BENEMÉRITA UNIVERSIDAD AUTÓNOMA DE PUEBLA

FACULTAD DE CIENCIAS DE LA ELECTRÓNICA

MAESTRÍA EN CIENCIAS DE LA ELECTRÓNICA
OPCIÓN EN AUTOMATIZACIÓN

"Diseño e instrumentación de un sistema de recepción OFDMA en plataforma FPGA"

T E S I S

Presentada para obtener el título de:
Maestro en Ciencias de la Electrónica Opción en Automatización

Presenta:

Lic. Juan Carlos Gutiérrez Ortega*

Directores:

Dra. Josefina Castañeda Camacho
Dr. Sergio Vergara Limon

Puebla, México **Diciembre 2016**

***BECARIO CONACYT**

BUAP

Agradecimientos

Agradezco al Consejo Nacional de Ciencia y Tecnología (CONACYT) por el apoyo económico otorgado durante mis estudios de Maestría en la Benemérita Universidad Autónoma de Puebla.

A la Dra. Josefina Castañeda Camacho y al Dr. Sergio Vergara Limon por brindarme su amistad, orientación, ayuda y principalmente por haber confiado en mí para el desarrollo de este proyecto.

A mis sinodales de tesis, el Dr. Jaime Cid Monjaráz, el Dr. Eligio Moisés Gutiérrez Arias y al Dr. Salvador Alcantar Iniesta por sus críticas constructivas para el mejoramiento de este proyecto.

A mis profesores de la Maestría en Ciencias de la Electrónica Opción Automatización, por ser parte importante de mi formación académica y personal.

A mi esposa, a mis padres, hermanas y amigos por su apoyo, consejos y por todos los momentos que hemos pasado juntos.

Resumen

Este documento presenta un trabajo de tesis que tuvo como finalidad el diseño e implementación de un sistema OFDMA (Acceso Múltiple por División Ortogonal de Frecuencia) basado en el estándar IEEE 802.16. En la primera parte del trabajo se presenta una descripción general de dicho estándar, en donde se incluyen varios conceptos y definiciones que son necesarias para comprender el funcionamiento del protocolo. Con ello, se pretende establecer los conocimientos básicos y necesarios, para facilitar la descripción de la metodología empleada en el diseño de la solución planteada en este trabajo.

En la segunda parte del trabajo se presenta el desarrollo y simulaciones de los diferentes bloques que componen un sistema básico de transmisión y recepción OFDMA realizadas con el software MATLAB, en donde se muestran los diagramas que describen el código implementado y los resultados obtenidos en cada etapa de la transmisión.

Después de haber obtenido un primer diseño de los bloques que conforman al sistema OFDMA a nivel simulación en el software MATLAB, se llevaron los programas realizados a nivel hardware. Donde la implementación del sistema se realizó en plataforma FPGA, en esta parte del trabajo se describen los bloques desarrollados mediante el software QUARTUS II.

Por último se presentan los resultados de pruebas realizadas al sistema tanto a nivel simulación con el software QUARTUS II, como con el dispositivo físico. En las simulaciones con QUARTUS II se buscó exponer el correcto funcionamiento de los bloques del receptor, en donde se verificó que el receptor recuperara la información enviada del transmisor. Seguido de la comprobación del correcto funcionamiento a nivel simulación del sistema, se realizaron pruebas físicas, primero con un FPGA y después dividiendo el transmisor y el receptor.

Índice general

Índice de figuras

Capítulo 1

Introducción

Hoy en día la cantidad de usuarios de los sistemas de comunicación inalámbrica ha ido en aumento y no solo eso, el incremento del requerimiento de mayores tasas de datos ha crecido drásticamente haciendo de gran importancia la renovación de los sistemas de comunicaciones para satisfacer dicha demanda.

El desarrollo de estas tecnologías debe enfrentar varios problemas que presentan los sistemas inalámbricos, entre otros debe considerarse por ejemplo las condiciones de transmisión son hostiles debido al desvanecimiento e interferencia provocada por el ambiente, además se debe hacer un uso eficaz del ancho de banda ya que es un recurso limitado.

El gran desafío de los sistemas de comunicación inalámbrica es proporcionar una velocidad de transmisión elevada y ofrecer un servicio de calidad garantizada. Una de las propuestas tecnológicas que ha tenido gran aceptación está basada en el uso del Multiplexaje por División Ortogonal de Frecuencia; que permite la transmisión de grandes cantidades de datos digitales a través de una onda de radio.

De manera conceptual OFDM ha existido durante décadas; pero su implementación real y con costos aceptables ha sido posible desde la llegada y propagación de tecnologías como los microprocesadores de alta velocidad y los dispositivos de lógica programable para poder hacer fiable el procesamiento digital requerido.

1.1 Antecedentes históricos

El primer esquema OFDM se remonta a 1966, cuando Robert W. Chang publicó su trabajo pionero sobre la síntesis de señales ortogonales con banda limitada para la transmisión de datos por multicanales. Posteriormente, fue emitida una patente en 1970 por su trabajo. Él presentó un nuevo esquema para la transmisión de señales de forma simultánea a través de un canal con banda limitada reduciendo la interferencia inter-canal (ICI) y la interferencia inter-simbólica (ISI) [1].

La idea principal de OFDM es transmitir paralelamente los símbolos de datos en múltiples subpotadoras que se reparten el ancho de banda disponible, en donde cada subportadora es ortogonal al resto de las subportadoras, en la Figura 1.1 se muestra el esquema básico OFDM implementado de forma analógica, en el cual era necesario un banco de osciladores para generar las señales sinusoidales correspondientes a cada subportadora, lo que imponía una alta complejidad para su implementación. Este inconveniente limitaba la aplicación de OFDM para sistemas militares [2].

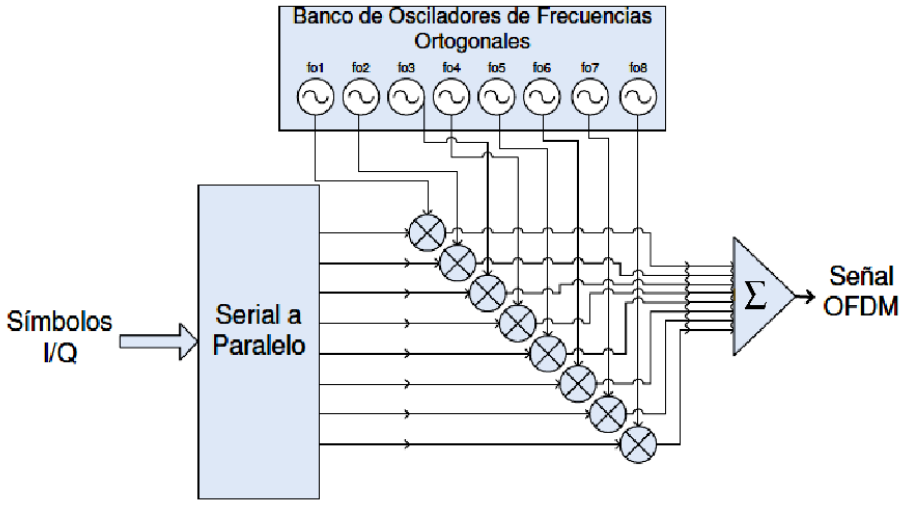


Figura 1.1: Esquema OFDM analógico.

Fue en 1971 cuando Weinstein y Ebert introdujeron la transformada discreta de Fourier y su inversa para realizar la modulación y la demodulación en banda base. Esto le daría viabilidad a OFDM, en lugar de los bancos de osciladores y la inmensa y costosa circuitería de radio frecuencia, que eran necesarios tanto en el transmisor como en el receptor, que provocaban serios problemas de sincronización y sintonización. Para combatir la ISI y la ICI ellos usaron tanto un intervalo de guarda vacío entre símbolos como un enventanado de tipo coseno alzado en el dominio del tiempo. Este sistema no conseguía una ortogonalidad perfecta entre subportadoras sobre un canal dispersivo pero era ya una mejora considerable para la época [1].

Otro gran avance fue el introducido por Peled y Ruiz en 1980, quienes introdujeron el prefijo cíclico (CP) o extensión cíclica, que resolvían los problemas que se producían en la ortogonalidad de las señales. En vez de utilizar un periodo de guarda vacío, ellos

propusieron transmitir en ese espacio una extensión cíclica del símbolo OFDM. Esto, efectivamente, simula un canal que realiza una convolución cíclica, lo que implica ortogonalidad sobre canales dispersivos cuando el CP es mayor que la respuesta impulsiva del canal. Sin embargo, esto introduce una pérdida de energía proporcional a la longitud del CP, pero que queda justificada por la nula ICI [3].

Fue hasta la década de los noventas que apareció el primer estándar comercial basado en OFDM, un estándar inalámbrico para la emisión de radio digital (DAB, Digital Audio Broadcasting). Desde ese entonces OFDM empezó a ocupar un sitio importante dentro de las comunicaciones, actualmente este esquema se utiliza en el estándar Europeo para difusión de video digital por redes terrestres (DVB-T, Digital Video Broadcasting Terrestrial), en algunas versiones del estándar IEEE 802.11 para redes inalámbricas de área local también conocido como Wi-Fi, y por supuesto en el estándar IEEE 802.16 para el acceso inalámbrico en redes metropolitanas, popularizado con el nombre comercial de WIMAX (Worldwide Interoperability for Microwave Access) [4]. A continuación se listan algunas de las fechas relevantes sobre el esquema OFDM:

- 1995, Se establece el estándar ETSI DAB: primer estándar inalámbrico basado en OFDM para la emisión de radio digital (DAB).
- 1997, Se establece el estándar DVB- T (Digital Video Broadcasting Terrestrial).
- 1999, Se establecen estándares IEEE 802.11a para redes inalámbricas LAN (Local Area Network).
- 2004, Se establece el estándar IEEE 802.16 para sistemas de banda ancha inalámbrica MAN (Metropolitan Area Network).
- 2005, Se desarrollan redes celulares móviles basadas en OFDM bajo el estándar IEEE 802.16e y el IEEE 802.20.
- 2010, Se elige la técnica OFDM para el estándar 3GPP Long Term Evolution (LTE), el más reciente estándar para comunicación de datos de alta velocidad [5].

1.2 Motivación del proyecto

OFDMA viene de las siglas en inglés Orthogonal Frequency Division Multiple Access, que significan Acceso Múltiple por División Ortogonal de Frecuencia. Esta técnica es la versión multiusuario del ampliamente reconocido esquema de modulación OFDM. Este esquema presenta ventajas sobre otros sistemas, de las cuales podemos mencionar su habilidad para funcionar correctamente en canales de comunicación que presentan condiciones difíciles, tales como atenuación en las frecuencias altas especialmente en cables de cobre de gran longitud, o la interferencia de banda angosta y el desvanecimiento selectivo de la señal con la frecuencia, que se presentan en las comunicaciones inalámbricas.

La cuarta generación de los sistemas celulares tendrá características óptimas para el desarrollo de una comunicación rápida y fluida. Dentro de estas características podemos mencionar que las velocidades serán mayores a los 2Mbps, se contara con telepresencia, seguridad, accesos multisistemas, terminales multimodo y multifunción [6-8].

Dentro del contexto de la cuarta generación las redes de comunicación móviles celulares consideran una interfaz de radio basada en OFDMA, debido a que mediante el uso de OFDMA, se ofrecen más canales de mayor ancho de banda y varios tipos de servicios. Actualmente, estos sistemas todavía están en fase de definición y desarrollo, pero importantes piezas de la tecnología 4G ya están en marcha y la transición a un nuevo nivel de expectativas en las comunicaciones inalámbricas está muy cerca [6].

Los métodos de acceso múltiple permiten que múltiples usuarios compartan un canal de comunicación para transmitir información a un receptor. Existen diferentes formas de compartir recursos de radio entre usuarios múltiples, asignando regiones en frecuencia, tiempo y espacio a diferentes usuarios. Algunos ejemplos incluyen los métodos clásicos como el acceso múltiple por división de frecuencia (FDMA, Frequency Division Multiple Access), acceso múltiple por división de tiempo (TDMA, Time Division Multiple Access), el acceso múltiple por división de código (CDMA, Code Division Multiple Access), los métodos desarrollados recientemente, como el acceso múltiple por división de espacio (SDMA, Space Division Multiple Access) y el acceso múltiple por división de frecuencias ortogonales (OFDM, Orthogonal Frequency Division Multiplex)[6].

Recientemente, la técnica OFDMA se ha introducido basando su esquema en la técnica de multiplexaje digital OFDM. Esta técnica de multiplexaje proporciona ventajas significativas en términos de eficiencia espectral, robustez frente al desvanecimiento de canales multitrayecto, resistencia a interferencia multiusuario, ecualización simplificada, técnicas de modulación y codificación adaptivas, entre otras.

Estas características han dado lugar a una amplia gama de aplicación en sistemas de comunicación inalámbrica, tales como la emisión de radio digital, en versiones del estándar IEEE 802.11 (Wi-Fi), en el estándar IEEE 802.16e (WIMAX) y en redes celulares. Sin embargo, la teoría, los algoritmos y las técnicas de implementación siguen siendo temas de interés en el área de comunicaciones digitales.

La base de este trabajo será el desarrollo de un sistema OFDMA, principalmente se enfocara en la implementación del receptor debido a que ya se tienen aportaciones al sistema de transmisión, de hecho previamente se desarrolló un trabajo sobre el diseño e implementación de un transmisor OFDMA [9], y la implementación de la Transformada Discreta de Fourier en plataforma FPGA [10], por lo que se plantean los siguientes objetivos.

1.3 Objetivos

Objetivo general:

Diseñar e implementar un sistema de recepción OFDMA en plataforma FPGA.

Objetivos particulares:

1. Estudiar el estado del arte del tópico, los principios de funcionamiento del esquema de acceso OFDMA y las normas que lo regulan.
2. Simular en MATLAB un sistema OFDMA en su etapa de transmisión y recepción.
3. Desarrollar el programa del receptor OFDMA en AHDL para su posterior implementación en el FPGA.
4. Realizar pruebas de confiabilidad del sistema de recepción implementado comparando su desempeño con el reportado en la literatura.

1.4 Metodología de la investigación

La presente tesis tuvo como propósito el diseño e implementación de un sistema OFDMA en el estándar IEEE 802.16. De modo que para el desarrollo del sistema se contemplaron las especificaciones y parámetros descritos en este estándar. Por lo que fue necesario un estudio previo de la técnica de acceso OFDMA y de los bloques que conforman al sistema. Por otra parte, se han utilizado las simulaciones para verificar los estudios teóricos presentados, así como para la validación práctica de algunas de las soluciones. Después de tener las simulaciones y propuestas de diseño para cada etapa del sistema se llevó esto a nivel hardware usando un FPGA. El diseño de la arquitectura se desarrolló siguiendo el esquema presentado en la Figura 1.2, el cual ilustra la secuencia de pasos a seguir para la obtención del sistema.

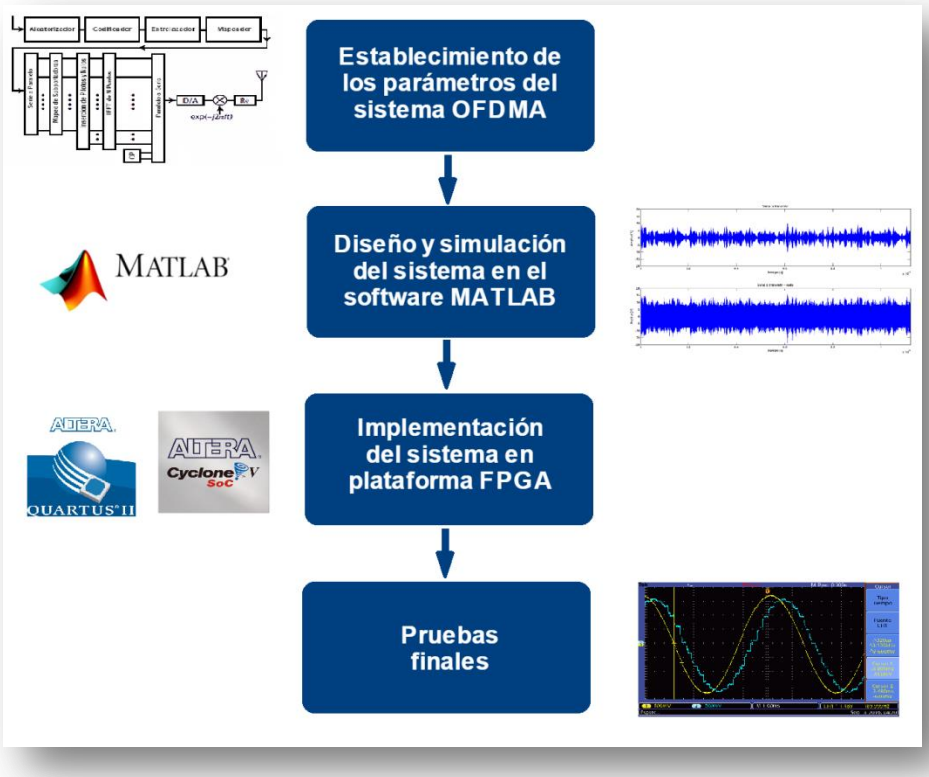


Figura 1.2: Metodología de diseño.

Capítulo 2

Fundamentos teóricos de un sistema OFDMA

El Multiplexaje por División Ortogonal de Frecuencia (OFDM) es una técnica de modulación popular y ampliamente aceptada en el área de comunicaciones inalámbricas. La versión multiusuario del sistema OFDM es llamada Acceso Múltiple por División Ortogonal de Frecuencia (OFDMA). La principal ventaja de estos sistemas es su robustez al desvanecimiento del canal en el entorno inalámbrico. En un sistema OFDMA la Transformada Rápida de Fourier (FFT) y su contraparte la Transformada Rápida de Fourier Inversa (IFFT) se utilizan para modular y demodular los símbolos de datos sobre las subportadoras, lo que hace viable su implementación. OFDM y OFDMA tienen muchas ventajas que contribuyen a su popularidad; por lo tanto, han sido usados en muchos sistemas inalámbricos y adoptados por diversas normas. El estándar IEEE 802.16 se basa en el empleo de OFDMA y ha surgido como un fuerte candidato para futuros sistemas inalámbricos.

El presente capítulo tiene el objetivo de presentar una descripción general de dicho estándar. Las especificaciones de la capa PHY que definen este estándar, incluyen un sin número de conceptos y definiciones que son necesarias para comprender el funcionamiento del protocolo, así como de las múltiples opciones y variantes que presenta. En este capítulo se describirán los conceptos que son fundamentales para entender la propuesta realizada en este trabajo de tesis. Con ello, se pretende establecer los conocimientos básicos y necesarios, para facilitar la descripción de la metodología empleada en el diseño de la solución planteada en este trabajo.

2.1 Principios básicos OFDM

Dentro de los sistemas de comunicaciones digitales convencionales, se tiene a los sistemas con una única portadora y a los sistemas de modulación multiportadora. En un sistema de comunicaciones digitales monoportadora cada símbolo se transmite serialmente ocupando todo el ancho de banda disponible en el canal. En un esquema de modulación multiportadora los símbolos son transmitidos paralelamente en múltiples subportadoras que se reparten el ancho de banda del canal. En un esquema multiportadora la interferencia y los desvanecimientos selectivos en frecuencia afectan en diferente grado a cada subportadora y, por ello, tiene una gran ventaja sobre un sistema monoportadora donde la interferencia y los desvanecimientos podrían causar que el enlace de comunicación se pierda completamente, como se muestra en la Figura 2.1 [11,12].

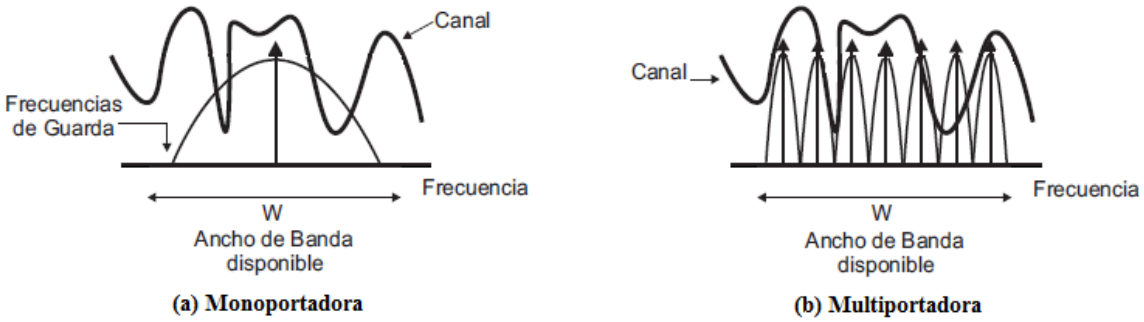


Figura 2.1: Efectos del canal en un sistema monoportadora y multiportadora.

Otra ventaja para un sistema multiportadora es que es eficaz en la lucha contra la interferencia entre símbolos (ISI) y el desvanecimiento por trayectorias múltiples. En el dominio del tiempo, trayectos múltiples conduce a expandir el tiempo de llegada de las señales recibidas debido a múltiples trayectorias de propagación a través de las cuales viajan las señales (retardo de propagación τ), lo anterior se describe en la Figura 2.2. Al dividir el ancho de banda disponible entre varias portadoras, el periodo de símbolo (T_s) se alarga, con lo que se tendrá una menor interferencia del símbolo anterior [7].

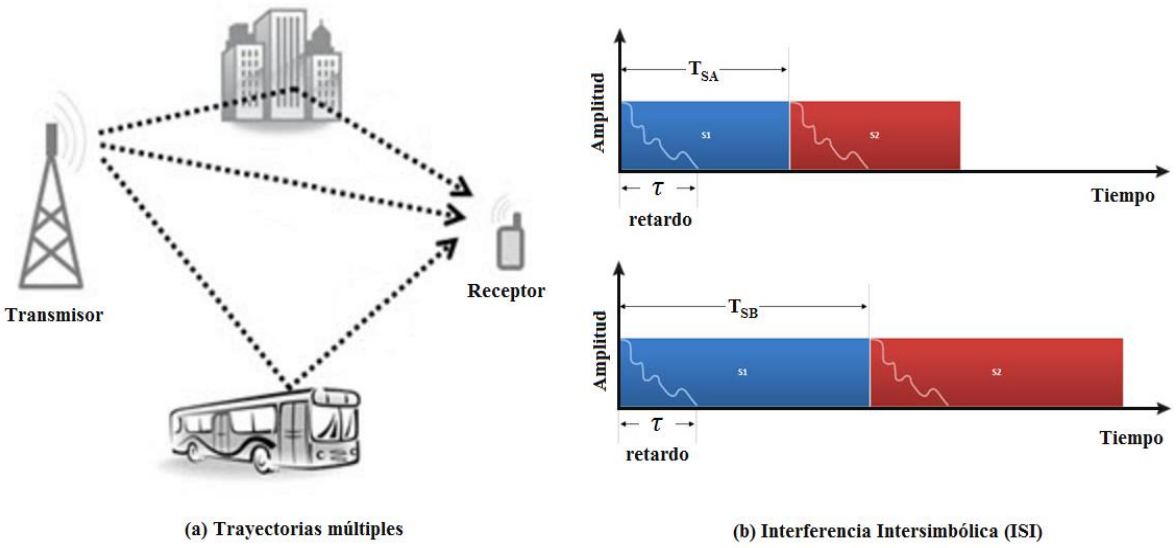


Figura 2.2: Reducción de la interferencia intersimbólica en un sistema multiportadora.

