Algoritmo CORDIC Unrolled

Trabajo práctico final CLP-CESE 10 Cohorte

Alberto Yapura

EL ALGORITMO DE CORDIC

El algoritmo de CORDIC se basa en las rotaciones de los vectores para encontrar una aproximación de una función no lineal. La principal ventaja del algoritmo es que es menos multiplicador. Utiliza sólo sumas y restas. CORDIC está construido con un número de etapas. Aumentando el número de etapas se mejora la precisión. La Fig. 1 muestra un ejemplo de tres rotaciones vectoriales, correspondientes a una CORDIC de tres etapas. El vector es el vector con un ángulo θ del cual, en este caso, se buscan los valores aproximados del coseno y el seno.

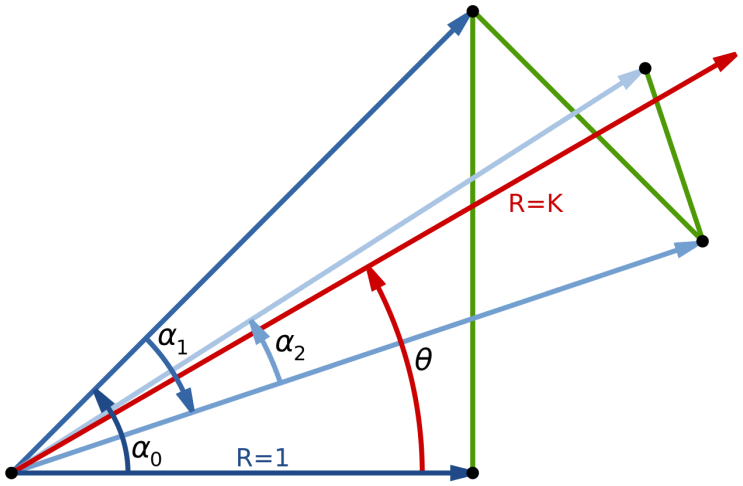

Figura 1. Tres rotaciones CORDIC.

La figura muestra tres rotaciones. La primera es positiva y se detecta que es demasiado grande. La segunda hará una rotación negativa y finalmente la tercera hará una positiva de nuevo.

Algoritmo CORDIC - modo de rotación

El objetivo principal de este trabajo es calcular las funciones trigonométricas Seno y Coseno desde un ángulo z_0. Para ello, CORDIC se implementó utilizando el modo rotacional para encontrar las coordenadas *x* e *y* dentro del cırculo trigonométrico unitario del ángulo z_0

considerado. Para comenzar, el algoritmo considera un vector inicial v_0 (Ecuación 1) con las siguientes coordenadas:

$$v_0 = \begin{bmatrix} 1 \\ 0 \end{bmatrix} \tag{1}$$

En cada iteración del algoritmo, el vector v_0 se gira en sentido antihorario o en sentido horario en pasos descendentes hasta llegar a obtener una aproximación de las magnitudes de *x* y *y* (Ecuación 2). Los pasos son definidos según la función trigonométrica arctan(2^{-i}) para i un número natural que comienza en 0.

$$\begin{cases} x_0 = \frac{1}{K} = 0.6072 \\ y_0 = 0 \\ z_0 = \theta \end{cases} \tag{2}$$

Para la Ecuación 2, la constante K se calcula, como se muestra a continuación, en base a las Ecuaciones 10 y 11. Formalmente, en cada iteración, el algoritmo calcula una rotación multiplicando el vector v_i con la matriz de rotación R_i:

$$v_i = R_i v_{i-1} \tag{3}$$

Dada la matriz de R_i de la Ecuación 4:

$$R_i = \begin{bmatrix} \cos(\gamma_i) & -\sin(\gamma_i) \\ \sin(\gamma_i) & \cos(\gamma_i) \end{bmatrix} \tag{4}$$

Con las siguientes propiedades trigonométricas de las ecuaciones 5 y 6:

$$\cos(\gamma_i) = \frac{1}{\sqrt{1+\tan^2(\gamma_i)}} \tag{5}$$

$$\sin(\gamma_i) = \frac{\tan(\gamma_i)}{\sqrt{1+\tan^2(\gamma_i)}} \tag{6}$$

El posible reduce la Ecuación 3 al siguiente formato:

$$v_i = \frac{1}{\sqrt{1+\tan^2(\gamma_i)}} \begin{bmatrix} 1 & -\tan(\gamma_i) \\ \tan(\gamma_i) & 1 \end{bmatrix} \begin{bmatrix} x_{i-1} \\ y_{i-1} \end{bmatrix} \tag{7}$$

