

Tecnológico de Costa Rica
Escuela de Ingeniería Electrónica

Diseño e Implementación de una Unidad de Punto Flotante FPU.

Informe de Proyecto de Graduación para optar por el título de
Ingeniero en Electrónica con el grado académico de Licenciatura

Jeffry Ignacio Quirós Fallas

Borrador de 9 de marzo de 2016

Declaro que el presente Proyecto de Graduación ha sido realizado enteramente por mi persona, utilizando y aplicando literatura referente al tema e introduciendo conocimientos propios.

En los casos en que he utilizado bibliografía he procedido a indicar las fuentes mediante las respectivas citas bibliográficas. En consecuencia, asumo la responsabilidad total por el trabajo de graduación realizado y por el contenido del correspondiente informe final.

Jeffry Ignacio Quirós Fallas

Cartago, 9 de marzo de 2016

Céd: 1-1439-0476

Instituto Tecnológico de Costa Rica
Escuela de Ingeniería Electrónica
Proyecto de Graduación
Tribunal Evaluador

Proyecto de Graduación defendido ante el presente Tribunal Evaluador como requisito para optar por el título de Ingeniero en Electrónica con el grado académico de Licenciatura, del Instituto Tecnológico de Costa Rica.

Miembros del Tribunal

Lic. Luis Carlos Rosales
Profesor Lector

Lic. Jose Alberto Díaz García
Profesor Lector

Dr. Alfonso Chacón Rodríguez
Profesor Asesor

Los miembros de este Tribunal dan fe de que el presente trabajo de graduación ha sido aprobado y cumple con las normas establecidas por la Escuela de Ingeniería Electrónica.

Cartago, 9 de marzo de 2016

Instituto Tecnológico de Costa Rica
Escuela de Ingeniería Electrónica
Proyecto de Graduación
Tribunal Evaluador
Acta de Evaluación

Proyecto de Graduación defendido ante el presente Tribunal Evaluador como requisito para optar por el título de Ingeniero en Electrónica con el grado académico de Licenciatura, del Instituto Tecnológico de Costa Rica.

Estudiante: Jeffry Ignacio Quiróa Fallas

Nombre del Proyecto: *Diseño e Implementación de una Unidad de Punto Flotante FPU*

Miembros del Tribunal

Lic. Luis Carlos Rosales
Profesor Lector

Lic. Jose Alberto Díaz García
Profesor Lector

Dr. Alfonso Chacón Rodríguez
Profesor Asesor

Los miembros de este Tribunal dan fe de que el presente trabajo de graduación ha sido aprobado y cumple con las normas establecidas por la Escuela de Ingeniería Electrónica.

Nota final del Proyecto de Graduación: _____

Cartago, 9 de marzo de 2016

Resumen

El resumen es la síntesis de lo que aparecerá en el tesis. Tiene que ser lo suficientemente consiso y claro para que alguien que lo lea sepa qué esperar del resto de la tesis si la leyera completamente. Puede concluir con palabras clave, que son los temas principales tratados en el documento. El resumen queda fuera de la numeración del resto de secciones.

No se acostumbra utilizar referencias bibliográficas, tablas, o figuras en el resumen.

Palabras clave: CORDIC, seno, coseno, FPU, precisión simple, precisión doble, FPGA

Abstract

The same as before, but in English.

Keywords: CORDIC, sine, cosine, FPU, simple precision, double precision, FPGA.

a mis queridos padres

Agradecimientos

Jeffry Ignacio Quirós Fallas

Cartago, 9 de marzo de 2016

Índice general

Índice de figuras	iii
Índice de tablas	v
Revisar	vii
1 Introducción	1
1.1 Objetivos y estructura del documento	2
2 Marco teórico	3
2.1 Estándar IEEE 754	3
2.1.1 Formatos de precisión en punto flotante.	5
2.2 Algoritmo CORDIC para el cálculo de operaciones trigonométricas.	8
2.2.1 Fundamento Teórico.	8
2.3 Arquitecturas de implementación del algoritmo CORDIC	12
2.3.1 Arquitectura Bit-Paralela Iterativa.	12
2.3.2 Arquitectura Bit-Paralela Desplegada.	13
2.3.3 Ecuaciones	14
2.3.4 Referencias bibliográficas	16
2.3.5 Extensión	16
2.4 Sobre esta plantilla \LaTeX	16
2.4.1 Marcar asuntos pendientes	16
2.4.2 Índices	17
3 Solución propuesta	19
4 Resultados y análisis	21
5 Conclusiones	23
Bibliografía	25
A Demostración del teorema de Nyquist	27

Índice de figuras

2.1	Acomodo de bits para la representación binaria de un número en punto flotante utilizando el estándar IEEE 754-2008.	6
2.2	Ordenamiento de bits en el formato de Precisión Simple.	7
2.3	Ordenamiento de bits en el formato de Precisión Doble.	7
2.4	Vector bidimensional antes y después de ser rotado.	8
2.5	Pseudorotación de un vector bidimensional.	9
2.6	Arquitectura Bit-Paralela Iterativa para la implementación en hardware del algoritmo CORDIC	13
2.7	Arquitectura Bit-Paralela Desplegada para la implementación en hardware del algoritmo CORDIC	14

Índice de tablas

2.1	Valores de los parámetros que definen los formatos básicos de números en punto flotante.7	4
2.2	Interpretación de patrones de bits del estándar IEEE 754 para el formato de precisión simple y doble.	6