El Multiplexaje por División Ortogonal de Frecuencia es una clase especial del esquema de modulación multiportadora que transmite una trama de datos digital de alta velocidad; dividiéndola en múltiples canales ortogonales de datos en paralelo, lo que da origen a líneas de datos de baja velocidad, estos canales se les denomina subcanales [13].

Lo que hace de OFDM una técnica ampliamente usada es el concepto de las subportadoras ortogonales. El uso de frecuencias ortogonales en OFDM permite la superposición de múltiples portadoras en una misma señal. Cada banda se sobrepone a su adyacente sin afectar la recuperación ni generar interferencia inter-canal (ICI) [12].

En la Figura 2.3 se muestra una representación de tres portadoras ortogonales, la ortogonalidad siempre se cumplirá cuando se tiene un conjunto de sinusoides de frecuencias $k\Delta f$, siendo Δf la frecuencia fundamental, entonces la frecuencia de cada una de las subportadoras debe ser un múltiplo de la frecuencia base. En un esquema OFDM Δf es la separación entre las subportadoras en el dominio de la frecuencia [14].

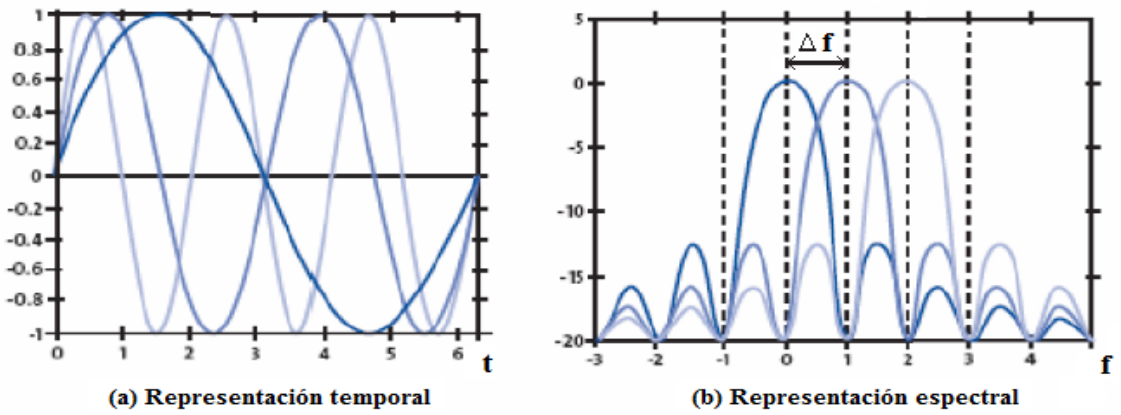


Figura 2.3: Representación de tres subportadoras ortogonales.

Una representación básica de un modulador OFDM es ilustrada en la Figura 2.4. Este consiste de un banco de moduladores complejos, donde cada modulador corresponde a una subportadora OFDM [15].

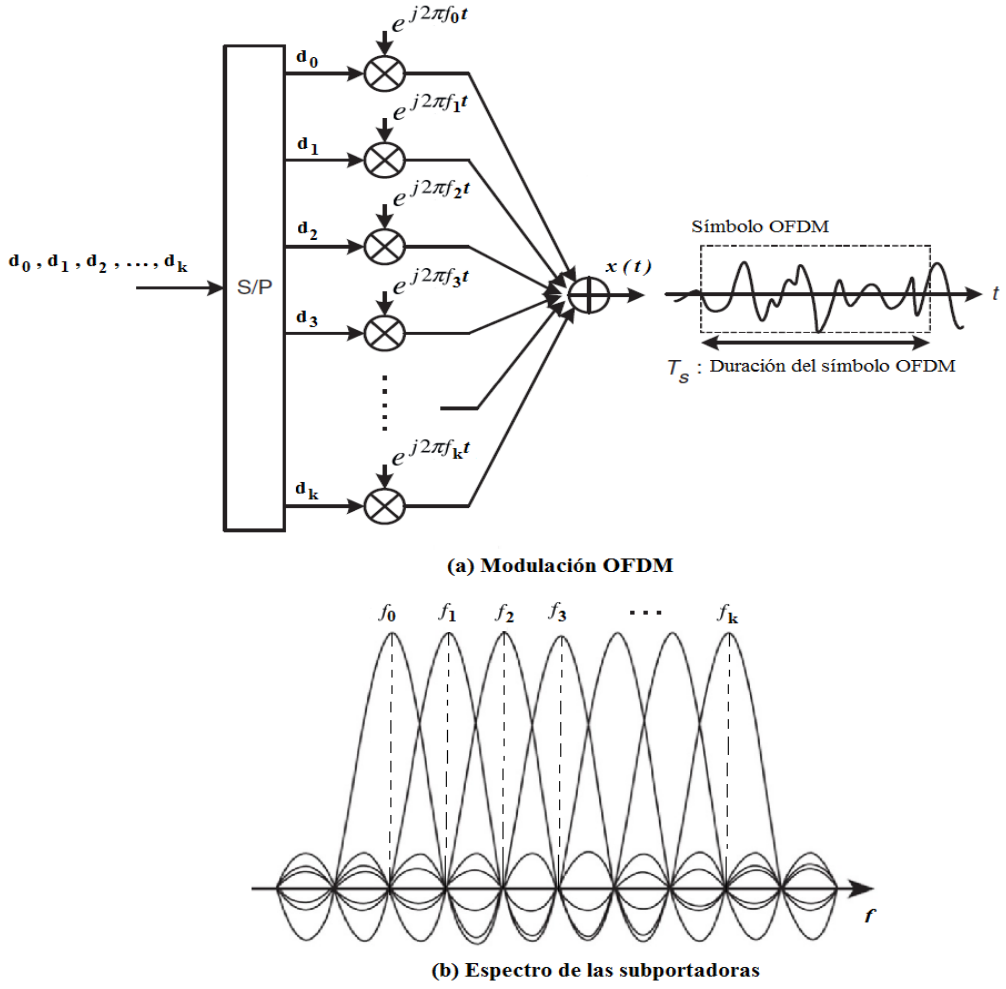


Figura 2.4: Diagrama ilustrativo de la modulación OFDM.

En el modulador OFDM se considera que se dispone de un conjunto de k símbolos complejos $d_0, d_1, \dots, d_k$ que se desean transmitir simultáneamente, cada uno de dichos símbolos se modula mediante una de las subportadoras. La señal resultante del proceso, denominada símbolo OFDM, vendrá dada por [14]:

$$x(t) = \sum_{k=0}^{K-1} d_k \, e^{j2\pi k \Delta f t} \tag{2.1}$$

Donde $f_k = k\Delta f$ es la frecuencia de la k-ésima subportadora, $\Delta f = 1/T_s$ es la separación entre subportadoras. La duración de un símbolo OFDM es igual a N veces el periodo de un bit individual (T_{sc}), es decir $T_s = NT_{sc}$. El ancho de banda se obtiene dividiendo el número de subportadoras entre la duración de símbolo OFDM, esto es $BW = N\Delta f = N/T_s$ [12].

Como se ilustra en la Figura 2.4 (b) en cada valor de f_k se presenta solo la contribución espectral de una única subportadora mientras que el resto presentan nulos, para implementar el sistema es conveniente tener el valor de la contribución espectral de las subportadoras en esos puntos, ya que en esos puntos no hay interferencia de las demás subportadoras, entonces para esto se deben tener muestras con una frecuencia $f_m = N\Delta f$ y por lo tanto un período de muestreo $T_m = 1/(N\Delta f) = T_s/N$, esto es, tomando N muestras de la señal en cada período de símbolo T_s [14].

En los sistemas OFDM y OFDMA actuales la Transformada Rápida de Fourier (FFT) y su contraparte la Transformada Rápida de Fourier Inversa (IFFT) se utilizan para modular y demodular los símbolos de datos sobre las subportadoras, en lugar de emplear múltiples osciladores lo que hace viable su implementación, y es por eso que el número de subportadoras está ligado al número de muestras de la transformada. También cabe mencionar que la diferencia entre OFDM y OFDMA es que en la última cada subportadora puede pertenecer a usuarios distintos [14].

2.2 IFFT / FFT

Uno de los principales atractivos de OFDMA es que existen algoritmos muy eficientes para calcular la Transformada Discreta de Fourier (DFT) y la Transformada Discreta de Fourier Inversa (IDFT). Este conjunto de algoritmos y técnicas se conocen colectivamente como FFT (Fast Fourier Transform) y fueron ampliamente estudiados desde su descubrimiento por Cooley y Tukey [10].

La IFFT y la FFT son la parte central de un sistema OFDMA, estos bloques nos permiten cambiar la señal del dominio de la frecuencia al dominio del tiempo y viceversa (ver Figura 2.5). Los bloques de la IFFT y FFT son necesarios para realizar los procesos de modulación y demodulación en múltiples trayectorias con la certeza de ser ortogonales entre sí, característica de suma importancia que permite la transmisión simultánea de múltiples símbolos [16,17].

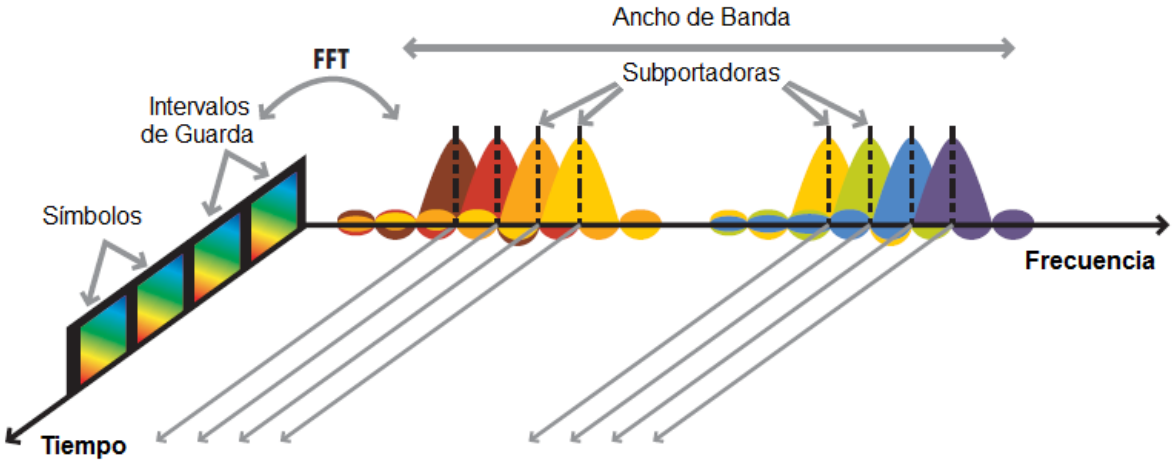


Figura 2.5: Aplicación de la Transformada Rápida de Fourier en OFDMA.

Las fórmulas para obtener la Transformada Discreta de Fourier y su inversa son:

$$X(k) = \sum_{n=0}^{N-1} x(n) e^{\left(-j\frac{2\pi}{N}nk\right)}, \qquad k = 0, 1, \dots, N-1. \tag{2.2}$$

$$x(n) = \frac{1}{N} \sum_{n=0}^{N-1} X(k) e^{\left(j\frac{2\pi}{N}nk\right)}, \qquad n = 0, 1, \dots, N-1. \tag{2.3}$$

Donde N es el número de muestras de las cuales se desea obtener la DFT. Las expresiones para la DFT e IDFT son muy similares, solo difieren en el signo del exponente y el factor de escalamiento $1/N$, por lo que cualquier algoritmo para DFT puede ser fácilmente adaptado para el cálculo de la transformada inversa.

Si desarrollamos de forma directa las expresiones para la DFT sería necesario hacer $(N-1)(N)$ sumas complejas y $(N)(N)$ multiplicaciones complejas. Esta forma de obtener la DFT no es eficiente y para su implementación se necesitarían demasiados recursos. Los algoritmos del tipo Cooley-Tukey reducen el número de sumas y de multiplicaciones necesarias para obtener la DFT. La idea básica de este tipo de algoritmos radica en poder descomponer una DFT en varias DFT de menor tamaño hasta llegar a transformadas de tamaño 2. Una vez resueltas las transformadas más simples se tienen que agrupar en otras de nivel superior que deben resolverse de nuevo y así sucesivamente hasta llegar al nivel más alto. Al final de este proceso los resultados obtenidos deben reordenarse para así tener el resultado completo de la DFT [10,17].

2.3 Estándar IEEE 802.16

El foro WIMAX (Worldwide Interoperability for Microwave Access) fue creado con el objetivo de certificar y promover la compatibilidad e interoperabilidad de los productos inalámbricos de banda ancha basados en los estándares IEEE 802.16. La meta de éste es acelerar la introducción de éstos sistemas dentro del mercado. El foro WIMAX certifica productos que son completamente interoperables y soportan servicios de banda ancha fija, móvil y portátil [18].

El grupo de trabajo IEEE 802.16 diseña normas y hace recomendaciones para el desarrollo y despliegue de las redes inalámbricas de banda ancha de área metropolitana (WMAN). El IEEE 802.16 es parte del comité de normas IEEE 802 LAN/MAN (Local Area Network/ Metropolitan Area Network). El objetivo del comité IEEE 802.16 fue diseñar un sistema de comunicación inalámbrica, que incorporara nuevas tecnologías de comunicación y procesamiento digital de señales para ofrecer Internet de banda ancha, telefonía por Internet, y demás servicios, a usuarios móviles en un área metropolitana [2].

El estándar IEEE 802.16 define cuatro especificaciones de capa física (PHY), cualquiera de las cuales puede ser usada con la capa de acceso al medio (MAC) para desarrollar un sistema inalámbrico de banda ancha. A continuación se describen brevemente dichas especificaciones [19,20]:

- WirelessMAN SC (Single-carrier), describe una capa PHY de una sola portadora para frecuencias entre 10 y 66 GHz, y requiere una condición de línea de vista (LOS). Esta capa PHY es parte de las especificaciones originales del estándar.
- WirelessMAN SCa, especifica una capa PHY de una sola portadora para frecuencias por debajo de los 11 GHz, que soporta una condición sin línea de vista (NLOS). Diseñada para operaciones punto a multipunto.
- WirelessMAN OFDM, describe una capa PHY basada en OFDM de 256 puntos para la FFT. Para operaciones punto a multipunto en condiciones NLOS (Non line of sight), a frecuencias por debajo de los 11 GHz. Esta capa física, finaliza en el estándar IEEE 802.16-2004, ha sido aceptada por el foro WIMAX para operaciones fijas y es a menudo referida como fixedWIMAX.
- WirelessMAN OFDMA, especifica una FFT de 2048 puntos basada en OFDMA empleada para operaciones en condiciones sin línea de vista (NLOS) y a frecuencias por debajo de los 11 GHz. En la versión móvil del estándar IEEE 802.16e, esta capa física ha sido modificada para usar S-OFDMA (Scalable OFDMA), donde el tamaño de la FFT es variable y puede tomar cualquiera de los siguientes valores: 128, 256, 512, 1024 o hasta 2048. El tamaño variable de la FFT permite una operación/implementación óptima del sistema, sobre un amplio rango de anchos de banda por canal y condiciones de radio. Esta capa física ha sido aceptada por el foro WIMAX para operaciones móviles y portátiles, y se conoce como mobile WIMAX.

Para el desarrollo de la propuesta de este trabajo de tesis, se consideró la especificación WirelessMAN OFDMA. Con el objetivo de comprender de mejor manera dicha propuesta es necesario revisar algunos conceptos y bloques especificados en la capa PHY, los cuales se describen a continuación.

2.4 Transmisor y Receptor OFDMA basados en el estándar IEEE 802.16

En la Figura 2.6 se muestran los bloques que conforman el transmisor y el receptor OFDMA basados en el estándar IEEE 802.16 [7].

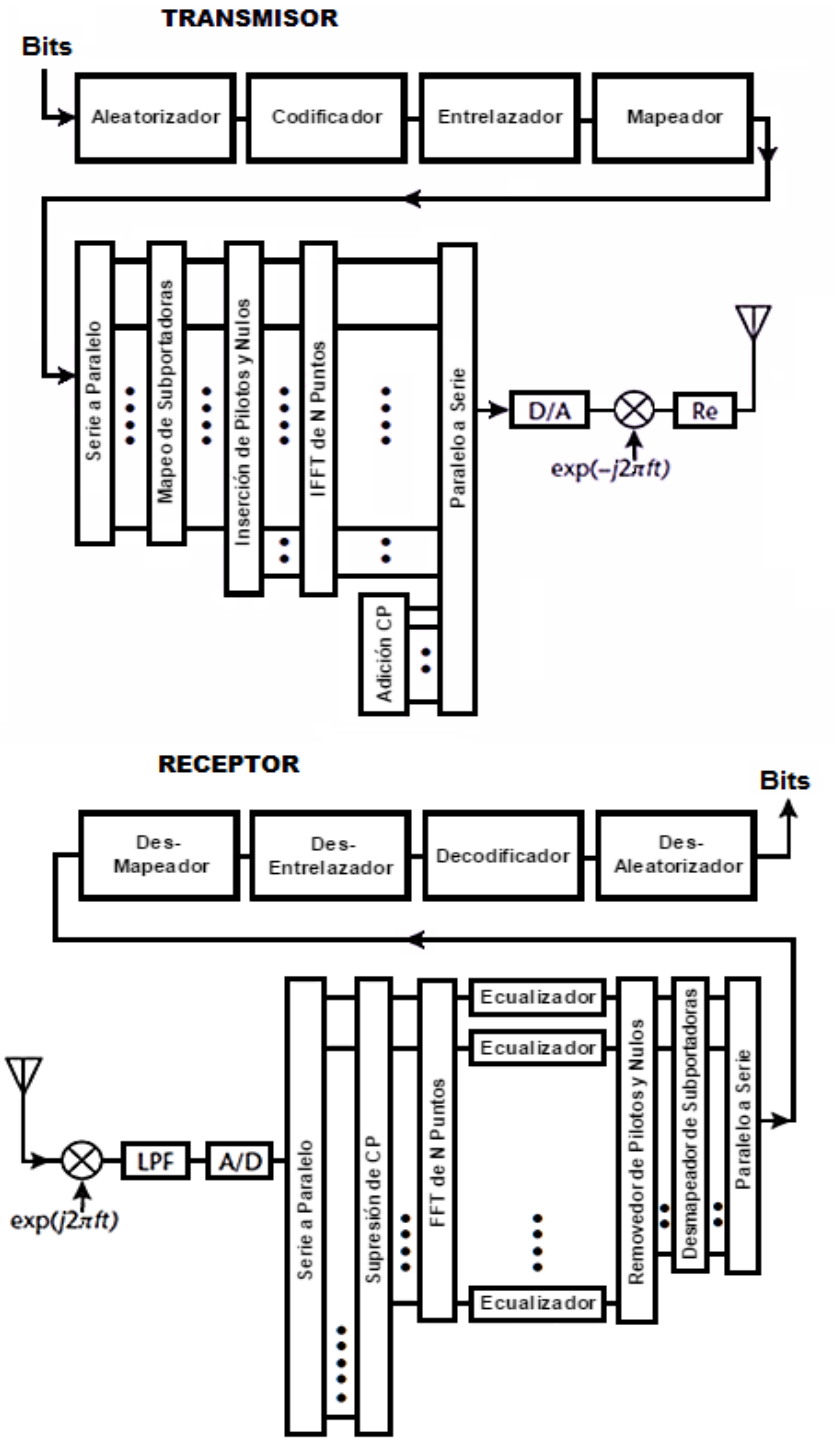


Figura 2.6: Transmisor y Receptor OFDMA.

2.4.1 Codificación del Canal

Todos los tipos de enlace: cable, fibra óptica, radio, entre otros; son susceptibles a sufrir interferencia de algún tipo. La codificación de canal se desarrolló con el fin de minimizar los daños de dichas interferencias sobre la información, sus funciones primarias son prevenir y corregir los errores de transmisión. Es por ello que el estándar IEEE 802.16 prevé estas situaciones y define una serie de procesos que se deben llevar al cabo antes de transmitir los datos por la interfaz del aire. El proceso de codificación del canal en la capa física OFDMA se compone de tres bloques: aleatorizador, codificador convolucional y entrelazador, implementandose en ese orden en la transmisión y en el orden inverso en la recepción [21].

2.4.2 Aleatorizador/Desaleatorizador

La aleatorización se emplea para minimizar la posibilidad de transmisión de una portadora no modulada y para garantizar un número adecuado de transiciones de bit. Además para distribuir los bits para evitar cadenas largas de ceros y unos [20].

El generador de la secuencia binaria pseudo-aleatoria (PRBS) usado para la aleatorización se muestra en la Figura 2.7. Los datos entran de forma secuencial en la aleatorización, el valor inicial de los registros será usado para calcular los bits aleatorios, que se combinan mediante una operación XOR con el flujo de bits en serie de cada ráfaga. La secuencia de aleatorización sólo se aplica a los bits de información [5]. Para este bloque en particular su contraparte en el receptor es el mismo, donde el valor inicial de los registros debe ser el mismo que en el transmisor para poder recuperar la secuencia original.

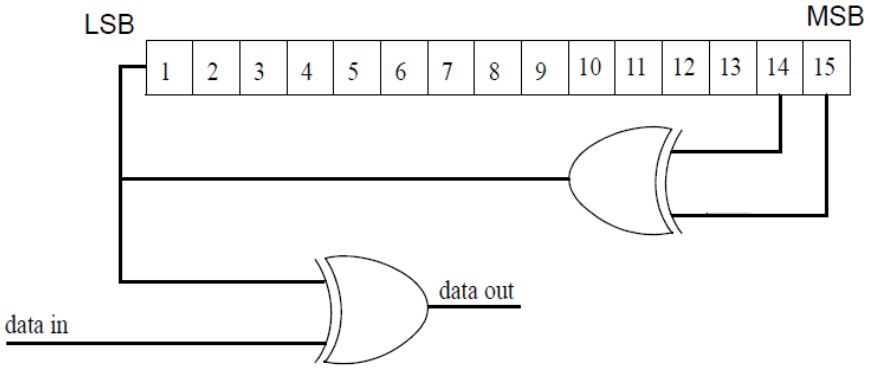


Figura 2.7: Bloque para la aleatorización de datos.

