El algoritmo de CORDIC

Desde la ecuación hasta la FPGA

Diseño de sistemas digitales con FPGA
1er Cuatrimestre 2023

DC-UBA

DEPARTAMENTO DE COMPUTACION
Facultad de Ciencias Exactas y Naturales - UBA

DEPARTAMENTO DE COMPUTACION

Facultad de Ciencias Exactas y Naturales - UBA

CORDIC

El redescubrimiento de los algoritmos de **CORDIC** (**CO**ordinate **R**otation **DI**gital **C**omputer) propuestos originalmente por Volder (en 1956), permitieron poner al **hardware** un paso adelante en cuanto a **procesamiento intensivo de señales**.

The CORDIC Trigonometric Computing Technique*

JACK E. VOLDER†

Summary—The *CO*ordinate *R*otation *DI*gital *C*omputer (CORDIC) is a special-purpose digital computer for real-time airborne computation. In this computer, a unique computing technique is employed which is especially suitable for solving the trigonometric relationships involved in plane coordinate rotation and conversion from rectangular to polar coordinates. CORDIC is an entire-transfer computer; it contains a special serial arithmetic unit consisting of three shift registers, three adder-subtractors, and special interconnections. By use of a prescribed sequence of conditional additions or subtractions, the CORDIC arithmetic unit can be controlled to solve either set of the following equations:

$$Y' = K(Y \cos \lambda + X \sin \lambda)$$
$$X' = K(X \cos \lambda - Y \sin \lambda),$$

or

$$R = K\sqrt{X^2 + Y^2}$$
$$\theta = \tan^{-1} Y/X,$$

where K is an invariable constant.

This special arithmetic unit is also suitable for other computations such as multiplication, division, and the conversion between binary and mixed radix number systems. However, only the trigonometric algorithms used in this computer and the instrumentation of these algorithms are discussed in this paper.

Introduction

THE CORDIC computing technique was developed especially for use in a real-time digital computer where the majority of the computation involved the discontinuous, programmed solution of the trigonometric relationships of navigation equations and a high solution rate for the trigonometric relationships of

* Manuscript received by the PGEC, May 25, 1959. Presented at the Western Joint Computer Conf., San Francisco, Calif.; March 3–5, 1959.

† Convair, a Div. of General Dynamics Corp., Fort Worth, Texas.

DEPARTAMENTO DE COMPUTACION
Facultad de Ciencias Exactas y Naturales - UBA

CORDIC

Estos algoritmos permiten, mediante un número determinado de **iteraciones**, evaluar funciones trigonométricas e hiperbólicas con un **error definido**.

$arccos(x)$

$cos(x)$

$sin(x)$

$arctan(x)$

$arcsin(x)$

rotación vectorial general

$\sqrt{x}$

polar ↔ rectangular

$cosh(x)$

$sinh(x)$

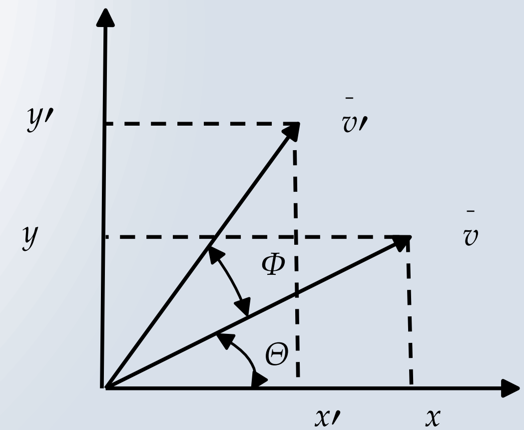
$tanh(x)$

CORDIC

¿Cómo surge el algoritmo?