Para que sea posible realizar las operaciones de aproximación de las funciones trigonométricas del coseno y del seno, es necesario limitar el ángulo y_i para que asuma los valores $\tan(y_i) = 2^{-i}$. De esta manera, la multiplicación con la tangente puede ser simplificada por una división de potencia de dos, por lo que es fácil de calcular con una

simple operación de desplazamiento de bits. Con esto, podemos simplificar la expresión para la Ecuación 8 de abajo:

$$v_i = \frac{1}{\sqrt{1+2^{-2i}}} \begin{bmatrix} 1 & -\sigma_{ii}2^{-i} \\ \sigma_i 2^{-i} & 1 \end{bmatrix} \begin{bmatrix} x_{i-1} \\ y_{i-1} \end{bmatrix} \tag{8}$$

A partir de este resultado es posible establecer las siguientes relaciones matemáticas de la Ecuación 9:

$$\begin{bmatrix} x_i \\ y_i \end{bmatrix} = \frac{1}{\sqrt{1+2^{-2i}}} \begin{bmatrix} x_{i-1} - \sigma_i 2^{-i} y_{i-1} \\ \sigma_i 2^{-i} x_{i-1} + y_{i-1} \end{bmatrix}$$

Omitiendo la constante de escala $\frac{1}{\sqrt{1+2^{-2i}}}$, obtenemos:

$$\begin{cases} x_i = x_{i-1} - \sigma_i 2^{-i} y_{i-1} \\ y_i = y_{i-1} - \sigma_i 2^{-i} x_{i-1} \end{cases} \rightarrow \begin{cases} x_{i+1} = x_i - \sigma_i 2^{-i} y_i \\ y_{i+1} = y_i - \sigma_i 2^{-i} x_i \end{cases} \rightarrow \begin{cases} x_{i+1} = x_i - \sigma_i 2^{-i} y_i \\ y_{i+1} = y_i - \sigma_i 2^{-i} x_i \\ z_{i+1} = z_i - \sigma_i \arctan(2^{-i}) \end{cases} \tag{9}$$

A partir de la Ecuación 9 el término $z_i+1 = z_i - \sigma i \arctan(2^{-i})$ tiene como función acumular el valor de todas las rotaciones y compararlo con su valor z_0, de manera que sea posible identificar el sentido de rotación a realizar en la siguiente iteración (en el sentido de las agujas del reloj o en el sentido contrario). El valor de (Ecuaciones 8 y 9) alterna entre -1 y 1 y sirve para indicar la dirección de rotación del vector. Durante las iteraciones del algoritmo y possıvel ignorar la siguiente expresión y aplicarla al final como un factor de escala, como se sugiere en la Ecuación 10:

$$K = \frac{1}{\sqrt{1+2^{-2i}}} \tag{10}$$

Como factor de escala, la expresión anterior debe trabajarse según la ecuación 11:

$$K(n) = \prod_{i=0}^{n-1} K_i = \prod_{i=0}^{n-1} \frac{1}{\sqrt{1+2^{-2i}}} \tag{11}$$

Si el número de iteraciones es limitado, el valor de K (Ecuación 12) puede aproximarse a una constante:

$$K = \lim_{n\to\infty} K(n) \approx 0.607252935008881256169 4$$

En algunos casos, para reducir la complejidad del algoritmo, la K no se corrige. Por esta razón, hay una ganancia de procesamiento A, calculada según la ecuación 13:

$$A = \frac{1}{K} = \lim_{n\to\infty} \prod_{i=0}^{n-1} \sqrt{1+2^{-2i}} \approx 1.64676025812107$$

Para el final del algoritmo los valores del seno y el coseno de z_0 serán, respectivamente, las coordenadas *y* y *x*.

Arquitectura unrolled

En una arquitectura unrolled todas las operaciones de rotación se realizan en un solo ciclo de reloj. El algoritmo CORDIC separa cada iteración en pasos. Cada paso está compuesto de los mismos componentes: tres sumadores/substractores, dos unidades aritméticas de desplazamiento y un LUT. Por lo tanto, la salida de un paso corresponde a la entrada del siguiente paso. El diagrama de la arquitectura paralela se muestra en la Figura 2.