2.4.3 Codificador/Decodificador

La corrección de errores hacia delante o FEC (Forward Error Correction), es un método que permite al receptor corregir datos erróneos sin necesidad de una retransmisión de la información original. Se utiliza para sistemas que operan en tiempo real, donde no se puede esperar a la retransmisión para corregir los datos. La corrección de errores se realiza agregando al mensaje original bits de redundancia. En el estándar IEEE 802.16 se define un FEC conformado por la concatenación de un código Reed-Solomon (externo), y un método de codificación convolucional (interno). Un código convolucional es un código lineal, donde la suma de dos palabras de código, no importa cuales, es también una palabra de código. La implementación de este tipo de código nos da una codificación continua, donde la secuencia de bits codificada depende, además de los bits actuales, de los bits previos. El código trabaja de la siguiente manera: por cada *k* bits de información, cuando se codifican se obtienen *n* bits, donde *k/n* es la tasa de codificación, por ejemplo en una tasa de codificación de 1/2 por cada bit de información se agrega un bit de redundancia. Esta codificación se realiza usando un registro de desplazamiento, formado por *m* flip-flops, donde *m* es el número de bits previos de los que depende la salida actual o memoria del codificador, y una lógica combinacional implementada por compuertas XOR, como se ilustra en la Figura 2.8 [20]:

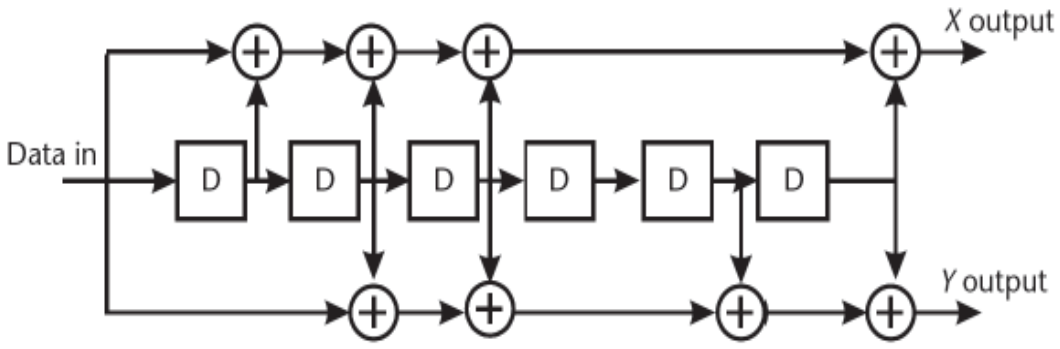


Figura 2.8: Codificador convolucional con una tasa de codificación de 1/2.

La implementación del decodificador está basada en el algoritmo de Viterbi. En el receptor el decodificador se encarga de la búsqueda de errores y la corrección de estos, cuyo principio es la generación de un algoritmo que encuentre el camino más probable de una secuencia de entrada a partir del camino con menor costo, esto se realiza mediante el cálculo de la distancia de Hamming, esta se obtiene de la comparación entre los bits recibidos y los esperados. Si los bits recibidos son iguales a los esperados se tiene una distancia de Hamming igual a cero, y si los bits no coinciden se tendrá una distancia igual al número de bits diferentes. Para el caso de los códigos convolucionales, los diferentes caminos se expresan mediante el uso de diagramas de Trellis, cada camino en el diagrama representa un costo (obtenido de la suma de las distancias de Hamming de cada comparación con los bits recibidos), el algoritmo debe comparar los costos de los caminos y decidir cuál tiene la mayor probabilidad de haber sucedido [22,23]. En la Figura 2.9 se muestra el diagrama de Trellis correspondiente al codificador en el transmisor, en donde se tienen 64 estados, que son todas las combinaciones posibles de los valores que pueden tener los flip-flops, además se inicia la decodificación considerando que todos los flip-flops comienzan con un valor de cero, y que siguiendo el algoritmo solo se tienen dos estados posibles. En este diagrama de Trellis se ejemplifica una secuencia de cuatro bits recibidos, en donde al inicio de la decodificación un bit no coincide con los esperados.

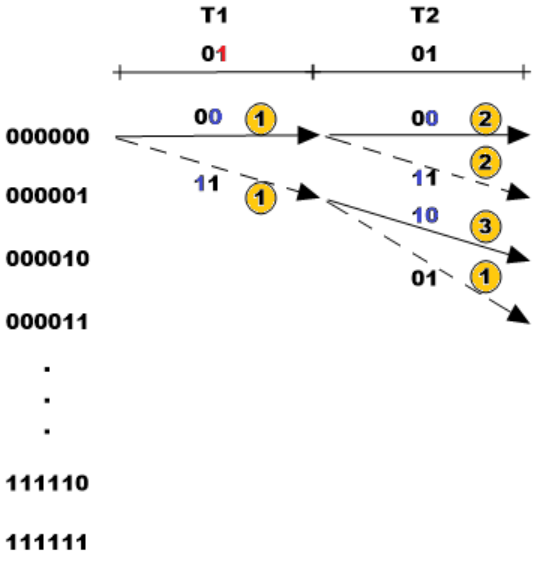


Figura 2.9: Diagrama de Trellis.

2.4.4 Entrelazador/Desentrelazador

El entrelazado se hace para proteger los datos de errores consecutivos durante la transmisión, la idea básica del entrelazado es variar el orden en que se encuentran originalmente los datos a enviar, y que posteriormente serán transmitidos a través de las diferentes subportadoras. De esta forma hablando en términos de OFDM, cada bloque de bits entrelazados se verá afectado por desvanecimientos independientes. Si la mayoría de los bits se reciben de forma correcta, los pocos errores que se produzcan podrán ser corregidos por el decodificador. En lo que se refiere al proceso de desentrelazado en el receptor, se reorganizan los datos en el orden original. En la Figura 2.10 se muestra el proceso de entrelazado.

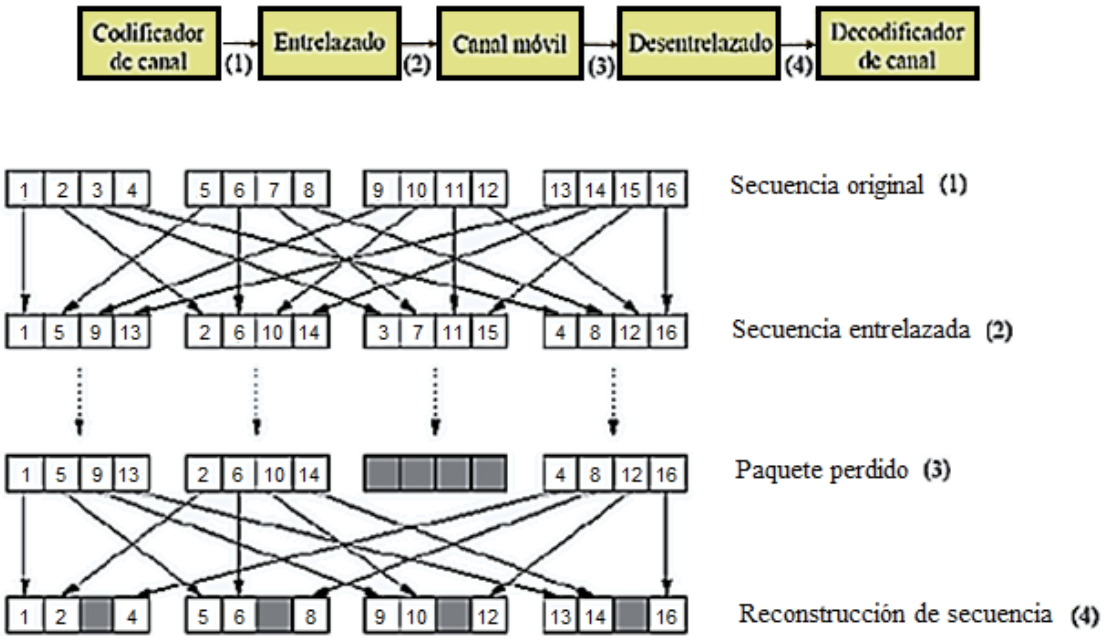
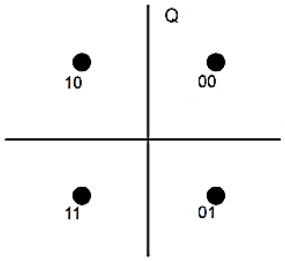


Figura 2.10: Proceso de entrelazado.

2.4.5 Mapeador/Desmapeador

El mapeo se realiza tomando grupos de bits que son asignados a puntos de una constelación. Una magnitud y fase específica representan una determinada combinación de bits. Así una secuencia de bits son mapeados a una secuencia de símbolos de modulación $S(k)$, donde k corresponde al número de símbolo. El número de bits por símbolo depende

del tipo de modulación. En el estándar IEEE 802.16 se especifica la modulación BPSK, QPSK, 16QAM y 64QAM. Un símbolo de modulación $S(k)$ tendrá la forma I+jQ. La Figura 2.11 describe el mapeo de bit para una modulación QPSK que es la empleada en este trabajo. El desmapeador en el receptor obtienen los bits de los símbolos modulados, considerando que estos estén dispersos debido a los efectos del canal durante la transmisión.



$B(1)$	$B(0)$	I	Q
0	0	1	1
0	1	1	-1
1	0	-1	1
1	1	-1	-1

Figura 2.11: Mapeo utilizando QPSK.

2.4.6 Inserción de Pilotos y Nulos.

En OFDMA no todas las subportadoras son usadas para transmitir datos, el estándar IEEE 802.16 exige el uso de subportadoras piloto y subportadoras nulas. Las subportadoras piloto tienen diversas utilidades, entre ellas el facilitar la sincronización y para la estimación del canal, así como la detección de desplazamientos en fase y frecuencia. Las subportadoras nulas son utilizadas como bandas de guarda y de DC. Las subportadoras de banda ayudan a contener el espectro de la señal en los bordes de la banda, su finalidad es evitar interferencia generada por el traslape con los canales adyacentes. Solo es usada una subportadora de DC necesaria para no introducir componente de DC a la señal OFDM. La Figura 2.12, muestra la estructura del símbolo OFDM [20].

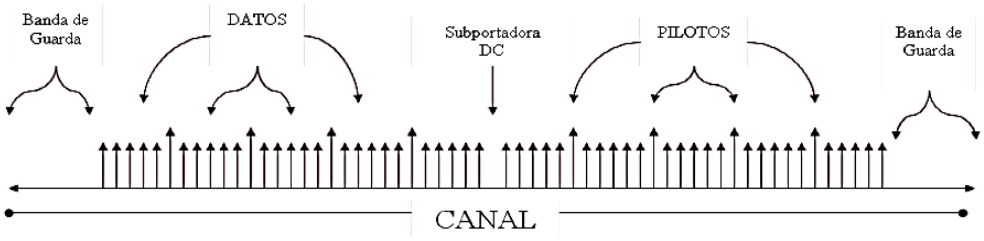


Figura 2.12: Estructura del símbolo OFDM en el dominio de la frecuencia.

2.4.7 Adición de un periodo de guarda para OFDM

Una de las propiedades más importantes de las transmisiones OFDM es la solidez en contra de los retardos de propagación por multitrayectoria. Esto se logra aumentando el período del símbolo, lo que minimiza las interferencias Inter-símbolo. De hecho, el nivel de robustez puede aumentar aún más mediante la adición de un período de guarda comprendido entre los símbolos trasmitidos. El período de guarda da tiempo a que las múltiples señales de los símbolos previos se desvanezcan antes de que la información del símbolo actual se una. La forma más eficaz de periodo de guarda usado es una extensión cíclica del símbolo. Si, en el dominio del tiempo, se agrega una copia espejo del final de la onda del símbolo al comienzo del mismo, como período de guarda, ésta extiende efectivamente la longitud del símbolo, al tiempo que se mantiene la ortogonalidad de la forma de onda ya que se tendrá un numero entero de periodos en el tiempo útil del símbolo. En la Figura 2.13 se ejemplifica la inserción del prefijo cíclico.

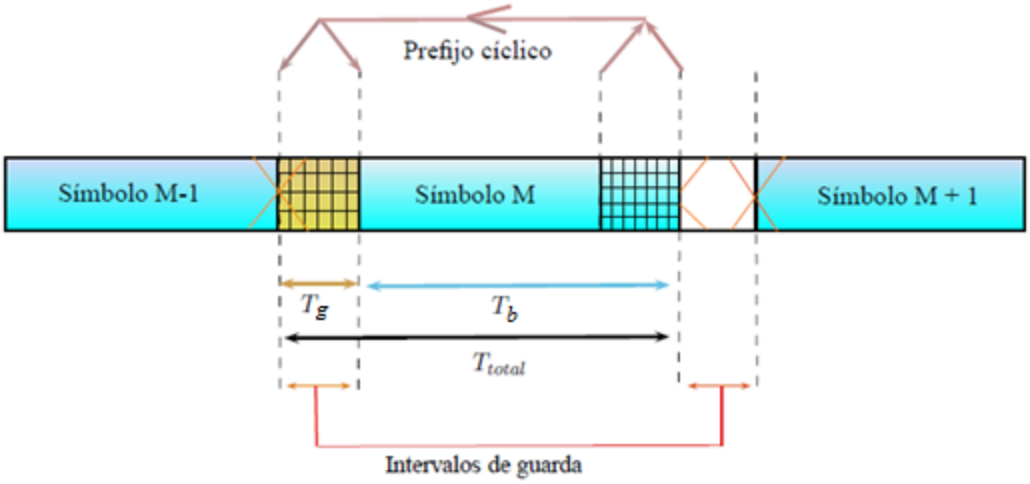


Figura 2.13: Inserción del prefijo cíclico.

El símbolo OFDM resultante tiene una duración total de $T_b + T_g$, donde T_g es la duración del intervalo de guarda y T_b es el tiempo útil del símbolo.

Para evitar totalmente la ISI, la respuesta al impulso del canal T_h debe ser menor que T_g del prefijo cíclico. Es decir se debe cumplir que $T_h < T_g$, como se ilustra en la Figura 2.14 [5].

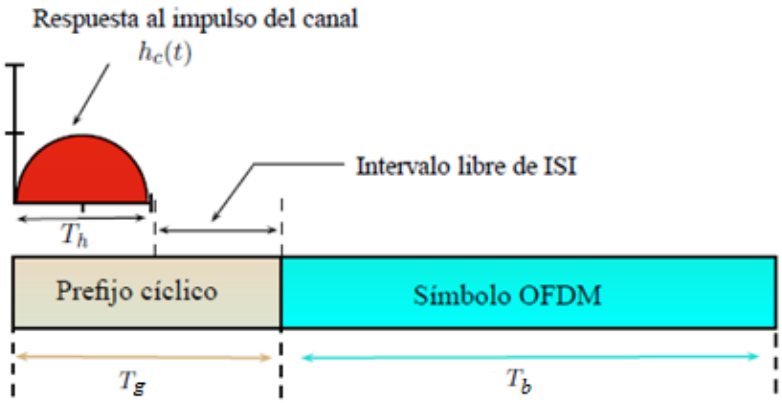


Figura 2.14: Prefijo cíclico para evitar interferencia entre símbolos.

2.4.8 Conversiones Digital-Analógico/Analógico-Digital

En el transmisor después de integrar todas las subportadoras como lo indica la estructura del símbolo OFDM en el dominio de la frecuencia, se procede al cálculo de la IFFT, a la salida de la IFFT obtenemos el símbolo en el dominio del tiempo, luego se hace la copia de la extensión cíclica, para obtener la señal de forma analógica se hace uso de un par de convertidores digital a analógico, donde los datos son introducidos de forma serial con un tiempo de duración T_m. Después de la conversión digital-analógica la componente real es multiplicada con una forma de onda senoidal y la componente imaginaria es multiplicada con una forma de onda cosenoidal. Las formas de onda resultantes de las multiplicaciones se suman para formar una única señal que es la que será transmitida por la interfaz del aire.

En el receptor se emplean un par de multiplicadores y filtros pasa-bajas para recuperar la componente real e imaginaria, después las señales obtenidas son enviadas a un par de convertidores analógico-digital, en donde se tienen que muestrear las señales con el mismo tiempo (T_m) empleado en el transmisor, antes de calcular la FFT en el receptor se elimina el prefijo cíclico, a la salida de la FFT se obtiene el símbolo OFDM nuevamente en el dominio de la frecuencia. Estando de nuevo en el dominio de la frecuencia, con la ayuda de las subportadoras piloto se realiza una estimación del canal y se trata de reducir los efectos del canal en las subportadoras de datos, después de esto se eliminan las subportadoras piloto y nulas, quedando solo las subportadoras correspondientes a los símbolos de datos, que son procesadas en forma digital por los bloques descritos anteriormente correspondientes al receptor.

2.5 Conclusión del capitulo

En este capítulo se presentó una descripción general de la técnica de acceso OFDMA. Se describieron brevemente los bloques que conforman a un transmisor y receptor OFDMA basados en el estándar IEEE 802.16, la descripción de la técnica de acceso y de los bloques son necesarios para comprender más fácilmente la propuesta realizada en este trabajo de tesis.

Hasta el momento ya se tienen los conocimientos básicos y necesarios para facilitar la descripción de los algoritmos desarrollados, que en primera instancia se evaluaron usando el software MATLAB. El sistema OFDMA será implementado mediante una simulación en MATLAB y ampliamente discutido en el siguiente capítulo.

Capítulo 3

Simulación del sistema OFDMA en MATLAB

En este capítulo se presentan las simulaciones realizadas con el software MATLAB de los diferentes bloques que componen un sistema básico de transmisión y recepción OFDMA, en donde se mostrara los diagramas que describen el código implementado y los resultados obtenidos en cada bloque. La simulación previa a la implementación real es necesaria para evaluar el correcto funcionamiento de los algoritmos propuestos y además nos permite visualizar la respuesta e información esperada en cada etapa del sistema.

3.1 Plataforma de simulación

MATLAB es un lenguaje de programación matemático de alto nivel integrado con entorno grafico amigable, visualización de datos, funciones, graficas 2D y 3D, procesamiento de imágenes, video, computación numérica para desarrollar algoritmos matemáticos con aplicaciones en ingeniería y ciencias exactas.

El ambiente de programación de MATLAB está compuesto por una interface gráfica con varias herramientas distribuidas en ventanas que permiten programar, revisar, analizar, registrar datos, utilizar funciones y desarrollar diversas aplicaciones. Además contiene un enorme número de funciones que facilitan la representación gráfica y visualización de variables, funciones, vectores, matrices y datos que pueden ser graficados en 2 y 3 dimensiones [24].

3.2 Parámetros de simulación

La etapa de simulación consistió en la creación de cada uno de los bloques que conforman el transmisor y receptor OFDMA, los parámetros característicos de la simulación fueron tomados del estándar IEEE 802.16-2009 para redes de área local y metropolitana aplicable a sistemas con acceso inalámbrico de banda ancha que utilizan la interfaz de aire, los parámetros se listan a continuación [20]:

- Tasa de codificación de 1/2.
- Modulación digital QPSK.
- Tamaño de la IFFT de 128.

3.3 Etapas del transmisor

En la Figura 3.1 se muestran los bloques que conforman al transmisor basado en el estándar IEEE 802.16 [7,20].

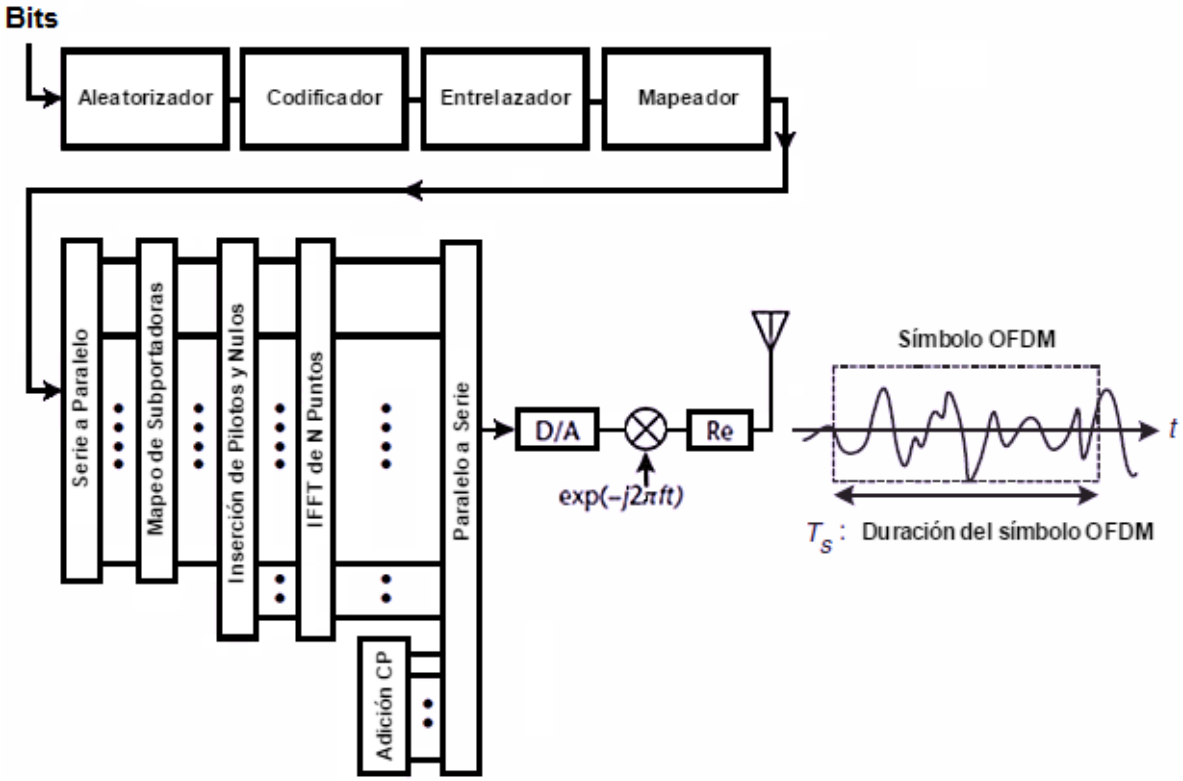


Figura 3.1: Estructura básica de un transmisor en el estándar IEEE 802.16.

En la primera parte de la simulación se presenta el procesamiento requerido para generar el símbolo OFDM que es la señal que se transmite mediante la interfaz del aire. El número de subportadoras empleadas para construir el símbolo es igual al tamaño de la IFFT, en el diseño propuesto la IFFT es de 128 puntos, entonces se tendrán 128 subportadoras de las cuales solo 96 se ocupan para la transmisión de datos. Considerando el tamaño de la IFFT, la tasa de codificación y el tipo de modulación digital a la entrada del sistema se tendrán 96 bits. Los datos de entrada para la simulación se muestran en la Figura 3.2, esta secuencia de bits representada en hexadecimal fue tomada del estándar IEEE 802.16, en este trabajo se usara la misma representación, en hexadecimal, la cual simplifica la notación de las secuencias de bits y además permite comparar los resultados de la simulación con los reportados en el estándar.

Figura 3.2: Datos de entrada al transmisor [20].

3.3.1 Aleatorizador

El aleatorizador es el primer bloque del transmisor y está conformado por 15 registros y 2 compuertas XOR. Al comienzo del proceso de la aleatorización los registros son cargados con un valor inicial, después el valor de los registros se desplaza en cada ciclo con lo que el valor del último registro se desprecia y el valor del primer registro es obtenido con una operación XOR entre los dos últimos registros, a la salida de esta compuerta XOR se obtiene una secuencia pseudo-aleatoria que se combina mediante otra operación XOR con los bits de entrada que son introducidos de forma serial. En la Figura 3.3 se presenta el diagrama de flujo que describe el proceso de la aleatorización y en la Figura 3.4 se muestran los resultados de la simulación en MATLAB de este bloque.