Se intenta resolver el problema de rotar un vector *v* un ángulo *Φ* determinado, para obtener un vector *v*', donde:

$$\overline{v} = (x,y)$$
$$\overline{v}' = (x',y')$$



$$x = \rho\cos\Theta$$
$$y = \rho\,sen\Theta$$

Entonces

$$x' = \rho\cos(\Theta + \Phi) = \rho[\cos\Theta\cos\Phi - sen\Theta\, sen\Phi]$$
$$y' = \rho\,sen(\Theta + \Phi) = \rho[sen\Theta\cos\Phi + sen\Theta\cos\Phi]$$

$$x' = x\cos\Phi - y\,sen\Phi$$
$$y' = y\cos\Phi + x\,sen\Phi$$

CORDIC

- El resultado anterior se puede escribir como:

$$x\prime = cos\Phi[x - y.tan\Phi]$$
$$y\prime = cos\Phi[y + x.tan\Phi]$$

- Lo que en sí no aporta mayor información...
- Sin embargo, si agregamos la restricción de que Φ varíe de forma que $tan(\Phi)=2^{-i}$, podemos acercarnos al ángulo deseado iterativamente.

$$x\prime = cos(tan^{-1}(2^{-i}))[x - y2^{-i}] = K_i[x - y2^{-i}]$$
$$y\prime = cos(tan^{-1}(2^{-i}))[y + x2^{-i}] = K_i[y + x2^{-i}]$$
$$Donde K_i = cos(tan^{-1}(2^{-i})) = \frac{1}{\sqrt{1 + 2^{-2i}}}$$

DEPARTAMENTO DE COMPUTACION
Facultad de Ciencias Exactas y Naturales - UBA

CORDIC

Resumiendo, el algoritmo de CORDIC queda:

$$x' = K_i[x - y \cdot d_i \cdot 2^{-i}]$$
$$y' = K_i[y + x \cdot d_i \cdot 2^{-i}] \qquad d_i = \pm 1$$

$$Donde K_i = \frac{1}{\sqrt{1 + 2^{-2i}}} \qquad Y\ sabemos\ que: K = \prod_{i=0}^{n} \frac{1}{\sqrt{1 + 2^{-2i}}} \rightarrow 0{,}6073..$$

$$Entonces: A_n = \prod_{i=0}^{n} \sqrt{1 + 2^{-2i}} \rightarrow 1{,}647.. \quad es\ la\ ganancia\ del\ algoritmo$$

Además, por cada iteración hay que guardar la información angular, lo que agrega una tercer ecuación:

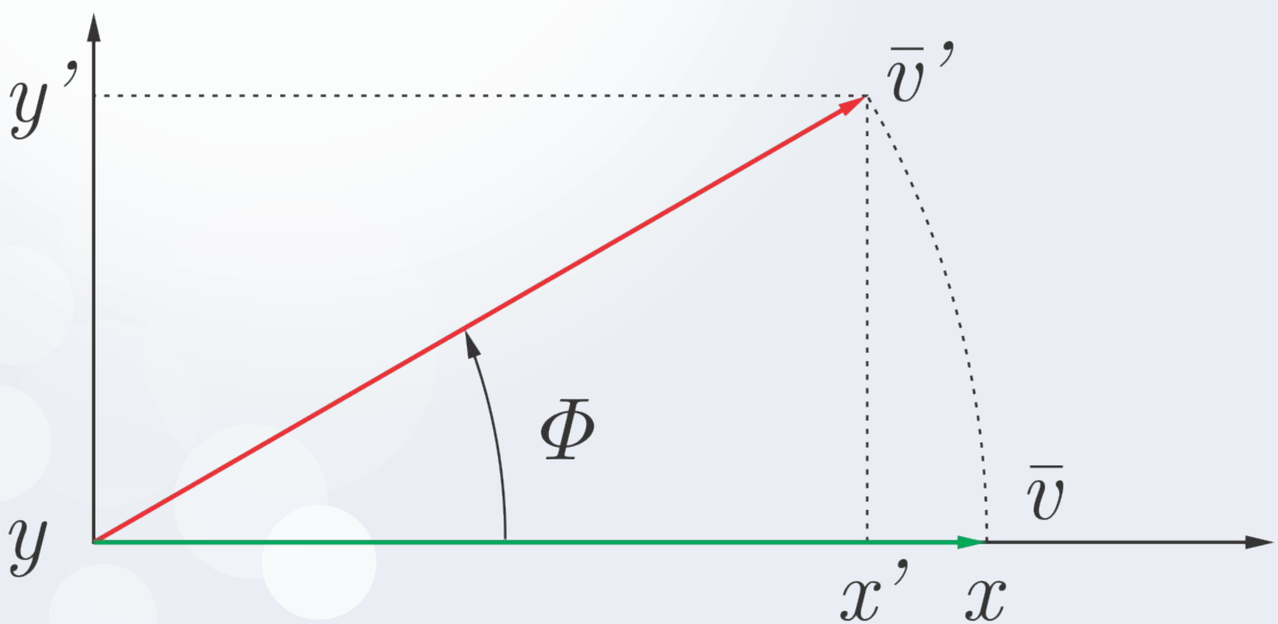
$$z' = z_i - d_i \cdot tan^{-1}(2^{-i})$$