Revisar

resultado de chk.....	17
explain.....	17
La caja simple	17
Por hacer:.....	17

Capítulo 1

Introducción

En la *introducción* deben quedar completamente claros los siguientes aspectos, cuyo significado depende del tipo concreto de tesis:

- Contexto
- Problema
- Esbozo de solución
- Objetivos y estructura

El contexto corresponde al entorno donde se desarrolla el proyecto, que puede ser el área general de aplicación, un dominio de problemas, etc. El problema concreto se sintetiza usualmente en una frase o pregunta. Esta pregunta debería ser una consecuencia a la que se llega después de realizar el desarrollo del contexto. Del planteamiento del problema se deriva cuál es el objetivo del trabajo en particular, que a su vez debe conducir al lector de forma natural al esbozo de la solución del problema a tratar en la tesis. Generalmente para aclarar la solución se hace uso de un diagrama de bloques o diagrama de flujo general, es decir, desde un nivel de abstracción alto, donde no sea necesario entrar en detalles técnicos. Usualmente este diagrama y su breve explicación dictan cuál debe ser la estructura del resto del tesis, que es mencionada siempre al final de la introducción.

Una buena introducción debe lograr que el lector tenga interés de leer el resto del tesis.

Es recomendable dividir la tesis en secciones, nombradas cada una de acuerdo a su contenido. **Jamás** utilice los nombres de la guía como “*Problema existente e importancia de su solución*”, sino algo como “La deforestación en Costa Rica” o lo que se adecúe a su problema en particular.

Recuerde que en español solo la primera letra del título va en mayúscula (exceptuando nombres propios, por supuesto).

1.1 Objetivos y estructura del documento

Esta plantilla LaTeX tiene como objetivo simplificar la construcción del documento de tesis, presentando ejemplo de figuras y tablas, así como otorgar una plataforma de compilación en GNU/Linux que simplifique la administración de todo el documento.

La última sección de la introducción usualmente sí tiene un título estandar que es “Objetivos y estructura del documento”, donde se presentan *en prosa* los objetivos general y específicos que ha tenido el proyecto de graduación, así como la estructura de la tesis (por ejemplo, “en el siguiente capítulo se esbozan los fundamentos teóricos necesarios para explicar en el capítulo 3 la propuesta realizada...”)

Capítulo 2

Marco teórico

En este capítulo se presentan los conceptos en los que se basa el sistema propuesto para la solución de operaciones trigonométricas, en el formato de representación estándar de números en coma flotante, IEEE 754-2008, en concreto las operaciones seno y coseno. A continuación se brinda información acerca del formato de palabra para la representación binaria de números en punto flotante, ya sea en precisión simple o precisión doble, 32 y 64 bits respectivamente, así como para el manejo de la excepción de resultados *Not a Number*, *NaN*.

Se presenta además las bases teóricas de la rotación de vectores, fundamento teórico en el que se sustenta el algoritmo CORDIC, investigado e implementado para la resolución de las operaciones trigonométricas antes mencionadas.

2.1 Estándar IEEE 754

El estándar IEEE 754 especifica formatos y métodos para aritmética en punto flotante en sistemas computacionales, permitiendo obtener el mismo resultado, dado el mismo dato de entrada, sin importar si el proceso es llevado a cabo en hardware, software, o una combinación de ambos [1]. Para lograr esto, los diseñadores del estándar utilizaron el método de notación científica, permitiendo representar un número factorizándolo en dos partes, la magnitud y la potencia [2].

Los formatos de representación de números en punto flotante del estándar, son usados para representar un subconjunto finito de los números reales. Los formatos se caracterizan por: su base (binaria, decimal), precisión (simple, doble, extendida) y por el rango de su exponente, y cada uno puede representar un conjunto único de datos en punto flotante [1].

El estándar define 5 formatos básicos:

- Tres formatos binarios, con codificaciones en longitud de 32, 64 y 128 bits.

- Dos formatos binarios, con codificaciones en longitud de 64 y 128 bits.

Las representaciones de un dato en punto flotante en un formato consiste en:

- Triples (signo, exponente y significando); en una base b , el numero en punto flotante representado por un tripe tiene la siguiente forma:

$$(-1)^{\text{signo}} \times b^{\text{exponente}} \times \text{significando}$$

- $+\infty, -\infty$
- $qNaN, sNaN$

El conjunto finito de números en punto flotante representables dentro de un formato en particular es determinado por determinados parámetros enteros, los cuales son:

- b = la base, 2 (binaria) o 10 (decimal).
- p = el número de dígitos en el significando, es decir, a precisión.
- $emax$ = el máximo exponente.
- $emin$ = el mínimo exponente.

Donde $emin = 1 - emax$ para cada uno de los formatos.

En la Tabla 2.1 se presentan los valores de cada uno de los parámetros para cada uno de los formatos básicos, en el cuál cada formato está identificado por su base y el número de bits de su codificación.