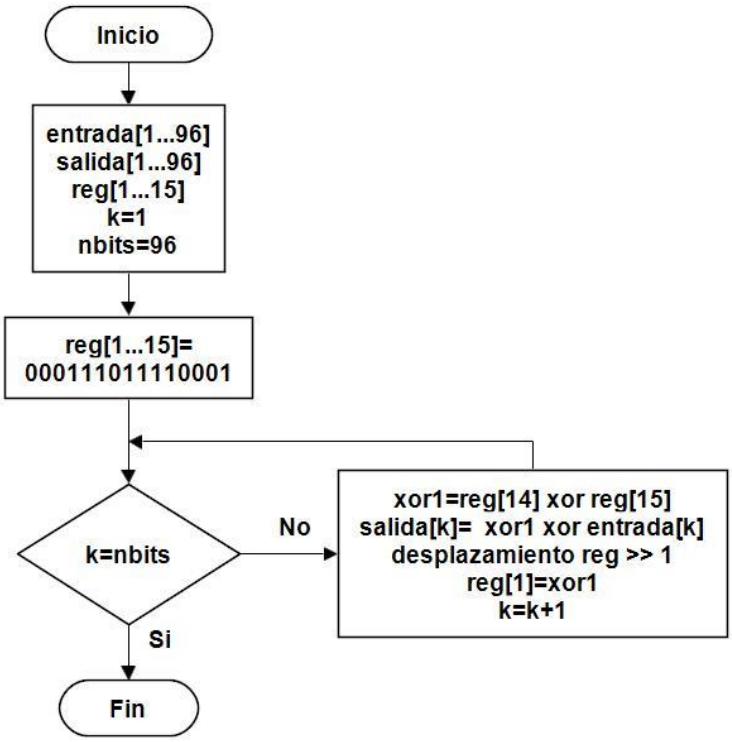


Figura 3.3: Diagrama de flujo para la aleatorización.

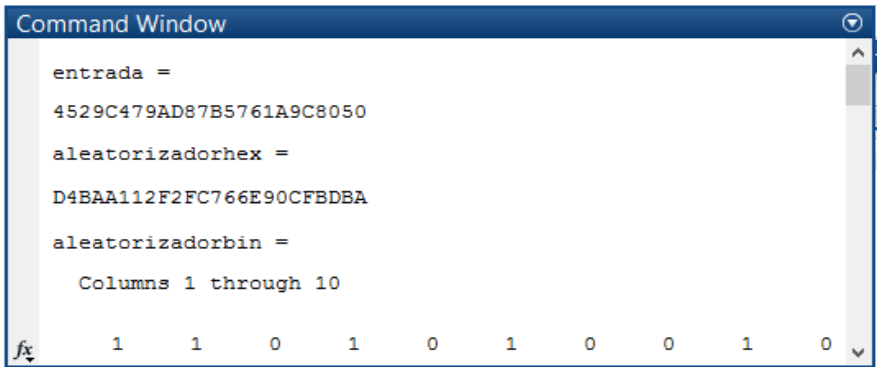


Figura 3.4: Simulación en MATLAB correspondiente a la aleatorización.

3.3.2 Codificador

La codificación es necesaria para poder corregir errores en la transmisión, y esta se realiza combinando mediante compuertas XOR o sumas binarias sin acarreo el bit de entrada con los bits de entrada previos que son almacenados usando registros, el valor de estos registros es cero al inicio de la codificación, en la Figura 3.5 se presenta el diagrama de flujo que describe el proceso de la codificación.

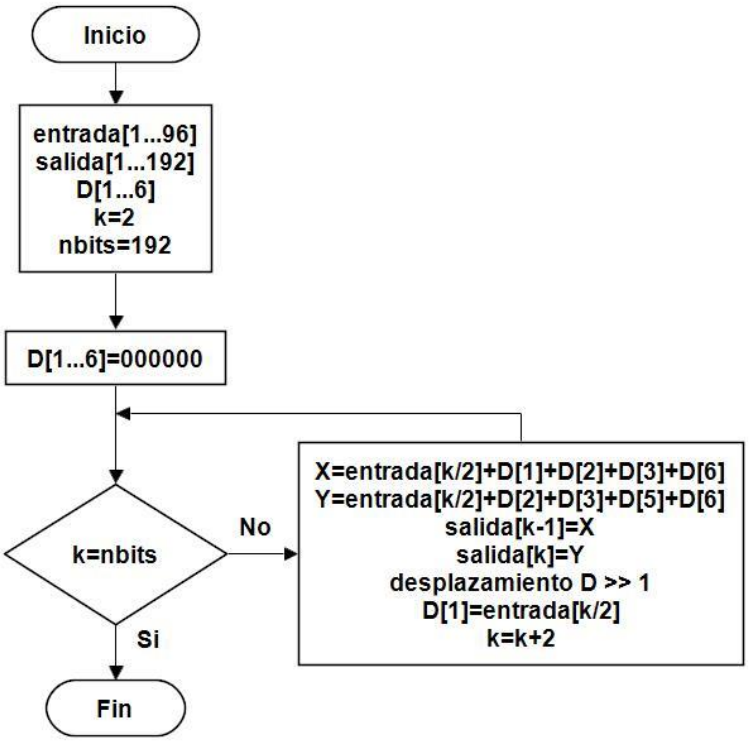


Figura 3.5: Diagrama de flujo para la codificación.

En el diagrama las expresiones para obtener X y Y corresponden al codificador convolucional seleccionado, entonces por cada bit de entrada se tendrá un valor para cada expresión, la salida de este bloque se obtiene combinando alternadamente los bits generados por los polinomios. A la entrada de este bloque se tienen 96 bits provenientes del aleatorizador, y a la salida 192 bits. En la figura 3.6 se muestra la simulación de la codificación en MATLAB.

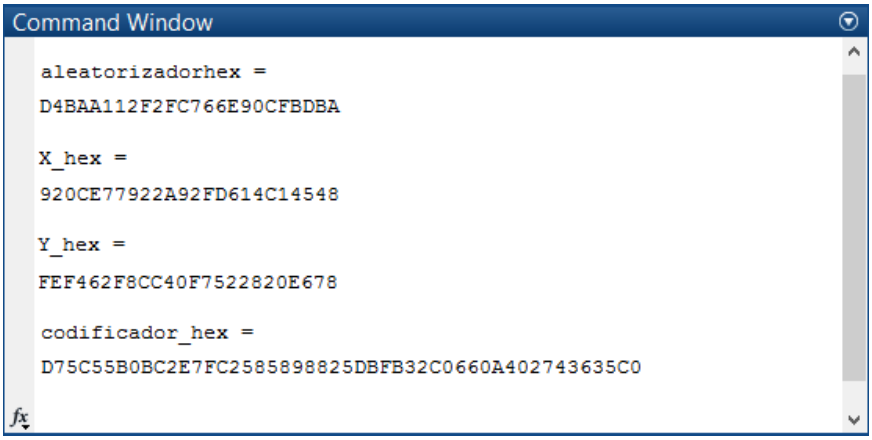


Figura 3.6: Simulación de la codificación en MATLAB.

3.3.3 Entrelazador

La idea básica del entrelazador es variar el orden en que se encuentran originalmente los datos a enviar, las expresiones para obtener las nuevas posiciones fueron tomadas del estándar IEEE 802.16 [20]. En la Figura 3.7 se ilustra el diagrama a flujo que describe el programa desarrollado para este bloque, y en la Figura 3.8 se presentan los resultados de la simulación.

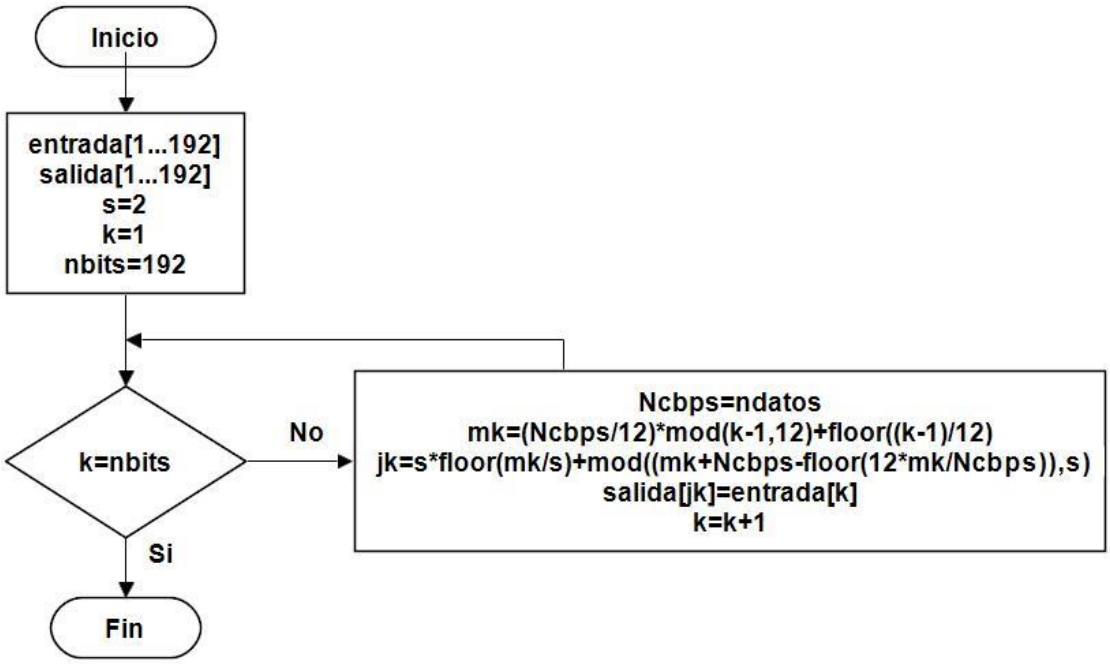


Figura 3.7: Diagrama de flujo para el entrelazado.

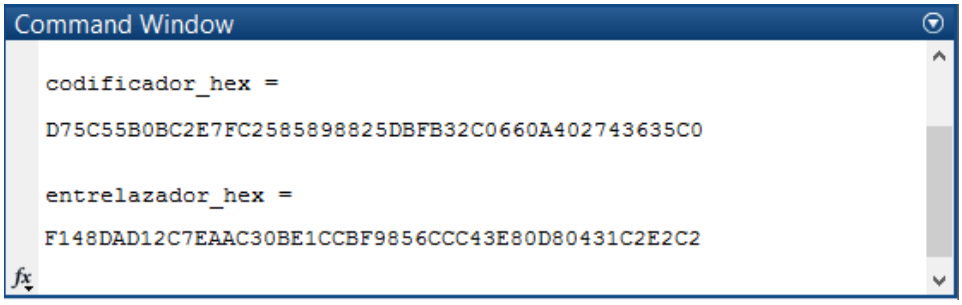


Figura 3.8: Resultado de la simulación del entrelazado en MATLAB.

3.3.4 Mapeador de Símbolo

El mapeo se realiza tomando grupos de bits que son asignados a puntos de una constelación especifica. Una magnitud y fase específica representan una determinada combinación de bits. En la Figura 3.9 se presenta el diagrama de flujo que describe el mapeo de símbolos usando una modulación QPSK, los bits son introducidos en pares a este bloque, y a cada par de bits se le asigna un numero complejo también llamado símbolo de dato.

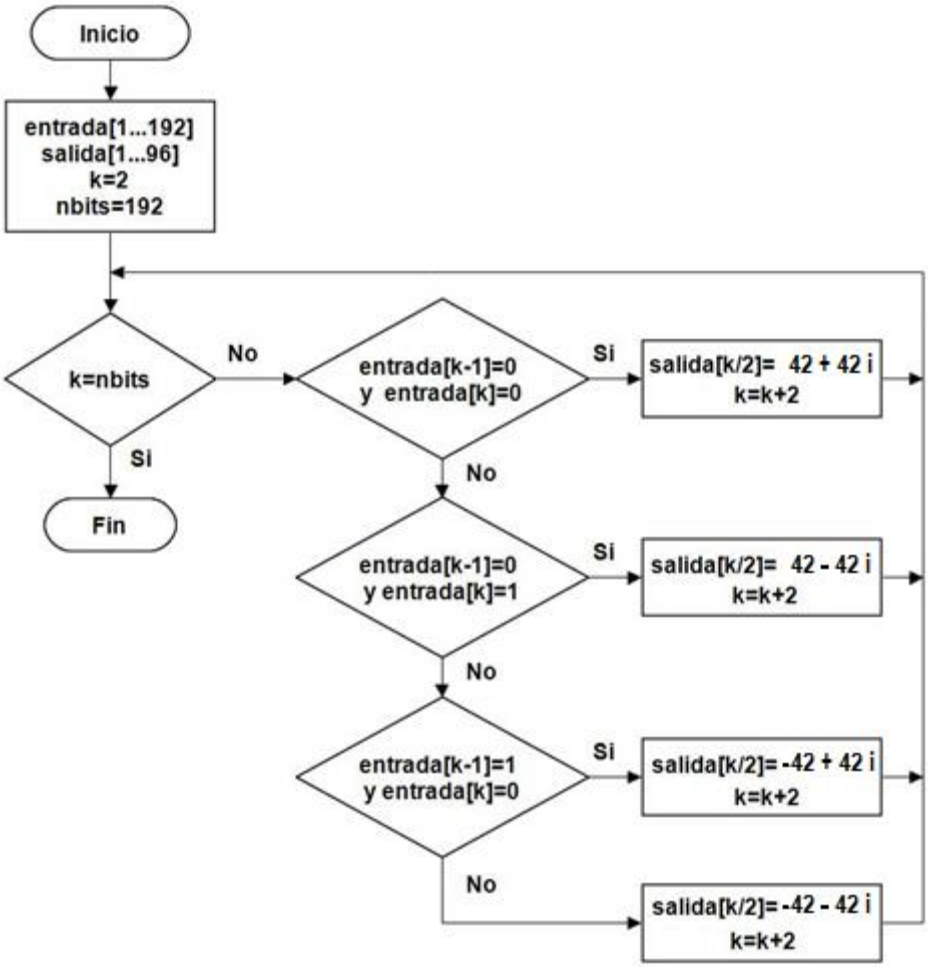


Figura 3.9: Diagrama de flujo para el mapeo de símbolos.

A la salida de este bloque se obtiene 96 datos complejos, que corresponden a las 96 subportadoras de datos, en la Figura 3.10 se muestran los datos mapeados.

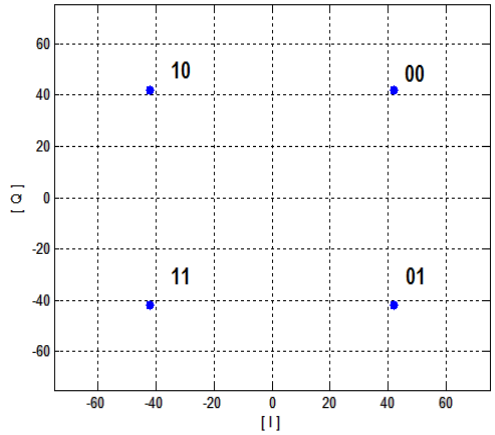


Figura 3.10: Resultado del Mapeo de Símbolos.

3.3.5 Estructura del símbolo OFDM

La estructura del símbolo OFDM está compuesto por sub-portadoras las cuales pueden ser: de datos; subportadoras pilotos que son necesarias para la estimación de canal; nulas que son utilizadas como bandas de guarda, y subportadora de DC. La Figura 3.11, muestra la estructura de un símbolo OFDM [20].

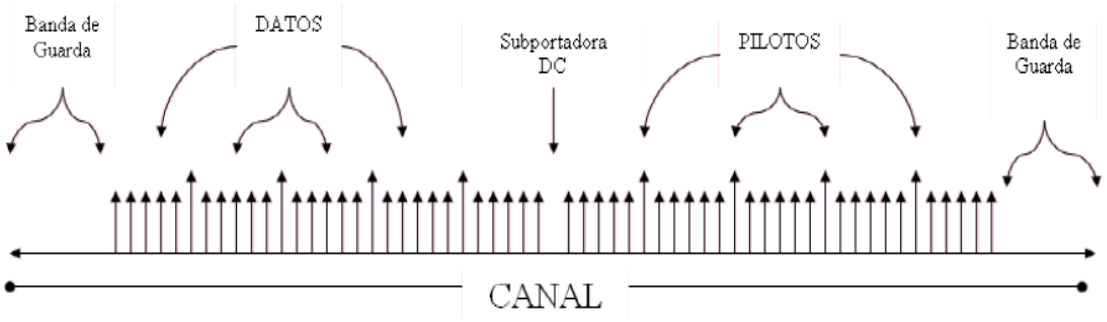


Figura 3.11: Estructura del símbolo OFDM.

El número total de puntos para el cálculo de la IFFT empleada en esta simulación es de 128, por lo tanto tendremos 128 subportadoras, de las cuales 96 corresponden a los datos, 8 a pilotos, 23 para la banda de guarda y una de DC. Los valores de las subportadoras de datos provienen del mapeador de símbolos, el valor de las subportadoras de banda de guarda y DC es cero, a las subportadoras pilotos se les asigna un valor que también debe ser conocido por el receptor. En la Figura 3.12 se presenta el diagrama de flujo que describe la construcción del símbolo OFDM, donde se asigna un valor a cada subportadora tomando los índices de la Tabla 3.1, este valor depende del tipo de subportadora.

Tipo de Subportadora	Índice
Subportadoras de Datos	12,13,14,15,16,17,19,20,21,22,23,24,25,27,28,29,30,31,32, 33,34,35,36,37,38,39,40,41,42,43,45,46,47,48,49,50,51,52, 53,54,55,57,58,59,60,61,62,63,65,66,67,68,69,70,71,73,74, 75,76,77,78,79,80,81,82,83,85,86,87,88,89,90,91,92,93,94, 95,96,97,98,99,100,101,103,104,105,106,107,108,109,111, 112,113,114,115,116
Subportadoras Piloto	18, 26, 44, 56, 72, 84, 102, 110
Banda de Guarda Izquierda	0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11,
Subportadora DC	64
Banda de Guarda Derecha	117,118,119,120,121,122,123,124,125,126,127

Tabla 3.1: Índices de las subportadoras para formar el símbolo OFDM

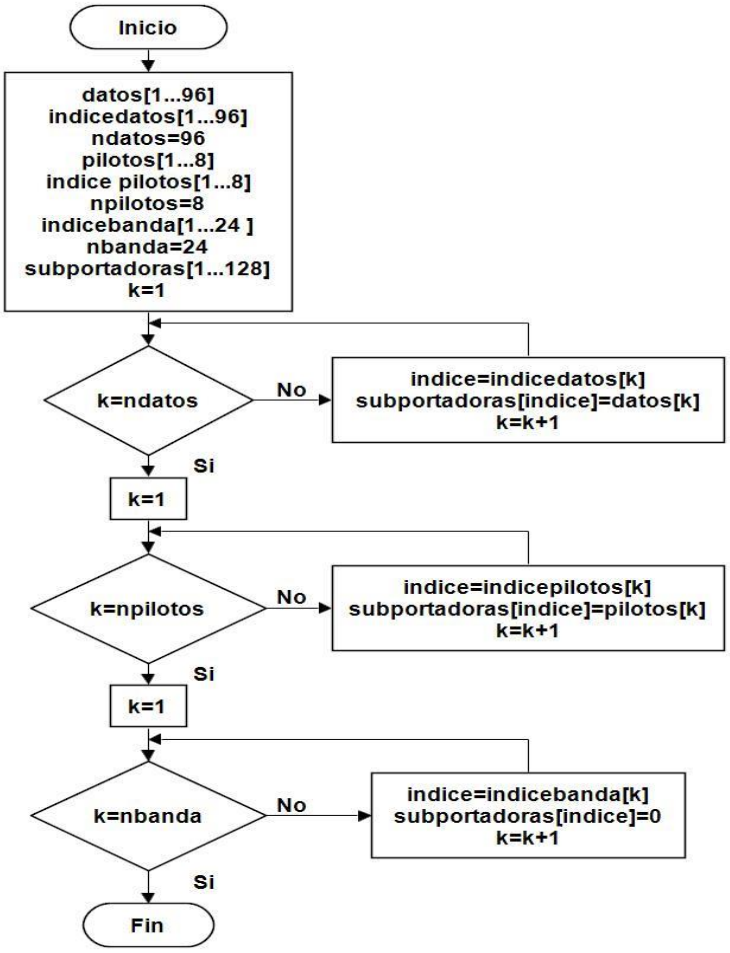


Figura 3.12: Diagrama para armar la estructura del símbolo OFDM.

3.3.6 Calculo de la IFFT

Para generar el símbolo OFDM a trasmitir es necesario obtener la IFFT. Los datos de entrada a la IFFT son los valores asignados a las subpotadoras. El resultado de la IFFT en MATLAB se muestra en la Figura 3.13, a la salida de este bloque también se obtienen números complejos, que se interpretan como muestras para generar el símbolo OFDM.

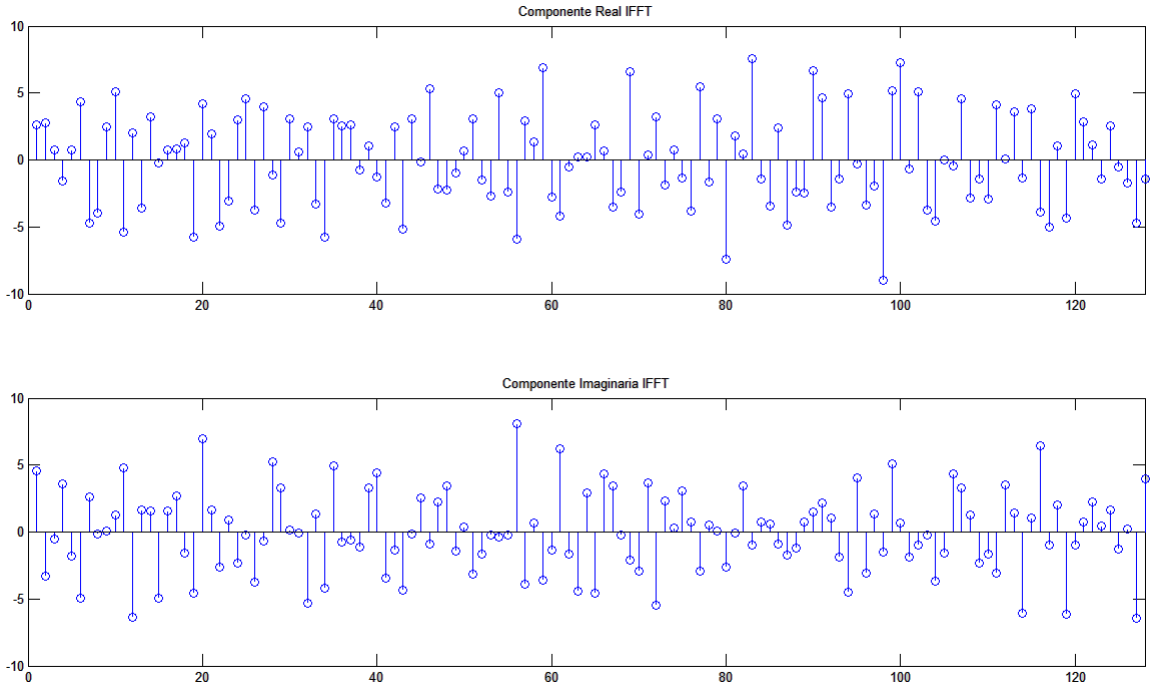


Figura 3.13: Muestras para generar el símbolo OFDM en el tiempo.

3.3.7 Adición del prefijo cíclico

En OFDM en lugar de emplear un tiempo de guarda vacío entre cada símbolo, se envía en ese tiempo una copia de las ultimas muestras que componen el símbolo útil OFDM, en la Figura 3.14 se muestra el resultado de la inserción del prefijo cíclico en MATLAB, la relación del tiempo de guarda con el tiempo útil del símbolo empleado en la simulación es $G = 1/4$, después de obtener la IFFT las ultimas 32 muestras del símbolo útil, resaltadas en color verde, son copiadas al comienzo de las 128 muestras que se tenían originalmente, esto último se visualiza en las muestras en color rojo, las 160 muestras totales son empleadas para generar el símbolo OFDM completo a transmitir.