CORDIC

Intentemos rotar el vector

$$\overline{v} = (x,y) = (1,0)$$
$$\Phi = 30^{\circ}$$

$$\overline{v}\,' = (x',y')$$



CORDIC

Ángulos a disposición:

$$\phi_i = tan^{-1}(2^{-i})$$

i	grados	radianes
0	45,00	0,7854
1	26,75	0,4636
2	14,04	0,2450
3	7,13	0,1244
4	3,58	0,0624
5	1,79	0,0312
6	0,90	0,0160
7	0,45	0,0080
8	0,22	0,0040
9	0,11	0,0020

CORDIC

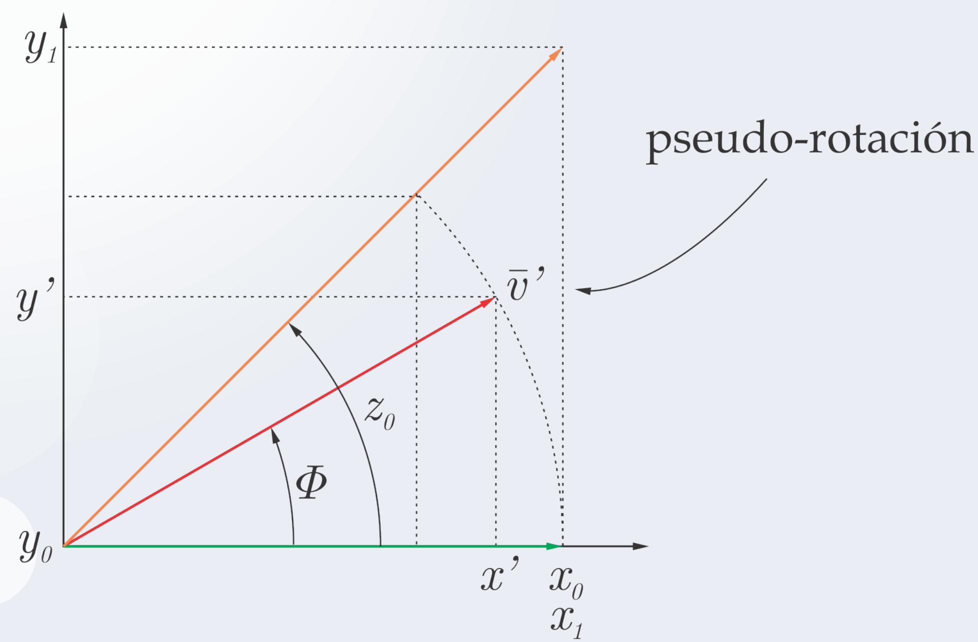
$$i = 0,\ \ x_0 = 1,\ y_0 = 0,\ z_0 = 30,\ d_0 = 1$$