Tabla 2.1: Valores de los parámetros que definen los formatos básicos de números en punto flotante.⁷

parametro	Formato Binario (b=2)			Formato Decimal (b=10)	
	binario(32 bits)	binario(64 bits)	binario(128 bits)	decimal64	decimal 128
p, digitos	24	53	113	16	34
$emax$	+127	+1023	+16383	+384	+6144

Sin importar el formato escogido, los siguientes datos deberán poder ser representados:

- Cualquier número en punto flotante igual o diferente a cero y con signo de la forma:

$$(-1)^s \times b^e \times m$$

, donde

- s es igual a 1 o 0, donde 1 representa a los números negativos y el 0 a los números positivos.
- e corresponde a cualquier numero entero entre $emin \leq e \leq emax$.
- m es un número representado por una cadena de dígitos de la forma $d0 * d1d2...d_{p-1}$, donde d_i es un dígito entero entre $0 \leq d_i \leq b$, por lo tanto m puede ser un valor entre $0 \leq m \leq b$.
- Dos valores infinitos, $+\infty, -\infty$.
- Dos representaciones NaN , $qNaN$ y $sNaN$.

El número positivo más pequeño representable en cada formato es b^{emin} , mientras que el más grande es $b^{emax} * (b - b^{1-p})$. Los números diferentes a cero con magnitudes menores a b^{emin} son llamados subnormales, debido a que sus magnitudes caen en valores entre cero y el número normal más pequeño.

En la notación científica, los números pueden tener exponentes tanto negativos como positivos. En la representación binaria, el exponente debe ser normalizado para ser utilizado, ya que en un sistema computacional un exponente negativo debería ser almacenado utilizando una representación en complemento a 2, pero para fines de comparación entre valores, la comparación de valores en complemento a 2 es más compleja que comparar valores que no posean signo, por lo que al exponente se le suma un valor de sesgo para normalizar dicho exponente, para ubicarlo dentro de los valores positivos siempre. Este valor de sesgo tiene un valor igual a $emin$, por lo que el valor del exponente normalizado vendría dado por (2.1) [3].

$$Exponente_normalizado = Exponente - emin \quad (2.1)$$

2.1.1 Formatos de precisión en punto flotante.

Como se menciona anteriormente, en el estándar IEEE 754 existen varios formatos de representación de números en punto flotante, ya sean binarios, decimales o formatos extendidos.

Los formatos de precisión, o almacenamiento, determinan la cantidad de espacio en memoria que un valor en punto flotante necesita en memoria para ser almacenado, y son utilizados para representar un subconjunto definido finito de números reales. Cada formato se diferencia de los demás por la ubicación del punto decimal, la precisión del dato y el rango de su exponente, así como el hecho de que cada uno puede representar solo un determinado conjunto de números en punto flotante [3].

Los formatos que se desarrollan en este documento tienen una distribución definida de bits, para representar un valor en punto flotante. En la Figura 2.1 se observa el acomodo de los diferentes campos que conforman la representación según el estándar IEEE 754 [1].

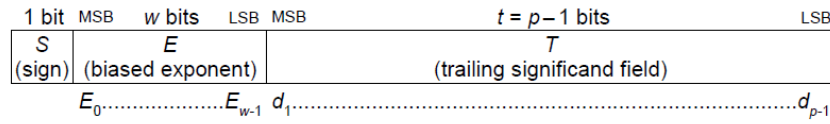


Figura 2.1: Acomodo de bits para la representación binaria de un número en punto flotante utilizando el estándar IEEE 754-2008.

En este documento solo se tratan dos formatos de precisión binarios, el formato de precisión simple y el de precisión doble, y al igual que los formatos decimales y extendidos, se caracterizan por tener un ancho de palabra definido.

Según el patrón de bits presente en la representación binaria de un número en punto flotante, así será su interpretación. En la Tabla 2.2 se presenta la interpretación de dichos patrones según los valores en los campos de la representación, ya sea para precisión simple o doble [3].

Tabla 2.2: Interpretación de patrones de bits del estándar IEEE 754 para el formato de precisión simple y doble.

Tipo de Número	Patrón de bits	
	Precisión Simple	Precisión Doble
Números Normales	Signo = 0 ó 1 $00_h < \text{Exponente Normalizado} < ff_h$ Fracción = Cualquier combinación de bits	Signo = 0 ó 1 $000_h < \text{Exponente Normalizado} < 7ff_h$ Fracción = Cualquier combinación de bits
Números Subnormales	Signo = 0 ó 1 Exponente Normalizado = 00_h Fracción $\neq 00000_h$	Signo = 0 ó 1 Exponente Normalizado = 000_h Fracción $\neq 0000000000000000_h$
Caso Especial Cero con Signo	Signo = 0 ó 1 Exponente Normalizado = 00_h Fracción = 000000_h	Signo = 0 ó 1 Exponente Normalizado = 000_h Fracción = 0000000000000000_h
Caso Especial Infinito Positivo	Signo = 0 Exponente Normalizado = ff_h Fracción = 00_h	Signo = 0 Exponente Normalizado = $7ff_h$ Fracción = 0000000000000000_h
Caso Especial Infinito Negativo	Signo = 1 Exponente Normalizado = ff_h Fracción = 00_h	Signo = 1 Exponente Normalizado = $7ff_h$ Fracción = 0000000000000000_h
Caso Especial Números no representables (Not a Number)	Signo = 0 ó 1 Exponente Normalizado = ff_h Fracción $\neq 00000_h$	Signo = 0 ó 1 Exponente Normalizado = $7ff_h$ Fracción $\neq 0000000000000000_h$

Formato de Precisión Simple.

Este formato de precisión tiene un ancho de palabra definido de 32 bits, de los cuales el campo de signo ocupa 1 bit, el bit más significativo de la cadena de bits, y puede tomar dos valores únicamente, 0 que representa a los números positivos, y 1 que representa a los números negativos. El campo de exponente ocupa los 8 bits después del bit de signo, siguiendo la dirección del MSB al LSB, por lo que el exponente puede tomar cualquier valor desde 0 a 255, donde los valores desde 0 a 127 representan los exponentes negativos, y desde 128 a 255 representan los exponentes positivos. Por último, el campo de mantisa ocupa 23 bits.