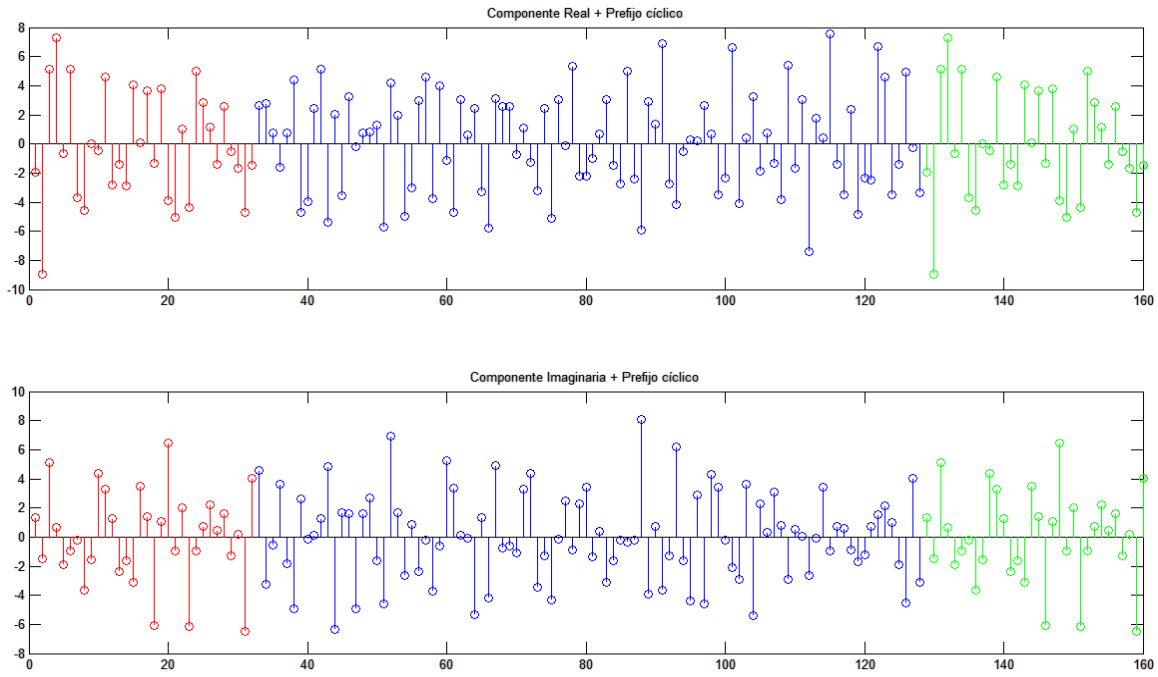


Figura 3.14: Inserción del prefijo cíclico en MATLAB.

3.3.8 Parámetros para generar el símbolo OFDM

Para generar el símbolo OFDM en el dominio del tiempo se tienen que considerar los siguientes parámetros tomados del estándar IEEE 802.16-2009 [20,25].

Parámetros primarios característicos del símbolo OFDM:

Ancho de banda de canal nominal: $BW = 1.25MHz$

Factor muestreo: $n = 8/7$

Tamaño de la FFT o IFFT: $NFFT = 128$

Relación del tiempo de Guarda con el tiempo útil del símbolo: $G = 1/4$

Parámetros derivados:

Frecuencia de muestreo: $Fs = (floor(n \cdot BW/8000)) \cdot 8000 = 1424000\ Hz$

Espacio entre subportadoras: $\Delta f = Fs/NFFT = 11125\ Hz$

Tiempo de símbolo útil: $T_b = 1/\Delta f = 8.9888 \times 10^{-5}\ s$

Tiempo de guarda (prefijo cíclico): $T_g = G \cdot T_b = 2.2472 \times 10^{-5}\ s$

Tiempo del símbolo OFDM: $T_s = T_b + T_g = 1.1236 \times 10^{-4}\ s$

Tiempo de muestreo: $T_m = T_b/NFFT = 7.0225 \times 10^{-7}\ s$

3.3.9 Conversión digital a analógica

Después de obtener la IFFT en el trasmisor el resultado pasa por un convertidor párelo a serie y seguido por un par de convertidores digital a analógico. El parámetro T_m se emplea para determinar el tiempo de duración de cada muestra en su forma analógica y T_s es el tiempo de duración del símbolo OFDM, el símbolo en banda base se muestra en la Figura 3.15, el cual está conformado por dos señales, una representa la parte real y la otra la parte imaginaria.

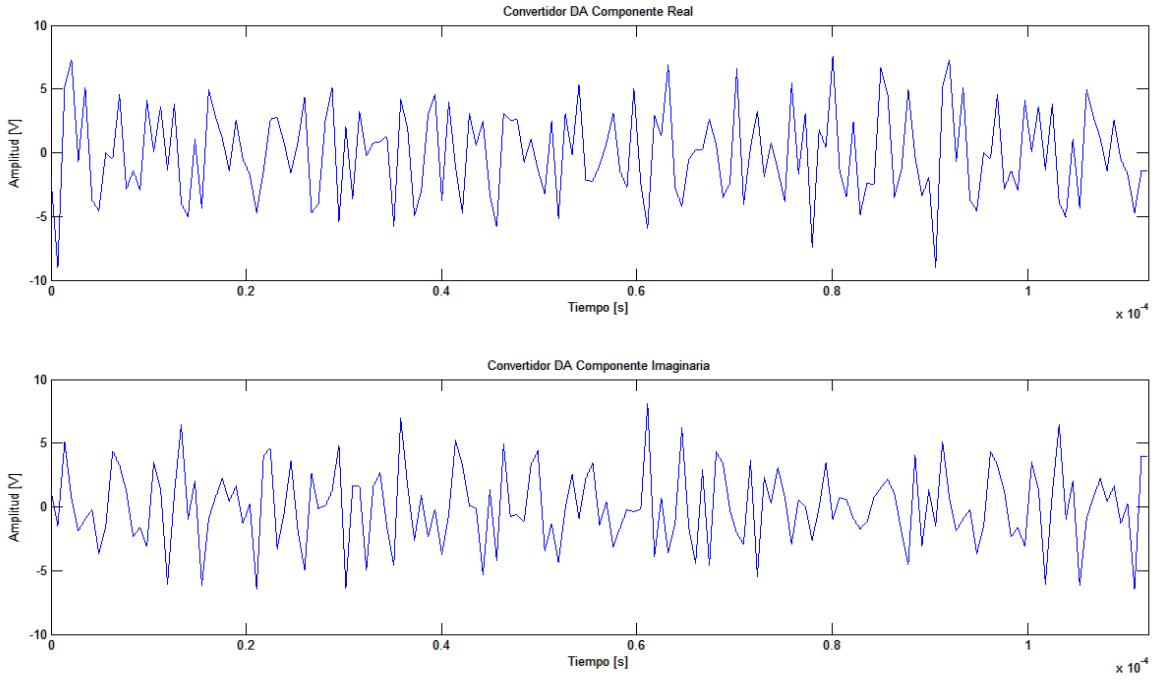


Figura 3.15: Obtención del símbolo OFDM en el dominio del tiempo.

3.3.10 Señal transmitida sobre la interfaz de aire

La última etapa implicada en la transmisión consiste en multiplicar cada componente del símbolo OFDM en el dominio del tiempo por una señal sinusoidal con una frecuencia a la que se desea transmitir. Para poder transmitir las dos señales al mismo tiempo sin que interfieran entre ellas se ocupan dos señales ortogonales, en la simulación se ocupó una función $\cos(2\pi ft)$ y una función $\text{sen}(2\pi ft)$, las dos con una frecuencia de 2.4 GHz, la función coseno multiplica la componente real y la función seno a la componente imaginaria, y después se suman para generar la señal a transmitir, lo anterior se describe en la Figura 3.16 [9].

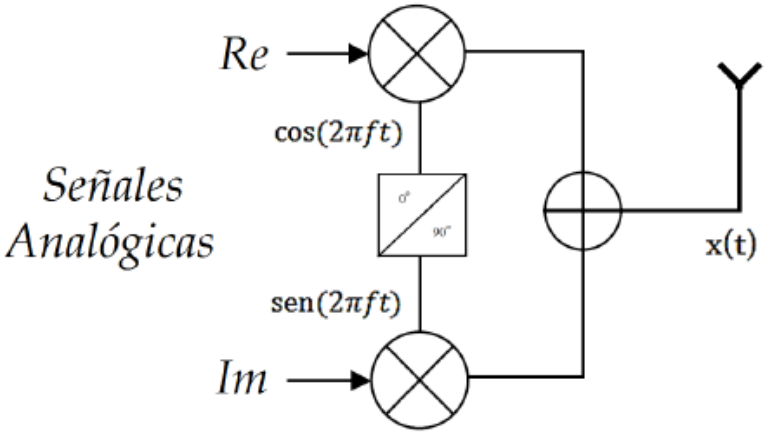


Figura 3.16: Obtención de la señal a transmitir.

En la Figura 3.17 se muestra en la parte superior el resultado obtenido en la simulación de la señal a transmitir y en la parte inferior se ilustra la señal corrompida por ruido blanco aditivo gaussiano (AWGN), el ruido se agregó con la finalidad de evaluar la capacidad de recuperación del receptor.

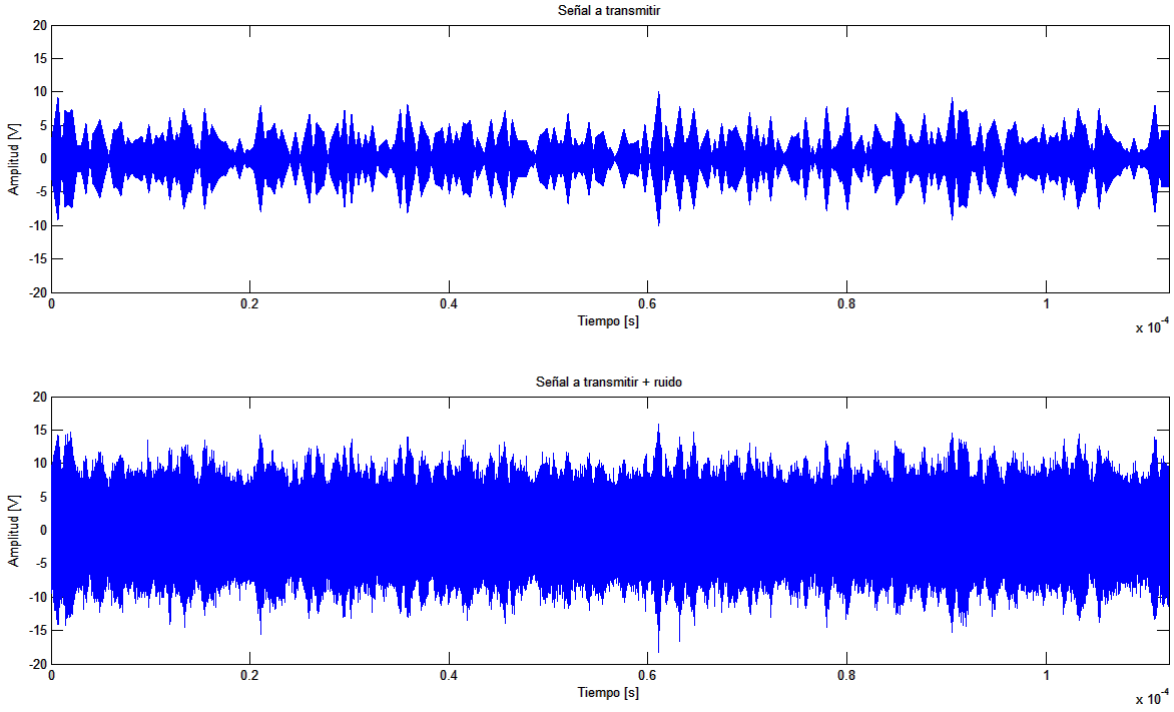


Figura 3.17: Señal a transmitir y señal a transmitir más ruido.

3.4 Etapas del Receptor

El estándar IEEE 802.16 específica en detalle cada bloque del transmisor. Sin embargo las normas no especifican explícitamente la arquitectura del receptor. Esto se hace para dejar algunos detalles de la aplicación de extremo a extremo para el vendedor, y muchos vendedores se diferencian por la forma en que implementan sus receptores. En la Figura 3.18 se presentan los bloques que conformarían a un receptor OFDMA [7].

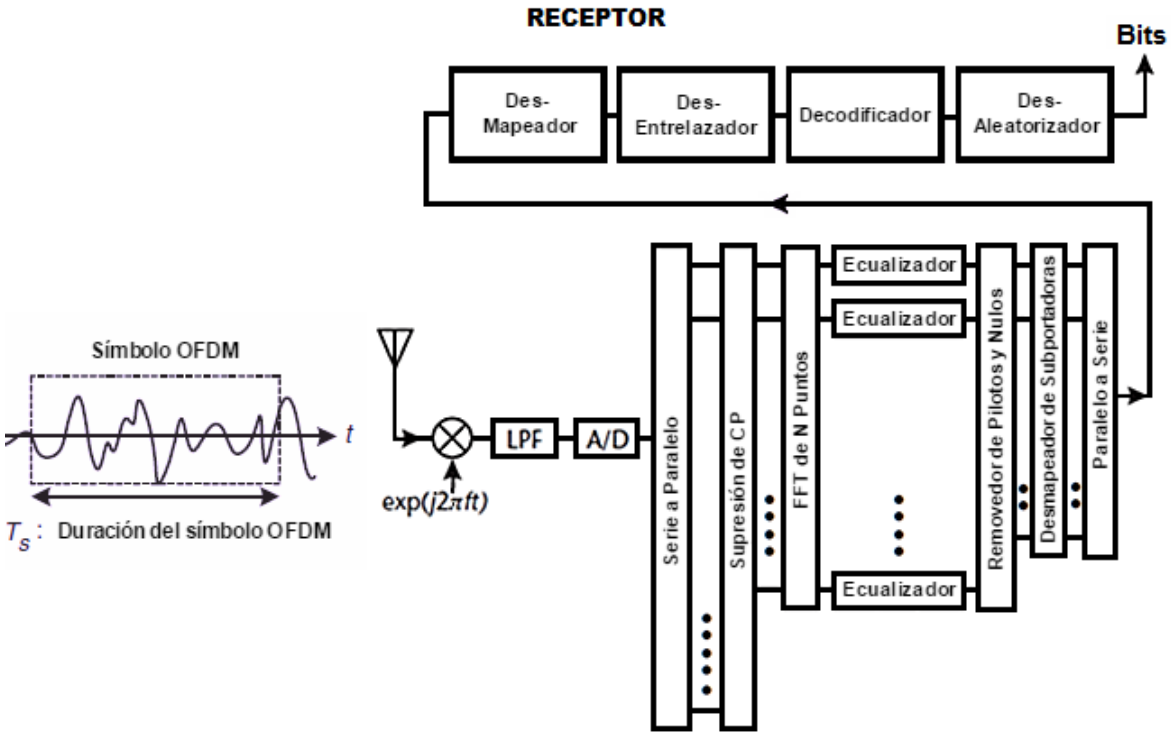


Figura 3.18: Estructura básica del receptor OFDMA en el estándar IEEE 802.16.

3.4.1 Multiplicadores y filtros

En el receptor los primeros bloques constan de un par de multiplicadores seguidos de filtros pasa bajas, necesarios para poder recuperar la componente real e imaginaria en banda base de la señal transmitida como se ilustra en la Figura 3.19 [9].

La señal que se transmite por la interfaz del aire se puede describir con la expresión en (3.1), donde la componente real e imaginaria eran obtenidas de la salida de los convertidores digital-analógico.

$$x(t) = Re(t)\cos(2\pi ft) + Im(t)sen(2\pi ft) \qquad (3.1)$$

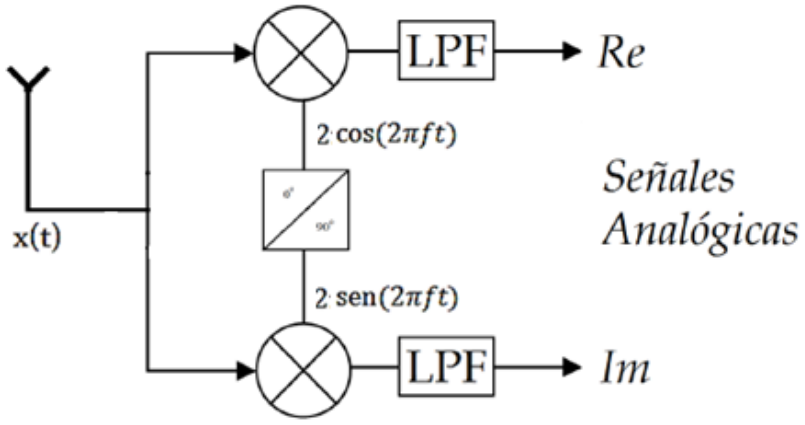


Figura 3.19: Bloque para obtener la forma de la señal original.

Entonces en el receptor lo primero que se tiene que hacer para poder recuperar la componente real e imaginaria es multiplicar (3.1) por $2\cos(2\pi ft)$, y (3.1) por $2\text{sen}(2\pi ft)$, el resultado de esto es (3.2) y (3.3).

$$x_{Re}(t) = Re(t) + Re(t)\cos(4\pi ft) + Im(t)sen(2\pi ft)2\cos(2\pi ft) \quad (3.2)$$

$$x_{Im}(t) = Re(t)\cos(2\pi ft)\,2\,\text{sen}(2\pi ft) + Im(t) + Im(t)\,\text{sen}(4\pi ft) \quad (3.3)$$

Después cada señal pasa por un filtro pasa bajas, el filtro cumple dos funciones: atenúa las frecuencias altas lo que nos permite recuperar la componente real e imaginaria y elimina parte del ruido en la señal transmitida. Con las funciones de MATLAB se diseñó un filtro Butteworth pasa bajas con las siguientes características:

Frecuencia de paso: $W_p = 1\ MHz$
Frecuencia de rechazo: $W_s = 1.5\ MHz$
Atenuación en la banda de paso: $R_p = 1\ dB$
Atenuación en la banda de paso: $R_s = 3\ dB$

Con lo que en la simulación se obtuvo un filtro digital con las características de un analógico, con una frecuencia de corte $f_c = 1.25\ MHZ$. En la Figura 3.20 se tiene la respuesta en magnitud del filtro pasa-bajas y en la Figura 3.21 las señales después del filtrado en color azul, y en rojo las señales que se tenían en el transmisor.

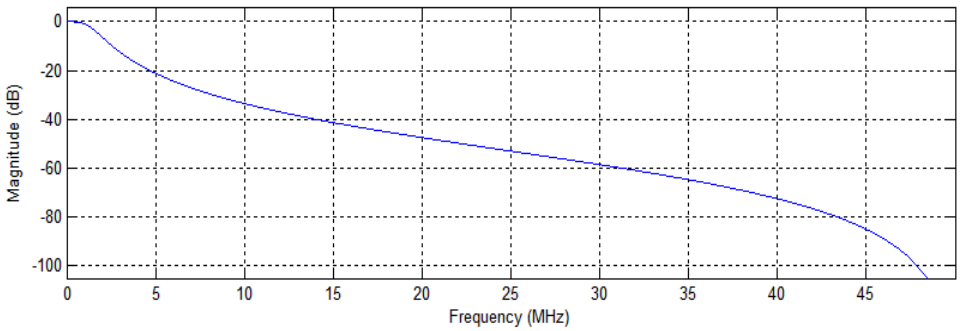


Figura 3.20: Respuesta en magnitud del filtro pasa-bajas.

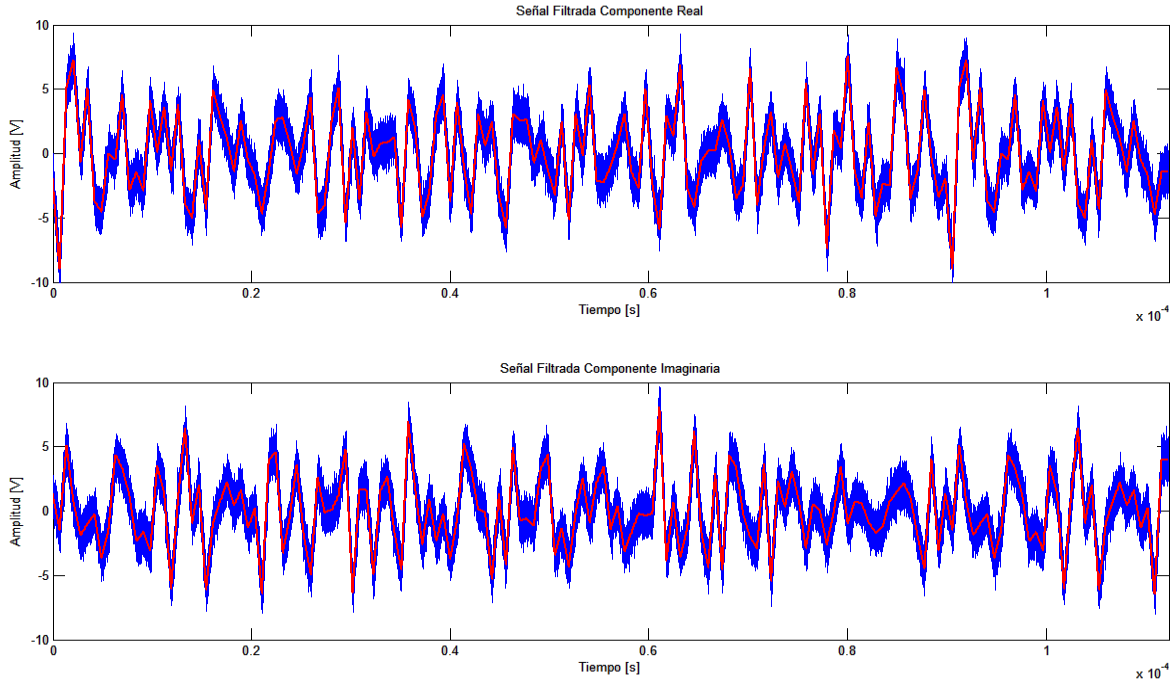


Figura 3.21: Señales obtenidas después del filtro pasa-bajas.

3.4.2 Conversión analógica a digital

Después del filtrado en el diagrama del receptor OFDMA, vienen los convertidores analógico-digital. En la simulación se tomaron 160 muestras de las amplitudes de cada señal filtrada, en la implementación se necesita que cada convertidor muestree las señales con un tiempo $T_m = 7.0225 \times 10^{-7}\ s$.

En la Figura 3.22 se ilustran los resultados en MATLAB del proceso de muestreo, en donde se graficaron en azul las muestras obtenidas en el receptor y para comparación en rojo las muestras que se tenían en el transmisor.