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

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

$$x_1 = 1 - 0 \cdot 2^{-0} = 1$$
$$y_1 = 0 + 1 \cdot 2^{-0} = 1 \qquad d_0 = +1$$

$$z_1 = 30 - 45 = -15$$



CORDIC

$$i = 1,\ x_1 = 1,\ y_1 = 1,\ z_1 = -15,\ d_1 = -1$$

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

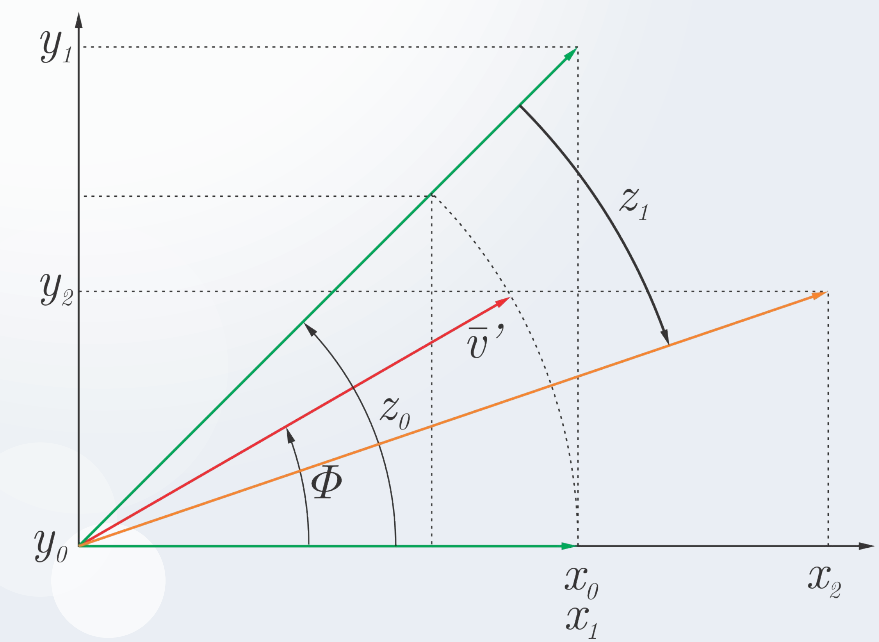
$$= y_i + x_i \cdot d_i \cdot 2^{-i} \qquad d_i = \pm 1$$

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

$$x_2 = 1 + 1 \cdot 2^{-1} = 1{,}5$$

$$y_2 = 1 - 1 \cdot 2^{-1} = 0{,}5 \qquad d_1 = -1$$

$$z_2 = -15 + 26{,}75 = 11{,}75$$



CORDIC

$$i = 2,\ x_2 = 1{,}5,\ y_2 = 0{,}5,\ z_2 = 11{,}75,\ d_2 = 1$$

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

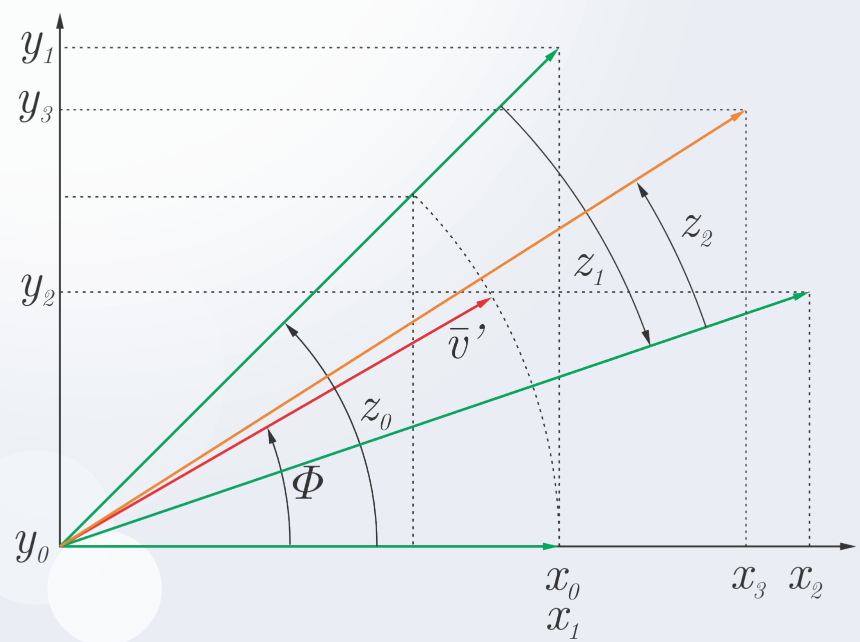
$$= y_i + x_i \cdot d_i \cdot 2^{-i} \qquad d_i = \pm 1$$

