UNIVERSIDAD TÉCNICA PARTICULAR DE LOJA

La Universidad Católica de Loja

ÁREA TÉCNICA

TÍTULO DE INGENIERO EN ELECTRÓNICA Y TELECOMUNICACIONES

Diseño e implementación en FPGA del algoritmo CORDIC vectorial y rotacional para sincronismo de un sistema OFDM

TRABAJO DE TITULACIÓN

AUTORES: Torres Delgado, Jonathan Patricio

Solano de la Sala León, Leonardo Rafael

DIRECTOR: Barragán Guerrero, Diego Orlando

LOJA- ECUADOR

2018

2018

APROBACIÓN DEL DIRECTOR DEL TRABAJO DE TITULACIÓN

M.Sc.

Barragán Guerrero, Diego Orlando.

DOCENTE DE LA TITULACIÓN

De mi consideración:

El presente trabajo de titulación: Diseño e implementación en FPGA del algoritmo CORDIC vectorial y rotacional para sincronismo de un sistema OFDM realizado por Torres Delgado, Jonathan Patricio y Solano de la Sala León, Leonardo Rafael, ha sido orientado y revisado durante su ejecución, por cuanto se aprueba la presentación del mismo.

Loja, enero de 2018

f) ……

DECLARACIÓN DE AUTORESÍA Y CESIÓN DE DERECHOS

"Nosotros, Jonathan Patricio Torres Delgado y Leonardo Rafael Solano de la Sala León, declaramos ser Autoreses del presente trabajo de titulación: **"Diseño e implementación en FPGA del algoritmo CORDIC vectorial y rotacional para sincronismo de un sistema OFDM"**, de la Titulación de Electrónica y Telecomunicaciones, siendo el M.Sc. Diego Orlando Barragán Guerrero director del presente trabajo; y eximimos expresamente a la Universidad Técnica Particular de Loja y a sus representantes legales de posibles reclamos o acciones legales. Además, certificamos que las ideas, conceptos, procedimientos y resultados vertidos en el presente trabajo investigativo, son de nuestra exclusiva responsabilidad.

Adicionalmente declaramos conocer y aceptar la disposición del Art. 88 del Estatuto Orgánico de la Universidad Técnica Particular de Loja que en su parte pertinente textualmente dice: "Forman parte del patrimonio de la Universidad la propiedad intelectual de investigaciones, trabajos científicos o técnicos y tesis de grado o trabajos de titulación que se realicen con el apoyo financiero, académico o institucional (operativo) de la Universidad"

f. ……..

Autores: Torres Delgado Jonathan Patricio

Cédula: 1105105348

f. ……..

Autores: Solano de la Sala León Leonardo Rafael

Cédula: 1105677635

DEDICATORIA

Este trabajo de investigación le queremos dedicar a Dios ya que por medio de su iluminación hemos podido culminar dicho trabajo, y a las personas más importantes en nuestras vidas como son nuestros padres, ya que sin el esfuerzo de ellos no hubiéramos podido finalizar con éxito nuestra formación profesional.

AGRADECIMIENTOS

En primer lugar le damos gracias a Dios, por habernos regalado los dones necesarios para culminar con éxito el presente trabajo.

A nuestros Padres por el apoyo y la confianza, especialmente a nuestras madres quienes fueron nuestro pilar fundamental.

A nuestro director de tesis y amigo M.Sc. Diego Barragán por sabernos guiar y corregir durante la realización de este trabajo, gracias por la paciencia y por los consejos.

A nuestra familia y amigos(as) que estuvieron apoyándonos durante toda la carrera.

ÍNDICE DE CONTENIDOS

LISTA DE FIGURAS

LISTA DE TABLAS

RESUMEN

En este trabajo se presenta la implementación de la estimación y corrección del desvío de frecuencia para un sistema OFDM en FPGA, usando los símbolos del preámbulo IEEE 802.11a. Para la detección del CFO fue implementado el algoritmo CORDIC en modo vectorial (para la estimación de la fase) y en modo rotacional (para la estimación de la exponencial compleja) teniendo en cuenta la relación entre el error del algoritmo (diferencia entre valor calculado y valor real) con el número de iteraciones de CORDIC. Para la implementación del algoritmo de sincronismo se utilizó los símbolos STS del preámbulo, realizando una estimación gruesa del desvío de frecuencia. Para verificar el desempeño del algoritmo se generaron símbolos OFDM con un desvío de frecuencia de 100 kHz (valor típico de desvío para este tipo de estándar). Los resultados obtenidos muestran la robustez del algoritmo implementado para el sincronismo de frecuencia de un sistema OFDM.

PALABRA CLAVES: OFDM, CORDIC, CFO, SINCRONISMO, PREÁMBULO, 802.11a, VHDL, FPGA.

ABSTRACT

In this thesis we present the implementation of the estimation and correction of frequency offset for an OFDM system in FPGA, using the IEEE 802.11a preamble symbols. For the detection of the CFO, the CORDIC algorithm was implemented in vectoring mode (for the estimation of the phase) and in rotational mode (for the estimation of the complex exponential) considering the relationship between the algorithm error (the difference between calculated value and real value) with the number of iterations of CORDIC. For the implementation of the synchronization algorithm, the STS symbols of the preamble were used, thus making a coarse estimation of the frequency offset. To verify the performance of the algorithm, OFDM symbols were generated with frequency deviation of 100 kHz (typical deviation value for this type of standard). The results obtained show the robustness of the algorithm implemented for the frequency synchronization of an OFDM system.

Keywords: OFDM, CORDIC, CFO, SYNCHRONISM, PREAMBLE, 802.11a, VHDL, FPGA.

TERMINOLOGÍA

AWGN: Additive White Gaussian Noise

BPSK: Binary Phase Shift Keying

CFO: Carrier Frequency Offset

CORDIC: COordinate Rotation DIgital Computer

dB: Decibelio

DSP: Digital Signal Processing

DFT: Discrete Fourier Transform

FFT: Fast Fourier Transform

FPGA: Field Programmable Gate Array

ICI: Inter Carrier Interference

IDFT: Inverse Discrete Fourier Transform

IFFT: Inverse Fast Fourier Transform

ISI: Inter Symbol Interference

LTS: Long Training Sequence

LTI: Linear Time-Invariant

LUT: Look Up Table

MATLAB: MATrix LABoratory

OFDM: Orthogonal Frequency Division Multiplexing

QPSK: Quadrature Phase Shift Keying

RTL: Resistor Transistor Logic

SNR: Signal to Noise Ratio

STS: Short Training Sequence

VHDL: VHSIC Hardware Description Language

INTRODUCCIÓN

La multiplexación por división de frecuencia ortogonal (OFDM) actualmente es utilizada en numerosos sistemas de comunicación como: WiFi (*Wireless Fifelity*), WiMax (*Worldwide Interoperability for Microwave Access*), LTE (*Long Term Evolution*) de telefonía móvil, TDT (Televisión Digital Terrestre), etc. OFDM es una técnica de modulación multiportadora que utiliza un conjunto de señales que se transmiten en paralelo con diferentes frecuencias, donde cada portadora es separada en varias subportadoras que se modula con un esquema de modulación convencional como QPSK, QAM, etc. Debido al gran número de señales sobrepuestas y ortogonales que se trasmiten en un mismo canal, esta técnica de modulación presenta una elevada eficiencia espectral. Al insertar tiempos más largos que la propagación de retardo del canal, los sistemas OFDM presentan robustez ante interferencias inter-simbólicas e inter-portadoras, por lo que son menos susceptibles ante fenómenos como la distorsión producida por los entornos multitrayectos (Oltean, M., 2004).

Sin embargo, uno de los principales inconvenientes que presenta los sistemas OFDM es su alta sensibilidad a los errores de sincronismo tanto en tiempo como en frecuencia. El sincronismo es una etapa fundamental para la correcta recepción de paquetes (Oltean, M., 2004).

Para superar estos inconvenientes, se busca innovar en el diseño de los bloques funcionales críticos, con el fin de satisfacer las aplicaciones en tiempo real de OFDM, y reducir al mínimo la complejidad del circuito y el coste computacional que se requiere (Alim, O., 2008).

En el presente trabajo se decidió implementar en FPGA (programado en VHDL) el algoritmo CORDIC (*COordinate Rotation DIgital Computer*), en modo rotacional y en modo vectorial, para la estimación y corrección de los errores de sincronismo de frecuencia. Debido a la utilización de sumadores y desplazadores de bits, CORDIC se convierte en un algoritmo simple de implementar y de bajo coste computacional.

Para la simulación del algoritmo de sincronismo de frecuencia se decidió utilizar un software libre de programación, denominado Python. Esta herramienta de programación utiliza un lenguaje de alto nivel, fácil de comprender y que puede ser utilizado en diferentes plataformas y sistemas operativos (Windows, IOS y Linux). Además, este lenguaje de programación no es dedicado a un ámbito específico de desarrollo, por lo que permite desarrollar diversas aplicaciones de software (Marzal, A., Luengo, I., 2004).

Finalmente, para la implementación del algoritmo de sincronismo de frecuencia se decidió utilizar un FPGA (programado en VHDL) usando los símbolos del preámbulo

definido en IEEEE 802.11a, debido a que este hardware digital permite implementar varios procesos al mismo tiempo (paralelismo) a diferencia de un DSP que es secuencial, lo cual convierte a un FPGA en un dispositivo bastante versátil para implementar sistemas de telecomunicaciones.

OBJETIVOS

Objetivo General

- Diseñar e implementar en FPGA el algoritmo CORDIC rotacional y vectorial para sincronismo de un sistema OFDM.

Objetivos específicos

- Describir el modelo matemático de CORDIC vectorial y rotacional.
- Simular en Python-Numpy el algoritmo de CORDIC rotacional y vectorial.
- Simular en VHDL a través de un testbench el algoritmo de CORDIC en modo rotacional y vectorial.
- A partir de los datos de esta simulación, implementar en FPGA el algoritmo CORDIC en modo vectorial para el cálculo de magnitud y fase de una variable compleja.
- Implementar en FPGA el algoritmo CORDIC en modo rotacional para calcular una exponencial compleja con frecuencia variable.
- Implementar el algoritmo propuesto para un sincronismo de frecuencia de un sistema OFDM.

JUSTIFICACIÓN

Es importante tener claros los principales puntos a tratar dentro de esta investigación para realizar su respectiva justificación. Estos puntos son:

1. Simulación del algoritmo de sincronismo en Python.
2. Implementación del algoritmo de sincronismo OFDM mediante VHDL.
3. Análisis del error del algoritmo versus número de iteraciones.
4. Análisis del algoritmo de sincronismo de frecuencia.

El primero tiene importancia debido a que CORDIC es un algoritmo fundamental para el desarrollo de este trabajo, permitiendo, a través de una simulación, comprender a profundidad el funcionamiento del algoritmo, para posteriormente llevarlo a la implementación en VHDL.

La importancia del segundo punto radica en que la implementación de este algoritmo de sincronismo de frecuencia de bajo coste computacional (debido a que CORDIC apenas utiliza sumadores y desplazadores de bits), posibilita la opción de mejorar el análisis de los inconvenientes presentados en el momento de implementarlo en varios sistemas de comunicación, permitiendo generar soluciones adaptables.

El tercer punto es importante debido a que OFDM está sujeto a errores de sincronismo en la recepción de paquetes, por lo que se busca un equilibrio entre número de iteraciones de CORDIC con el error que presenta el algoritmo y poder solucionar este problema.

El último punto es fundamental porque permite analizar si la detección y corrección del desvío de frecuencia fue correcto, a través de la estimación y detección del desvío de frecuencia que presenta el preámbulo IEEE 802.11a al momento de pasar por un canal AWGN.

CAPÍTULO I: MARCO TEÓRICO

1.1 Introducción a FPGA

Cuando se aborda el diseño e implementación de un sistema electrónico con hardware dedicado, son varias las opciones que hay, como: CPLD (*Complex Programmable Logic Device*), MCU (*Microcontroller Unit*), ASIC (*Application Specific Integrated Circuit*), DSP (*Digital Signal Processor*), etc., considerando un buen compromiso coste-prestaciones, una de las opciones son los dispositivos de lógica programable versátiles, denominados FPGA (*Field Programmable Gate Array*).

Los FPGA están compuestos por un conjunto de bloques de lógica configurable (CLBs), junto con conexiones entre esos bloques dispuesto de forma regular, como se muestra en la Figura 1.1, (Patricia, B., 2015). Cada uno de los bloques contiene pequeñas memorias RAM y flips-flops que se pueden configurar para realizar todo tipo de circuitos combinacionales y secuenciales, permitiendo obtener un tiempo de diseño relativamente corto (reduciendo el parámetro: *Time to market*) en comparación con los DSP's (Ayala R., 2015). Los FPGA permiten implementar tablas LUT, que resulta de utilizar parte de una memoria RAM estática, para almacenar información que permita al usuario desarrollar el sistema electrónico deseado.

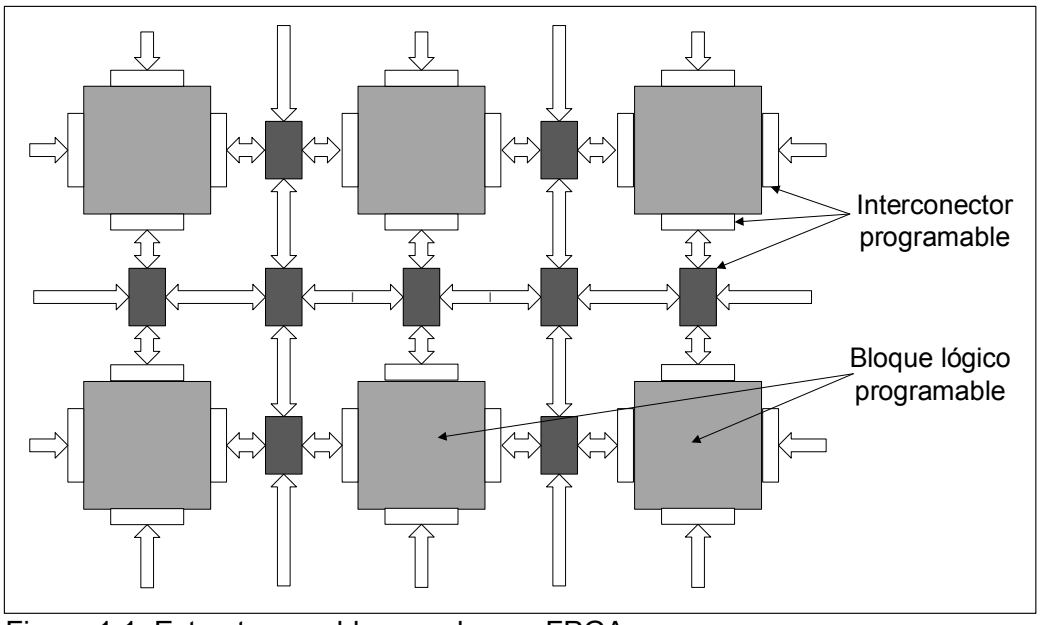


Figura 1.1. Estructura en bloques de una FPGA
Fuente: reproducido de (Ayala R., 2015)
Elaboración: Autores

1.1.1 Ventajas al usar un FPGA

- **Rendimiento:**

  Debido al diseño paralelo del hardware de un FPGA, la latencia de procesamiento computacional se reduce en comparación con los DSP's, permitiendo ejecutar varios procesos al mismo tiempo en cada ciclo de reloj,

funcionalidad que coincide con los requerimientos de cualquier aplicación (National Intruments, 2011).

- **Fiabilidad:**

  Mientras que el núcleo de un DSP sólo puede ejecutar una instrucción a la vez, poniendo siempre en riesgo de que sus tareas se obstruyan entre sí, los FPGA's, que no necesitan sistemas operativos, minimizan los retos de fiabilidad con ejecución paralela y hardware preciso dedicado a cada tarea (National Intruments, 2011).