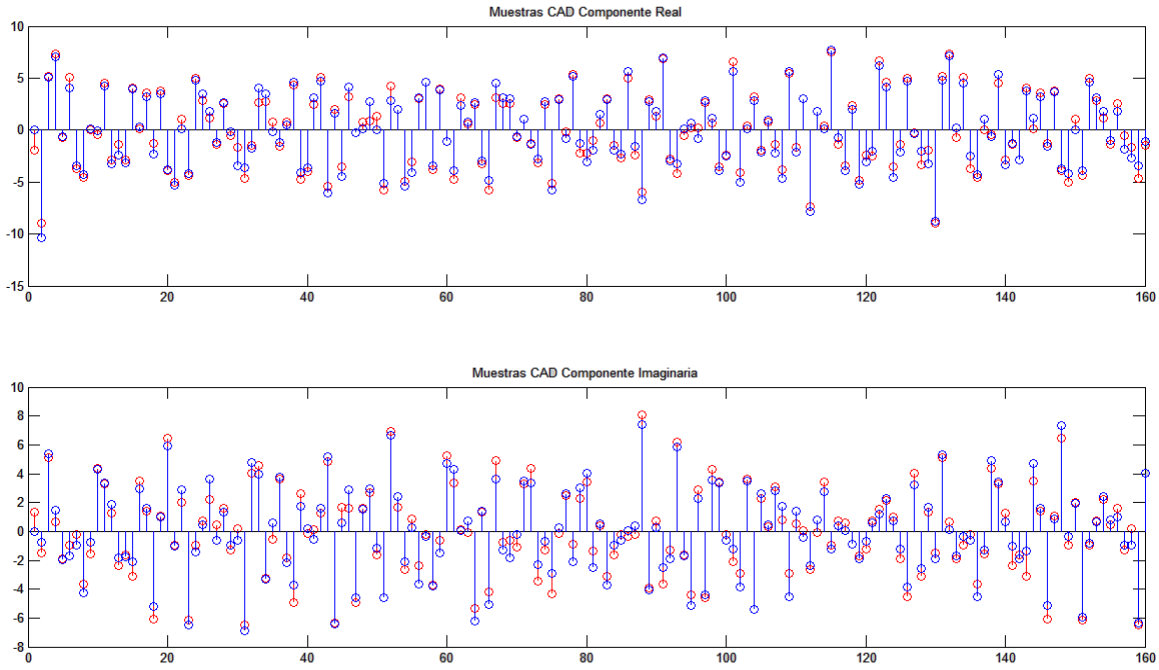


Figura 3.22: Resultados del proceso de muestreo en MATLAB.

3.4.3 Supresión del CP y cálculo de la FFT

Después de la conversión analógica-digital, los símbolos en banda base entran al convertidor serie-paralelo. El prefijo cíclico es removido, se eliminan las primeras 32 muestras correspondientes al prefijo cíclico, y los K símbolos restantes entran en la función FFT de MATLAB. La función FFT transforma los K símbolos del dominio del tiempo al dominio de la frecuencia.

3.4.4 Ecualización

Otra de las ventajas que presenta OFDM, es su sencilla ecualización, un sistema de comunicación lineal puede ser modelado por el diagrama mostrado en la Figura 3.23 [7].

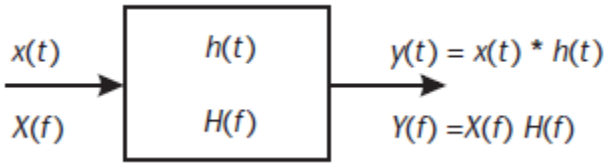


Figura 3.23: El efecto del canal h(t) cambia la señal de entrada x(t) en la señal de salida y(t).

En el trasmisor se parte que el símbolo OFDM se encuentra en el dominio de la frecuencia, después se obtiene la IFFT para generar el símbolo OFDM en el dominio del tiempo, que es transmitido a través de la interfaz del aire, por último en el receptor se calcula la FFT regresando al dominio de la frecuencia, entonces para la estimación del canal en el dominio de la frecuencia tenemos [7]:

$$H(f) = \frac{Y(f)}{X(f)} \tag{3.4}$$

En la simulación para la estimación del canal en el receptor se ocupó la expresión anterior, donde $X(f)$ son los datos de las subportadoras piloto, que también son conocidos por el receptor, $Y(f)$ son los pilotos recibidos, entonces $H(f)$ se obtuvo como:

$$H(f) = H_k = \frac{kPiloto\ recibido}{kPiloto\ conocido} \tag{3.5}$$

Entonces como tenemos una estimación del canal H_k, podemos reducir los efectos del canal en cada símbolo transmitido de la siguiente manera:

$$X(f)_{Tx} = \frac{Y(f)_{Rx}}{H_k} = \ Y(f)_{Rx} \frac{kPiloto\ conocido}{kPiloto\ recibido} \tag{3.6}$$

En la simulación se estimo H_k para los ocho pilotos, entonces el ecualizador (en cada ruta) reduce los efectos del canal de cada símbolo de datos recibidos (corrige los símbolos de datos). Cada piloto se utiliza solo para corregir a un número asignado de subportadoras de datos, que son las más cercanas a la subportadora piloto, en la Figura 3.24 se describe lo anterior.

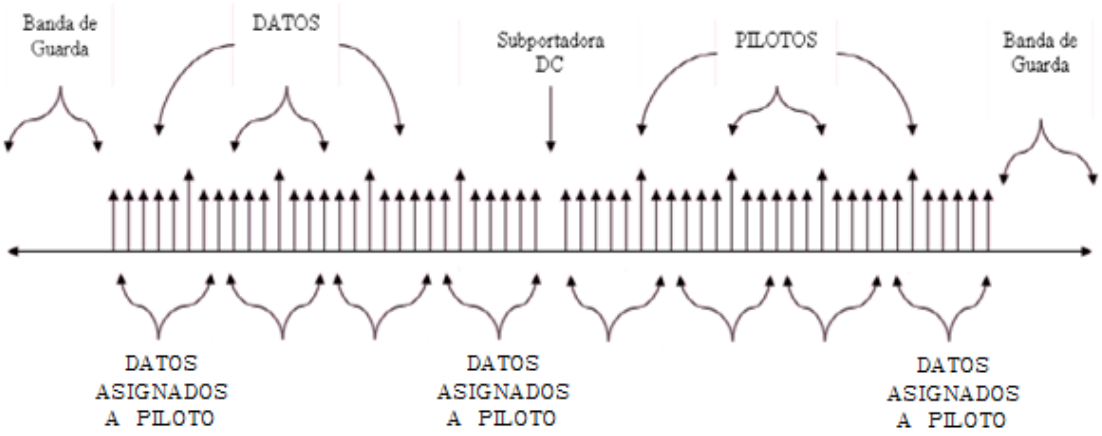


Figura 3.24: Subportadoras de datos asignadas a cada piloto.

Después de que los pilotos son leídos y usados para la estimación de canal, tanto los símbolos piloto y los símbolos nulos son eliminados. Lo que queda son los símbolos de datos corregidos que son dirigidos al desmapeador.

3.4.5 Desmapeador de símbolos

En la Figura 3.25 se muestran un gráfico en MATLAB de los símbolos de datos obtenidos en el receptor después de una transmisión con ruido, el desmapeo de símbolos consiste en asignar al dato complejo recibido una pareja de datos binarios, dependiendo de la región en la que se encuentre el dato recibido.

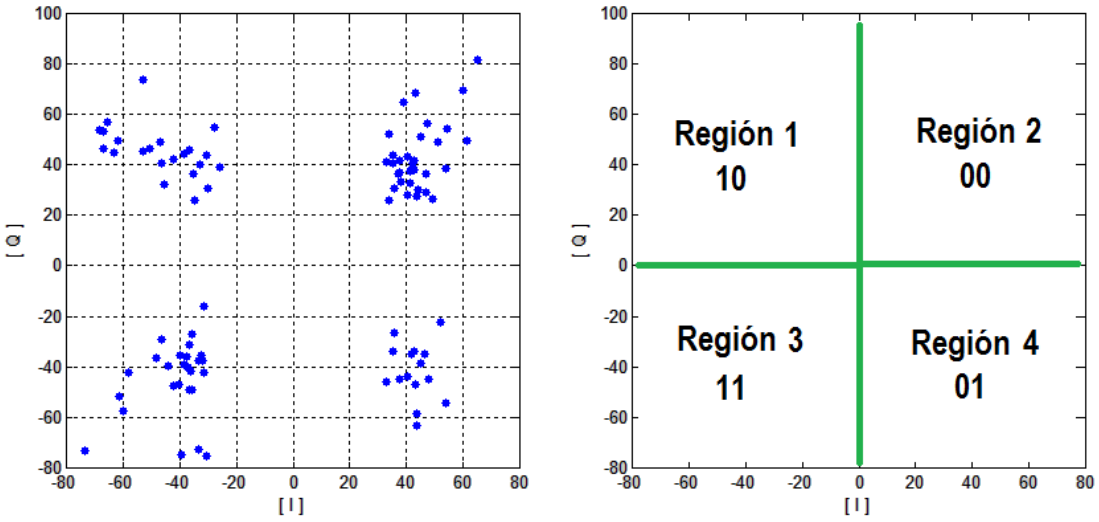


Figura 3.25: Descripción del demapeo de símbolos.

En la Figura 3.26 se presenta el resultado de la simulación correspondiente al desmapeo de símbolos, a la salida de este bloque se obtiene nuevamente una secuencia de bits. En la Figura 3.27 se muestra el diagrama de flujo empleado para implementar el programa en MATLAB.

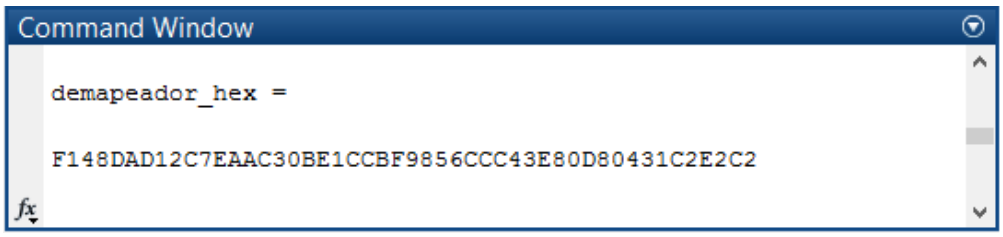


Figura 3.26: Resultado del demapeo de símbolos en MATLAB.

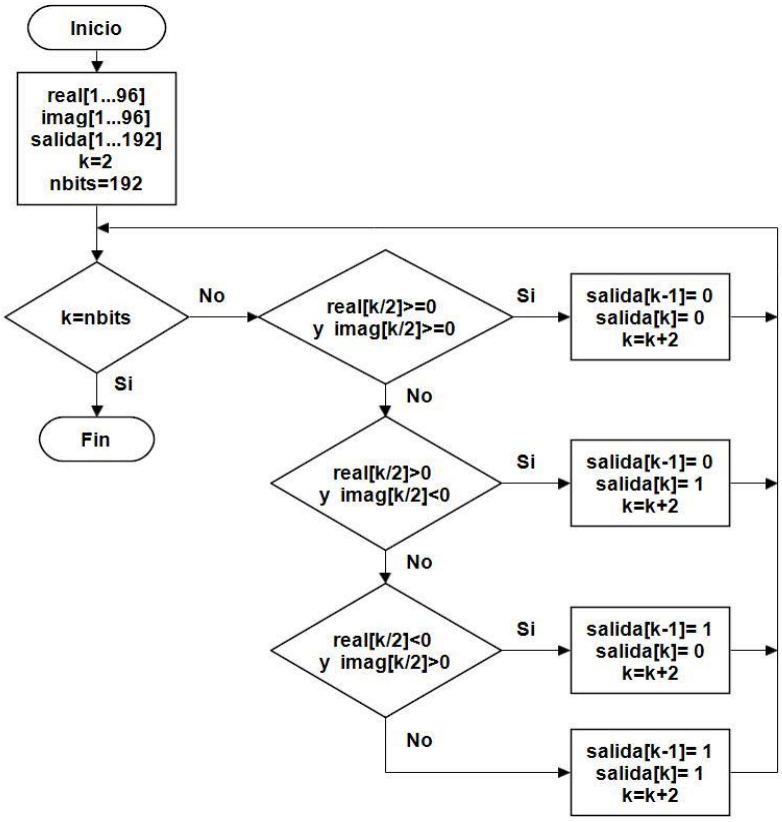


Figura 3.27: Diagrama de flujo para el desmapeo de símbolos.

3.4.6 Desentrelazador

El objetivo de este bloque es reordenar los bits recibidos de tal forma que tomen la misma posición que tenían antes de ser entrelazados, con el fin de que si los datos de alguna subportadora son erróneos, los errores se dispersen (se evita que sean consecutivos) y puedan ser corregidos por el decodificador. En la Figura 3.28 se tiene el resultado en MATLAB del desentrelazado y en la Figura 3.29 se muestra el diagrama que describe el código ocupado en la simulación.

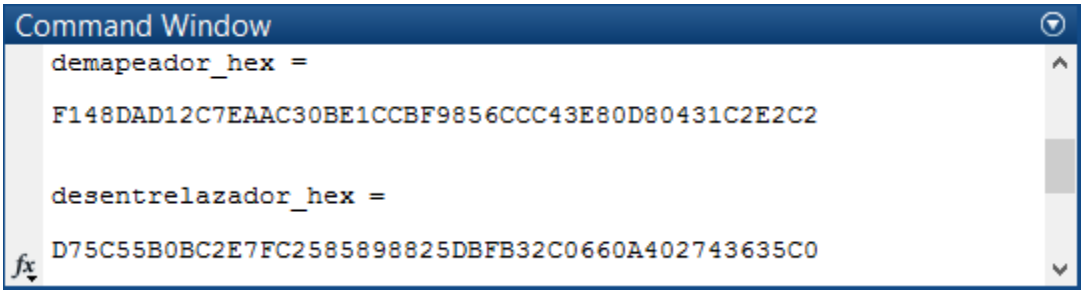


Figura 3.28: Resultado del desentrelazador en MATLAB.

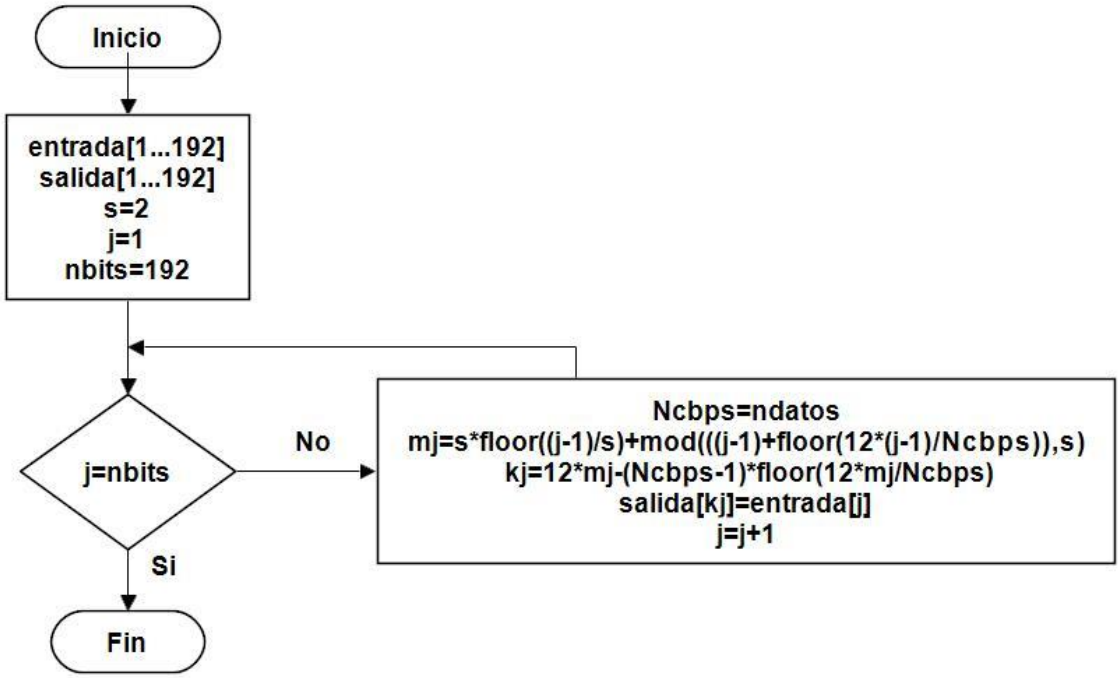


Figura 3.29: Diagrama para reordenar los bits recibidos.

3.4.7 Decodificador

En el receptor el decodificador se encarga de la búsqueda de errores y la corrección de estos. La implementación de este bloque está basada en el algoritmo de Viterbi, en el cual se hace uso de la distancia de Hamming. Esta se obtiene de la comparación entre los bits recibidos y los esperados. Si los bits recibidos son iguales a los esperados se tiene una distancia de Hamming igual a cero, y si los bits no coinciden se tiene una distancia igual al número de bits diferentes. El algoritmo debe comparar los costos de las diferentes combinaciones posibles de los datos recibidos y decidir cuál tiene la mayor probabilidad de haber sucedido. En el codificador empleado se tienen 64 estados, que son todas las combinaciones posibles de los valores que pueden tener los flip-flops, además se inicia la decodificación considerando que todos los flip-flops comienzan con un valor de cero, y que siguiendo el algoritmo solo se tienen dos estados posibles. En la Figura 3.30 se muestra el diagrama que describe el algoritmo para la decodificación desarrollado y en la Figura 3.31el resultado en MATLAB.

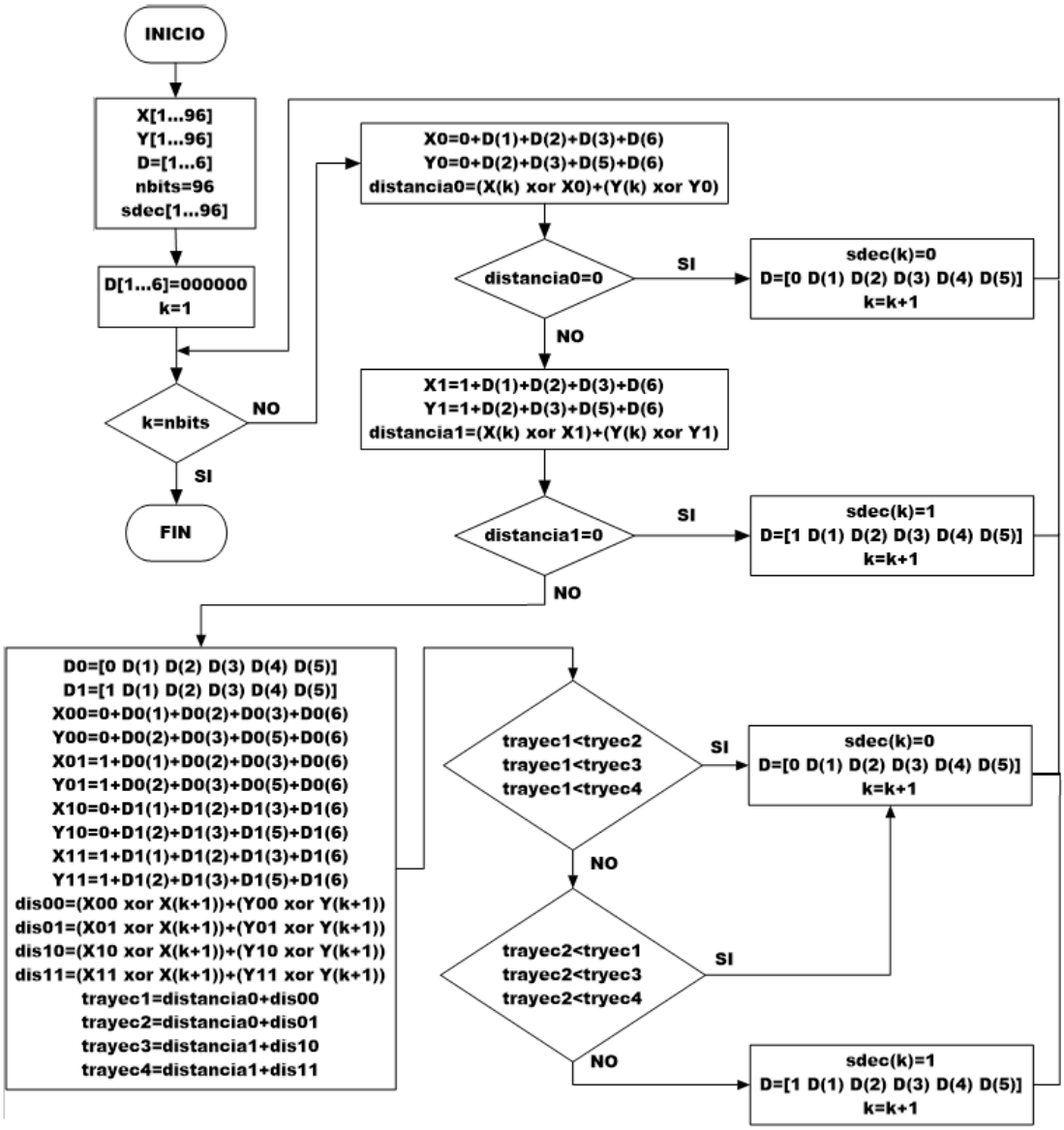


Figura 3.30: Diagrama de flujo para la decodificación.

Figura 3.31: Resultado de la decodificación en MATLAB.

3.4.8 Desaleatorizador

El último bloque del receptor es el desaleatorizador en donde se tiene los datos recibidos finales que corresponden a la entrada del transmisor. Este bloque es igual que el aleatorizador, pero para poder recuperar los datos, el estado inicial de los registros debe ser igual que el del receptor al comienzo de la aleatorización. A continuación se muestra el resultado que se obtuvo de este bloque en MATLAB, donde a la salida del desaleatorizador se recupera la secuencia de bits que se introdujeron en el transmisor (ver Figura 3.32).

Figura 3.32: Bits obtenidos a la salida del receptor.

3.5 Conclusión del capitulo

En este capítulo se establecieron los parámetros del sistema OFDMA, de acuerdo con el estándar IEEE 802.16, que permitieron el diseño de un proceso de transmisión y recepción utilizando la técnica OFDM a nivel simulación. El desarrollo de esta simulación permitió la compresión de las etapas y de los bloques que conforman el transmisor y receptor OFDMA. Además nos permitió evaluar los algoritmos propuestos y visualizar la respuesta e información esperada en cada etapa del sistema, que fue de gran ayuda para implementar el sistema en plataforma FPGA.

En el siguiente capítulo se presentara la implementación del sistema en plataforma FPGA, donde se tomaron como base los algoritmos propuestos, diagramas de flujo que describen los procesos y resultados expuestos en este capítulo.

Capítulo 4

Implementación del sistema OFDMA en plataforma FPGA

En el capítulo anterior se presentó el desarrollo del sistema OFDMA a nivel simulación mediante el software MATLAB, en esta parte se tiene un primer diseño de los bloques que conforman al sistema, lo siguiente es llevar los programas realizados en MATLAB a nivel hardware. En este capítulo se presenta la implementación del sistema en plataforma FPGA (Field Programmable Gate Array), los bloques que conforman al sistema fueron desarrollados utilizando el software QUARTUS II, producido por Altera. El software QUARTUS II ofrece un entorno completo que se adapta fácilmente a los requisitos del diseño.

4.1 Plataforma de desarrollo

Un proyecto en hardware de un sistema experimental de comunicaciones de este tipo requiere una plataforma de desarrollo que contenga las facilidades que permitan:

- Alcanzar los requisitos exigidos por el diseño, en velocidad y tamaño. Un dato importante obtenido de la simulación en MATLAB es que el procesamiento de la información en el receptor debe ser menor a $112.36\ \mu s$.
- Diseñar las partes del sistema y comprobar su funcionalidad en forma gradual.
- Interconectar los subsistemas como "cajas negras", que impulsen un diseño jerárquico.
- Realizar simulaciones y obtener datos que se asemejen a su comportamiento en el dispositivo físico real, tanto en tiempo como en precisión numérica.
- Implementar el sistema en una plataforma física real y comprobar su funcionalidad.
- Documentar el desarrollo para futuras referencias.