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

$$x_3 = 1{,}5 - 0{,}5 \cdot 2^{-2} = 1{,}375$$

$$y_3 = 0{,}5 + 1{,}5 \cdot 2^{-2} = 0{,}875 \qquad d_2 = +1$$

$$z_3 = 11{,}75 - 14{,}04 = -2{,}29$$



DEPARTAMENTO DE COMPUTACION
Facultad de Ciencias Exactas y Naturales - UBA

CORDIC

Modo de rotación:

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

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

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

$$Donde\ \ d_i = -1\ \ si\ \ \ z_i < 0,\ 1\ \forall\ otro\ \ z_i$$

Resultando:

$$x_n \simeq A_n\left[x_0 cos(z_0) - y_0 sen(z_0)\right]$$

$$y_n \simeq A_n\left[y_0 cos(z_0) + x_0 sen(z_0)\right]$$

$$z_n \simeq 0 \qquad\qquad A_n = \prod_{i=0}^{n} \sqrt{1 + 2^{-2i}}$$

CORDIC

Modo de vectorización (rota el vector al eje x):

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

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

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

$$Donde\ \ d_i = 1\ \ si\ \ y_i < 0, -1\ \forall\ \ otro\ \ y_i$$

Resultando:

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

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

$$y_n \simeq 0 \qquad A_n = \prod_{i=0}^{n} \sqrt{1 + 2^{-2i}}$$

CORDIC

Módulo y fase de un vector

La conversión de coordenadas rectangulares a polares resulta inmediata de los resultados del modo de vectorización, ya que:

$$x_n \simeq A_n \sqrt{x_0^2 + y_0^2} \quad \textit{Es el módulo del vector escalado}$$

$$z_n \simeq z_0 + tan^{-1}\left(\frac{y_0}{x_0}\right) \quad \textit{Es el ángulo del vector original si } z_0 = 0$$

$$A_n = \prod_{i=0}^{n} \sqrt{1 + 2^{-2i}} \quad \textit{Es la ganancia del algoritmo}$$

CORDIC

Cálculo de seno y coseno:

Recordando el modo de rotación:

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

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

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

$$Donde\ \ d_i = -1\ si\ \ z_i < 0, 1\ \forall\ \ otro\ z_i$$

Resulta que si hacemos *y=0* y *x=1* queda:

$$x_n \simeq A_n x_0 cos(z_0)$$

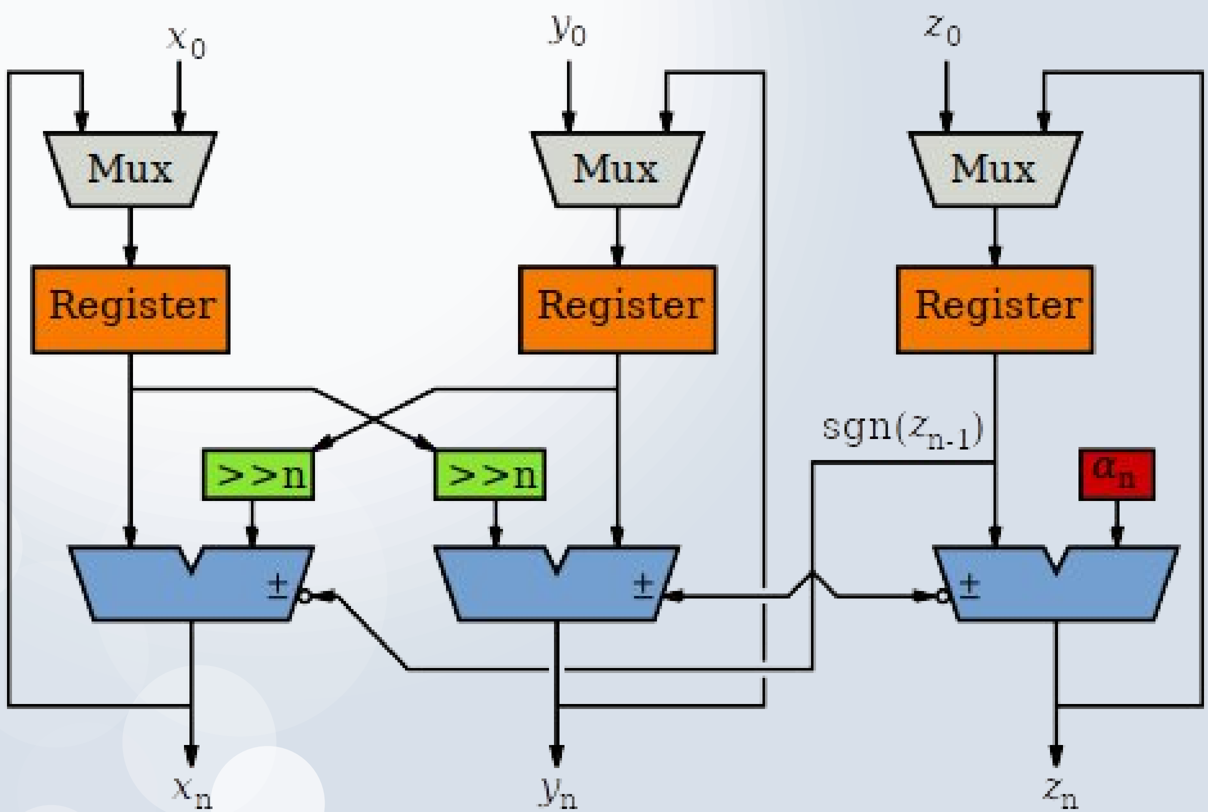
$$y_n \simeq A_n x_0 sin(z_0)$$

$$A_n = \prod_{i=0}^{n} \sqrt{1 + 2^{-2i}}$$

CORDIC

Topologías

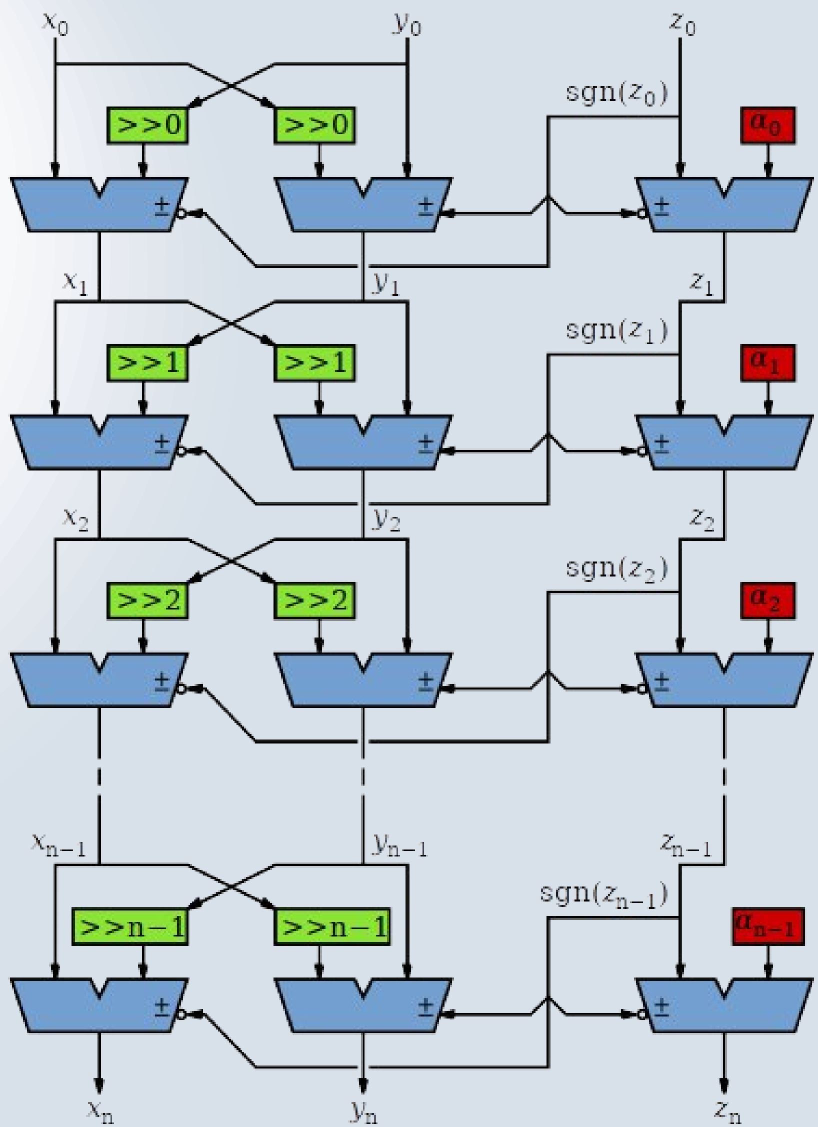
Bit parallel iterative:



CORDIC

Topologías

Bit parallel *unrolled:*
Más área
¡Oportunidad de pipeline!



CORDIC

Implementación en VHDL:

Definición de la entidad. Interfaz y puertos.

```
entity cordic is
    generic(N    :natural := 16;  --Ancho de la palabra
            ITER :natural := 16); --Numero de iteraciones
    port ( clk : in std_logic;
           rst : in std_logic;
           x0  : in std_logic_vector(N-1 downto 0);
           y0  : in std_logic_vector(N-1 downto 0);
           z0  : in std_logic_vector(N-1 downto 0);
           xn  : out std_logic_vector(N-1 downto 0);
           yn  : out std_logic_vector(N-1 downto 0);
           zn  : out std_logic_vector(N-1 downto 0)
    );
end entity;
```

CORDIC

Implementación en VHDL:

Componente “Iteración de CORDIC”

```
component cordic_iter is
    generic(N     : natural := 8;  --Ancho de la palabra
            SHIFT : natural := 1); --Desplazamiento
    port( clk   : in std_logic;
          rst   : in std_logic;
          xi    : in std_logic_vector(N-1 downto 0);
          yi    : in std_logic_vector(N-1 downto 0);
          zi    : in std_logic_vector(N-1 downto 0);
          ci    : in std_logic_vector(N-1 downto 0);
          xip1 : out std_logic_vector(N-1 downto 0);
          yip1 : out std_logic_vector(N-1 downto 0);
          zip1 : out std_logic_vector(N-1 downto 0)
        );
end component;
```

CORDIC

Implementación en VHDL: uso del **for/if generate.**

```
type ConnectVector is array(ITER-1 downto 0) of std_logic_vector(N-1 downto 0);
signal wirex, wirey, wirez : ConnectVector;

generic_inst:
for j in 0 to ITER-1 generate
begin
  it0:
  if j = 0 generate
  begin
    iter0: cordic_iter
      generic map(N,0)
      port map(
        clk => clk,
        rst => rst,
        xi => x0,
        yi => y0,
        zi => z0,
        ci => atanLUT(0),
        xip1 => wirex(j+1),
        yip1 => wirey(j+1),
        zip1 => wirez(j+1)
      );
  end generate;
```

CORDIC

Implementación en VHDL: uso del **for/if generate.**

```
itj:
  if j>0 and j<(ITER-1) generate
  begin
   iterj: cordic_iter
     generic map(N,j)
     port map(
       clk => clk,
       rst => rst,
       xi => wirex(j),
       yi => wirey(j),
       zi => wirez(j),
       ci => atanLUT(j),
       xip1 => wirex(j+1),
       yip1 => wirey(j+1),
       zip1 => wirez(j+1)
     );
  end generate;
```

DEPARTAMENTO DE COMPUTACION
Facultad de Ciencias Exactas y Naturales - UBA

CORDIC

Implementación en VHDL: uso del **for/if generate.**

```
itM_1:
  if j=(ITER-1) generate
  begin
   iterM_1: cordic_iter
     generic map(N,j)
     port map(
       clk => clk,
       rst => rst,
       xi => wirex(j),
       yi => wirey(j),
       zi => wirez(j),
       ci => atanLUT(j),
       xip1 => xn,
       yip1 => yn,
       zip1 => zn
     );
  end generate;
end generate;
```