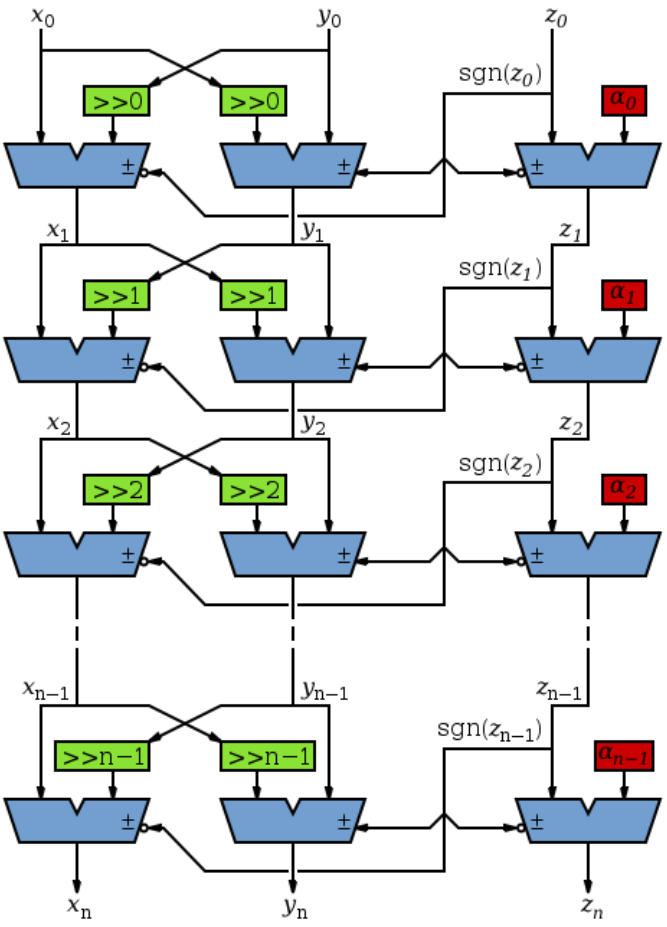

Figura 2. Arquitectura Unrolled.

Archivos

El algoritmo CORDIC fue desarrollado utilizando el lenguaje VHDL, se compone de los siguientes archivos:

Componentes

- Cordic.vhd
- CordicUnrolled.vhd
- Mux.vhd
- RightShifter.vhd
- Sumador1b.vhd
- SumadorRestador.vhd
- utility.vhd

Simulación y pruebas

- SinCos_tb.vhd

Simulación

Para realizar las simulaciones, síntesis e implementación se utilizó el software Vivado 2018.1 Se simularon los componentes Mux.vhd, RightShifter.vhd, Sumador1b.vhd, SumadorRestador.vhd y el SinCos_tb.vhd que engloba al CordicUnrolled.vhd y Cordic.vhd.

Se muestra a continuación capturas de pantallas de la simulación:

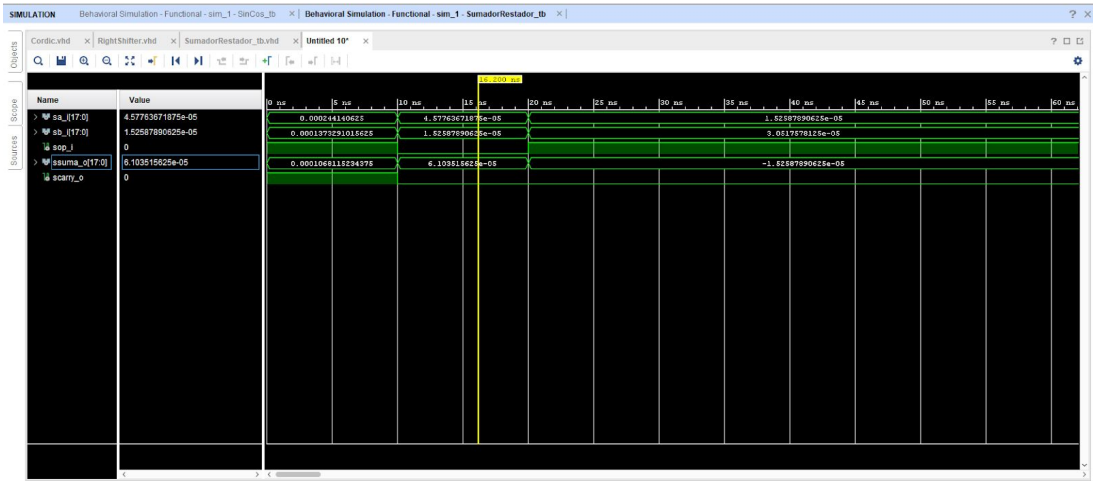

Figura 3. Simulación componente Sumador/Restador

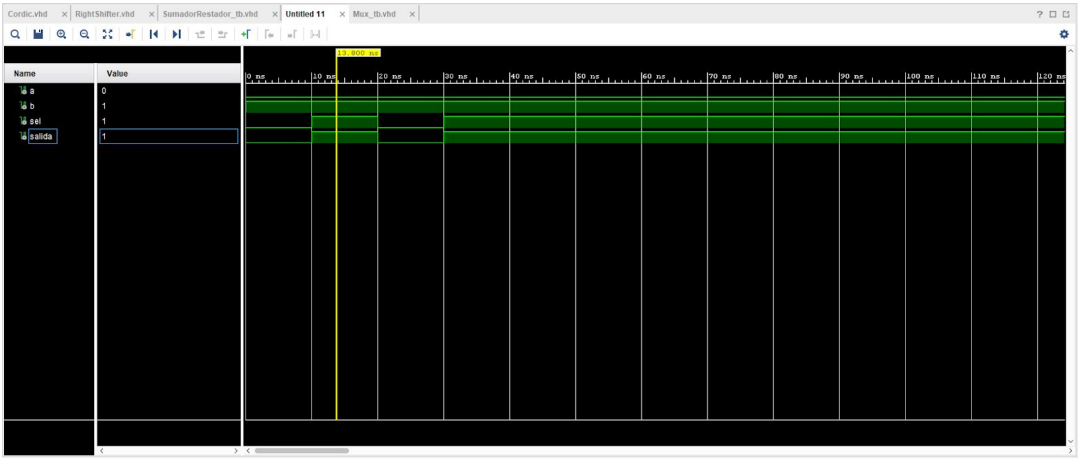

Figura 4. Simulación componente multiplexor

Figura 5. Simulación componente de desplazamiento aritmético

