Conclusiones

El presente trabajo planteó como objetivo primario el de estudiar el algoritmo CORDIC y utilizar el lenguaje de descripción de hardware VHDL para describir algunas de sus arquitecturas. Como objetivo secundario se propuso simular las descripciones realizadas modificando los parámetros de interés: ancho de palabra y número de iteraciones, para determinar la exactitud que se obtiene en los resultados. La simulación se llevó a cabo con el cálculo de las funciones seno, coseno y arcotangente.

Acerca del uso de VHDL

- VHDL permite diseñar, modelar y verificar un sistema desde un alto nivel de abstracción, refinando el diseño con la posibilidad de culminar con la descripción del mismo hasta el nivel de compuertas. La posibilidad de describir un sistema digital con un alto nivel de abstracción es importante para comprender inicialmente el funcionamiento del mismo, sin necesidad de codificarlo previamente en otro lenguaje.

- Al estar basado en un estándar (IEEE Std 1076-1987 y IEEE Std 1076-1993) puede utilizarse para minimizar problemas de portabilidad.

- El lenguaje permite, de ser necesario, verificar cada componente del sistema por separado.

- Las simulaciones funcionales en VHDL se pueden llevar a cabo fácilmente mediante la descripción algorítmica de los bancos de prueba. Si bien las simulaciones exhaustivas llevan mucho tiempo, se pueden realizar de manera sencilla, ya que las diferentes metodologías existentes para describir bancos de prueba permiten generar gran cantidad de patrones de verificación.

- Los diseños realizados con VHDL no sólo pueden ser simulados, sino también sintetizados con herramientas adecuadas.

- Los componentes descriptos con VHDL para un diseño pueden reutilizarse posteriormente en otros diseños.

- Con el lenguaje VHDL el diseño de sistemas digitales no sólo está limitado a la ingeniería, sino que abre inmensas posibilidades para programadores en general.

- El uso de VHDL no sólo es importante en la industria, sino también para la enseñanza ya que provee una forma legible y estándar de especificar y describir sistemas digitales.

Acerca de las descripciones del algoritmo CORDIC

- La descripción funcional algorítmica con un alto nivel de abstracción, posibilitó la comprensión del funcionamiento del algoritmo CORDIC. El correcto funcionamiento de ésta, facilitó la descripción de las dos arquitecturas particulares.

- Los resultados obtenidos al calcular las funciones matemáticas para ambas arquitecturas particulares fueron los mismos, como era de esperar ya que ambos diseños poseen idéntica funcionalidad.

- Los bancos de prueba permitieron llevar a cabo la simulación de las descripciones en forma sencilla. Sin embargo, el tiempo de ejecución de la simulación resultó prolongado en la mayoría de las pruebas y se necesitó gran potencia de cómputo.

- Para realizar la descripción de las dos arquitecturas particulares del algoritmo se tuvieron que estudiar aspectos relacionados con el diseño lógico de circuitos combinatorios y secuenciales. La descripción modular y simulación en VHDL del algoritmo CORDIC permitieron comprender el funcionamiento de muchos de los componentes básicos que lo constituyen, como por ejemplo el flip-flop D, y su interacción con otros componentes del sistema.

- La potencia y expresividad del lenguaje VHDL permitieron describir en forma compacta componentes con cierto nivel de complejidad como por ejemplo el Barrel Shifter y la tabla de búsqueda para la arquitectura iterativa del algoritmo.

Acerca de la exactitud de los resultados numéricos

El estudio analítico detallado de los errores del algoritmo CORDIC no era objetivo del presente trabajo. Sin embargo al analizar los resultados de las simulaciones, pudo inferirse lo siguiente respecto de la exactitud de los mismos.

- La exactitud del algoritmo es dependiente del ancho de palabra utilizado para las tres componentes *X, Y* y *Z*, así como del número de iteraciones efectuadas. Las operaciones fundamentales que efectúa un procesador CORDIC son de suma y desplazamiento en las cuales la longitud finita en el ancho de palabra introduce errores, que resultan acrecentados por la representación de punto fijo.

  Si se considera por ejemplo a las componentes x_i e y_i con un ancho de palabra m, en la etapa i, si $i \leq m$ entonces x_{i+1} e y_{i+1} se actualizan con el valor truncado de $y_i \cdot 2^{-i}$ y $x_i \cdot 2^{-i}$ respectivamente. Si $i > m$ entonces $x_{i+1} = x_i$ e $y_{i+1} = y_i$, y la actualización valdrá cero. Por lo tanto, el ancho de palabra limita el número máximo de iteraciones útiles, y en principio la mayor exactitud se consigue luego de m iteraciones.

- La exactitud de la rotación de un ángulo α está determinada por la exactitud de la aproximación que se obtiene con la suma de los ángulos α_i. Los errores en aquellas sumas, que se producen por operar con valores truncados, pueden cancelar el aporte de un mayor número de iteraciones.

- Además existe otro tipo de error que tiene lugar cuando alguna de las componentes *X* o *Y* es negativa. En el caso de las iteraciones finales, la actualización de las componentes debería acercarse siempre a cero debido a la gran cantidad de desplazamientos a derecha que se efectúan. Esto ocurre únicamente cuando las componentes *X* o *Y* son positivas. Sin embargo cuando son negativas el resultado se acerca a *–1*.

  A modo de ejemplo se puede considerar el número *1Eh* o decimal positivo *30* representado con un ancho de palabra de 8 bits. El valor desplazado a derecha varias veces se acerca a cero. Si ahora se considera el negativo de dicho número con la representación en complemento a dos, su valor sería *E2h*. El valor desplazado a derecha varias veces ya no se acerca a cero, sino a *FFh* que en complemento a dos es igual a *–1*.

Perspectivas sobre trabajos futuros

- Si bien se hicieron simulaciones funcionales, en un futuro sería interesante realizar también simulaciones digitales teniendo en cuenta retrasos propios de los componentes del sistema para una plataforma específica.

- Plantear la implementación del algoritmo sobre una plataforma de lógica programable.

- Extender el algoritmo considerando los casos hiperbólico y lineal.

- Analizar una representación en punto flotante para las componentes *X*, *Y* y *Z*.

- Analizar arquitecturas alternativas y modificaciones a las arquitecturas descriptas. En particular, la arquitectura bit-paralela desplegada puede modificarse agregando un registro entre cada iteración o luego de varias iteraciones originando una arquitectura desplegada tipo “pipeline”. De esta manera se observa un retardo inicial hasta llenar el pipeline pero para el cálculo de secuencias resulta útil ya que se obtiene un resultado por cada ciclo de reloj. Esta modificación resulta sencilla de realizar con la arquitectura descripta. Simplemente se debe intercalar el registro descripto para la versión iterativa entre cada par de etapas. La arquitectura resultante deja de ser puramente combinatoria debido a la inclusión de elementos de memoria.