En la Figura 2.2 se muestra la distribución de bits de cada campo para el formato de precisión simple.

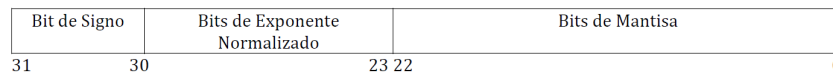


Figura 2.2: Ordenamiento de bits en el formato de Precisión Simple.

La normalización del exponente para este formato se lleva a cabo con un valor de sesgo de 127.

Formato de Precisión Doble.

En el formato de precisión doble se tiene un ancho de palabra definido de 64 bits, de los cuales el campo de signo, al igual que en el formato de precisión simple, ocupa 1 bit, el bit más significativo de la cadena de bits, y puede tomar dos valores únicamente, 0 que representa a los números positivos, y 1 que representa a los números negativos. El campo de exponente ocupa los once bits después del bit de signo, siguiendo la dirección del MSB al LSB.. Por último, el campo de mantisa ocupa 52 bits.

En la Figura 2.3 se muestra la distribución de bits de cada campo para el formato de precisión doble.

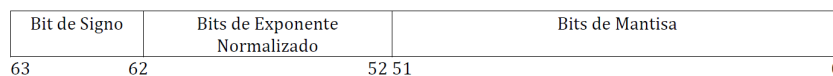


Figura 2.3: Ordenamiento de bits en el formato de Precisión Doble.

La normalización del exponente para este formato se lleva a cabo con un valor de sesgo de 1023.

2.2 Algoritmo CORDIC para el cálculo de operaciones trigonométricas.

El algoritmo de Computación Digital para Rotación de Coordenadas, CORDIC por sus siglas en inglés (**CO**ordinate **R**otation **D**igital **C**omputer), fue desarrollado especialmente para usarse en computadoras digitales en tiempo real, donde la mayoría de la computación involucraba la resolución de las relaciones trigonométricas de las ecuaciones de navegación y una alta tasa de exactitud para las relaciones trigonométricas de las transformaciones de coordenadas [4].

Este algoritmo fue originalmente propuesto por Jack Volder en el año 1959, con el propósito de calcular funciones trigonométricas mediante la rotación de vectores. La rotación de vectores es usada también para realizar la conversión de sistemas coordenados (polar a cartesiano y viceversa), y como parte de funciones matemáticas más complejas como lo son la Transformada Rápida de Fourier (FFT), y la transformada Coseno Discreta (DCT).

El CORDIC es un algoritmo iterativo para rotar vectores a ciertos ángulos determinados, que efectúa operaciones matemáticas a alta velocidad en sistemas coordenados lineales, circulares e hiperbólicos y que solo utiliza operaciones de suma y desplazamiento para efectuar operaciones trigonométricas, exponenciales, logarítmicas, y otras funciones trascendentales [5].

2.2.1 Fundamento Teórico.

El algoritmo originalmente propuesto [4], describe la rotación de un vector bidimensional en el plano cartesiano, como el mostrado en la Figura 2.4. El funcionamiento del algoritmo se deduce de la formula general de rotación de vectores:

$$\begin{aligned} x' &= x \cos \theta - y \sin \theta \\ y' &= y \cos \theta + x \sin \theta \end{aligned} \quad (2.2)$$

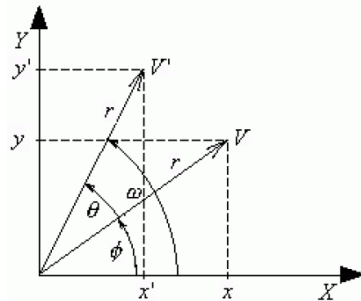


Figura 2.4: Vector bidimensional antes y después de ser rotado.

Sacando a factor común el $\cos \theta$ en la Ecuación (2.2), asumiendo que $\cos \theta \neq 0$, se obtiene

$$\begin{aligned}x' &= \cos\theta(x - y\tan\theta) \\y' &= \cos\theta(y + x\tan\theta)\end{aligned}\tag{2.3}$$

Y tomando en cuenta que,

$$\cos\theta = \frac{1}{\sqrt{1 + \tan^2\theta}}\tag{2.4}$$

Entonces la Ecuación (2.3) se puede expresar como

$$\begin{aligned}x' &= \frac{x - y\tan\theta}{\sqrt{1 + \tan^2\theta}} \\y' &= \frac{y + x\tan\theta}{\sqrt{1 + \tan^2\theta}}\end{aligned}\tag{2.5}$$

En el algoritmo CORDIC, las rotaciones son substituidas por pseudorotaciones vectoriales, tal y como se muestra en la Figura 2.5.

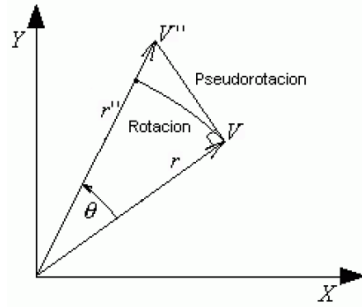


Figura 2.5: Pseudorotación de un vector bidimensional.