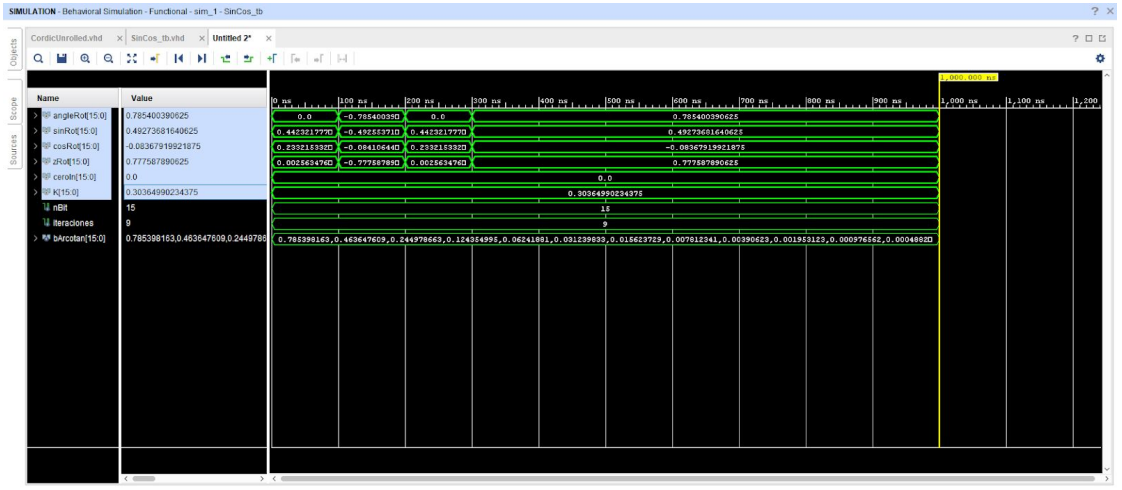

Figura 6. Simulación componente principal - CORDIC

Síntesis e implementación

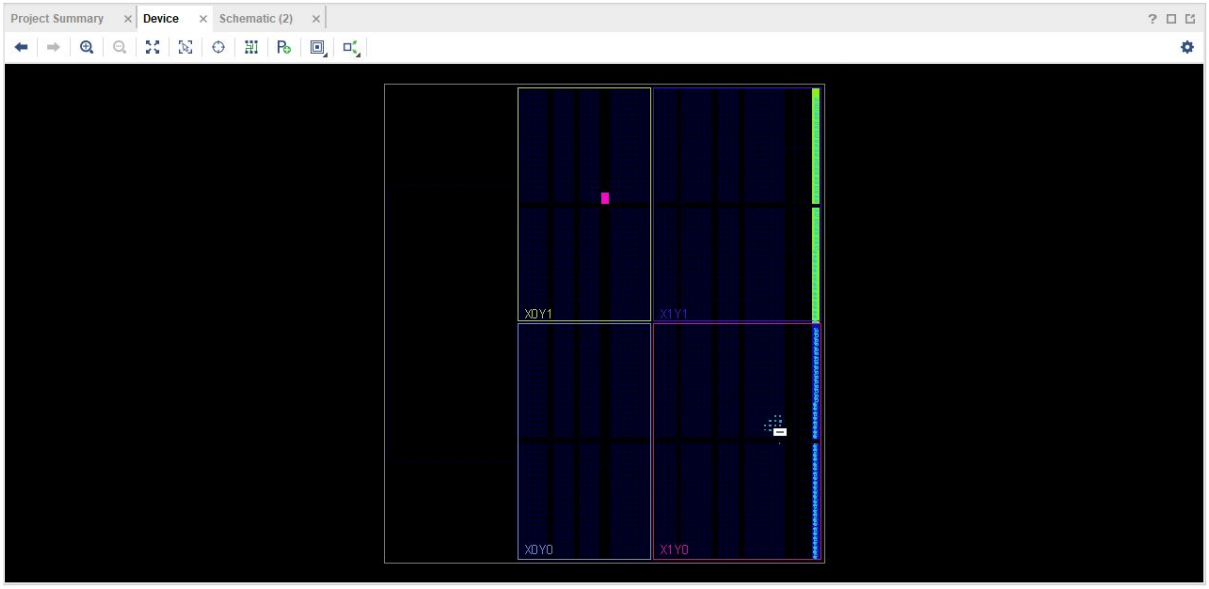

Figura 7. Síntesis del dispositivo

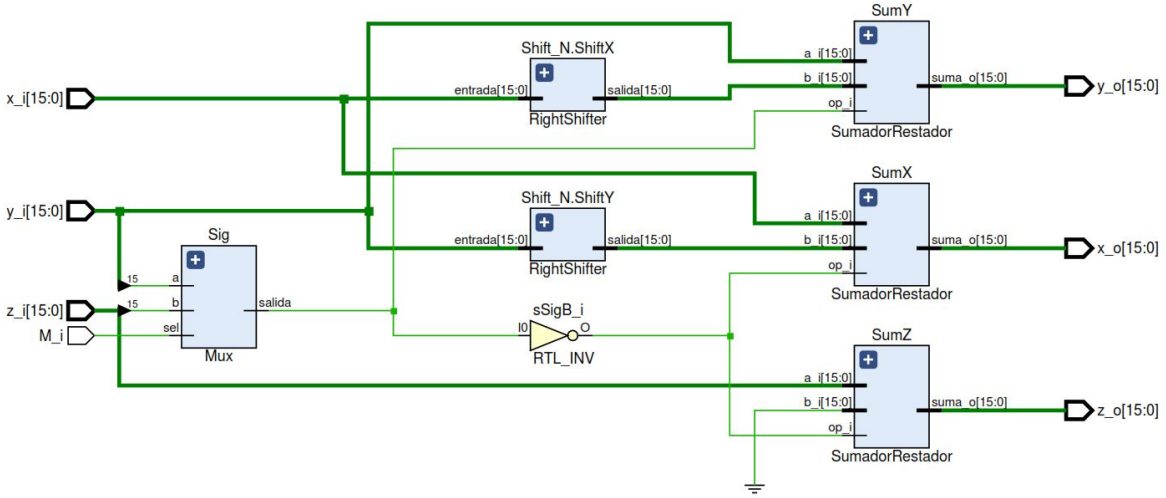

Figura 8. Esquemático del dispositivo