1.2 Algoritmo CORDIC

El algoritmo CORDIC (*COordinate Rotation DIgital Computer)*, fue planteado por Jack Volder en 1959 (Volder, J.,1959). Este algoritmo fue desarrollado especialmente para computadoras digitales en tiempo real, que a través de registros de desplazamiento y sumadores (en lugar de multiplicadores) permite realizar el cálculo de funciones trigonométricas, conversión de sistemas de coordenadas, etc. La simplicidad de su diseño permite ahorrar muchos recursos en hardware.

El algoritmo CORDIC consiste en girar un vector de entrada de dos dimensiones, como se muestra en la Figura 1.2, permitiendo calcular el ángulo del vector recibido de una constelación por medio de la función $arctan$ y se usa en receptores para encontrar desplazamiento de fase y frecuencia. Este algoritmo resulta de la fórmula general para rotación de vectores, que se detalla a continuación (Joachim, R.,2002):

$$\begin{aligned} x' &= xcos\theta - ysen\theta \\ y' &= ycos\theta + xsen\theta \end{aligned} \tag{1}$$

Usando las identidades trigonométricas en la ecuación (1), se llega a:

$$V:\begin{cases} x = rcos\phi \\ y = rsen\phi \end{cases} \text{ (a)} \qquad V':\begin{cases} x' = rcos\omega \\ y' = rsen\omega \end{cases} \text{ (b)} \tag{2}$$

Si $r = 1$ y $\theta = \omega - \phi$ entonces $\omega = \theta + \phi$. Se tiene:

$$\begin{aligned} sen(\alpha + \beta) &= \text{sen}\,\alpha\cos\beta + \cos\alpha\text{sen}\,\beta \\ cos(\alpha + \beta) &= \cos\alpha\cos\beta - \text{sen}\,\alpha\text{sen}\,\beta \end{aligned} \tag{3}$$

Luego, reemplazando $\omega = \theta + \phi$ en (2b), y usando la identidad trigonométrica de la ecuación (3), se obtiene:

$$\begin{aligned} x' &= \cos(\theta + \phi) = cos\theta cos\phi - sen\theta sen\phi = xcos\theta - ysen\theta \\ y' &= sen(\theta + \phi) = sen\theta cos\phi + cos\theta sen\phi = xsen\theta + ycos\theta \end{aligned} \tag{4}$$

Remplazando $sen\theta = cos\theta tan\theta$ y aplicando factor común se obtiene la ecuación (5) (Edwards, B.H., 1998).

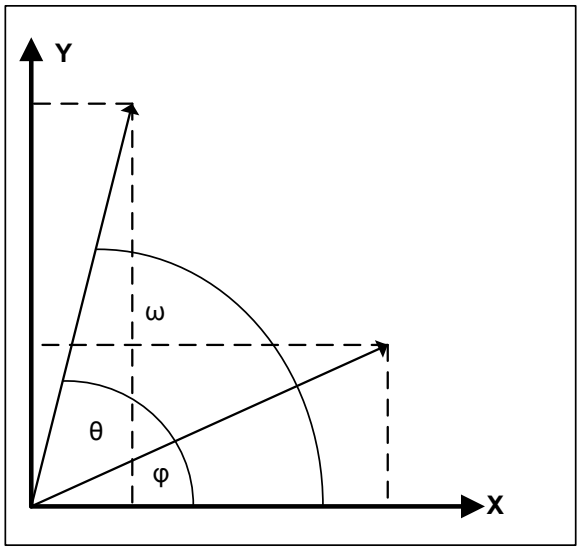


Figura 1.2. Rotación normal de un vector
Fuente: reproducido de (Joachim, R.,2002)
Elaboración: Autores

$$x^{'} = cos\theta(x - ytan\theta)$$
$$y^{'} = cos\theta(y + xtan\theta) \quad (5)$$

Si además:

$$cos\theta = \sqrt{\cos^2\theta} = \sqrt{\frac{1}{\frac{1}{\cos^2\theta}}} = \frac{1}{\sqrt{\frac{1}{\cos^2\theta}}} = \frac{1}{\sqrt{\frac{sen^2\theta + \cos^2\theta}{\cos^2\theta}}} = \frac{1}{\sqrt{\frac{\cos^2\theta}{\cos^2\theta} + \frac{sen^2\theta}{\cos^2\theta}}} \quad (6)$$
$$= \frac{1}{\sqrt{1 + tan^2\theta}}$$

Entonces, reemplazando la ecuación (6) en la ecuación (5), se obtiene:

$$x^{'} = \frac{x - ytan\theta}{\sqrt{1 + tan^2\theta}}$$
$$y^{'} = \frac{y + xtan\theta}{\sqrt{1 + tan^2\theta}} \quad (7)$$

En el algoritmo CORDIC se realiza una pseudorotación en lugar de una rotación como se muestra en la Figura 1.3 (Parhami, B., 2000):

Mientras una rotación normal mantiene la magnitud del vector, en una pseudorotación la magnitud incrementa a la siguiente razón:

$$V^{''} = V\sqrt{1 + tan^2\theta} \quad (8)$$

Y sus componentes son representadas en la ecuación (9).

$$x^{''} = x - ytan\theta$$
$$y^{''} = y + xtan\theta \quad (9)$$

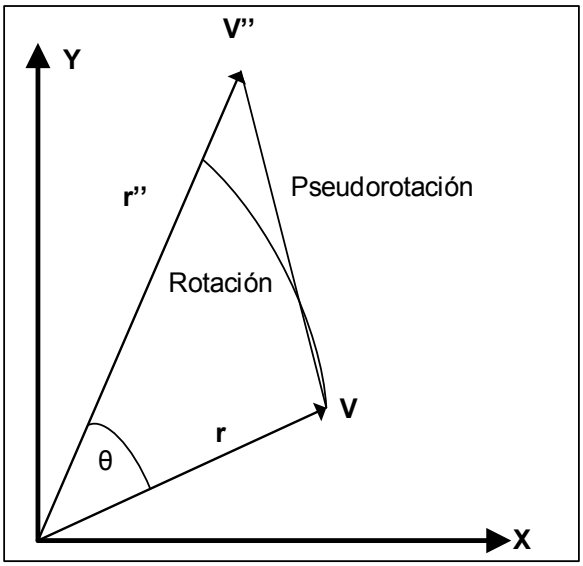


Figura 1.3. Pseudorotación de un vector
Fuente: reproducido de (Joachim, R.,2002)
Elaboración: Autores

La rotación de un ángulo se puede representar como la suma de rotaciones en cada iteración. Sí $x = x_0,\ y = y_0$ y $z = z_0$, y realizando n iteraciones se obtiene:

$$x_n^{'} = xcos\left(\sum_{i=0}^{n-1}\theta_i\right) - ysen\left(\sum_{i=0}^{n-1}\theta_i\right)$$

$$y_n^{'} = ycos\left(\sum_{i=0}^{n-1}\theta_i\right) + xsen\left(\sum_{i=0}^{n-1}\theta_i\right) \qquad (10)$$

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

Y para una pseudorotación se representa con la siguiente ecuación:

$$x_n^{''} = \left(xcos\left(\sum_{i=0}^{n-1}\theta_i\right) - ysen\left(\sum_{i=0}^{n-1}\theta_i\right)\right)\prod_{i=0}^{n-1}\sqrt{1+tan^2\theta_i}$$

$$y_n^{''} = \left(ycos\left(\sum_{i=0}^{n-1}\theta_i\right) + xsen\left(\sum_{i=0}^{n-1}\theta_i\right)\right)\prod_{i=0}^{n-1}\sqrt{1+tan^2\theta_i} \qquad (11)$$

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

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

Si se determina que los ángulos de rotación son iguales a $tan\theta = \pm 2^{-i}, i \epsilon N$, la multiplicación por la tangente se transforma en una operación de desplazamiento.

Entonces se puede dar sentido a la rotación, en lugar de sólo rotar (Parhami, B., 2000). De la ecuación (5), cada iteración se expresa como:

$$x_{i-1} = K_i\left(x_i - y_i d_i 2^{-i}\right)$$
$$y_{i-1} = K_i\left(y_i + x_i d_i 2^{-i}\right) \quad (12)$$

Donde $K_i = cos\theta_i = cos\left(arctan2^{-i}\right) = \frac{1}{\sqrt{1+tan^2\theta_i}} = \frac{1}{\sqrt{1+2^{-2i}}}$ y $d_i = \pm 1$ dependiendo del sentido de la rotación.

K_i al final de las iteraciones se comportará como una constante K_n:

$$K_n = \lim_{n\to\infty} \prod_{i=0}^{n} K_i \cong 0{,}6073$$

K_n varía dependiendo del número de iteraciones, otra forma de representar:

$$A_n = \frac{1}{K_n} = \prod_{i=0}^{n-1} \frac{1}{K_i} = \prod_{i=0}^{n-1} \sqrt{1+2^{-2i}} \cong 1{,}6467$$

Como se menciona anteriormente el signo decide el sentido de la rotaciones, y estos ángulos se almacenan en una tabla de búsqueda (Lookup Table, LUT). Modificando la componente z_n de la ecuación (11) se obtiene la siguiente ecuación:

$$z_{i+1} = z_i - d_i arctan\left(2^{-i}\right) \quad (13)$$

Dependiendo del sistema angular y las condiciones para la rotación se generan dos algoritmos: CORDIC modo vectorial (para el cálculo de la fase) y CORDIC modo rotacional (para el cálculo de las componentes seno y coseno).

1.2.1 Modo Vectorial

En CORDIC modo vectorial, se acumula el ángulo de rotación del vector de entrada, que gira hasta el eje X, aproximando a cero la magnitud de la componente Y del vector (Parhami, B., 2000). Por consiguiente se pueden deducir las siguientes ecuaciones (Joachim, R.,2002):

$$x_{i+1} = x_i - y_i d_i 2^{-i}$$
$$y_{i+1} = y_i + x_i d_i 2^{-i} \quad (14)$$
$$z_{i+1} = z_i - d_i arctan\left(2^{-i}\right)$$

En donde, $d_i = \begin{cases} -1 & , si\ y_i \geq 0 \\ 1 & , si\ y_i < 0 \end{cases}$

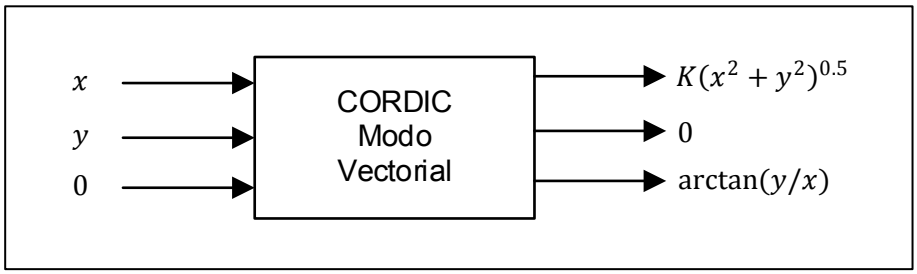


Figura 1.4. Bloque CORDIC vectorial
Fuente: reproducido de (Valls J., 2006)
Elaboración: Autores

1.2.2 Modo Rotacional

En CORDIC modo rotacional, el vector de entrada gira un ángulo que dependiendo de cada iteración adquiere un valor específico, de esta manera se aproxima la magnitud del ángulo a cero. Por este motivo, el sentido de rotación del valor de dicho ángulo se obtiene a continuación (Joachim, R.,2002):

$$\begin{aligned} x_{i+1} &= x_i - y_i d_i 2^{-i} \\ y_{i+1} &= y_i + x_i d_i 2^{-i} \\ z_{i+1} &= z_i - d_i arctan(2^{-i}) \end{aligned} \tag{15}$$

En donde, $d_i = \begin{cases} -1 & , si\ z_i < 0 \\ 1 & , si\ z_i \geq 0 \end{cases}$

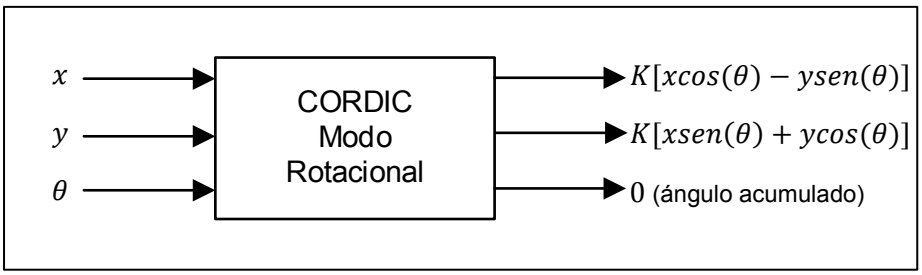


Figura 1.5. Bloque CORDIC rotacional
Fuente: reproducido de (Valls J., 2006)
Elaboración: Autores

1.3 OFDM

La multiplexación por división de frecuencia ortogonal (OFDM) es un esquema de modulación digital que expande el concepto de modulación de subportadora única, mediante el uso de múltiples subportadoras en un mismo canal. Al insertar un tiempo de guarda (que es más largo que la propagación de retardo del canal), los sistemas OFDM son menos susceptibles a la interferencia entre símbolos introducida en el entorno multitrayecto, siendo eficaz contra este tipo de distorsión (Moose, 1994).

La idea de las comunicaciones *multicarrier* es transmitir un flujo de datos, utilizando un gran número de subportadoras ortogonales estrechamente espaciadas que se transmiten en paralelo. Cada subportadora se modula con un esquema de modulación digital común (como QPSK , 16QAM, etc.). Sin embargo, la combinación de varias subportadoras permite velocidades de datos similares a los esquemas de modulación de portadora única convencionales dentro de anchos de banda equivalentes (Poggi, G, 2009).

Las subportadoras se muestrean con una tasa de N/T_s, donde N es el número de subportadoras y T_s es la duración del símbolo OFDM, de esta manera se obtiene un símbolo generado por N subportadoras (Moose, 1994).

En OFDM se transmiten N muestras X_n con $n = 0,1,\ldots,N-1$, donde X_n es el símbolo transmitido en la n-ésima frecuencia de portadora, por la cual la salida del transmisor de multiportadora de valor complejo está dada por (Barry, Lee, Messerschmitt, 2004):

$$x(t) = \sum_{n=0}^{N-1} X_n e^{j2\pi f_n t} \tag{26}$$

La salida del transmisor digital va a tener datos discretos en instantes $t = kT_s$, donde k es un valor entero. Así, la salida del transmisor digital de multiportadoras es dada por:

$$x(kT_s) = \sum_{n=0}^{N-1} X_n e^{j2\pi f_n kT_s} \tag{17}$$

Además, considerando que la separación de frecuencia entre dos subportadoras adyacentes es $N/2\pi$ y que fueron uniformemente espaciadas por un espacio de frecuencias de f_s $(m = 0;1;\ldots;N-1)$ se tiene:

$$x(kT_s) = \sum_{n=0}^{N-1} X_n e^{j2\pi nmkT_s} \tag{18}$$

Para evitar destruir la ortogonalidad de las señales subportadoras, el espacio de frecuencias tiene una separación de $f_s = 1/(NT_s)$, por lo tanto, la señal OFDM viene dada por:

$$x(kT_s) = \sum_{n=0}^{N-1} X_n e^{\frac{j2\pi k_n}{N}} \tag{19}$$

El espectro de las subportadoras OFDM es mostrado en la Figura 1.6.

De la misma manera, considerando una señal OFDM $X_{n,q}$ transmitida en la n-ésima portadora del q-ésimo símbolo OFDM, obtenemos un símbolo complejo de dominio de tiempo, que se representa en la ecuación (20) (Xiao, Cowan, Ratnarajah, Fagan, 2009).

$$x_q(kT_s) = \frac{1}{\sqrt{N}} \sum_{n=0}^{N-1} X_{n,q}\, e^{\frac{j2\pi nk}{N}} \tag{20}$$

Donde $\frac{1}{\sqrt{N}}$ es el factor para que la energía de la señal en el dominio del tiempo sea igual a la energía de la señal en el dominio de la frecuencia (Barry, Lee, Messerschmitt, 2004).

