ if } y_i < 0,\ \ -1 \text{ otherwise.}$$

e si aggiornano i valori fintanto che y=0

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

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

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

Alla fine

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

$$y_n = 0$$

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

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

Limiti

Come esposto l'algoritmo è limitato ad angoli compresi tra $\pm\ \pi/2$ in quanto al passo 0: tan(1) = $\pi/2$)

Per estenderlo si può introdurre una ulteriore rotazione

- di $\pm\pi/2$

$$x' = -d \cdot y$$

$$y' = d \cdot x$$

$$z' = z + d \cdot \pi/2$$

- Oppure di π

$$x' = d \cdot x$$

$$y' = d \cdot y$$

$$z' = z \text{ if } d = 1, \text{ or } z - \pi \text{ if } d = -1$$

Sin e Cos

Partendo con $y_0=0$

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

- Genera comtemporaneamente sin e cos
- Se $x_0 = 1/A_n$ si ottiene il risultato pulito
- Spesso (nei modulatori) l'uscita viene moltiplicata per l'ampiezza. Con questo si può sfruttare x_0 per ottenere la modulazione

Conversione
Coord. Polari – Coord. Cartesiane

$$x = r\cos\theta$$

$$y = r\sin\theta$$

- Valori di partenza:
  - x_0 : modulo nelle coord. polari (r)
  - z_0: fase nelle coord. polari
  - y_0: $=0$
- Il risultato va corretto in base al valere di An (calcolo di An e moltiplicatore)

Rotazione Generica

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

- Valori di partenza:
  - $x_{0,}$ y_0: valori iniziali
  - z_0: angolo di rotazione

- Il risultato va corretto in base al valere di An (calcolo di An e due moltiplicatori)

Arcotangente

- E' il modo di funzionamento principale del "vectoring mode"
  - Si inizializza con $z_0 = 0$
  - L'input va fornito in forma di rapporto (vantaggio: si può fornire un input = inf)
  - An non influenza il risultato

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

Modulo di un vettore

- E' implicito nel vectoring mode

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

- Il risultato deve venir scalato di An, ma è un sistema molto economico per calcolare il modulo,

Conversione
Coord. Cartesiane– Coord. Polari

- Sono implicite nel “vectoring mode”
- Sono state analizzate separatamente nei due lucidi precedenti
  - Calcolo dell’arcotangente
  - Modulo del vettore

Funzioni inverse

- In genere se una funzione è calcolabile attraverso l'algoritmo cordic, lo è pure la sua inversa
- Es: arcsin
  - Si parta con un vettore unitario posto sull'asse x
  - Si ruoti fintanto che y risulta uguale all'argomento
  - Si deve agire opportunamente sulla funzione decisionale

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

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

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

where

$d_i = +1$ if $y_i < c$, -1 otherwise, and

c = input argument.

Realizzazione

La decisione d_i si può collegare

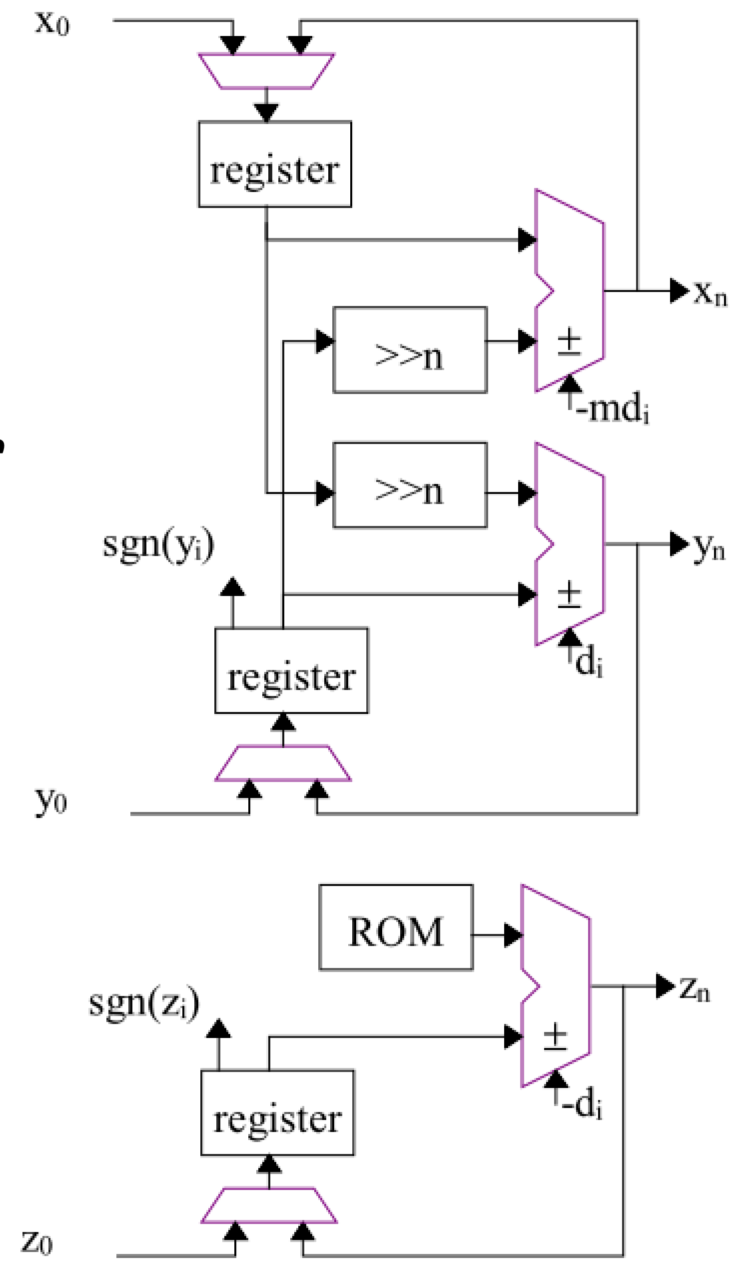
- Al bit di segno di z_i nel *Rotation mode*
- Al bit di segno di y_i nel *Vectoring mode*

- Una opportuna FSM conta le iterazioni ed opera sullo shift e sull'indirizzo della ROM

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

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

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



NCO su Altera

- NCO: Numeric Controlled Oscillator