Debido a que se cuenta con modelos orientados a este tipo de proyecto, con el software apropiado y además con kits de desarrollo que facilitan la puesta en marcha de prototipos, el tipo de plataforma elegida fue el FPGA.

4.1.1 Tecnología FPGA

Un Arreglo de Compuertas Lógicas Programables (FPGA, Field Programmable Gate Array) es un dispositivo semiconductor que puede ser programado. Un FPGA puede ser utilizado para implementar cualquier función lógica que un circuito de aplicación específica podría realizar, pero con la capacidad de actualizar su funcionalidad para ofrecer ventajas sobre diversas aplicaciones.

4.1.2 AHDL

El lenguaje de descripción de hardware de Altera (AHDL, Altera Hardware Description Language) es un lenguaje de alto nivel modular que está completamente integrado en el sistema QUARTUS. AHDL es una poderosa herramienta para describir arquitecturas digitales. Su principal aplicación es la descripción de hardware, cada parte puede operar en tiempo real de forma paralela. Aunque la sintaxis es similar al lenguaje C, un archivo AHDL describe operaciones simultáneas, no una secuencia de operaciones. Cada declaración define la salida de una subunidad con relación a sus entradas. Los archivos AHDL son compilados en datos de bits que definen las conexiones internas de los chips Altera.

Para realizar una descripción de hardware en QUARTUS II se deben tomar en cuenta elementos básicos para la estructura del diseño de una función.

En la Figura 4.1 se ilustra un ejemplo de descripción de hardware el cual se conforma de elementos como el nombre del diseño, declaración de entradas y salidas, declaración de variables, asignación de señales y la descripción del diseño [26].

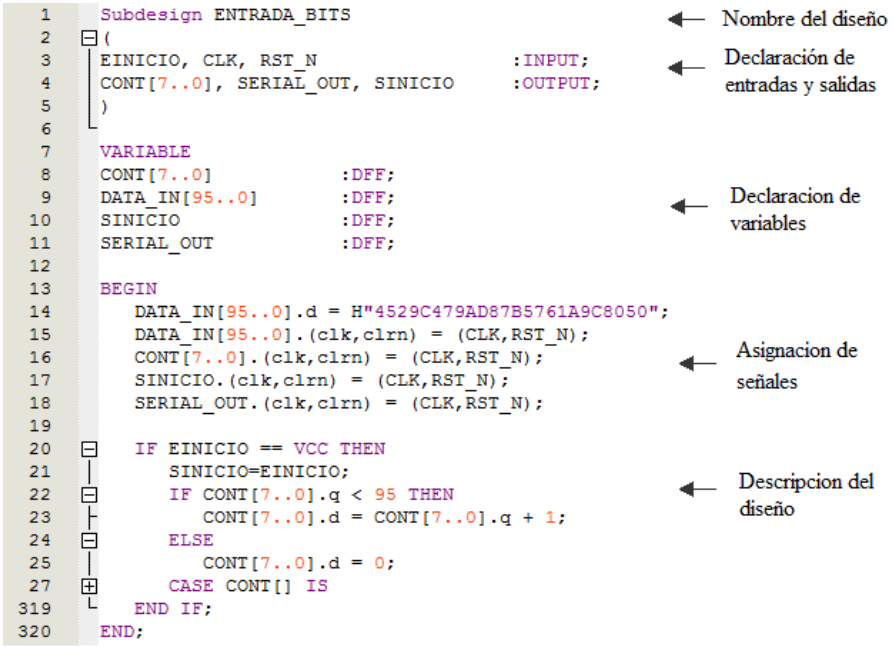


Figura 4.1: Estructura básica para descripción de hardware.

4.1.3 Cyclone V

La familia de FPGA Cyclone V ofrecen una combinación de alta funcionalidad, baja potencia y costo. Ofrecen soluciones ideales por su gran volumen, baja potencia de consumo y aplicaciones sensibles al costo [27].

En el caso particular de esta tesis, para la parte de implementación, se utilizaron dos tarjetas DEO-Nano-SoC que cuentan con el dispositivo FPGA Cyclone V 5CSEMA4U23C6N de Altera (ver Figura 4.2). Las características con las que cuenta el dispositivo FPGA 5CSEMA4U23C6N son las siguientes [28]:

- Dual-core ARM Cortex-A9 (HPS).
- 40K Elementos Lógicos Programables.
- 2,460 Kbits de Memoria Integrada.
- 5 PLLs.

Adicionalmente, este diseño de tarjeta permite un total de 94 entradas o salidas y tres fuentes de reloj de 50 MHz.

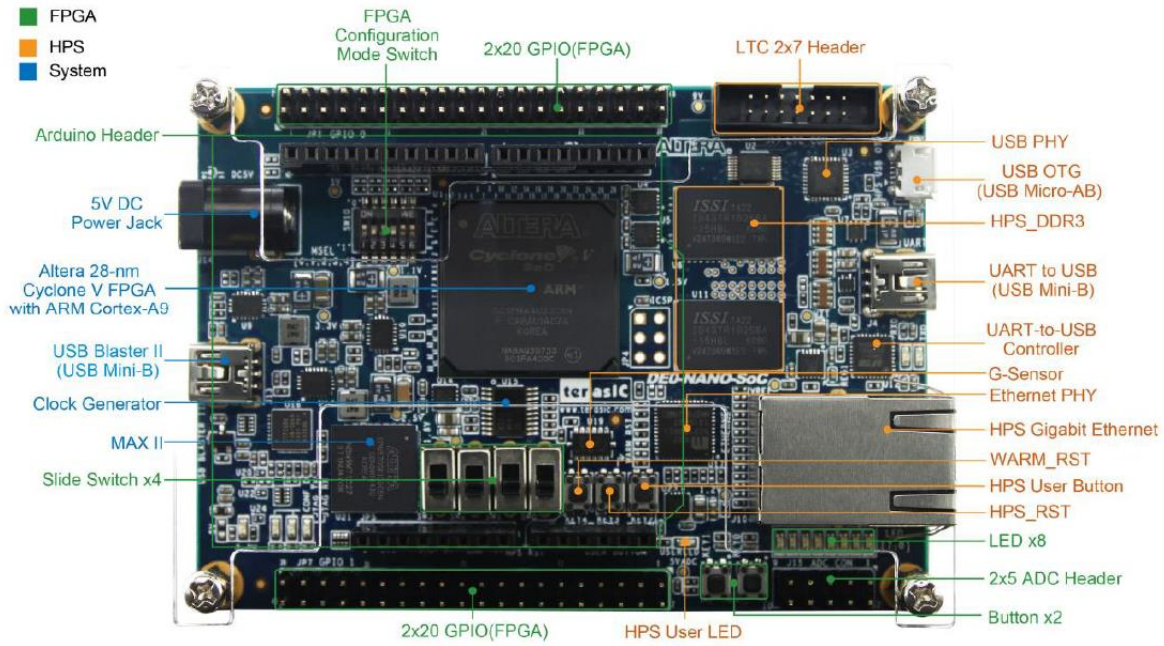


Figura 4.2: Tarjeta DEO-Nano-SoC con FPGA 5CSEMA4U23C6N.

4.2 Diseño del Transmisor y Receptor

En la Figura 4.3 se ilustra el diagrama general que describe la implementación del transmisor y receptor OFDMA. Los bloques que conforman al transmisor son: Aleatorizador, Codificador, Entrelazador, Mapeador, Inserción de Pilotos y Nulos, IFFT, Adición del CP, Almacenamiento de Datos y Señales de reloj. El receptor está conformado por los siguientes bloques: Supresión del CP, Almacenamiento de Muestras, FFT, Supresión de Pilotos y Nulos, Desmapeador, Desentrelazador, Decodificador, Aleatorizador y Señales de reloj.

La forma para verificar el correcto funcionamiento de los bloques que conforman al sistema se realizó mediante la comparación con los resultados obtenidos en la simulación en MATLAB, que está basada en el estándar IEEE 802.16. También para verificar el correcto funcionamiento de cada bloque del receptor se emplearon los bloques correspondientes del transmisor, donde al bloque del transmisor se introducía una secuencia de datos o bits y su salida se conectaba a la entrada de su contraparte en el receptor, a la salida del bloque del receptor se recuperaba la secuencia de datos introducida en el bloque del transmisor.

Después esta metodología para comprobar los bloques del receptor se siguió empleando pero agregando cada vez más bloques a la prueba, hasta realizar la verificación con todos los bloques que conforman al sistema. En la siguiente parte se describirán los bloques desarrollados que conforman al sistema OFDMA.

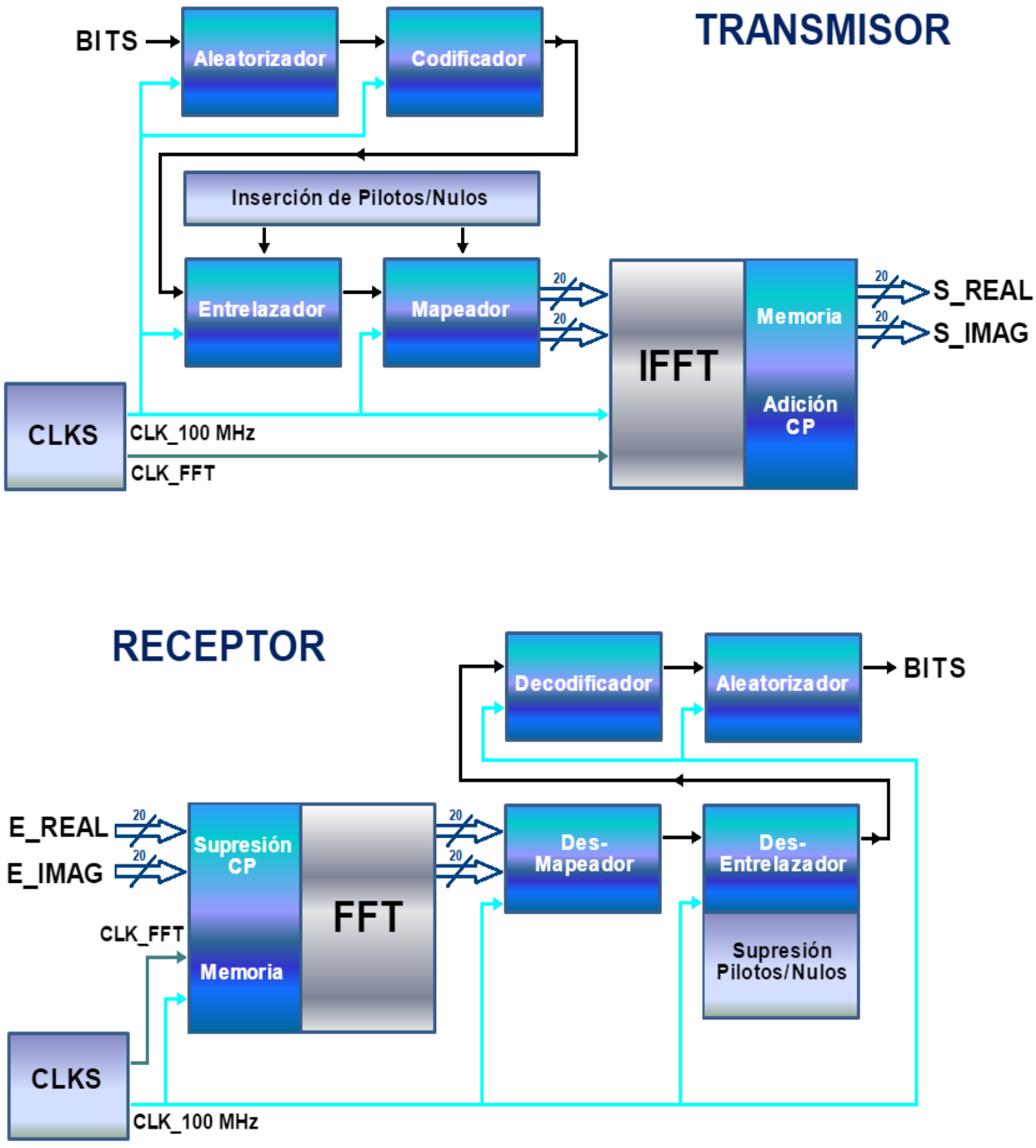


Figura 4.3: Diagrama a bloques del diseño del transmisor y receptor.

4.2.1 Señales de reloj

En la figura 4.4 se ilustran los bloques empleados para generar las señales de reloj. El primer bloque corresponde a una megafunción proporcionada por altera llamada ALTPLL, con este bloque generamos una señal de reloj de 100 MHz a partir de la fuente de reloj de 50 MHz que proporciona la tarjeta Deo-Nano-SoC. La señal de reloj de 100 MHz nos permite procesar la información con mayor velocidad, además esta frecuencia de reloj fue empleada en el diseño para el cálculo de la FFT, y es necesaria para cumplir con los requisitos del sistema, de esta señal depende el tiempo que tardara el bloque de la FFT en realizar el cálculo. En el transmisor después de realizar el cálculo de la IFFT se necesita enviar el resultado en serie a los convertidores digital-analógico con una duración T_m, y en el receptor es necesario tener muestras de las señales recibidas con ese mismo tiempo de muestreo (T_m), el segundo bloque fue desarrollado para generar la señal de reloj que nos permitiera cumplir con lo anterior. El segundo bloque es un contador de pulsos, en el cual cuando se llega a cierto conteo de pulsos la cuenta se reinicia, este conteo nos permite generar la señal CLK_FFT la cual permanece en cero durante la mitad de la cuenta y después cambia a un nivel en alto durante el resto de la cuenta.

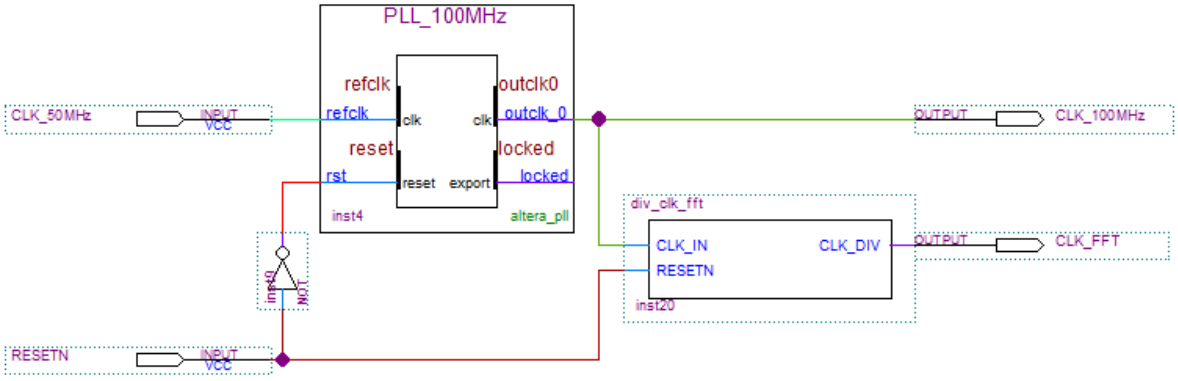


Figura 4.4: Bloques para generar las señales de reloj.

4.2.2 Aleatorizador

En la Figura 4.5 se ilustra la estructura del aleatorizador exigido en el estándar, este se compone de 15 registros de desplazamiento y 2 compuertas XOR, su implementación en lenguaje AHDL resulta más sencilla porque estos elementos están incluidos en la librería de QUARTUS II, los registros empleados para su implementación son flip-flop del tipo D, donde al inicio del proceso de la aleatorización los registros son inicializados con el valor

$FFD[14..0] = B"000111011110001"$. Para este bloque en particular su contraparte en el receptor es el mismo, donde el valor inicial de los registros debe ser el mismo que en el transmisor para poder recuperar la secuencia original.

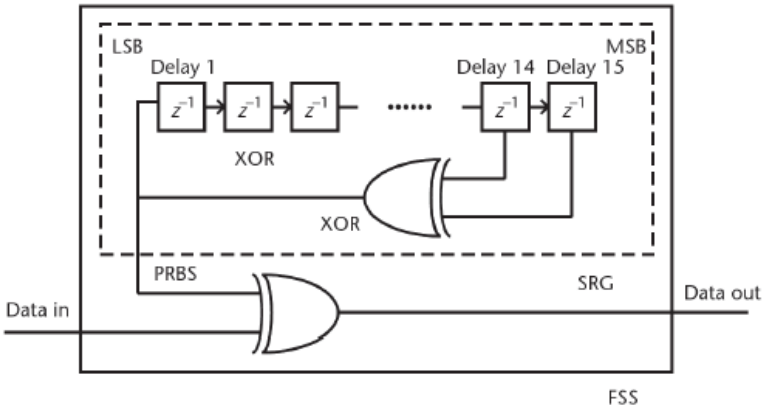


Figura 4.5: Diagrama aleatorizador.

En la Figura 4.6 se muestra el bloque desarrollado para la aleatorización de bits, donde el bloque consta de cuatro entradas y dos salidas, las cuales se describen a continuación:

RAN_IN	Recibe los bits en forma serial para su posterior aleatorización.
$EINICIO$	La función de esta entrada es dar inicio al proceso de la aleatorización.
CLK	Entrada para la señal de reloj.
RST_N	Esta entrada sirve de reset general, además permite reinicializar los registros con el valor requerido al inicio de la aleatorización.
RAN_OUT	Salida correspondiente a la señal aleatorizada.
$SINICIO$	Señal de habilitación para procesos posteriores al bloque.

Figura 4.6: Bloque desarrollado para la aleatorización.

4.2.3 Codificador/Decodificador

En la figura 4.7 se presenta el codificador empleado en el transmisor, la base de este bloque son 6 registros de desplazamiento y dos sumas, para su implementación al no necesitar el acarreo de las sumas, estas se pueden sustituir con operaciones XOR en cascada o anidadas.

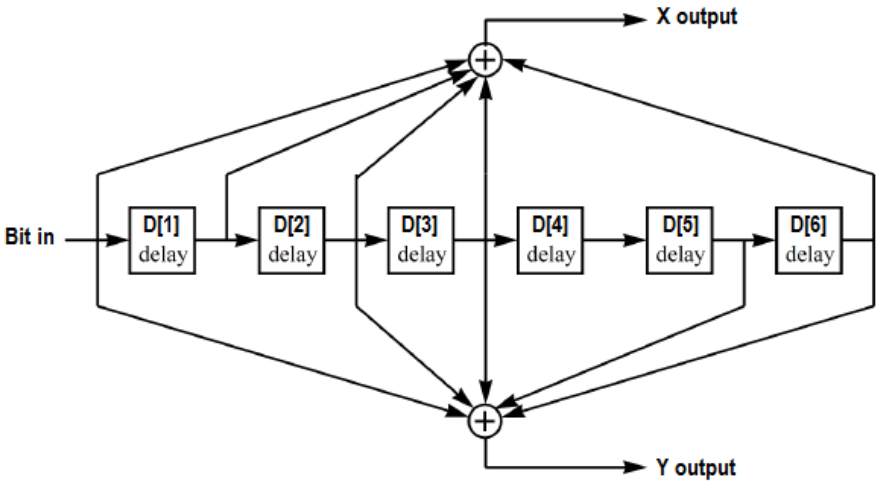


Figura 4.7: Esquema del Codificador en el transmisor.

Al inicio de la decodificación los registros se tienen que reinicializar a cero. Por cada bit de entrada al codificador se tienen dos bits de salida, donde las salidas X y Y son obtenidas de las siguientes expresiones:

$$X = Bit\_in + D[1] + D[2] + D[3] + D[6] \quad (4.1)$$

$$Y = Bit\_in + D[2] + D[3] + D[5] + D[6] \quad (4.2)$$

Para su diseño en hardware se utilizó como base la Figura 4.7, los registros empleados para su implementación son flip-flop del tipo D y para realizar las sumas compuertas XOR. En la Figura 4.8 se ilustra el bloque para la codificación.

Figura 4.8: Bloque para codificación.

Las entradas y salidas del decodificador se definen de la siguiente manera:

BIT_IN	Entrada para los bits a codificar.
$EINICIO$	La función de esta entrada es dar inicio al proceso de la codificación.
CLK	Entrada para la señal de reloj.
RST_N	Esta entrada sirve de reset general, además permite reinicializar a cero el valor de los registros para cada nueva codificación.
SX	Salida codificada X.
SY	Salida codificada Y.
$SINICIO$	Señal de habilitación para procesos posteriores al bloque.

Para el desarrollo en AHDL del decodificador se usó como base el algoritmo de Viterbi implementado en el software de MATLAB. En el receptor se parte de que en el codificador las salidas dependen tanto de la entrada actual como de los bits previos, y puesto que el codificador está conformado por seis registros, se pueden tener 64 combinaciones posibles en los valores de los registros, para la implementación del decodificador se tomaron los 64 casos y dependiendo de las entradas anteriores se decide el valor que pudo ser enviado. Para este bloque en el receptor en el FPGA ya no se realizará el cálculo de las distancias de Hamming, estas se obtuvieron directamente del programa realizado en MATLAB. Previo a la implementación se realizó un análisis de cada uno de los 64 estados, y dependiendo de los cuatros bits previos se seleccionaba el valor del bit que pudo ser enviado. Después de tener este análisis resultó más sencillo implementar el bloque, en AHDL se formaron los 64 estados con sentencia $CASE$, y en cada estado se tenían 16 combinaciones posibles de los 4 bits previos, implementadas con sentencias $IF - ELSIF$, en cada combinación se especificaba el estado siguiente. La ventaja de esta propuesta es que se redujo la complejidad de la implementación y la decodificación es más rápida ya que no se necesita hacer demasiado procesamiento.

En la Figura 4.9 se muestran los bloques desarrollados para la decodificación, el primero permite tener el valor de 4 bits anteriores en el mismo ciclo de reloj, necesarios para la decodificación, el segundo es el correspondiente al decodificador.

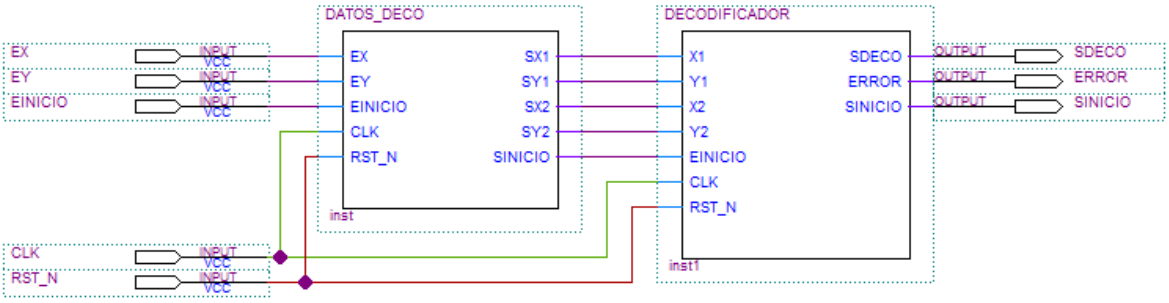


Figura 4.9: Bloques desarrollados para la decodificación.

Las entradas y salidas involucradas en el proceso de la decodificación se describen a continuación:

EX	Recibe los bits a decodificar correspondientes a la señal X del codificador.
EY	Recibe los bits a decodificar correspondientes a la señal Y del codificador.
$EINICIO$	La función de esta entrada es dar inicio al proceso de la decodificación.
CLK	Entrada para la señal de reloj.
RST_N	Esta entrada sirve de reset general, además permite llevar el estado actual del bloque al estado cero para iniciar una nueva decodificación.
$SDECO$	Salida correspondiente a la decodificación.
$ERROR$	Indica cuando los valores recibidos no concuerdan con los esperados.
$SINICIO$	Señal de habilitación para procesos posteriores al bloque.