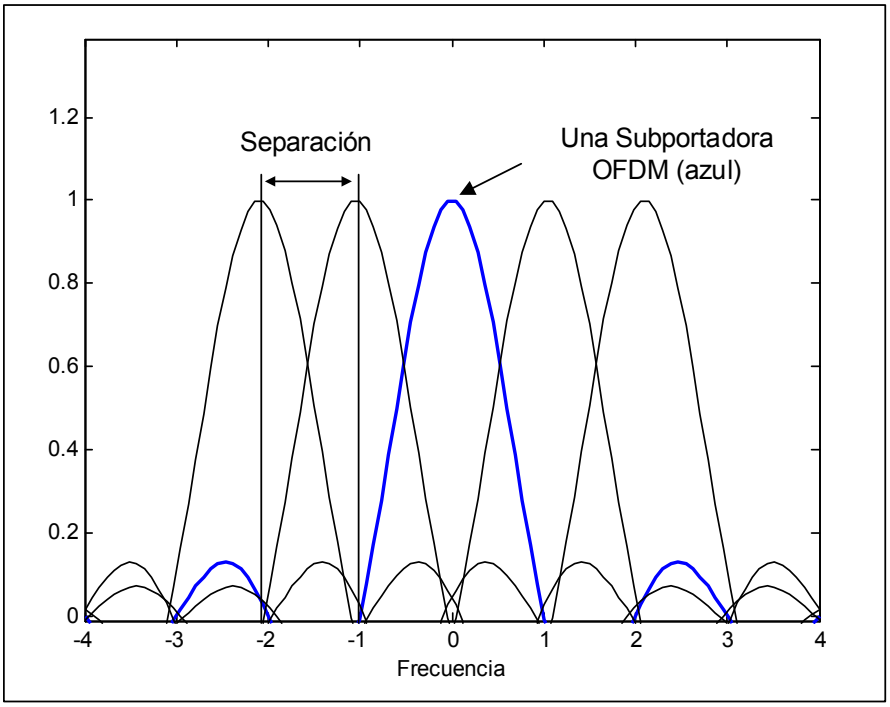


Figura 1.6. Multiplexación por División de Frecuencia Ortogonal (OFDM)
Fuente: reproducido de (Panta, K., Armstrong, J., 2003)
Elaboración: Autores

1.4 Problemas de sincronismo en OFDM

La demodulación y la detección de señales OFDM requieren una sincronización precisa. Por ejemplo, los tiempos de símbolo y los errores de estimación de CFO (*Carrier Frequency Offset*) pueden causar ISI (*Inter-Symbol Interference*) e ICI (*ICI - Intercarrier Interference*) y pueden conducir a una grave degradación del rendimiento del sistema. Para el presente trabajo sólo vamos a considerar la descripción general del esquema de sincronismo CFO.

Como se dijo previamente en la Sección 1.3, uno de los inconvenientes de los sistemas *multicarrier* es su alta sensibilidad a los errores de sincronismo. De hecho, los errores

de estimación y sincronización de CFO inducen una reducción de la amplitud de la señal útil y provoca interferencia entre subportadoras adyacentes (Poggi. G., 2009).

La interferencia inter-simbólica (ISI) se debe a la utilización de un conjunto de muestras del símbolo anterior o posterior en la DFT para obtener la señal en el dominio de la frecuencia, en cambio, la interferencia entre subportadoras (ICI) es consecuencia de la desalineación entre la ventana de la DFT y la señal del símbolo recibido (Poggi, G., 2009).

1.4.1 Diferencia de frecuencia entre portadoras

Considerando un sincronismo temporal donde $\Delta\theta = 0$, la señal recibida en presencia de un canal AWGN, está dada por:

$$r_q(n) = s_q(n)e^{j\left(\frac{2\pi}{N}\varepsilon k + \emptyset\right)} + w_q(n) \tag{21}$$

Donde $\varepsilon = \Delta f T_c N$ es el desvío de frecuencia normalizada debido al espaciamiento entre portadoras $\Delta\theta = \hat{\theta} - \theta$, N es el número de portadoras de la señal OFDM y $\emptyset = \frac{2\pi}{N}\varepsilon q M$ la fase acumulada de cada q-ésimo símbolo OFDM, debido a una diferencia de frecuencia ε igual a una fracción del espacio interportador, puede causar ICI y atenuación en la señal transmitida, por lo tanto, la señal en el receptor se representa como (Poggi, G., 2009):

$$\hat{X}_{i,q} = \sum_{n=0}^{N-1}\left[s_q(n)e^{j\left(\frac{2\pi}{N}\varepsilon k + \emptyset\right)} + w_q(n)\right]e^{-j\frac{2\pi ki}{N}} \tag{22}$$

Luego, teniendo en cuenta la expresión de la señal transmitida en la ecuación (20) tenemos:

$$\hat{X}_{i,q} = e^{j\left[\pi\varepsilon\left(\frac{N-1}{N}\right) + \emptyset\right]}\frac{sen(\pi\varepsilon)}{Nsen\left(\frac{\pi\varepsilon}{N}\right)}X_{i,q} + \xi_{i,q} + W_{i,q} \tag{23}$$

Donde $\xi_{i,q}$, representa la interferencia entre portadoras (ICI), que viene dada por:

$$\xi_{i,q} = \frac{e^{j\phi}}{N}\sum_{h=-,\,h\neq i}^{N-1} X_{h,q}\sum_{k=0}^{N-1} e^{j\frac{2\pi}{N}k(h-i+\varepsilon)} \tag{24}$$

Y donde $W_{i,q}$ representa al ruido blanco en el dominio de la frecuencia y viene dada por:

$$W_{i,q} = \sum_{k=0}^{N-1}[w_q(k)]e^{-\frac{j2\pi ki}{N}} \tag{25}$$

De la ecuación (23) podemos observar que la señal recibida está dada por la suma de tres términos diferentes: el término útil $X_{i,q}$ que presenta atenuación y rotación de fase,

la interferencia entre portadoras (ICI) $\xi_{i,q}$, y la adición de un ruido blanco $W_{i,q}$. El efecto de los errores de sincronización se presenta en la Figura 1.7, donde se muestra el espectro de la señal OFDM en ausencia de errores de sincronismo (líneas continuas) y en presencia de un desvío de frecuencia de portadora CFO $\varepsilon = 0.2$ (líneas punteadas). Como podemos ver, la presencia de un CFO provoca una reducción en la amplitud de la señal y presencia de ICI. En (Pollet, T., 2010), evalúan analíticamente la degradación de la tasa de error de bit (BER) causada por la presencia de CFO para un canal AWGN (Poggi, G., 2009) . Se encuentra que un sistema *multicarrier* es mucho más sensible que el sistema de una sola portadora y, en particular, la degradación en SNR (en dB) se puede aproximar mediante:

$$D(dB) \cong \frac{SNR}{SNR_{e(\varepsilon)}} \approx \frac{10(\pi\varepsilon)^2 SNR}{3\ln 10} = \frac{10(\pi \Delta f T_c N)^2 SNR}{3\ln 10} \tag{26}$$

Por lo tanto, los errores de la degradación en dB, inducida por la presencia de CFO, están limitados por:

$$D(dB) \cong \frac{SNR}{SNR_{e(\varepsilon)}} \leq 10\log_{10}\left[\frac{1 + 0.0.5947\text{SNR}\sin^2(\pi\varepsilon)}{sinc^2(\varepsilon)}\right] \tag{27}$$

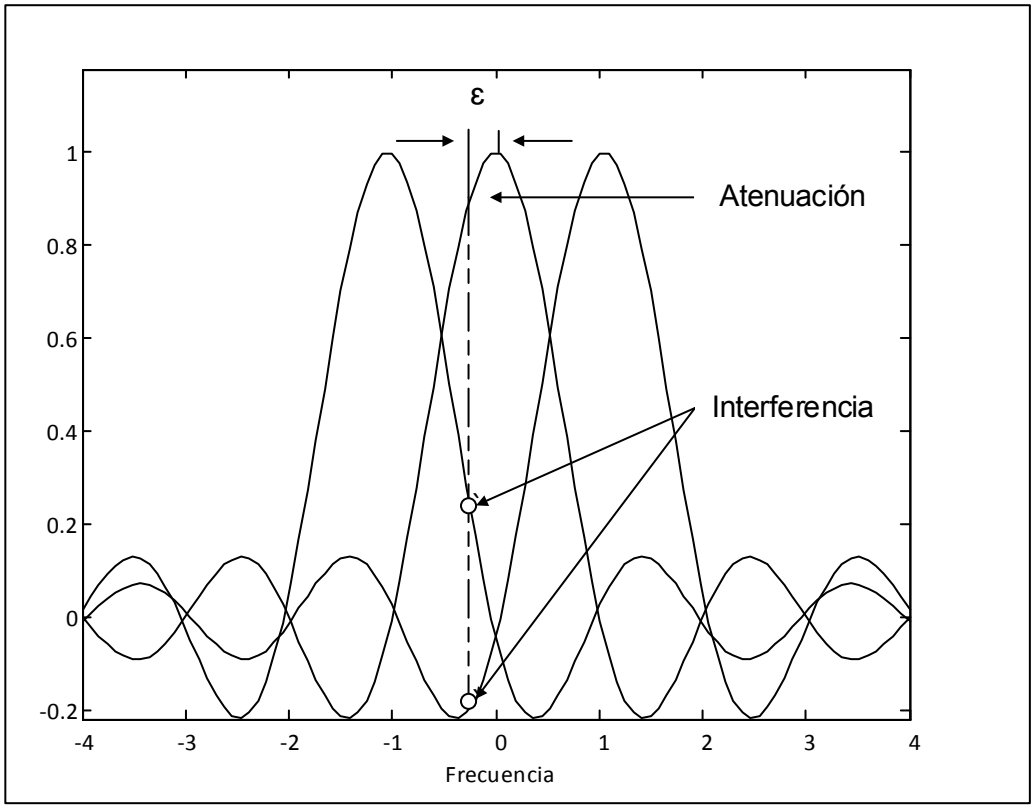


Figura 1.7. Interferencia entre subportadoras OFDM
Fuente: reproducido de (Panta, K., Armstrong, J., 2003)
Elaboración: Autores

1.5 Estándar IEEE 802.11a

El estándar IEEE 802.11a utiliza una multiplexación ortogonal por división de frecuencia (OFDM) conformada por 52 subportadoras con una tasa máxima de datos de 54 Mbit/s. El estándar IEEE 802.11a usa de 12 a 13 canales que no se superponen, 8 dedicados a interiores y 4 punto a punto (Keysight Technologies, 2011).

1.5.1 Descripción general de la señal OFDM del estándar IEEE 802.11a

Las señales IEEE 802.11a son señales de tipo pulsado (o de ráfaga). El ancho de banda total del canal es de 20 MHz, con un ancho de banda ocupado de 16,6 MHz. Un único símbolo OFDM contiene 52 subportadoras; 48 son subportadoras de datos y 4 son subportadoras piloto, como se muestra en la Figura 1.8 (Daund, Vyas, 2016).

Todas las subportadoras de datos utilizan el mismo formato de modulación dentro de una ráfaga dada. Sin embargo, el formato de modulación puede variar de ráfaga a ráfaga.

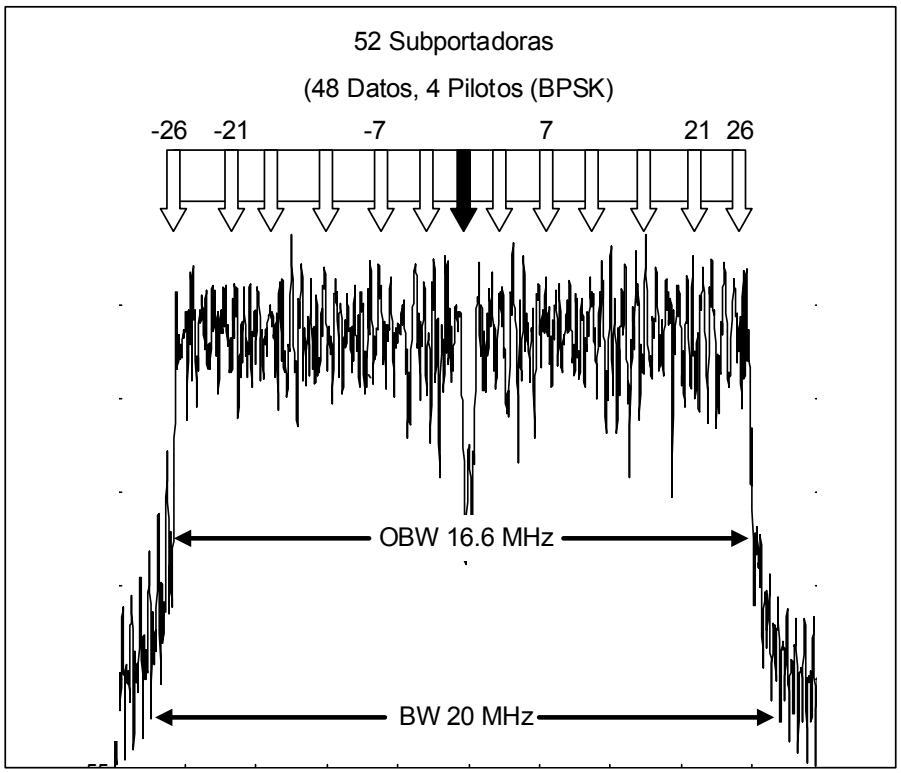


Figura 1.8. Parámetros físicos OFDM del estándar IEEE 802.11a
Fuente: reproducido de (Keysight Technologies, 2011)
Elaboración: Autores

1.5.2 Estructura de la trama OFDM del estándar IEEE 802.11a

La estructura de la trama básica del estándar IEEE 802.11a contiene 3 campos principales: preámbulo, *signal* y datos. Al inicio de la trama, se transmite un preámbulo con una magnitud y fase conocida. El campo *signal* contiene la longitud, el tipo de

modulación y la información de velocidad de los datos que se transmiten y finalmente se añaden un conjunto de símbolos OFDM que contienen los datos de entrada para completar la ráfaga, como se muestra en la Figura 1.9 (Daund, Vyas, 2016).

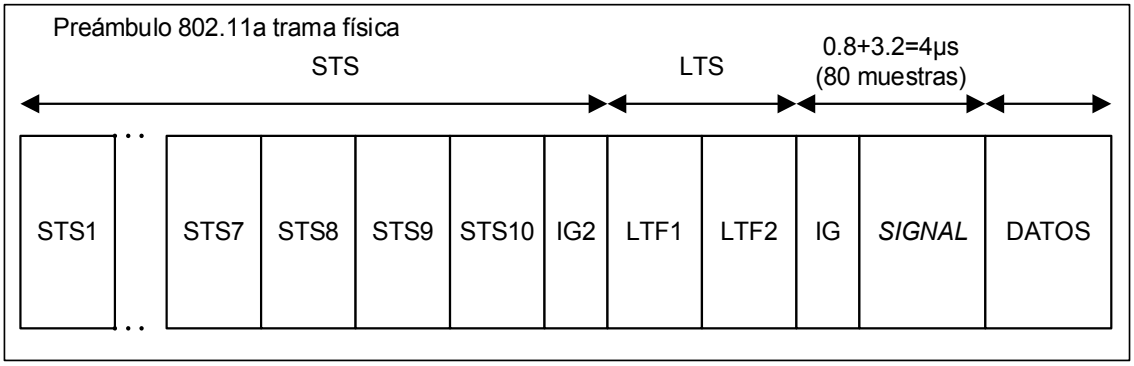


Figura 1.9. Estructura de la trama OFDM del estándar IEEE 802.11a
Fuente: reproducido de (Batra, A., 2010).
Elaboración: Autores

1.5.3 Parámetros de sincronización de señales OFDM del estándar IEEE 802.11a

En la Tabla 1.1 se presenta los principales parámetros del estándar IEEE 802.11a utilizados a lo largo de este trabajo.