En una rotación real no se cambia la magnitud del vector V una vez realizada la rotación, pero en una pseudorotación esto no aplica, ya que incrementa su magnitud en

$$V'' = V\sqrt{1 + \tan^2\theta}\tag{2.6}$$

La rotación de un ángulo se puede descomponer en una sumatoria de rotaciones más pequeñas. Asumiendo inicialmente que $x = x_0$, $y = y_0$ y $z = z_0$, después de realizadas n iteraciones para una rotación real, se obtiene

$$x'_n = x\cos\theta\left(\sum_{i=0}^{n-1}\theta_i\right) - y\sin\theta\left(\sum_{i=0}^{n-1}\theta_i\right)\tag{2.7}$$

$$y'_n = y\cos\theta\left(\sum_{i=0}^{n-1}\theta_i\right) + x\sin\theta\left(\sum_{i=0}^{n-1}\theta_i\right)\tag{2.8}$$

$$z'_n = z - \left(\sum_{i=0}^{n-1} \theta_i \right) \quad (2.9)$$

En cambio para una pseudorotación se obtiene

$$x''_n = (x \cos \theta \left(\sum_{i=0}^{n-1} \theta_i \right) - y \sin \theta \left(\sum_{i=0}^{n-1} \theta_i \right)) \prod_{i=0}^{n-1} \sqrt{1 + \tan^2 \theta_i} \quad (2.10)$$

$$y''_n = (y \cos \theta \left(\sum_{i=0}^{n-1} \theta_i \right) + x \sin \theta \left(\sum_{i=0}^{n-1} \theta_i \right)) \prod_{i=0}^{n-1} \sqrt{1 + \tan^2 \theta_i} \quad (2.11)$$

$$z''_n = z - \left(\sum_{i=0}^{n-1} \theta_i \right) \quad (2.12)$$

, en donde

$$\sum_{i=0}^{n-1} \theta_i = \theta$$

Las Ecuaciones (2.9) y (2.12) son conocidas como el parámetro *acumulador angular*, ya que incluye las rotaciones realizadas hasta el momento.

Si se restringen los ángulos de rotación de tal manera que $\tan \theta = \pm 2^{-i}$, $i \in N$, la operación multiplicación, que es tan costosa para un sistema computacional en términos de área y tiempo, se ve reducida a una operación de desplazamiento. Además, los diversos ángulos se pueden obtener al realizar una sucesión de rotaciones elementales cada vez mas pequeñas, y en cada iteración, en vez de de determinar si se debe rotar o no, lo que hace es escoger el sentido de rotación.

Tomando como punto de partida las Ecuaciones (2.3), sustituyendo $\tan \theta = \pm 2^{-i}$, y además teniendo en cuenta que $\cos(\theta) = \cos(-\theta)$, las iteraciones se pueden expresar como

$$\begin{aligned} x_{i+1} &= K_i(x_i - y_i d_i 2^{-i}) \\ y_{i+1} &= K_i(y_i + x_i d_i 2^{-i}) \end{aligned} \quad (2.13)$$

en donde $K_i = \frac{1}{\sqrt{1 + 2^{-2i}}}$ y $d_i = \pm 1$ dependiendo del sentido de rotación.

Este valor K_i , debe ser multiplicado por las ecuaciones correspondientes a las pseudorotaciones, para obtener la rotación real, ya que como se menciono anteriormente, las pseudorotaciones aumentan el tamaño del vector, por lo que el valor K_i contrarrestaría este efecto.

Eliminando el valor K_i de las ecuaciones iterativas, se obtiene un algoritmo basado en sumas y desplazamientos, y factor K_i puede aplicarse al final o al principio del proceso como una constante K_n , definida como

$$K_n = \lim_{n \rightarrow \infty} \prod_{i=0}^n K_i \cong 0.607253 \quad (2.14)$$

El valor exacto de la constante K_n es determinado por la cantidad de iteraciones realizadas.

En cada paso de iteración, el algoritmo, en vez de decidir si se rota o no, lo que decide es el signo o sentido de rotación que se efectuara. por esto mismo, cada ángulo final se puede representar mediante un vector de signos, en donde cada componente corresponde a un ángulo de la secuencia de ángulos elementales. Estos ángulos se almacenan en una Look-Up Table(LUT). Dependiendo del sistema angular, se almacenan en la tabla las arcotangentes correspondientes a ese sistema. Con esta último punto, se puede modificar el acumulador angular, obteniendo

$$z_{i+1} = z_i - d_i \arctan(2^{-i}) \quad (2.15)$$

El algoritmo CORDIC puede trabajar u operar en dos modos, *rotación*(rotation) o *vectorización*(vectoring). En primero rota el vector de entrada un ángulo específico que se introduce como parámetro. El segundo modo rota el vector de entrada hacia el eje X , acumulando el ángulo necesario para efectuar dicha rotación.

En el caso del modo de *rotación*, el acumulador angular es inicializado con el ángulo a rotar, y la decisión sobre el sentido de rotación en cada iteración, se realiza para minimizar la magnitud del ángulo acumulado, hasta llegar a cero, con lo que el signo que determina el sentido de rotación, se obtiene del valor del acumulador angular en cada iteración.