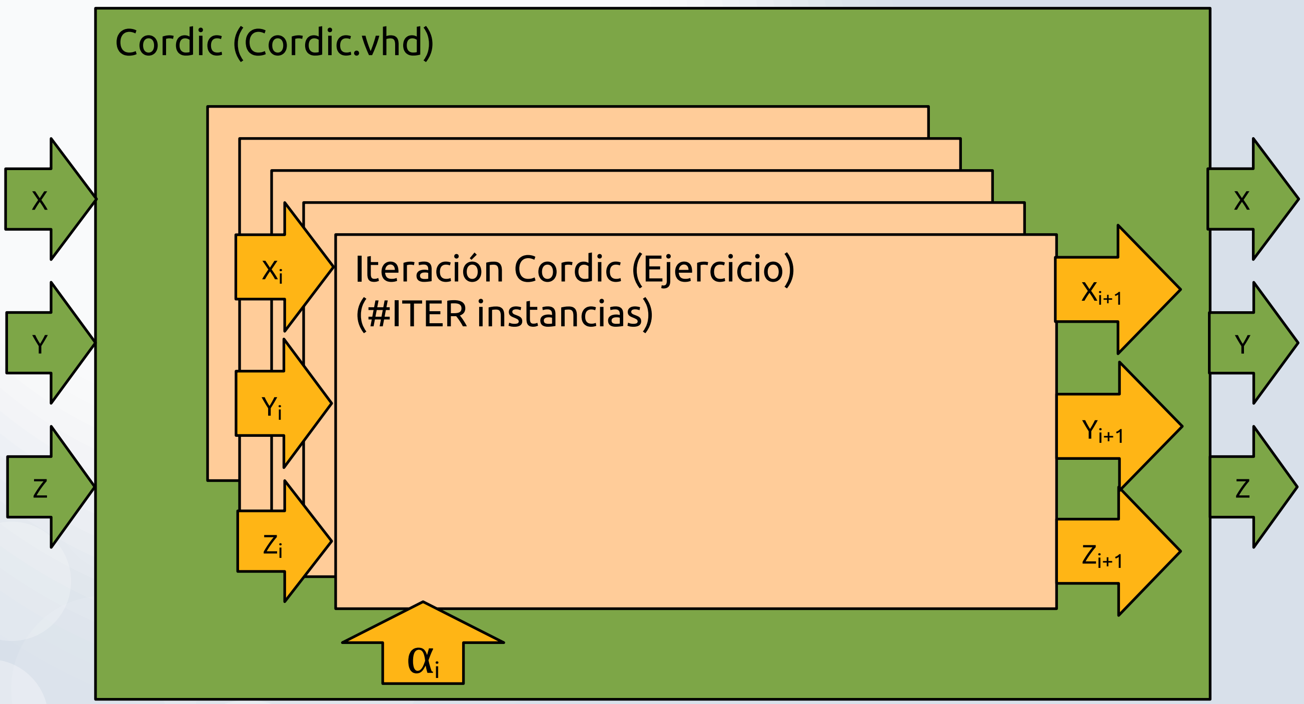
VHDL: Manos a la obra

CORDIC

Consigna:
Utilizando el código provisto, implementar y simular un algoritmo de CORDIC.
Pasos a seguir:
a) Escribir la unidad "cordic_iter"
b) Sintetizar el código y simular con datos arbitrarios a la entrada. Comparar con una implementación de alto nivel (Python (provista))
c) Agregar corrección por cuadrante.
d) Eviarle datos desde la PC a través de FPGALink, calcular en la FPGA y devolver los resultados.

VHDL: Manos a la obra

CORDIC



Bibliografía

- *The design Warrior's guide to FPGA* – C. Maxfield
- *RTL hardware design using VHDL* – P. Pong
- *Digital signal processing with FPGA* – Meyer-Baese

¿Dudas?