Tabla 1.1 Parámetros del estándar IEEE 802.11a

Parámetro	Símbolo	Valor
Ancho de banda	W	$20\ MHz$
Periodo de muestreo	T_a	$50ns$
Total de subportadoras	N_{Total}	52
Subportadoras de datos	N_{datos}	48
Subportadoras piloto	N_{piloto}	4
Tamaño de muestras FFT	N_{FFT}	$64\ puntos$
Espaciado entre subportadoras	$FSP = W/N_{FFT}$	$312.5\ kHz$
Tiempo de Intervalo de Datos	$T_{FFT} = 1/FSP$	$3.2\mu s$
Intervalo de guarda	$T_{GI} = T_{FFT}/4$	$0.8\mu s$
Símbolo de entrenamiento	$T_{TS} = T_{FTT}/2$	$1.6\mu s$
Tiempo del intervalo de símbolos	$T_{SYM} = T_{GI} + T_{FFT}$	$4\mu s$
Secuencia corta de entrenamiento	$T_{SHORT} = 10 * T_{FFT}/4$	$8\mu s$
Secuencia larga de entrenamiento	$T_{LONG} = T_{TS} + 2 * T_{FFT}$	$8\mu s$
Preámbulo	$T_{SHORT} + T_{LONG}$	$16\mu s$

Fuente: reproducido de (Keysight Technologies, 2011)
Elaboración: Autores

1.5.4 Preámbulo IEEE 802.11a

El campo del preámbulo está compuesto por dos conjuntos de símbolos: símbolos cortos (STS, *Short Training Sequence*) y los símbolos largos (LTS, *Long Training Sequence*) como se muestra en la Figura 1.10 (Batra, A., 2010).

Los símbolos cortos OFDM (STS) constan de 10 repeticiones cortas, cada una de 16 muestras de longitud ($0.8\mu s$) y utilizan 12 de las 52 subportadoras que son moduladas por los elementos de la secuencia S, dada por la ecuación (28).

$$X(S)_{[-26,26]} = \sqrt{\frac{13}{6}}\,[0,0,1+j,0,0,0,-1-j,0,0,0,1+j,0,0,0,-1-j,0,0,0,-1-j,0,0,0,1 \\ +j,0,0,0,0,0,0,0,-1-j,0,0,0,-1-j,0,0,0,1+j,0,0,0,1+j,0,0,0,1 \\ +j,0,0,0,1+j,0,0] \tag{28}$$

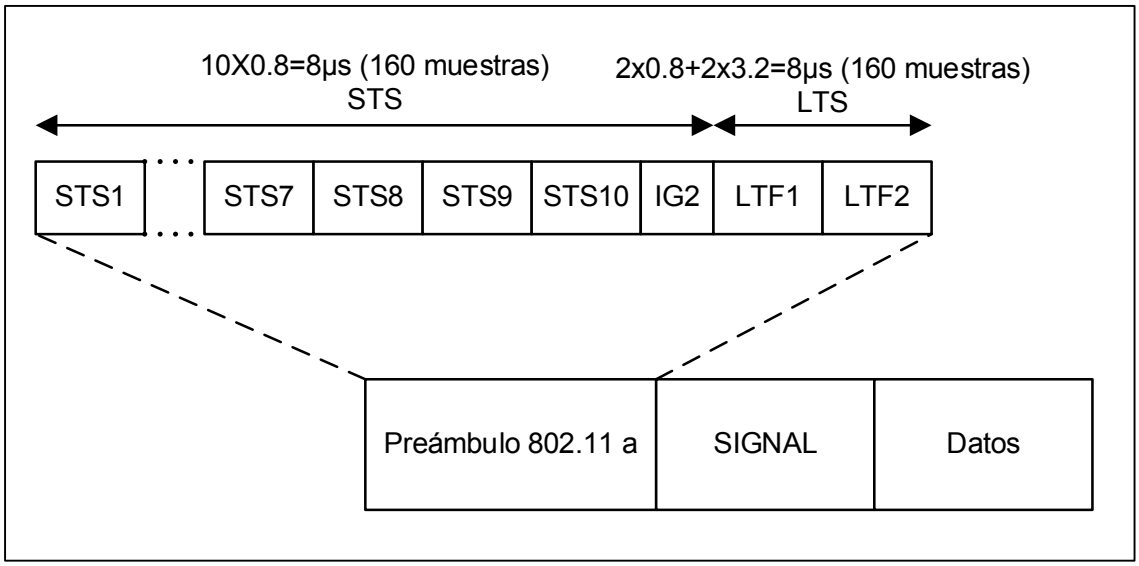


Figura 1.10. Preámbulo IEEE 802.11a
Fuente: reproducido de (Keysight Technologies, 2011)
Elaboración: Autores

La multiplicación por un factor de $\sqrt{\frac{13}{6}}$ es para normalizar la potencia promedio del símbolo OFDM resultante (Batra, A., 2010).

Por otro lado, los símbolos largos de entrenamiento OFDM (LTS) contiene 2 repeticiones largas, cada una compuesta por 64 y utiliza 53 subportadoras (incluido un valor cero en DC), que están moduladas por los elementos de la secuencia L, dada por:

$$X(L)_{[-26,26]} = [1,1,-1,-1,1,1,-1,1,-1,1,1,1,1,1,1,-1,1,1, \\ -1,1,-1,1,1,1,1,1,0,1,-1,-1,1,1,-1,1,-1,1,-1, \\ -1,-1,-1,-1,1,1,-1,-1,1,-1,1,-1,1,1,1,1] \tag{29}$$

Donde $X(k), k = 0{:}N-1$, representa los símbolos de entrenamiento STS y LTS en el dominio de la frecuencia. A continuación se lleva ambas ecuaciones al dominio del tiempo, aplicando una IFFT, donde los símbolos transmitidos vienen dados por (Cong, L., 2014):

$$x(t) = W_{T_W}(t)\frac{1}{N}\sum_{k=0}^{N-1} X(k)e^{j2\pi k\Delta f t} \tag{30}$$

Donde $x(t)$ es la muestra en el dominio del tiempo en la salida de la IFFT, N es el número de puntos FFT/IFFT y k es el índice de subportadora. La muestra se multiplica por una función en el dominio de tiempo discreto para dar forma a un símbolo OFDM, esta función en el dominio discreto se expresa por la ecuación (31) (Cong, L., 2014).

$$W_{N_W}(n) = \begin{cases} 1 & si\ 0 \leq n < 160 \\ 0.5 & si\ n = 0, 160 \\ 0 & caso\ contrario, \end{cases} \tag{31}$$

Como se observa que las ecuaciones (28) y (29) sólo contienen 52 y 53 subportadoras respectivamente, se utiliza una IFFT de 64 puntos para insertar N puntos IFFT a cada una para completar la señal, de forma que ambas señales contengan el mismo número de puntos utilizados en la IFFT. De esa forma, aplicando una IDFT a las señales de las ecuaciones (28) y (29), se obtiene (Cong, L., 2014):

$$x(n) = \frac{1}{N}\sum_{k=0}^{N-1} X(k) e^{j2\pi k \frac{n}{N}} \tag{32}$$

Por lo tanto para la ecuación (28) la señal creada posee una duración de $3.2\mu s$, con 64 puntos en el tiempo discreto, con una tasa de muestreo de $20\ MHz$ (Barragán, D., 2013).

1.6 Sincronismo de Frecuencia

El desvío de frecuencia en un sistema OFDM ocurre cuando la señal del oscilador en el receptor no se sincroniza con la señal portadora recibida. Este fenómeno se puede atribuir a dos factores importantes, desajuste de la frecuencia del oscilador en el transmisor y en el receptor, o bien por el efecto Doppler a medida que el transmisor y el receptor se mueven (Timothy M., 2007). Cuando esto ocurre la señal recibida obtendrá un desvío de frecuencia Δf que será proporcional al desvío de fase según se muestra a continuación:

$$\phi = \pi T \Delta f \tag{33}$$

Esa diferencia de fase puede ser calculada a través de la observación de las muestras recibidas, separadas por la duración de un símbolo. El sincronismo se realiza con una secuencia de entrenamiento de dos símbolos OFDM, donde se toma un solo símbolo corto de entrenamiento s_n de la señal discreta transmitida, que está definida por:

$$y_n = s_n \exp\left(j2\pi f_{tx} n\, T_s\right) \tag{34}$$

Donde f_{tx} es la frecuencia de portadora y T_s es el tiempo de muestreo. El símbolo corto en banda base recibido es definido por la ecuación (35).

$$r_n = s_n \exp\left(j2\pi f_{\Delta} n\, T_s\right) \tag{35}$$

Donde el desvío de frecuencia de la portadora $f_{\Delta} = f_{tx} - f_{rx}$ es la frecuencia de portadora recibida (Barragán, D., 2013).

Considerando este primer símbolo de entrenamiento donde la primera mitad es idéntica a la segunda mitad (en orden de tiempo), excepto por un desplazamiento de fase causado por el desplazamiento de frecuencia de la portadora (Timothy M., 1997), al multiplicar el conjugado de la muestra de la primera mitad con la muestra correspondiente de la segunda mitad, la suma de los productos con L muestras complejas en la mitad del primer símbolo de entrenamiento (STS), se puede expresar en la siguiente ecuación:

$$R(d) = \sum_{m=o}^{L-1} r_{m+d}^{*} r_{m+d+L}$$

$$R(d) = \sum_{m=o}^{L-1} \left[s_{m+d} \exp\left(j2\pi f_{\Delta}(m+d)T_s\right)\right]^{*} \left[s_{m+d+L} \exp\left(j2\pi f_{\Delta}(m+d+L)T_s\right)\right] \quad (36)$$

$$R(d) = \exp\left(-j2\pi f_{\Delta} L T_s\right) \sum_{m=o}^{L-1} s_{m+d}^{*} s_{m+d+L}$$

El resultado de esta multiplicación será la anulación del efecto del canal, y el resultado tendrá una fase aproximada a la de la ecuación (33), por lo que la relación entre ambas ecuaciones se presenta a continuación:

$$\phi \approx \arg\left[R(d)\right] \quad (37)$$

Para 16 muestras complejas ($L = 16$) existe una diferencia de tiempo total de $16T_s$, donde T_s es la duración de una muestra. Esa fase es calculada con el algoritmo CORDIC en modo vectorial. Por lo tanto, la estimación aproximada del desvío de frecuencia viene dada por:

$$f_{\Delta} \approx \frac{\arg\left[R(d)\right]}{2\pi * T_s * 0.8 * f_s} \quad (38)$$

En la implementación, el valor de $\arg\left[R(d)\right]$ es calculado con el algoritmo CORDIC, y su valor se encontrará entre $-\pi$ y π, por lo que los valores límites del desvío de frecuencia serán:

$$-625\ kHz \leq f_{\Delta} \leq 625\ kHz \quad (39)$$

Por lo tanto, el desvío máximo identificable equivale a dos veces el espaciamiento entre cada subportadora.

El estándar 802.11a establece que la máxima tolerancia para la frecuencia central es de ± 20 partes por millón (ppm), lo que corresponde a un desvío de $200\ kHz$ cuando la frecuencia de portadora es $5\ GHz$ (Barragán, D., 2013).

Para corregir el desvío de frecuencia se genera una señal a partir de la ecuación (40).

$$s_d = r_d e^{-j2\pi f_{\Delta} n T_s} \quad (40)$$

Luego, reemplazando el valor de Δf en la ecuación (40), se obtiene la ecuación (41).

$$S_d = r_d \exp\left[-j2\pi\left(\frac{arg\,[R(d)]}{2\pi T_s(0.8 * f_s)}\right)nT_s\right]$$
$$s_d = r_d \exp\left[-j\left(\frac{arg\,[R(d)] * n}{0.8 * f_s}\right)\right] \quad (41)$$

Reemplazando el valor del ancho del canal, se obtiene la siguiente ecuación:

$$s_d = r_d \exp\left[-j\left(\frac{arg\,[R(d)] * n}{16}\right)\right] \quad (42)$$

Donde la señal es multiplicada por una exponencial compleja con frecuencia idéntica al desvío estimado pero de signo contrario.

En la implementación se utiliza el algoritmo CORDIC en modo rotacional para generar la exponencial compleja, conforme a la fórmula de Euler (Barragán, D., 2013).

CAPÍTULO II: SIMULACIÓN DEL ALGORITMO

2.1 Introducción

En esta sección se muestra las diferentes herramientas de software utilizadas para generar el modelo del canal, la secuencia de entrada con los símbolos del preámbulo OFDM y la cuantización de estas muestras. De igual manera, se simula el procedimiento para corregir el desvío de frecuencia en Python, que permite un mayor entendimiento al momento de implementar el algoritmo en VHDL.

2.2 Software y Hardware

Para simular e implementar el algoritmo se usa el software Xilinx ISE y una FPGA Virtex 5. Para programar la FPGA se utiliza el lenguaje de programación VHDL. Este permite crear componentes que pueden ser reutilizados en el código; además permite distintas alternativas de implementación y por medio de simulaciones verificar el comportamiento del sistema.

Para simular y verificar los resultados obtenidos se utiliza el software MATLAB, que permite interactuar con los circuitos generados en VHDL por medio de archivos .txt; estos archivos de texto de entrada tienen los símbolos OFDM con desvío de frecuencia en binario y usando la herramienta testbench la respuesta del circuito se observa en un archivo de salida txt. Esta respuesta ingresa a MATLAB para verificar el comportamiento del circuito, de esta manera el testbench brinda el ambiente para observar y analizar los resultados obtenidos de la implementación (Barragán, D., 2013).

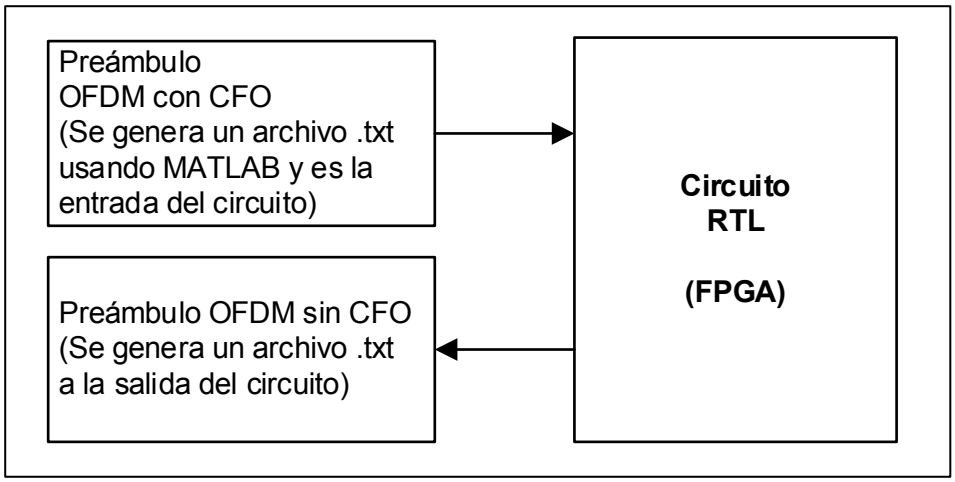


Figura 2.1. Estructura de un testbench
Fuente: reproducido de (Barragán, D., 2013)
Elaboración: Autores

2.3 Modelo del Canal

Cuando se aborda el diseño de un sistema de comunicación OFDM, es importante saber el tipo de canal por el que las señales se van a transmitir. OFDM al ser una técnica de modulación para sistemas inalámbricos, se puede considerar un canal con

desvanecimiento causado por la propagación multitrayecto que afecta a la señal transmitida. Debido a la presencia de la propagación multitrayecto, el receptor obtiene múltiples copias de la señal transmitida, este efecto viene representador por (Cong, L., 2014):

$$y(t) = \sum_i a_i(f,t)\phi(t - \tau_i(f,t)) \tag{43}$$

Donde $a_i(f,t)$ *y* $\tau_i(f,t)$ representan respectivamente la atenuación y el retardo en un instante t.