Para el modo de *rotación*, las ecuaciones iterativas son

$$\begin{aligned} x_{i+1} &= x_i - y_i d_i 2^{-i} \\ y_{i+1} &= y_i + x_i d_i 2^{-i} \\ z_{i+1} &= z_i - d_i \arctan(2^{-i}) \end{aligned} \quad (2.16)$$

$$\text{en donde, } d_i = \begin{cases} -1, & z_i < 0 \\ +1, & z_i \geq 0 \end{cases}$$

Para el modo de *vectorización*, el ángulo que ingresa se rota hasta quedar alineado con el eje X . Este resultado se logra minimizando la magnitud del componente y , en vez de la magnitud del acumulador angular. a diferencia del modo de *rotación*, se utiliza el signo de la variable y para definir la dirección de rotación de la siguiente iteración. Si se inicia el acumulador angular con el valor cero, al finalizar el proceso esta variable contendrá el ángulo de rotación adecuado.

Para el modo de *vectorización*, las ecuaciones iterativas son

$$\begin{aligned}
x_{i+1} &= x_i - y_i d_i 2^{-i} \\
y_{i+1} &= y_i + x_i d_i 2^{-i} \\
z_{i+1} &= z_i - d_i \arctan(2^{-i})
\end{aligned} \tag{2.17}$$

en donde, $d_i = \begin{cases} -1, y_i \geq 0 \\ +1, y_i < 0 \end{cases}$

Cabe destacar que el algoritmo CORDIC está restringido a rango de ángulos $\frac{-\pi}{2} \leq \theta \leq \frac{\pi}{2}$, esto debido a que los ángulos elementales convergen sólo dentro de este rango, y para tratar ángulos por fuera de ese rango, estos se deben de reducir al primer o cuarto cuadrante.

2.3 Arquitecturas de implementación del algoritmo CORDIC.

2.3.1 Arquitectura Bit-Paralela Iterativa.

La arquitectura bit-paralela es una de las más utilizadas a la hora de implementar el algoritmo CORDIC en hardware. Se denomina paralela por la forma en que se opera con las componentes X , Y y Z . Cada etapa de esta implementación consiste en un registro para almacenar los valores de salida, una unidad del desplazamiento y un sumador algebraico.

Cuando se inicia el cálculo, los valores iniciales para x_0, y_0 y z_0 ingresan de manera paralela a los registros a través de los multiplexores de 2x1. El bit más significativo de la variable X o Y , dependiendo del modo, en cada paso iterativo determina la operación a efectuar por el sumador. Seguidamente, las señales de las variables X y Y son desplazadas, y luego sumadas o restadas a las mismas señales sin desplazar, correspondiente a la variable opuesta.

La variable Z suma o resta los valores almacenados en el registro con los valores almacenados en una Look-Up Table, (LUT), de arcotangentes precalculadas con una cantidad de entradas proporcional a la cantidad de iteraciones. Dicha LUT puede ser implementada en una memoria ROM. Cuando se alcanza la iteración n , el valor a calcular se puede obtener en la salida.

Esta arquitectura tiene como ventaja el uso eficiente de hardware, debido a que los recursos son reutilizados en cada iteración. En la Figura 2.6 se muestra el diagrama correspondiente a la arquitectura antes descrita. A esta arquitectura además, se le debe implementar una máquina de control que controle el flujo de los datos, además de controlar las iteraciones realizadas, el canal de mux a activar, entre otras cosas.

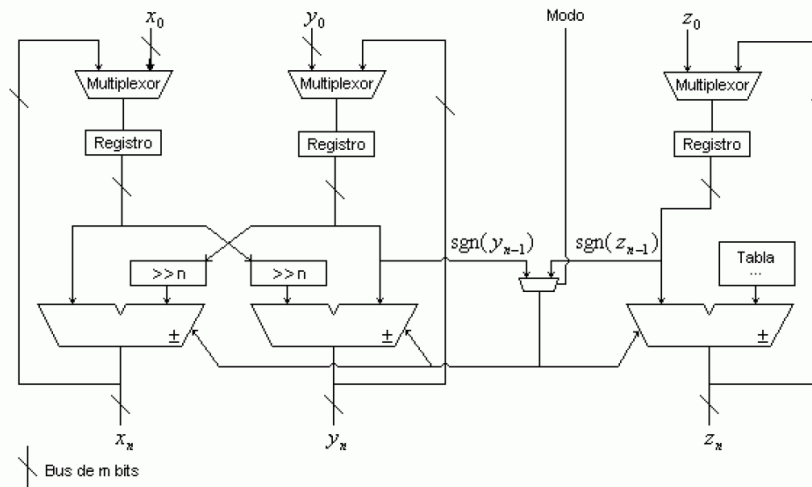


Figura 2.6: Arquitectura Bit-Paralela Iterativa para la implementación en hardware del algoritmo **CORDIC**.

2.3.2 Arquitectura Bit-Paralela Desplegada.

Esta arquitectura, en vez de almacenar el resultado de cada iteración en registros y volver a utilizar los mismos recursos, tal como lo hacía la arquitectura anterior, despliega su diseño de manera que se separa en etapas correspondientes a cada iteración. Cada una de las etapas está compuesta por los mismos componentes, las dos unidades de desplazamiento y tres sumadores algebraicos, por lo que las salidas de una etapa corresponden a la entrada de la siguiente, y el diseño se vuelve completamente combinatorio.

El desarrollar el algoritmo CORDIC por medio de esta arquitectura provee dos ventajas:

- Las unidades de desplazamiento y las constantes correspondientes a cada iteración pueden ser cableadas.
- Al ser el diseño completamente combinatorio, no se requiere de una máquina de control, lo que simplifica bastante el diseño.