Si suponemos que la atenuación y el retardo no dependen de la frecuencia f obtenemos la siguiente ecuación:

$$y(t) = \sum_i a_i(t)x(t - \tau_i(t)) \tag{44}$$

En la ecuación (44) se puede observar que el canal es lineal, por lo que la salida obtenida en un instante t a consecuencia de un impulso transmitido en el instante $t - \tau$, puede ser descrita por la respuesta $h(\tau,t)$ por lo que la relación entre la entrada y la salida viene dada por (Casado, F,, 2008):

$$y(t) = \int_{-\infty}^{\infty} h(\tau,t)x(t - \tau)d\tau \tag{45}$$

Si se compara la ecuación (45) con la ecuación (44), se puede observar que el canal tiene una respuesta al impulso descrita por:

$$h(\tau,t) = \sum_i a_i(t)\delta(\tau - \tau_i(t)) \tag{46}$$

En la ecuación (46) el efecto Doppler ya viene incluido, pero no es inmediatamente mostrado en dicha representación. Al realizar la convolución de la señal transmitida $x(t)$ con la respuesta al impulso del canal y adicionando el ruido $n(t)$, la señal en banda base recibida está dada por (Casado, F., 2008):

$$z(t) = \sum_i a_i(t)x\big(\tau - \tau_i(t)\big) + n(t) \tag{47}$$

Además, considerando el caso particular de que un sistema esté conformado por un transmisor, receptor y un canal lineal, cuando las atenuaciones $a_i\,(t)$ y los retrasos de propagación $\tau_i(t)$ no depende del tiempo t, obtenemos un canal lineal invariante en el tiempo (LTI) con una respuesta al impulso representada por:

$$z(t) = \sum_i a_i\delta\big(\tau - \tau_i\big) \tag{48}$$

De esta manera se introduce el desvío de frecuencia en la secuencia de entrada, detallada en la ecuación (49). Para este trabajo se toma un desvío de frecuencia moderado $\Delta f = 100\ kHz$ (Casado, F., 2008).

$$r_n = z(t)e^{j2\pi\Delta f t}|_{t = nT_s} \tag{49}$$

El esquema del modelo del canal utilizado se muestra a continuación

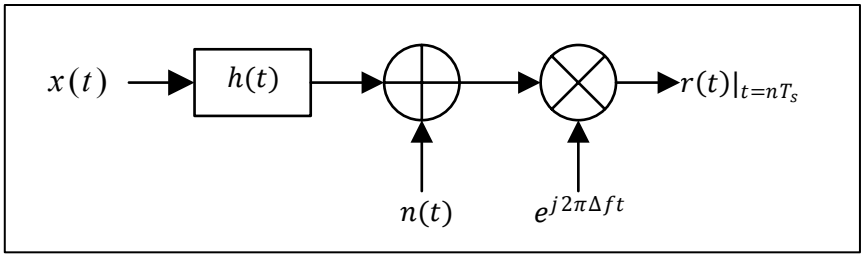


Figura 2.2. Modelo del Canal
Fuente: reproducido de (Barragán, D., 2013)
Elaboración: Autores

2.4 Secuencia de Entrada

Para la implementación, la secuencia de entrada está conformada por 320 muestras del preámbulo. Los símbolos cortos (STS) están conformados por 10 símbolos y cada símbolo es compuesto por 16 muestras complejas. Los símbolos largos (LTS) están conformados por 2 símbolos y cada símbolo es compuesto por 64 muestras complejas y 36 muestras corresponden al intervalo de guarda.

Una vez generadas las muestras, los símbolos STS, LTS y el intervalo de guarda son arreglados para formar el preámbulo. Luego, al multiplicar el preámbulo con una señal que posee un desvío de frecuencia de $100\ kHz$ y agregando ruido con la función AWGN de MATLAB se obtiene los valores de la Tabla 2.1 y Tabla 2.2.

Tabla 2.1 Muestras de los símbolos STS

N	Re	Im	N	Re	Im
1	0,046	0,046	9	0,033	0,056
2	-0,132	-0,002	10	0,039	-0,127
3	-0,009	-0,079	11	-0,071	-0,037
4	0,143	0,001	12	-0,060	0,130
5	0,091	0,012	13	-0,034	0,086
6	0,143	0,010	14	-0,068	0,126
7	0,001	-0,080	15	-0,065	-0,046
8	-0,130	-0,027	16	0,062	-0,117

Fuente: reproducido de (Batra, A., 2010)
Elaboración: Autores

En la entrada del bloque de estimación sólo ingresaron las muestras de los símbolos STS, debido a que se realiza una estimación gruesa. Al momento de realizar la compensación se utiliza todas las muestras del preámbulo.

Tabla 2.2 Muestras de los símbolos LTS

N	Re	Im	N	Re	Im	N	Re	Im	N	Re	Im
1	0,151	0,039	17	0,076	0,045	33	-0,114	0,107	49	-0,040	-0,079
2	-0,031	0,116	18	0,008	-0,105	34	0,078	0,060	50	0,029	-0,116
3	0,018	0,117	19	-0,067	-0,020	35	0,140	0,006	51	0,150	0,062
4	0,109	-0,067	20	-0,146	-0,017	36	0,026	0,145	52	-0,002	-0,061
5	0,024	-0,025	21	0,042	-0,116	37	0,040	0,036	53	-0,053	-0,035
6	0,051	0,093	22	0,058	-0,041	38	-0,012	-0,105	54	-0,068	0,128
7	-0,118	0,048	23	-0,089	-0,048	39	-0,091	0,091	55	-0,114	-0,015
8	-0,042	0,105	24	-0,040	0,045	40	-0,082	0,091	56	0,009	-0,053
9	0,098	0,026	25	0,042	0,149	41	-0,146	-0,051	57	-0,020	-0,099
10	0,053	-0,006	26	-0,096	0,076	42	-0,048	0,037	58	-0,107	0,035
11	0,008	0,115	27	-0,097	0,086	43	0,042	0,092	59	-0,055	0,115
12	-0,132	0,060	28	0,104	0,019	44	0,044	-0,056	60	-0,090	-0,057
13	0,032	0,055	29	-0,034	-0,042	45	0,119	-0,035	61	0,027	-0,023
14	0,060	0,006	30	-0,143	-0,035	46	0,008	0,146	62	0,073	-0,104
15	-0,052	-0,154	31	0,003	-0,140	47	0,015	0,068	63	-0,115	-0,026
16	0,117	-0,022	32	-0,055	-0,081	48	0,105	-0,001	64	-0,118	0,024

Fuente: reproducido de (Batra, A., 2010)
Elaboración: Autores

2.5 Cuantización de las Muestras

Con el fin de minimizar el consumo de potencia y economizar los recursos destinados para registro y decodificación, cuando se trabaja con una representación en punto flotante, se decidió utilizar la representación en punto fijo con complemento a dos para realizar la cuantización de las muestras. De esta forma se va a poder utilizar los bits para indicar cuando un número es positivo y cuando es negativo (Barragán Diego, 2013).

Considerando una palabra binaria de M bits, cuando es interpretada con un número entero racional con señal en punto fijo de forma $A_i/2^a$, donde A_i y a son enteros, se puede asumir que el conjunto de valores racionales P viene dado por:

$$P = -2^{m-1} \leq A_i \leq 2^{m-1} - 1 \quad (50)$$

Eligiendo un valor de a para calcular A_i a partir de la muestra exacta a_i, viene dada por:

$$A_i = [a_i x 2^a] \quad (51)$$

Por lo tanto, para determinar el valor del entero más próximo, utilizamos la siguiente ecuación:

$$a_i^{'} = \frac{A_i}{2^a} \quad (52)$$

Entonces, para obtener el error de cuantización entre la muestra aproximada $a_i^{'}$ y la muestra exacta a_i, se utiliza la ecuación (53).

$$error = a_i^{'} - a_i \quad (53)$$

Finalmente, se utiliza una palabra binaria M de 12 bits, para garantizar un error mínimo de cuantización y evitar el desbordamiento de bits. Si el valor de la muestra es un número negativo, la operación binaria que se debe incluir es el complemento a dos (Barragán, D., 2013).

2.6 Simulación en Python

En este punto se explica de qué manera se realiza la implementación del algoritmo en VHDL, utilizando el lenguaje de programación Python, con el fin de entender el procedimiento y comportamiento del algoritmo.

2.6.1 Símbolos OFDM con CFO

Inicialmente los símbolos OFDM tienen un desvío de frecuencia de $0\ kHz$. Al multiplicar estos símbolos por una exponencial compleja que posee un CFO de $100\ kHz$ y ruido AWGN, como se muestra en la Figura 2.3. Se obtienen los símbolos OFDM con un desvío de frecuencia de $100\ kHz$.

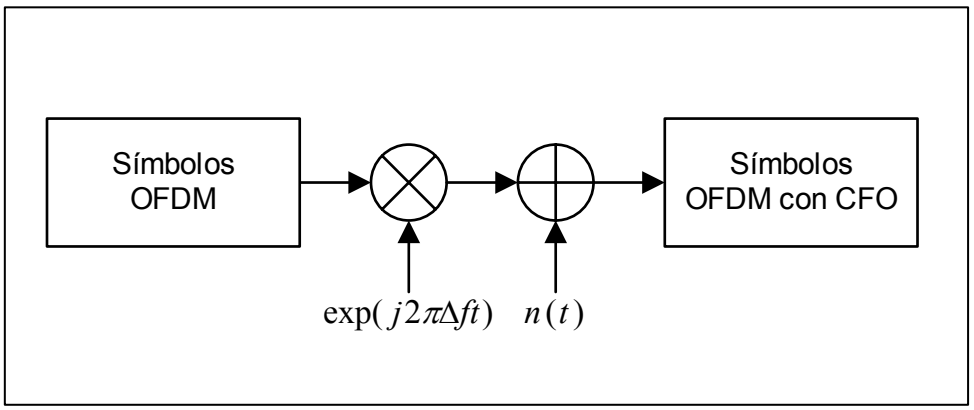


Figura 2.3. Generar símbolos OFDM con CFO
Fuente: reproducido de (Barragán, D., 2013)
Elaboración: Autores

2.6.2 Detección de CFO

Para detectar el desvío de frecuencia se aplica la autocorrelación de las muestras de los símbolos STS, este resultado sirve como entrada al algoritmo CORDIC en modo vectorial para determinar el desvío de frecuencia de los símbolos de entrada. Este proceso se observa en la Figura 2.4.

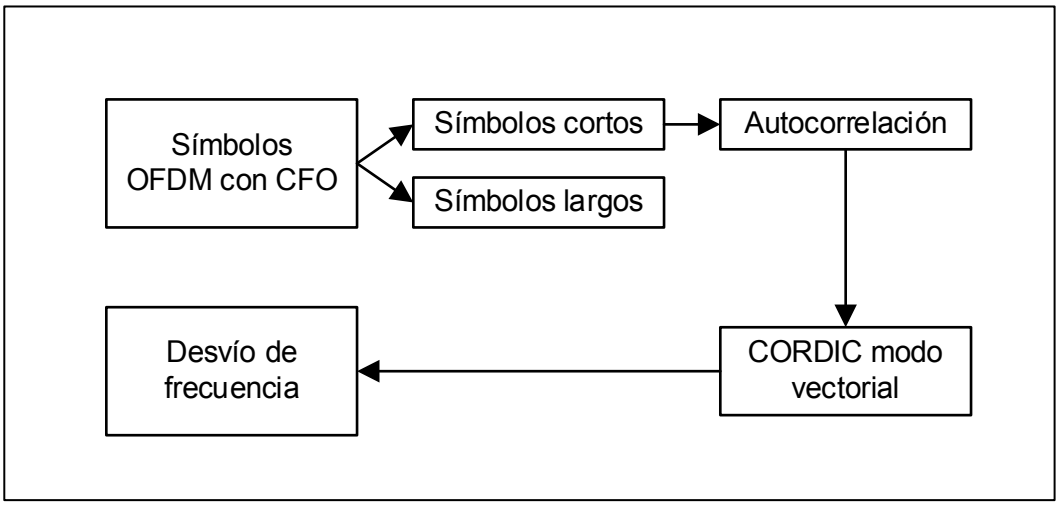


Figura 2.4. Proceso para estimar el CFO
Fuente: Autores
Elaboración: Autores

2.6.3 Corrección del desvío de frecuencia

Para corregir el desvío de frecuencia se utiliza CORDIC en modo rotacional. La entrada de este bloque son los valores que se obtuvo en CORDIC modo vectorial, para obtener una exponencial compleja con el mismo desvío de frecuencia pero con signo contrario.

La exponencial compleja se multiplica con los símbolos del preámbulo OFDM con CFO, obteniendo como resultado el preámbulo sin desvío de frecuencia.

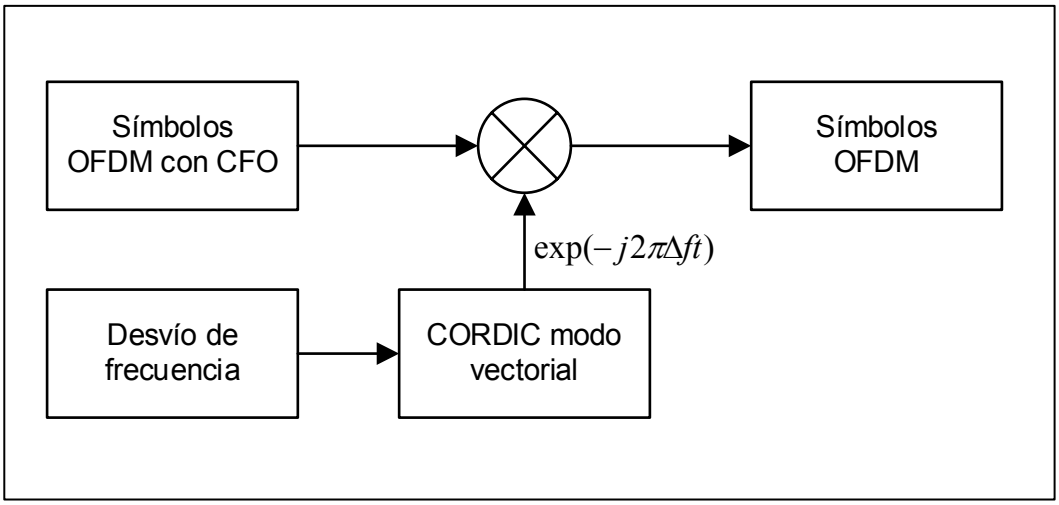


Figura 2.5. Proceso para compensar el CFO
Fuente: Autores
Elaboración: Autores

CAPÍTULO III: IMPLEMENTACIÓN EN FPGA

3.1 Introducción

En el presente capítulo se presenta la implementación del algoritmo CORDIC en modo vectorial y en modo rotacional para un sistema de sincronismo de frecuencia OFDM. En la etapa de sincronismo de frecuencia se realiza operaciones de desplazamientos de bits, registros de desplazamiento, multiplicaciones complejas, acumulaciones (sumatoria recursiva), estimación de fase y generación de las funciones seno y coseno para realizar la estimación y compensación del desvío de frecuencia de la portadora.

En el desarrollo del algoritmo CORDIC, en modo vectorial y rotacional, fue importante determinar la relación del número de iteraciones del algoritmo con el error en el cálculo de la fase y de las funciones seno y coseno, además de tomar en cuenta las condiciones para girar los vectores correctamente.

3.2 Sincronismo de frecuencia

Como se menciona en el Capítulo II, el desvío de frecuencia en un sistema OFDM es causado cuando la señal del oscilador del receptor no se sincroniza con la señal portadora recibida. Este fenómeno puede aparecer debido al ruido AWGN, efecto Rayleigh o bien por el efecto Doppler (Timothy M., 2007). El desvío de frecuencia tiene como efecto la destrucción de la ortogonalidad entre las portadoras, aumentando el riesgo de que aparezca el efecto ICI. El sincronismo de frecuencia tiene como objetivo estimar ese desvío de frecuencia y corregirlo. En este caso sólo se desarrolló una estimación gruesa porque el desvío de frecuencia no era tan severo ($100\ kHz$). La arquitectura del sincronizador de frecuencia se presenta en la Figura 3.1.

3.2.1 Estimación Gruesa

En esta sección se explica cómo estimar el valor del CFO. Para ello es necesario utilizar las muestras de los símbolos STS, que sirven como entrada al bloque Operación A para luego pasar al bloque CORDIC vectorial

3.2.1.1 *Operación A*

En este bloque se realiza el cálculo de la correlación de los símbolos de entrenamiento cortos (STS). Para realizar esta operación se implementó un registro de desplazamiento conformado por un conjunto de Flip-flops tipo D, que proporciona almacenamiento para un bit cada uno (Barragán, D., 2013). De esta manera, el registro de desplazamiento, que tiene un retardo de 16 muestras, está compuesto de L registradores en cascada, como se observa en la Figura 3.2, cada uno con N bits de entrada y un camino para las señales de entrada tanto real e imaginaria. En cada subida de reloj una nueva muestra es registrada y la muestra más antigua es descartada.

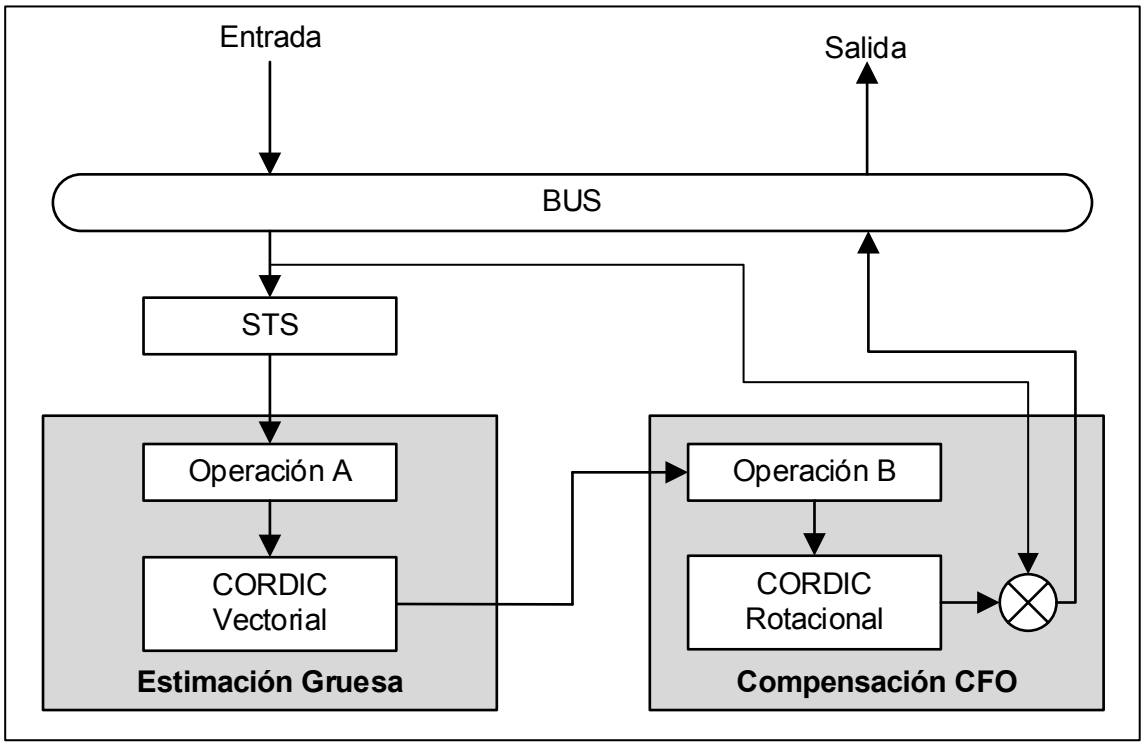


Figura 3.1. Arquitectura de estimación y compensación CFO
Fuente: reproducido de (Barragán, D., 2013)
Elaboración: Autores

A la entrada del registro de desplazamiento ingresan señales digitales reales e imaginarias, que después de 16 pulsos de reloj las muestras más nuevas se colocan en la salida del registro para luego ingresar a un multiplicador complejo (Barragán, D., 2013)

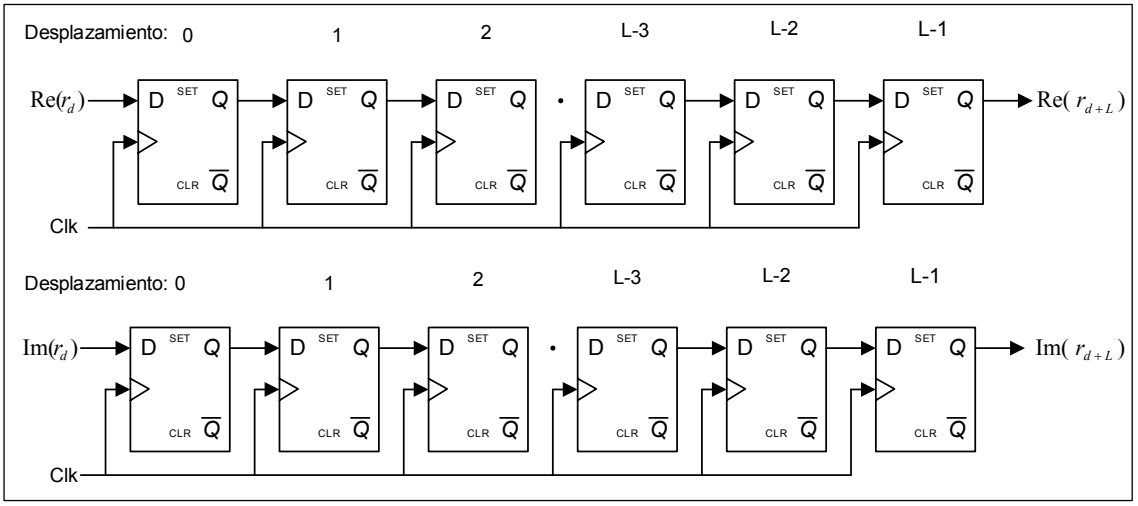


Figura 3.2. Registro de desplazamiento con retardo de 16 muestras
Fuente: reproducido de (Martínez, I. A., 2007).
Elaboración: Autores

El multiplicador complejo está conformado por cuatro multiplicadores reales y dos sumadores algebraicos, como se muestra en la Figura 3.3. Cada entrada tiene una

longitud de 12 bits, al realizar la multiplicación se obtiene una salida con una palabra binaria de 24 bits y al momento de pasar por el sumador algebraico se aumenta 1 bit de carreo, obteniendo una salida de 25 bits.

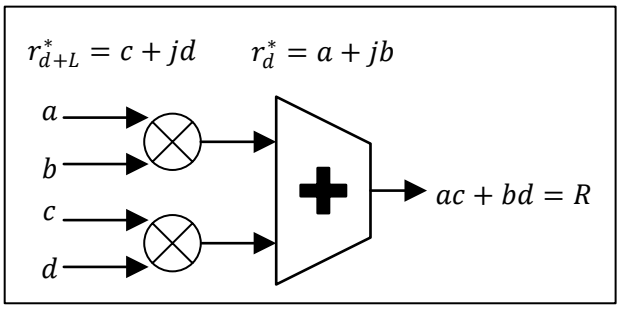


Figura 3.3. Bloque de un Multiplicador Complejo
Fuente: reproducido de (Barragán, D., 2013)
Elaboración: Autores

Luego de la multiplicación se realiza una operación recursiva que es un acumulador de las muestras obtenidas a la salida del multiplicador, teniendo una longitud de ventana de 16 pulsos de reloj, donde en cada pulso un nuevo valor resultante de la multiplicación es ingresado a un registro de desplazamiento, como se muestra en la Figura 3.4. Por lo tanto, el valor obtenido en la salida del acumulador es una palabra binaria de 29 bits.

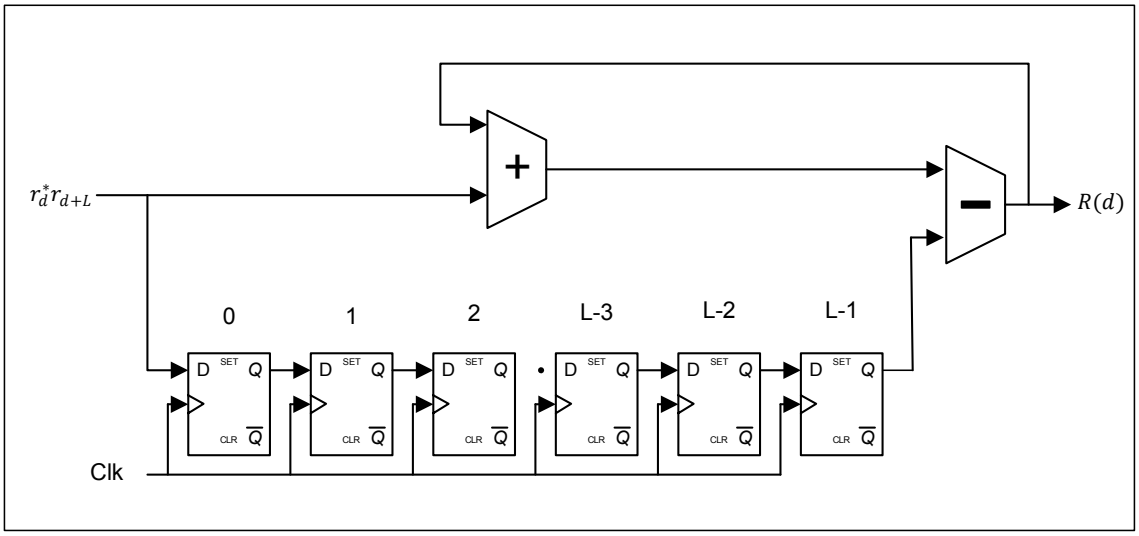


Figura 3.4. Diagrama del acumulador
Fuente: reproducido de (Barragán, D., 2013)
Elaboración: Autores

Finalmente, utilizando una operación de desplazamiento a la derecha, que produce una división para 2, la palabra binaria de 29 bits se reduce a 16 bits con el fin de economizar recursos en hardware. Este valor será la entrada al bloque CORDIC en modo vectorial.

3.2.1.2 *CORDIC vectorial*

El CORDIC en modo vectorial funciona para ángulos entre $-\pi/2$ y $\pi/2$, por lo que es necesario que el valor de entrada al algoritmo necesite un rotación inicial, de forma que se ubique en el primer o cuarto cuadrante, el circuito que realiza esta operación se observa en la Figura 3.5.

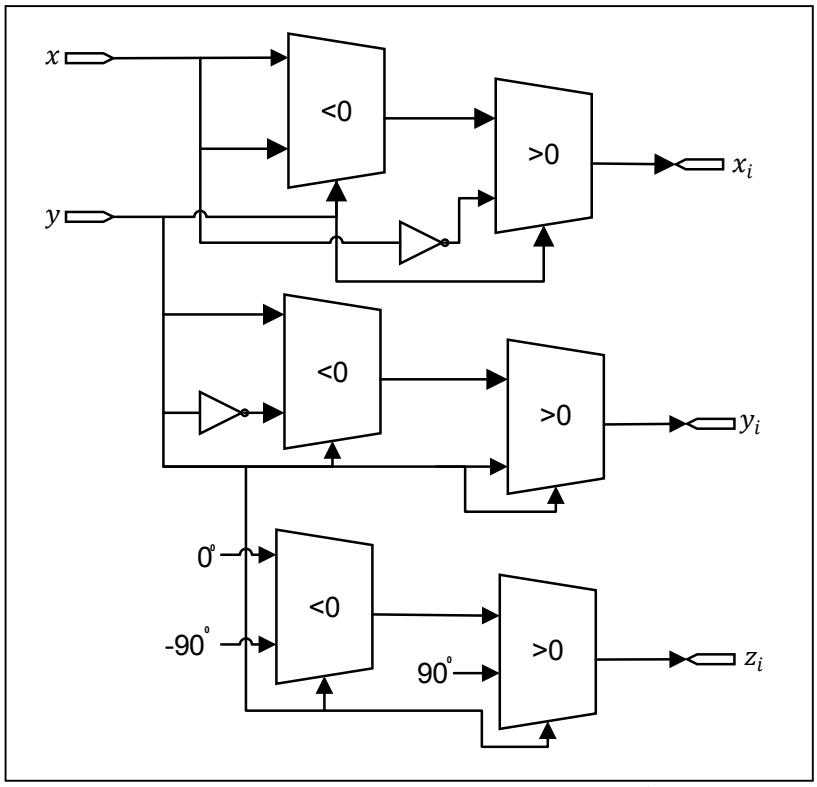


Figura 3.5. Diagrama del circuito de rotación $\pm\ 90^{\circ}$
Fuente: reproducido de (Barragán, D., 2013)
Elaboración: Autores

A continuación, se utiliza una unidad de pseudorotación con la finalidad de girar el vector en cada una de las iteraciones en un valor determinado por a_i. En la Tabla 3.1 se observan los ángulos $a_i = tan^{-1}(2^{-i})$ que son almacenados en la FPGA.

Los valores de la Tabla 3.1 fueron calculados y aproximados en Python. En VHDL fueron almacenados en registros.

Para obtener el valor de x_{i+1} y de y_{i+1} es necesario calcular el valor de $y_i 2^{-i}$ y $x_i 2^{-i}$ respectivamente, que esto implementado en VHDL es un desplazamiento de bits a la derecha. Luego, se realiza una operación algebraica de suma o resta con el valor de entrada x_i o y_i, según sea el caso, donde finalmente la salida de esta operación se almacena en un registro para ser usado en la siguiente iteración.

Tabla 3.1 Valores de a_i y su representación en binario.

Iteración	Valor decimal (grados)	Valor Binario
0	45.0000	0010000000000000
1	26.5650	0001001011100000
2	14.0362	0000100111110000
3	7.1250	0000010100010000
4	3.5763	0000001010000000
5	1.7899	0000000100110000
6	0.8952	0000000010100000
7	0.4476	0000000001010000
8	0.2238	0000000000101001
9	0.1119	0000000000010100
10	0.0559	0000000000001010
11	0.0279	0000000000000101
12	0.0137	0000000000000011
13	0.0068	0000000000000010
14	0.0039	0000000000000001
15	0.0019	0000000000000001

Fuente: reproducido de (Hampson G. Paplinski A, 2011)
Elaboración: Autores

Para obtener el valor de z_{i+1} se realizan operaciones similares a las explicadas, teniendo en cuenta que para este caso el valor de entrada z_i se va a sumar o restar con los valores de la Tabla 3.2. En VHDL los valores de esta tabla son llamados a través de un registro, esta arquitectura se observa en la Figura 3.6.