Pero, presenta una gran desventaja, debido a la gran cantidad de recursos de hardware que consume, ya que en este diseño se requiere replicar cada etapa n veces, igual a la cantidad de iteraciones que tomaría en la arquitectura bit-paralela iterativa, para conseguir un resultado similar.

En la Figura 2.7 se observa el esquema de la arquitectura Bit- Paralela Desplegada, donde se observa que cada componente presente en la arquitectura Bit-Paralela Iterativa, a excepción de los registros y de la *LUT*, que es cambiada por constantes en cada sumador algebraico utilizado en el acumulador algebraico, la variable z , se debe repetir n veces.

Imágenes vectoriales

Las imágenes vectoriales **deben** ser empleadas en todo tipo de diagrama. En ellas no se almacenan píxeles, sino las estructuras geométricas que componen la figura como círculos (representado por posición de su centro y su radio), rectángulos (representados por sus esquinas), líneas, texto, etc. La mayoría de programas para elaborar este tipo de diagramas, como Inkscape, XFig, OpenOffice.org Draw, MS Visio, Adobe Illustrator, etc. proveen varios formatos vectoriales que pueden ser insertados tanto en LaTeX como en OpenOffice.org Writer (o MS Word). Los formatos más empleados son los llamados metafiles, que incluyen al WMF, EMF. En LaTeX se utiliza por lo general EPS. Recientemente se ha incrementado el soporte al formato SVG.

No debe cometerse el error de generar una imagen vectorial a partir de una imagen raster, pues una vez realizada la discretización espacial no es posible reconstruir los elementos geométricos que componen la imagen. Por ello, no tiene ningún sentido generar un archivo EPS o WMF a partir de una imagen ya almacenada en BMP, JPG, o PNG, pues lo único que ocurrirá es que se inserta la figura raster tal cual en la imagen vectorial, sin implicar ninguna ganancia en la calidad.

Esta plantilla de LaTeX administra la generación de ciertas figuras por usted. Puede colocar en el directorio `fig/` archivos EPS, JPG, PNG o GP (de GNUPlot) y el Makefile se encarga de hacer todas las conversiones necesarias. En las siguientes subsecciones se describen dos casos adicionales que resultan útiles para realizar figuras más complejas.

Figuras `ltxfig`/`psfrag`

Cuando en el subdirectorio `fig/` se encuentran dos archivos con el mismo nombre pero extensiones `ltxfig` y `psfrag`, por ejemplo `prueba.ltxfig` y `prueba.psfrag`, entonces el Makefile asume que usted desea crear una figura a partir del archivo `prueba.ltxfig`, creado con el programa XFig, sustituyendo los textos ahí presentes con texto formateado con LaTeX.

La figura ?? ha sido creada con este esquema. Revise los archivos correspondientes en el directorio de figuras `fig/ltxfig_prototipo.*` para más detalles sobre su uso.

Figuras `pstricks`

Los archivos con extensión `.pstricks` en el directorio `fig` se utilizan para generar cualquier tipo de imágenes según el código que se contenga. Es un concepto más general que el anterior. La figura ?? ha sido creada con este esquema. Puede revisar los archivos `prototipo_gnuplot*` como un ejemplo de su uso, en donde de un archivo `gnuplot` (`..gp`) se genera un archivo `..eps`, el cual es incluido en el archivo `.pstricks` sustituyendo cadenas de texto por código LaTeX.

Entradas en el índice de figuras

El índice de figuras debe servir para encontrar rápidamente dónde se encuentra cierta figura. El pie de la figura, indicado en L^AT_EX con `caption` puede ser extenso, en especial para indicar detalles de las figura, y es la entrada por defecto que aparecerá en el índice de figuras, la cual no debe superar la extensión de una línea y debe únicamente dar la idea del contenido de la figura para poder ser encontrada. Para lograr esto en L^AT_EX se utiliza

```
\caption[Texto en el índice]{Texto al pie de la figura}
```

2.3.4 Referencias bibliográficas

2.3.5 Extensión

Una tesis de licenciatura no debe sobrepasar las 120 páginas incluyendo apéndices y los formalismos desde portada hasta índices.

El cuerpo de la tesis (desde introducción hasta conclusiones) usualmente se extiende desde 45 páginas hasta no más de 80, dependiendo de la problemática tratada.

No es necesario reproducir contenidos de otras fuentes: agregue las referencias a dichas fuentes, y límitese a enunciar lo estrictamente necesario para comprender sus propuestas de solución.

2.4 Sobre esta plantilla L^AT_EX

Esta plantilla L^AT_EX pretende simplificar varios pasos en la creación del documento de tesis.

2.4.1 Marcar asuntos pendientes

La plantilla tiene dos “*modos*” de operación: normal y borrador (*draft*). En el archivo `main.tex` a partir de la línea 41 usted encuentra el código

```
%
% DRAFT MODE
%
\newboolean{draftmode}           % boolean used to control draft-mode
% Ensure that only one of the next two lines is active:
\setboolean{draftmode}{true}      % turn draft mode on
%\setboolean{draftmode}{false}    % turn draft mode off
```


Con el modo borrador, se activan ciertos comandos y funcionalidades útiles en el proceso de elaboración de la tesis, pero que deben ser desactivados al final, antes de entregar la tesis. Por ejemplo, se activa el pie de página que dice “*Borrador: fecha*”, y se activa el índice titulado “Revisar”. En dicho índice aparecen las páginas en donde se hayan utilizado alguno de los siguientes comandos:

- `\boxcomment{comentario}` Crea una caja en el margen de página con el comentario indicado.
- `\explain{comentario}` Crea una caja en el margen de página con el comentario indicado, con una flecha hacia la derecha para indicar qué en concreto debe ser revisado.
- `\chk{comentario}` Crea una caja en el margen con símbolo de “chequeado” y el comentario indicado.
- `\TODO{comentario}` Crea una caja grande de fondo sombreado con el comentario indicado.