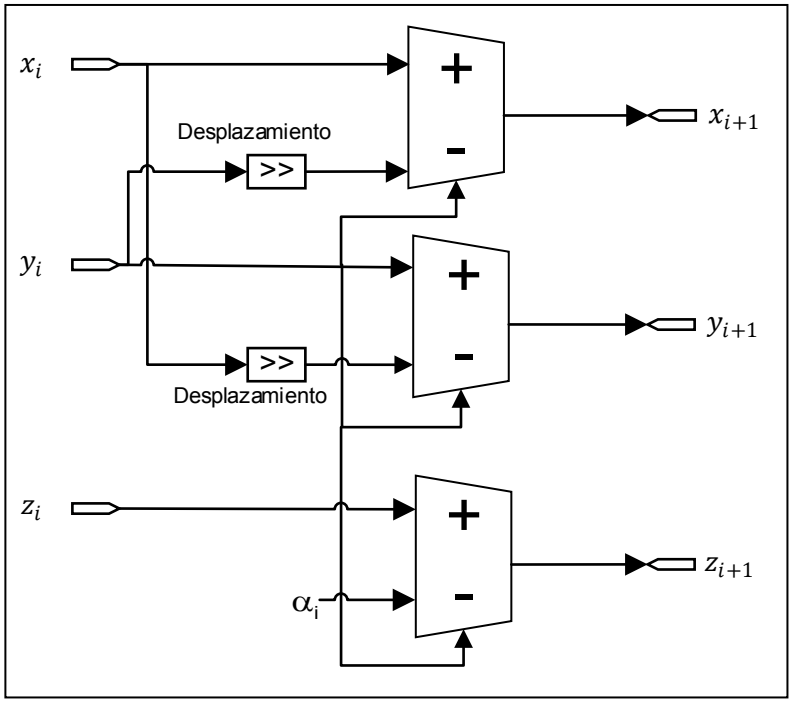


Figura 3.6. Diagrama de una iteración en CORDIC vectorial
Fuente: reproducido de (Barragán, D., 2013)
Elaboración: Autores

3.2.2 Compensación CFO

3.2.2.1 *Operación B*

En este bloque se realiza el cálculo de la acumulación de los valores de los ángulos provenientes de la salida del CORDIC en modo vectorial. Antes de implementar esta operación, se utiliza cuatro veces el bloque de desplazamiento a la derecha que produce una división para 16. A continuación, se implementa el bloque de acumulación que representa la multiplicación de estos valores con n. La arquitectura de este proceso se observa en la Figura 3.7.

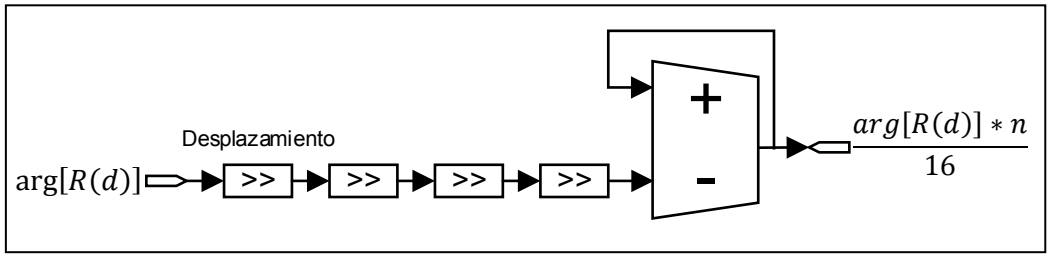


Figura 3.7. Diagrama del bloque B
Fuente: reproducido de (Barragán, D., 2013)
Elaboración: Autores

Para realizar estas operaciones tanto la entrada como la salida deben tener el mismo tamaño de bits, teniendo en cuenta que el desbordamiento de bits fue importante para que estos valores varíen entre $-\pi$ y π.

3.2.2.2 *CORDIC Rotacional*

El CORDIC en modo rotacional funciona para ángulos entre $-\pi$ y π, por lo que es necesario que el valor de entrada al algoritmo necesite un rotación inicial de forma que se ubique en el cuadrante correcto, el circuito que realiza esta operación se observó en la Figura 3.6.

Para obtener el valor de x_{i+1}, y_{i+1} y de z_{i+1} se realiza un proceso similar al implementado en CORDIC modo vectorial, con la diferencia de que ahora la condición depende del valor de z_i y del valor de entrada al algoritmo. Esta arquitectura se observa en la Figura 3.8.

Finalmente, la señal s_d se multiplica con el preámbulo de entrada, como se muestra en la Figura 3.9, obteniendo como resultado el preámbulo son desvío de frecuencia.

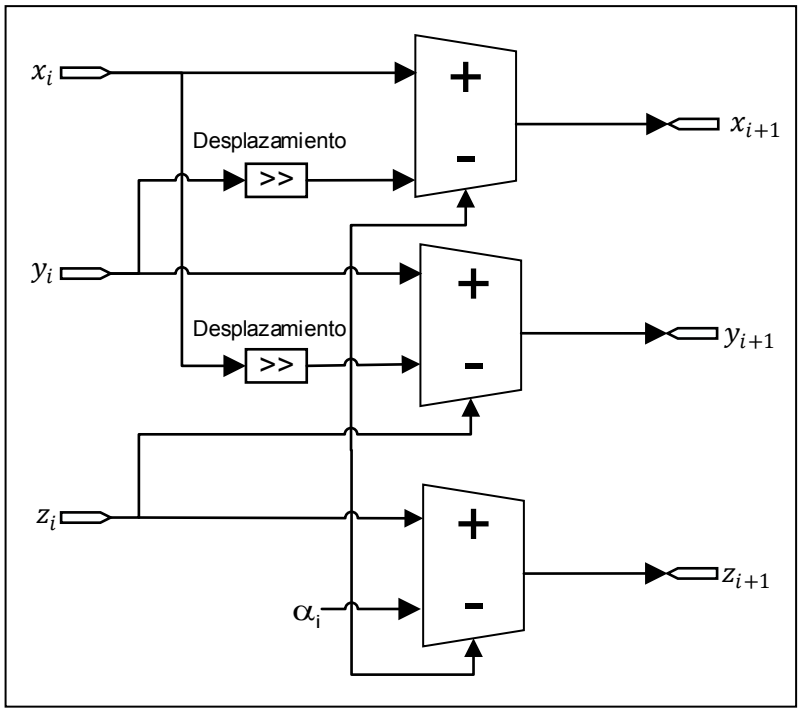


Figura 3.8. Diagrama de una iteración en CORDIC rotacional
Fuente: reproducido de (Barragán, D., 2013)
Elaboración: Autores

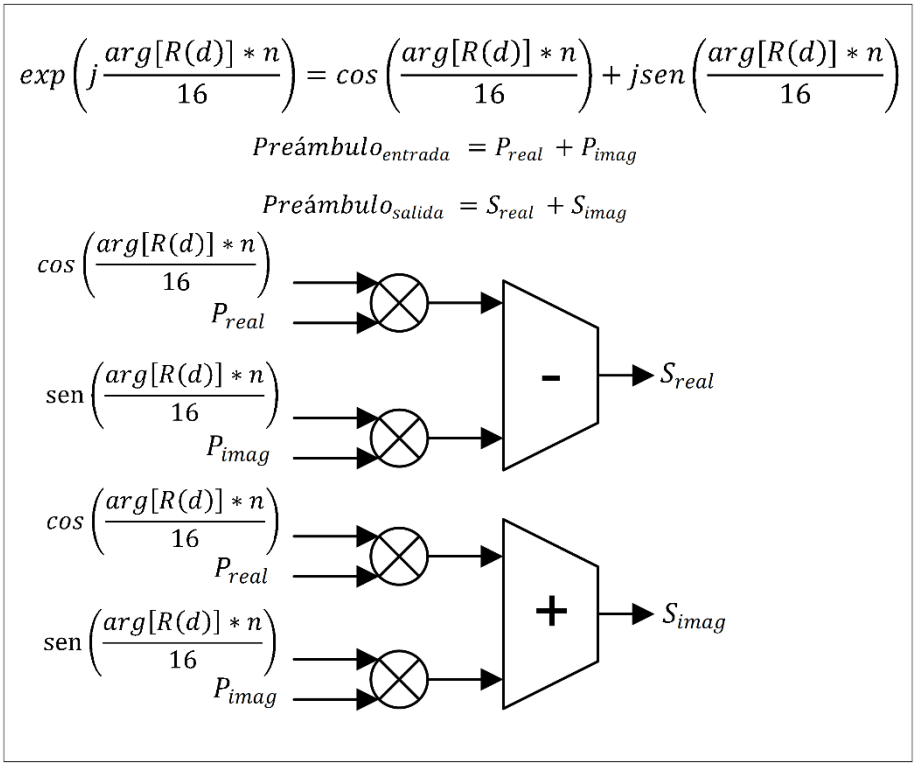


Figura 3.9. Diagrama del multiplicador para la corrección del desvío de frecuencia
Fuente: reproducido de (Barragán, D., 2013)
Elaboración: Autores

CAPÍTULO IV: ANÁLISIS DE RESULTADOS

4.1 Introducción

En el presente capítulo se presenta los resultados obtenidos luego de haber implementado el algoritmo de sincronismo de frecuencia OFDM en VHDL. Se presenta la relación entre el error del algoritmo CORDIC con el número de iteraciones y finalmente se presentan los histogramas de los resultados obtenidos al compensar los símbolos del preámbulo OFDM, con sus respectivos valores estadísticos.

4.2 Resultado de simular el error de CORDIC versus el número de iteración

En este apartado se presenta el resultado obtenido del error en dB del algoritmo CORDIC versus el número de iteraciones, como se observa en la Figura 4.1.

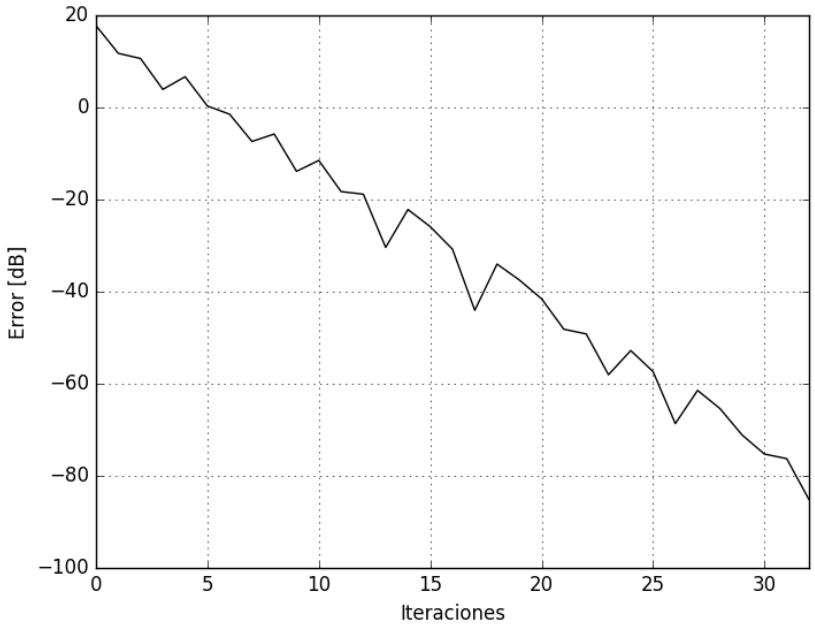


Figura 4.1. Error del algoritmo CORDIC versus Número de Iteraciones
Fuente: Autores
Elaboración: Autores

En la Figura 4.1 se observa que el error del algoritmo CORDIC se reduce cuando el número de iteraciones aumenta, por lo que también el coste computacional será mayor. Considerando las prestaciones en hardware que nos ofrece la FPGA Virtex 5, se busca un equilibrio entre ambas variables, y teniendo en cuenta que el número de iteraciones es igual al número de bits que se utiliza, se tiene como resultado final que en la iteración 16 se obtiene el valor más conveniente del error para poder implementar el algoritmo CORDIC en modo vectorial y en modo rotacional.

4.3 Resultado de implementar el algoritmo de CORDIC

En este apartado se presenta el resultado obtenido en la implementación del algoritmo CORDIC en modo vectorial y en modo rotacional, como se observa en la Figura 4.2 y 4.3 respectivamente.

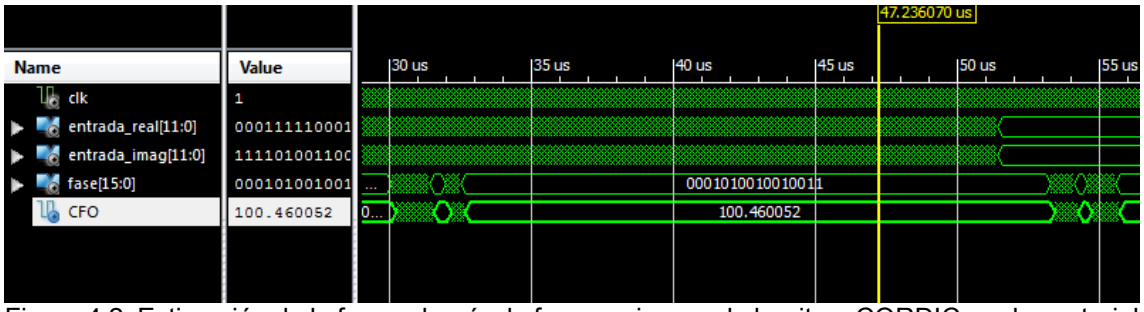


Figura 4.2. Estimación de la fase y desvío de frecuencia con el algoritmo CORDIC modo vectorial
Fuente: Autores
Elaboración: Autores

En la Figura 4.2 se observa que el bloque tiene como entradas la parte real y la parte imaginaria de las muestras de los símbolos del preámbulo OFDM, con ruido AWGN y CFO. A la salida del algoritmo se obtiene la fase de cada una de las muestras de entrada, que permite estimar el desvío de frecuencia. Este bloque se lo denomina estimación gruesa, debido que solo utiliza los símbolos STS del preámbulo para estimar la fase.

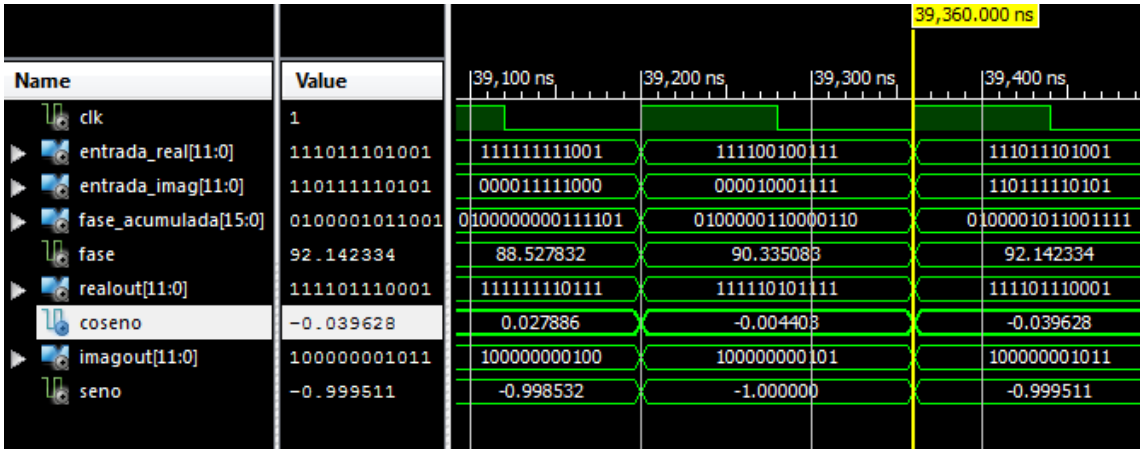


Figura 4.3. Estimación del seno y coseno con el algoritmo CORDIC modo rotacional
Fuente: Autores
Elaboración: Autores

En la Figura 4.3 se observa que el bloque tiene como entrada la fase acumulada, que se obtiene al ingresar la fase de salida del algoritmo CORDIC modo vectorial en la operación B. La fase acumulada varía entre $-\pi$ a π y es el valor que ingresa en el algoritmo CORDIC modo rotacional, el cual nos da como resultado el valor de las funciones seno y coseno. Para una fase de 90° se obtiene un coseno de – 0.039 y seno de – 1. El signo de la función seno varía para poder generar la exponencial compleja con signo contrario y realizar la compensación de los símbolos del preámbulo OFDM.