En este párrafo se utilizan algunos de estos comandos para ilustrar su efecto. El `\chk` como puede observar tiene sentido usarlo para marcar que algo está casi listo. Por otro lado el comando `\explain` permite marcar algo que requiere ser revisado en redacción, valores, etc. El `\boxcomment` solo pone una marca al margen.

✓ resultado de chk
↩ explain
La caja simple

Por hacer:

Finalmente el comando `TODO` coloca esta caja gris.

Si usted desativa el modo draft, desaparecen todas las páginas, y desaparece el índice “Revisar”. En éste índice aparecen todas las páginas en donde se utilizaron estos comandos con los respectivos comentarios, lo que permite encontrar rápidamente detalles que usted indicó que debe revisar.

2.4.2 Índices

Como índice se conoce la lista de términos claves con su respectiva página, al final del documento. La plantilla ofrece varios comandos para simplificar el uso estandar del comando de \LaTeX `\index{termino}` que coloca al término indicado en el índice. Con `\nt[indice]{termino}` (*new term*) usted indica la entrada principal del término, que aparece en el texto en el índice, es decir, en el índice aparece lo que indique en vez de “índice” y en el texto aparece lo que indique “termino”; `\ot{termino}` agrega una entrada secundaria al término.

Capítulo 3

Solución propuesta

Primero que todo, jamás utilice el título indicado arriba, sino algo relacionado con su solución: “Sistema de corrección de distorsión” o lo que competa a su tesis en particular.

Este capítulo puede separarse en varias secciones, dependiendo del problema concreto. Aquí los algoritmos o el diseño del sistema deben quedar lo suficientemente claros para que otra persona pueda re-implementar al sistema propuesto. Sin embargo, el enfoque no debe nunca concentrarse en los detalles de la implementación particular realizada, sino del diseño conceptual como tal.

Recuerdese que toda figura y tabla deben estar referenciadas en el texto.

Capítulo 4

Resultados y análisis

En tesis formales en este capítulo se exponen los diseños experimentales realizados para comprobar el funcionamiento correcto del sistema. Por ejemplo, si se realiza algún sistema con reconocimiento de patrones, usualmente esta sección involucra las llamadas *matrices de confusión* donde se compactan las estadísticas de reconocimiento alcanzadas. En circuitos de hardware, experimentos para determinar variaciones contra ruido, etc. También pueden ilustrarse algunos resultados concretos como ejemplo del funcionamiento de los algoritmos. Puede mostrar por medio de experimentos ventajas, desventajas, desempeño de su algoritmo, o comparaciones con otros algoritmos.

Recuerde que debe minimizar los “saltos” que el lector deba hacer en su documento. Por tanto, usualmente el análisis se coloca junto a tablas y figuras presentadas, y debe tener un orden de tal modo que se observe cómo los objetivos específicos y el objetivo general del proyecto se han cumplido.

Capítulo 5

Conclusiones

Las conclusiones no son un resumen de lo realizado sino a lo que ha llevado el desarrollo del proyecto, no perdiendo de vista los objetivos planteados desde el principio y los resultados obtenidos. En otras palabras, qué se concluye o a qué se ha llegado después de realizado el proyecto de graduación. Un error común es “concluir” aspectos que no se desarrollaron en la tesis, como observaciones o afirmaciones derivadas de la teoría directamente. Esto último debe evitarse.

Es usual concluir con lo que queda por hacer, o sugerencias para mejorar los resultados.

Bibliografía

- [1] Ieee standard for floating-point arithmetic. *IEEE Std 754-2008*, pages 13–21, Aug 2008. 3, 5
- [2] The Oxford Math Center. Ieee 754 format [online]. 2015 [visitado el 9 de marzo de 2016]. URL <http://www.oxfordmathcenter.com/drupal7/node/43>. 3
- [3] Diego Andrés Rodríguez Valverde. Diseño e implementación de una unidad aritmético-lógica de coma flotante para un procesador de aplicación específica., 2016. 5, 6
- [4] J. E. Volder. The cordic trigonometric computing technique. *IRE Transactions on Electronic Computers*, EC-8(3):330–334, 1959. 8
- [5] Jie Zhou, Yong Dou, Yuanwu Lei, Jinbo Xu, and Yazhuo Dong. Double precision hybrid-mode floating-point fpga cordic co-processor. In *High Performance Computing and Communications, 2008. HPCC '08. 10th IEEE International Conference on*, pages 182–189, 2008. ID: 1. 8

Apéndice A

Demostración del teorema de Nyquist

El título anterior es solo un ejemplo ilustrativo. Éste teorema no ameritaría un apéndice pues es parte normal del currículum de Electrónica, pero apéndices usualmente involucran aspectos de esta índole, que se salen de la línea de la tesis, pero que es conveniente incluir por completitud.

Los anexos contienen toda información adicional que se considere pertinente agregar, como manuales de usuario, demostraciones matemáticas que se salen de la línea principal de la tesis, pero que pueden considerarse parte de los resultados del trabajo.