4.4 Resultado de compensar los símbolos del preámbulo OFDM

Finalmente en este punto se presenta el resultado de la compensación de los símbolos del preámbulo OFDM como se muestra en la Figura 4.4.

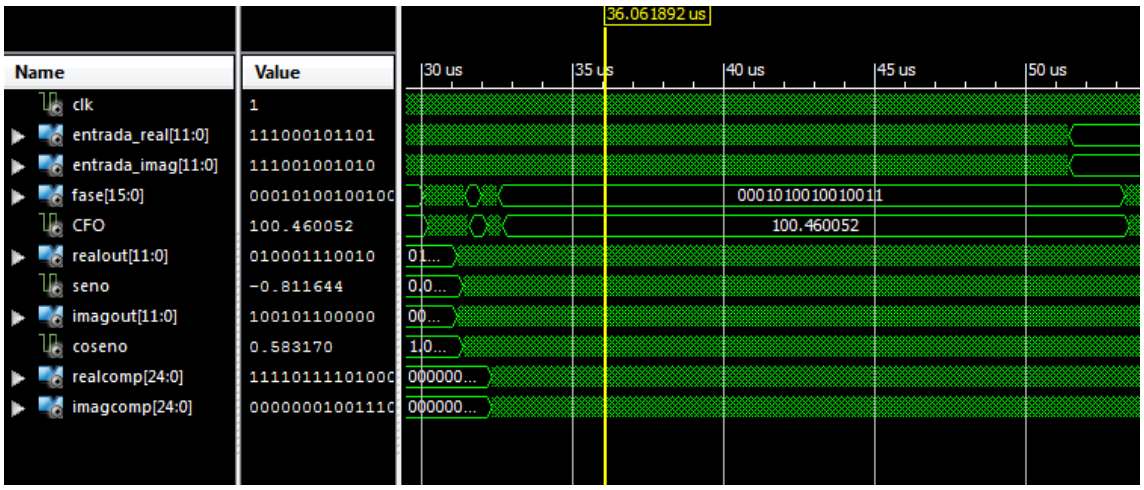


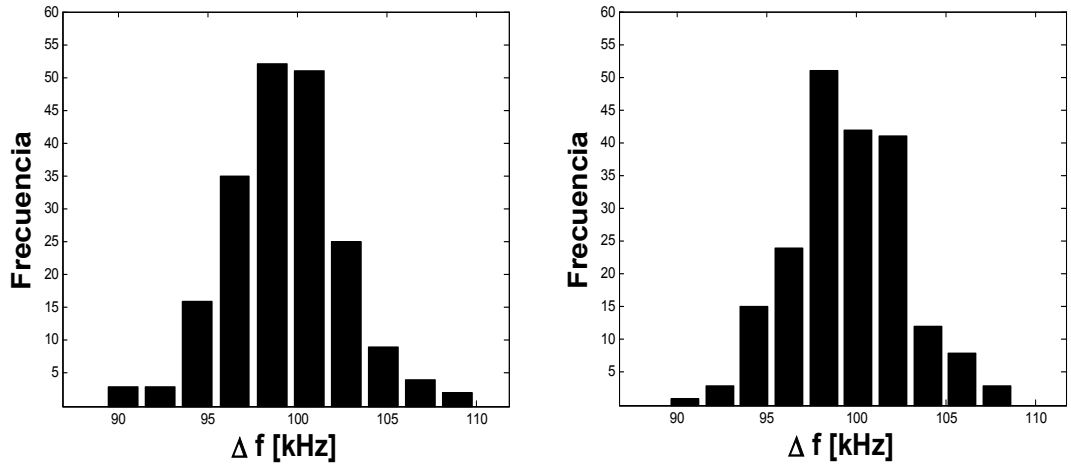
Figura 4.4. Compensación de los símbolos OFDM
Fuente: Autores
Elaboración: Autores

Para realizar la compensación se multiplica la exponencial compleja generada en el CORDIC modo rotacional, con los símbolos de entrada del preámbulo OFDM. A la salida de este bloque se obtiene los símbolos del preámbulo sin desvío de frecuencia con un tamaño de 24 bits.

4.5 Resultado de implementar el algoritmo de sincronismo de frecuencia

En este apartado se presenta el resultado obtenido de la implementación del algoritmo de sincronismo de frecuencia OFDM. Para comprobar el comportamiento y desempeño del algoritmo se realizaron un conjunto de pruebas utilizando 400 señales de entrada con un desvío de frecuencia de $100\ kHz$, un SNR de $10\ dB$ y atrasos en el canal de 50 y $150\ ns$. En la Figura 4.5, se observa el histograma de los resultados obtenidos de las pruebas realizadas.

En la Figura 4.5 (a) se utilizó 200 señales de entrada con un desvío de frecuencia de $100\ kHz$, ruido AWGN y un atraso en el canal de $50\ ns$, estos valores utilizados son valores típicos de una estimación y compensación gruesa, mientras que, para la Figura 4.5 (b) se utilizó 200 señales con el mismo desvío de frecuencia, ruido AWGN pero con un atraso en el canal de $150\ ns$.



(a) Atraso de 50 ns (b) Atraso de 150 ns

Figura 4.5. Histograma con desvío de frecuencia de 100 kHz.
Fuente: Autores
Elaboración: Autores

Los valores obtenidos de la estimación total se presentan en la siguiente tabla:

Tabla 4.1 Estimación del desvío de frecuencia

Δf (kHz)	Atraso (ns)	Min. Estimado (kHz)	Max. Estimado (kHz)	Media	Varianza
100	50	89.2067	110.0159	99.0772	10.8358
100	150	89.4547	108.8333	99.5350	10.4127

Fuente: Autores
Elaboración: Autores

El resultado obtenido de los recursos utilizados en la implementación se presenta en la Tabla 4.2.

Tabla 4.2 Recursos utilizados en FPGA Virtex 5 de la estimación y compensación de CFO

Recursos	Usado
Número de Slice Flip-Flop	995
Registros de Desplazamientos	122
Número de Slices	1529
Máxima Frecuencia	192.25 MHz

Fuente: Autores
Elaboración: Autores

Los resultados de la implementación del algoritmo de sincronismo de frecuencia son comparados con los resultados mostrados en la Sección 4.3 de (Barragán, D., 2013), como se muestra en la Tabla 4.3.

Tabla 4.3 Comparación de Resultados.

	Algoritmo Comparativo	Algoritmo Propuesto
Media	100.4710 kHz	99.0772 kHz
Número de Slice Flip-Flop	4881	995
Número de Slices	5125	1529
Máxima Frecuencia	77.561 MHz	192.25 MHz

Fuente: reproducido de (Barragán, D., 2013)
Elaboración: Autores

Debido a que en (Barragán, D., 2013) se utiliza una estimación fina del algoritmo de sincronismo de frecuencia el valor de la media es mejor en comparación con el valor obtenido en la estimación gruesa que se implementó en esta tesis, pero la cantidad de recursos utilizados en FPGA al implementar nuestro algoritmo se reducen.

CONCLUSIONES

El principal objetivo de esta propuesta fue desarrollar un algoritmo de sincronismo OFDM para la estimación y corrección del desvío de frecuencia, basado en el estándar IEEE 802.11a. Para desarrollar el algoritmo de sincronismo de frecuencia se utilizó el lenguaje de programación de hardware VHDL para luego implementarlo en una FPGA.

Se eligió utilizar una FPGA, porque permite ejecutar varios procesos al mismo tiempo debido al paralelismo de su hardware.

La primera parte para desarrollar el algoritmo de sincronismo de frecuencia fue describir el modelo matemático de CORDIC, conforme se explica en el Capítulo I. Esto con el objetivo de entender la rotación de los vectores y la utilidad del modo vectorial (para la detección de la fase) y del modo rotacional (para la generación de la exponencial compleja). Fue importante tener en cuenta la relación entre el error del algoritmo CORDIC con el número de iteraciones, ya que, debido a esto se tuvo en cuenta el parámetro coste-prestación que ofrece el hardware de una FPGA. Para esto fue importante simular en el lenguaje de programación Python esta relación y cada uno de los bloques del algoritmo de sincronismo de frecuencia para entender su funcionamiento y concluir que el valor óptimo del error se encontró al utilizar 16 iteraciones.

Para implementar el algoritmo de sincronismo de frecuencia en VHDL, fue importante entender la estructura del preámbulo IEEE 802.11a, conformado por símbolos STS y LTS.

Para determinar el CFO se realizó el cálculo de la correlación de los símbolos STS que fueron la entrada de CORDIC modo vectorial para determinar el desvío de frecuencia que es proporcional a la fase de la señal recibida, esta estimación se denomina estimación gruesa.

Una vez estimada la fase de la señal, se generó una exponencial compleja con signo inverso a partir de CORDIC modo rotacional. Esta operación permitió eliminar el desvío de frecuencia multiplicando el preámbulo de entrada por la exponencial compleja generada.

Para verificar el rendimiento del algoritmo se ingresaron señales con desvíos de frecuencia de $100\ kHz$, con atrasos en el canal de 50 y $150\ ns$, y a través de la media y la varianza se determinó bajo qué condiciones el algoritmo se desempeñó de manera óptima, concluyendo que para una frecuencia de $100\ kHz$ y un atraso de $150\ ns$ se obtuvieron los mejores resultados.

RECOMENDACIONES

- Primero se debe comprender la teoría que involucra el algoritmo a implementar, por lo que es importante realizar una simulación en Python que permite comprender adecuadamente el funcionamiento de cada bloque del algoritmo, para luego implementarlo en un FPGA.
- Es importante conocer las prestaciones y limitaciones del hardware a utilizar para poder implementar el algoritmo sin inconvenientes.
- Es importante conocer la versión del *software* a utilizar, ya que algunas versiones de Xilinx ISE funcionan mejor en Windows 7, para verificar si Xilinx ISE trabaja de manera correcta en el SO se puede simular un circuito y verificar su correcto funcionamiento usando la herramienta del testbench.
- Al momento de programar en VHDL es necesario tener claro cuáles serán las entradas y salidas de cada componente, para luego poder reutilizar cada uno de estos.
- Para programar en VHDL es necesario tener conocimiento de sistemas digitales, ya que se basa en la programación de compuertas lógicas programables, de esta manera se podrá entender mejor el comportamiento de las señales digitales obtenidas.
- Una vez comprendido el funcionamiento básico del algoritmo, se debe buscar información adicional y observar trabajos relacionados de distintas fuentes como texto, imágenes, etc. Por lo que es mejor tomar una fuente de información que no conlleve mayores dificultades, puesto que esto requerirá un mayor consumo de tiempo que desviará la atención del objetivo principal de la investigación y desarrollo.
- Se puede probar la robustez del algoritmo de sincronismo de frecuencia implementado un canal con desvanecimiento Rayleigh y efecto Doppler.
- A partir de este trabajo se puede seguir desarrollando el algoritmo de sincronismo de frecuencia, mejorando el algoritmo planteado y realizando una estimación fina para obtener mejores resultados.

BIBLIOGRAFÍA

Alim, O. A., Elboghdadly, N., Ashour, M. A., & Elaskary, A. M. (2008). FPGA implementation for an optimized CORDIC module for OFDM system. En *Computer Engineering & Systems, 2008. ICCES 2008. International Conference on* (pp. 21–26). IEEE.

Ayala, R. López, V. (2015). FPGA: Nociones básicas e implementación. Universidad Politécnica de Madrid, Laboratorio de Diseño Microelectrónico.

Barragán, D. (2013) IMPLEMENTACAO EM FPGA DE ALGORITMOS DE SINCRONISMO PARA OFDM. IWT International Workshop on Telecommunication.

Barry, J., Lee, E., Messerschmitt, D. (2004). Digital Communication. Third Edition. Pp. 203-284.

Batra, A., Balakrishnan, J., Dabak, A. G., Gharpurey, R., Fontaine, P. H., & Lin, H.-C. (2010). White Paper. IEEE Standards Board. Edition. Piscatawa, USA.

Casado, F. (2008) ESTUDIO DE LOS EFECTOS DEL MOVIMIENTO EN SEÑALES OFDM. Universidad Autónoma de Barcelona. pp. 330-334.

Cong, L. (2014) Robust time and frequency synchronization in 802.11a communication wireless system. Signal and Image Processing. Université Paris-Nord - Paris XIII.

Daund, A., Vyas, P. K. (2016). Wireless Broadband Access with the Application of IEEE 802.11 b based Wi-Fi Model. International Journal of Computer Applications, 154(5).

Edwards, B.H., Underwood, J.M. (1998), How Do Calculators Calculate Trigonometric Functions?, Proceedings of the Ninth Annual International Conference on Technology in Collegiate Mathematics, Reno, Nevada.

Hampson, G. Paplinski A. (2011), A VHDL Implementation of a CORDIC Arithmetic Processor Chip. Robotics and Digital Technology, Monash University, Australia. Proceedings of the Ninth Annual International Conference on Technology in Collegiate Mathematics, Reno, Nevada.

Joachim, R. (2002). Descripción en VHDL de arquitecturas para implementar el algoritmo CORDIC. Facultad de Informática.

Keysight Technologies. (2011). 802.11 OFDM Overview. Retrieved from http://rfmw.em.keysight.com/wireless/helpfiles/89600b/webhelp/subsystems/wlan-ofdm/content/ofdm_80211-overview.htm

Martínez, I. A, Zubia, J. G, & Usategui, J. M. A (2007). Sistemas Digitales y Tecnología de Computadores. Editorial Parainfo. Vol 8, pp. 215-229.

Marzal, A., Luengo, I. G. (2004). Introducción a la Programación con Python y C. Publicacions de la Universitat Jaume I.

Moose, P. (1994). A technique for orthogonal frequency division multiplexing frequency offset correction. IEEE Transactions on communications. Vol. 42. pp. 2908-2914.

National Intruments. (2011). Introducción a la Tecnología FPGA: Los Cinco Beneficios Principales. Retrieved from http://www.ni.com/white-paper/6984/es/

Oltean, M. (2004). An Introduction to Orthogonal Frequency Division Multiplexing. Analele Universitatii Oradea, 2004, Fascicola Electrotehnica, Sectiunea Electronica, 180-185.

Panta, K., Armstrong, J. (2003). Spectral analysis of OFDM signals and its improvement by polynomial cancellation coding. IEEE Transactions on Consumer Electronics, 49(4), 939-943.

Parhami, B. (2000), Computer Arithmetic - Algorithms and Hardware Designs, Oxford University Press.

Patricia, B. (2015). Diseño de Sistemas con FPGA. Retrieved from https://www.dc.uba.ar/materias/disfpga/2015/c1/descargas/Introduccion.pdf

Poggi, G. (2009). Synchronization Technics For OFDM Systems. Universita Degli'studi Di Napoli Federico II Dottorato di Ricerca in Ingegneria Elettronica e delleTelecomunicazioni (XVIII ciclo).

Pollet, T., Van Bladel, M., Moeneclaey. M. (2010). BER sensitivity of OFDM systems to carrier frequency o_set and Wiener phase noise. Communications, IEEE Transactions on, 43(234):191-193.

Timothy, M. Schmidl and Donald C. Cox, Fellow, IEEE. (2007) Robust Frequency and Timing Synchronization for OFDM.

Valls, J., Sansaloni, T., Pérez-Pascual, A., Torres, V., & Almenar, V. (2006). The use of CORDIC in software defined radios: A tutorial. IEEE communications magazine, 44(9), 46–50.

Volder, J.,(1959) , The CORDIC Trigonometric Computing Technique, IRE Trans. Electronic Computing, Vol EC-8, pp. 330-334.

Xiao, P., Cowan, C., Ratnarajah, T., Fagan, A. (2009). Time synchronization algorithms for IEEE 802.11 OFDM systems.