4.2.4 Entrelazador/Desentrelazador

Para el desarrollo en lenguaje AHDL del proceso de entrelazado y desentrelazado, fue necesario el apoyo de los programas realizados en MATLAB. El cálculo de las nuevas posiciones se realizaron con esta herramienta empleando las expresiones tomadas del estándar IEEE 802.16, evitando que se tuvieran que realizar los cálculos en el FPGA. Para la implementación de cada bloque fueron necesarios 192 flip-flops tipo D, estos son empleados para guardar el valor de toda la secuencia de bits para después ser mostrada en forma serial en el orden calculado. En la Figura 4.10 se muestran los bloques desarrollados para el proceso del entrelazado.

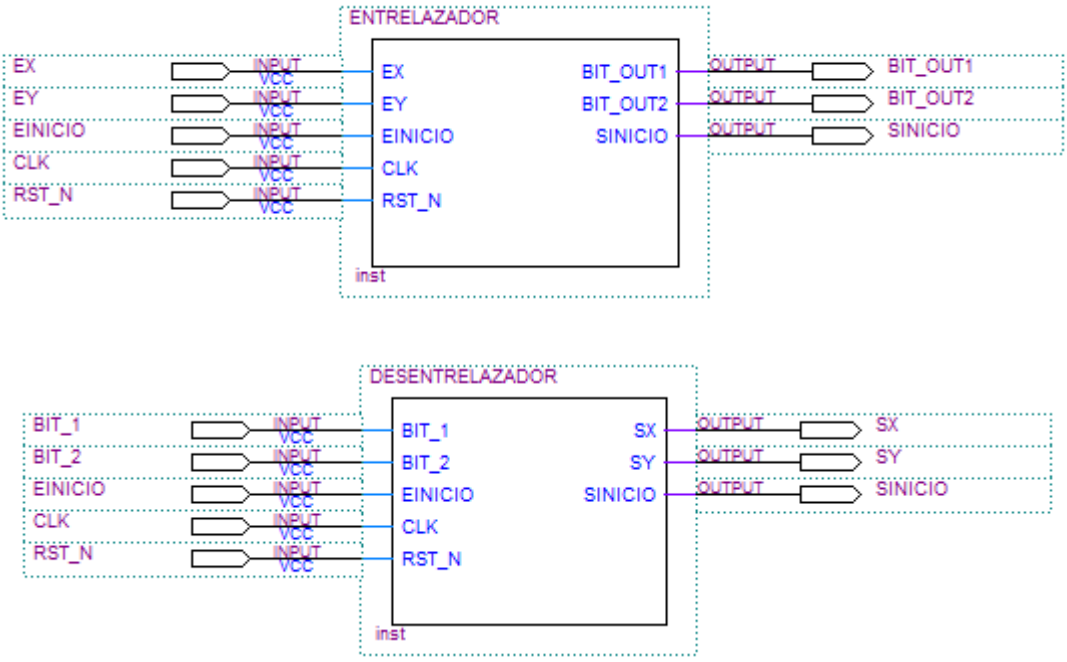


Figura 4.10 : Bloques de entrelazado y desentrelazado desarrollados en QUARTUS II.

La implementación de estos bloques es muy similar, la única diferencia es el valor para las posiciones nuevas. Las entradas y salidas empleadas en el proceso de entrelazado se describen a continuación:

EX/BIT_1	Recibe los bits a entrelazar/desentrelazar correspondientes a la señal X del codificador.
EY/BIT_2	Recibe los bits a entrelazar/desentrelazar correspondientes a la señal Y del codificador.
$EINICIO$	La función de esta entrada es dar inicio al proceso de entrelazado/desentrelazado.
CLK	Entrada para la señal de reloj.
RST_N	Esta entrada sirve de reset general.
BIT_OUT/SX	Salida entrelazada/desentrelazada correspondiente a la señal X del codificador.
BIT_OUT/SY	Salida entrelazada/desentrelazada correspondiente a la señal Y del codificador.
$SINICIO$	Señal de habilitación para procesos posteriores al bloque.

En la simulación en MATLAB las salidas X y Y correspondientes a la codificación son mezcladas para formar una sola salida, después esta es llevada al bloque de entrelazado. En la implementación de estos bloques es más conveniente tener dos salidas en lugar de una, ya que los datos procesados hasta el momento son manejados en forma serial. El tener dos señales a la entrada del entrelazador, en lugar de una, nos permite ir almacenando los datos de dos en dos, por ello el proceso requerirá la mitad de tiempo. A la salida se empleó la misma idea, es decir, los datos entrelazados son mostrados de dos en dos además que el siguiente bloque necesita que los datos entren en pares.

4.2.5 Mapeador/Desmapeador

Para desarrollar estos bloques en AHDL se consideró el formato de los números empleados en la IFFT y FFT (ver Figura 4.11), en donde se tiene que un número ya sea real o complejo se representa con una cadena de 20 bits, en donde el bit de signo es el bit más significativo y para representar un numero negativo se debe emplear el complemento a 2.

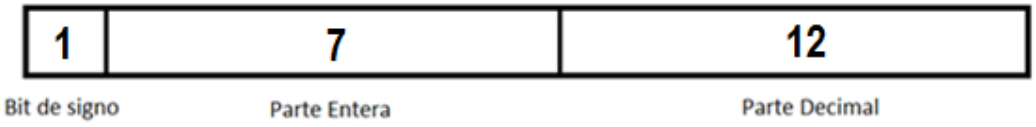


Figura 4.11: Formato de la distribución de bits de la palabra a utilizar.

Esta asignación es abstracta debido a que el sistema lo tratará como un número entero. Para el desarrollo de estos bloques se tomó como base el resultado de las simulaciones en el software MATLAB, en la Figura 4.12 se muestra el resultado de la simulación correspondiente al mapeo y desmapeo de símbolos. En el transmisor el mapeo se realiza tomando grupos de bits que son asignados a puntos de una constelación especifica. Una magnitud y fase específica representan una determinada combinación de bits. Para el mapeo de símbolos se empleó una modulación QPSK, los bits son introducidos en pares a este bloque, y a cada par de bits se le asigna un numero complejo también llamado símbolo de dato, entonces para el mapeo de símbolos el valor asignado para los números real o imaginario podía ser 42 o -42, estos valores se seleccionaron porque de ellos depende el resultado de la IFFT, con esos valores el resultado de la IFFT puede ser representado con el formato seleccionado. Para el bloque en el receptor se consideró que en la transmisión existe ruido, por lo que los valores se esperan dispersos como se observa en la Figura 4.12,

el desmapeo de símbolos consiste en asignar al dato complejo recibido una pareja de datos binarios, dependiendo de la región en la que se encuentre el dato recibido. Para la propuesta presentada para el desmapeo de símbolos, no es necesaria la ecualización, y solo es válida para una modulación QPSK. Si para posteriores trabajos se requiere emplear otro tipo modulación digital como 16QAM o 64QAM será necesario la estimación del canal y la corrección de los símbolos.

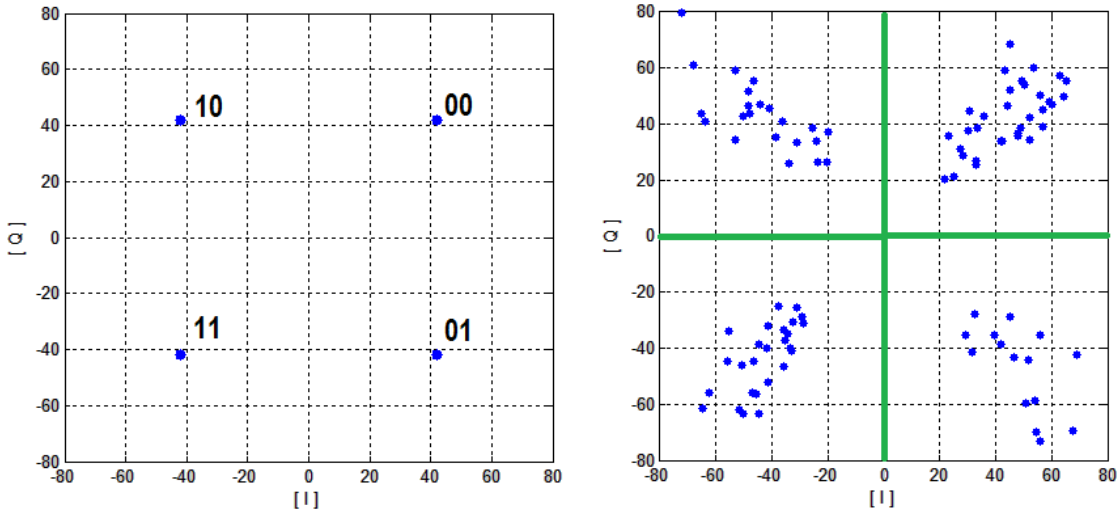


Figura 4.12: Resultado en MATLAB correspondiente al mapeo de símbolos.

En la Figura 4.13 se ilustran los bloques desarrollados para el mapeo y desmapeo de símbolos.

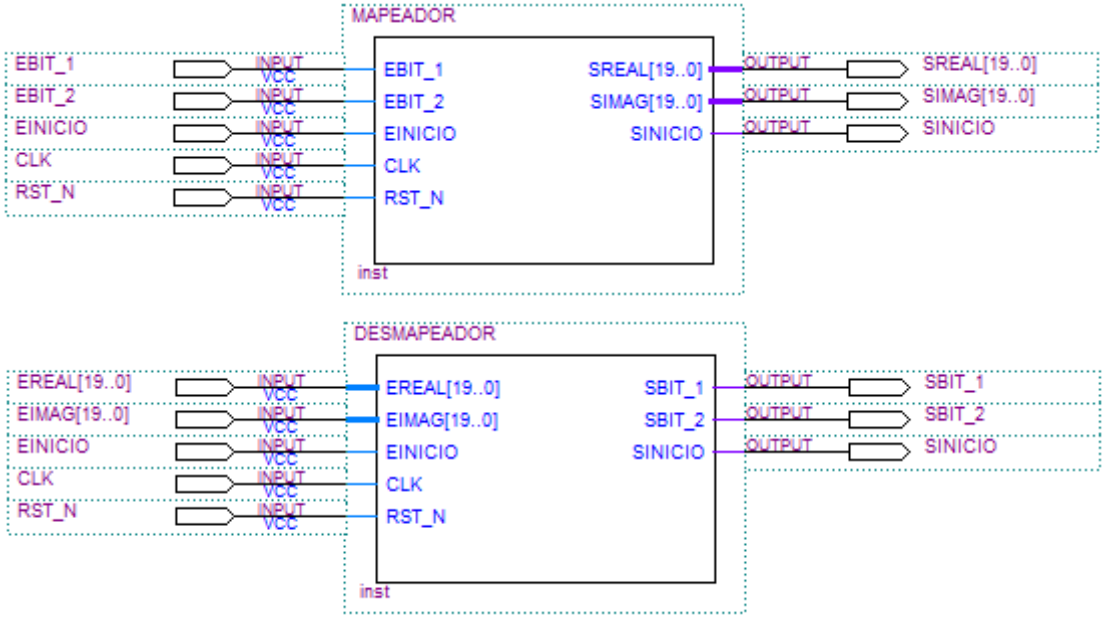


Figura 4.13: Mapeador y Desmapeador de símbolos desarrollados en QUARTUS II.

Las entradas y salidas involucradas en el mapeo y desmapeo de símbolos se describen a continuación:

$EBIT1/EREAL$	Recibe los datos a mapear/desmapear correspondientes a la señal X del codificador.
$EBIT2/EIMAG$	Recibe los datos a mapear/desmapear correspondientes a la señal Y del codificador.
$EINICIO$	La función de esta entrada es dar inicio al mapeo/desmapeo de símbolos.
CLK	Entrada para la señal de reloj.
RST_N	Esta entrada sirve de reset general.
$SREAL/SBIT1$	Salida mapeada/desmapeada correspondiente a la señal X del codificador.
$SIMAG/SBT2$	Salida mapeada/desmapeada correspondiente a la señal X del codificador.
$SINICIO$	Señal de habilitación para procesos posteriores al bloque.

4.2.6 IFFT/FFT

Los bloques correspondientes a la IFFT y FFT son los módulos centrales en el sistema OFDMA. Su función es realizar la IDFT y DFT empleando un algoritmo computacionalmente eficiente. Este algoritmo ya fue estudiado e implementado en plataforma FPGA [10]. Pero es de suma importancia comprender su implementación para poder integrar estos bloques al sistema desarrollado.

Las ecuaciones para la DFT e IDFT pueden ser reescritas de la siguiente forma [10]:

$$\mathrm{X(k)} = \sum_{\mathrm{n}=0}^{\mathrm{N}-1} \mathrm{x(n)} \mathrm{W}_{\mathrm{N}}^{\mathrm{nk}}, \qquad \mathrm{k} = 0, 1, \dots, \mathrm{N}-1 \tag{4.3}$$

$$\mathrm{x(n)} = \frac{1}{N} \sum_{\mathrm{n}=0}^{\mathrm{N}-1} \mathrm{X(k)} \mathrm{W}_{\mathrm{N}}^{-\mathrm{nk}}, \qquad \mathrm{n} = 0, 1, \dots, \mathrm{N}-1 \tag{4.4}$$

Donde N es el número de muestras de las cuales se desea obtener la DFT y W es denominado como factor de rotación y cumple con:

$$W_N^{nk} = e^{-j\frac{2\pi}{N}nk} \tag{4.5}$$

$$W_N^{-nk} = e^{j\frac{2\pi}{N}nk} \tag{4.6}$$

Las expresiones para la DFT e IDFT son muy similares, solo difieren en el signo del exponente y el factor de escalamiento $1/N$, por lo que cualquier algoritmo para DFT puede ser fácilmente adaptado para el cálculo de la transformada inversa.

La propuesta para realizar el cálculo de la DFT está basada en los algoritmos del tipo Cooley-Tukey, los cuales permiten reducir el número de sumas y de multiplicaciones necesarias para obtener la DFT gracias a algunas propiedades de los factores de rotación. La idea básica de este tipo de algoritmos radica en poder descomponer una DFT en varias DFT de menor tamaño hasta llegar a transformadas de tamaño 2 y 4 [10].

En la Figura 4.14 se describe el diagrama general desarrollado para obtener la FFT de tamaño 128, la propuesta en este bloque es descomponer la transformada de tamaño 128 en transformadas de tamaño 8 y 16, con esto se tienen que hacer 16 transformadas de tamaño 8 y 8 transformadas de tamaño 16. La ventaja de esta propuesta es que solo se emplea un bloque para la FFT de tamaño 8 y un bloque para la FFT de tamaño 16, la desventaja de esto es que se requiere de más tiempo para calcular la FFT pero que queda justificado ya que permite reducir considerablemente los elementos lógicos para su implementación. Los demás bloques que aparecen en el esquema son necesarios para realizar el cálculo de la FFT, entre estos se destacan los multiplicadores donde se incluyen los factores de rotación y los arreglos de memorias que tienen la finalidad de almacenar los datos de entrada y el resultado de las operaciones, con lo que se tiene acceso a la información cuando es requerida según el algoritmo.

El esquema para la FFT de tamaño 8 se muestra en la Figura 4.15, en este bloque se ocupa la FFT de tamaño 2 y 4 para obtener la transformada de tamaño 8.

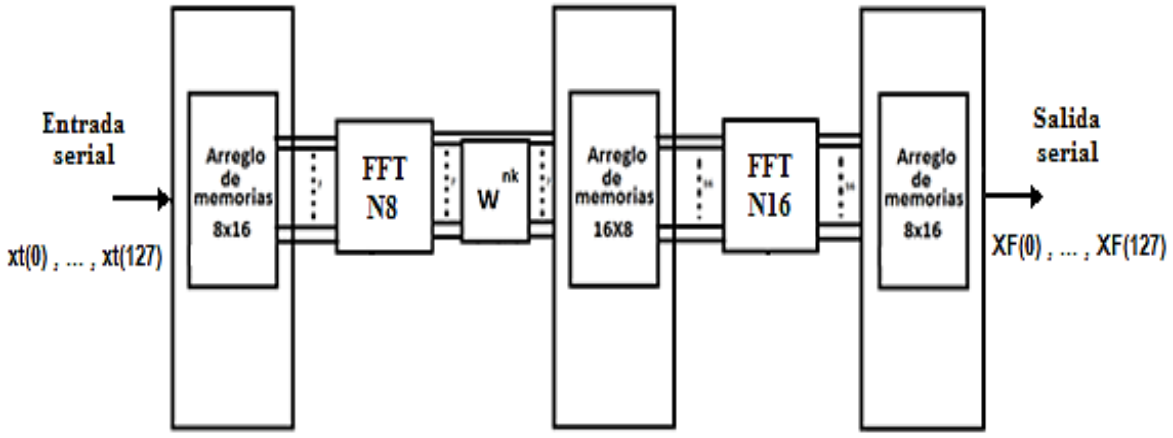


Figura 4.14: Diagrama general a bloques de la FFT de 128 muestras.

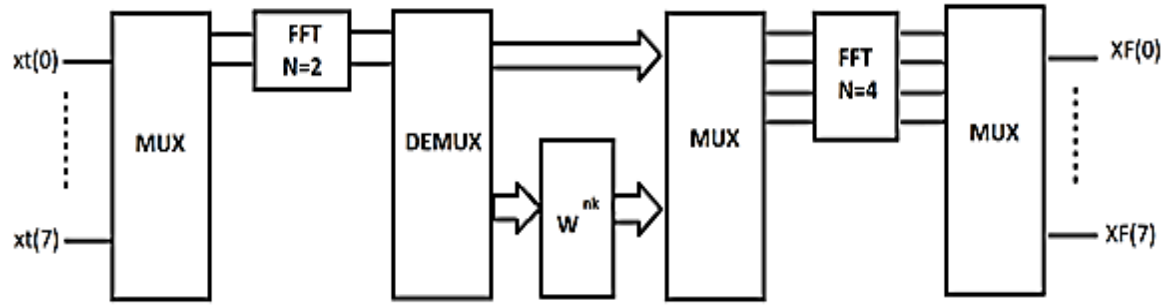


Figura 4.15: Diagrama a bloques del firmware del algoritmo FFT para N=8 base-2 y base-4.

En las Figuras 4.16 y 4.17 se presentan otras propuestas para obtener la IFFT de tamaño 128 y la IFFT de tamaño 8, el objetivo de esta mención es seguir dejando bases para trabajos posteriores.

En la propuesta original para obtener la IFFT de 128 era necesario cambiar el signo de los factores de rotación, pero se usaban los bloques de la FFT de tamaño 8 y 16 para el cálculo de la IFFT y el factor de escalamiento $1/\mathrm{N} = 1/128$ era introducido al final. Pero al integrar los bloques al sistema el resultado de la IFFT no concordaba con lo esperado, por lo que se propuso introducir el factor de escalamiento $1/8$ en la FFT de tamaño 8 y cambiar de signo los factores de rotación para así obtener la IFFT de tamaño 8 como se ilustra en la Figura 4.17, lo mismo se realizó con la FFT de tamaño 16 pero introduciendo el factor de escalamiento $1/16$. Esto permitió ir comprobando por partes el cálculo de la IFFT, donde la comprobación consistía en introducir una secuencia de datos para la FFT y después la salida

de este bloque era introducida al bloque de la IFFT, lo que se esperaba era recuperar la secuencia que se tenía a la entrada de la FFT. Los resultados para la IFFT y FFT de tamaño 8 eran los esperados, pero los resultados de la IFFT y FFT de tamaño 16 no concordaban en su mayoría, al verificar operación por operación se encontró que el formato empleado para la representación de los números no era suficiente para representar los valores de los factores de rotación y los resultados, en el formato original se empleaban 10 bits para representar la parte fraccionaria lo que permitía representar números hasta milésimas, pero los factores de rotación también contenían diezmilésimas, por lo que se cambió el formato para representar los números, en este nuevo formato se emplearon 12 bits para representar la parte fraccionaria. Con estos cambios realizados el resultado de la IFFT de tamaño 128 concordó con los esperados.

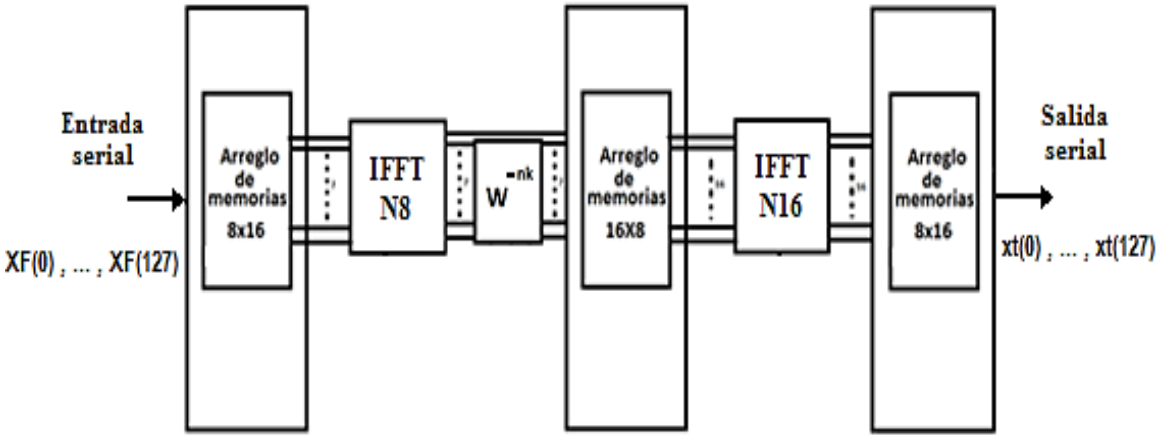


Figura 4.16: Diagrama general a bloques de la IFFT de 128 muestras.

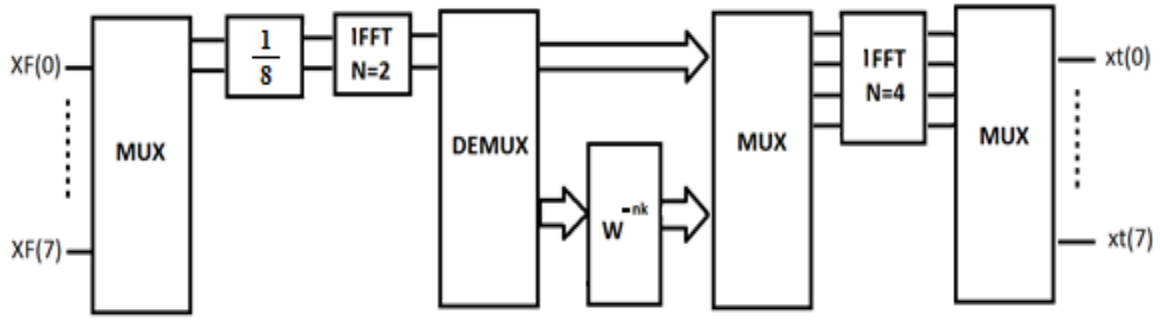


Figura 4.17: Diagrama a bloques del firmware del algoritmo IFFT para N=8 base-2 y base-4.

En la Figura 4.18 se muestra los bloques desarrollados para obtener la IFFT y FFT.

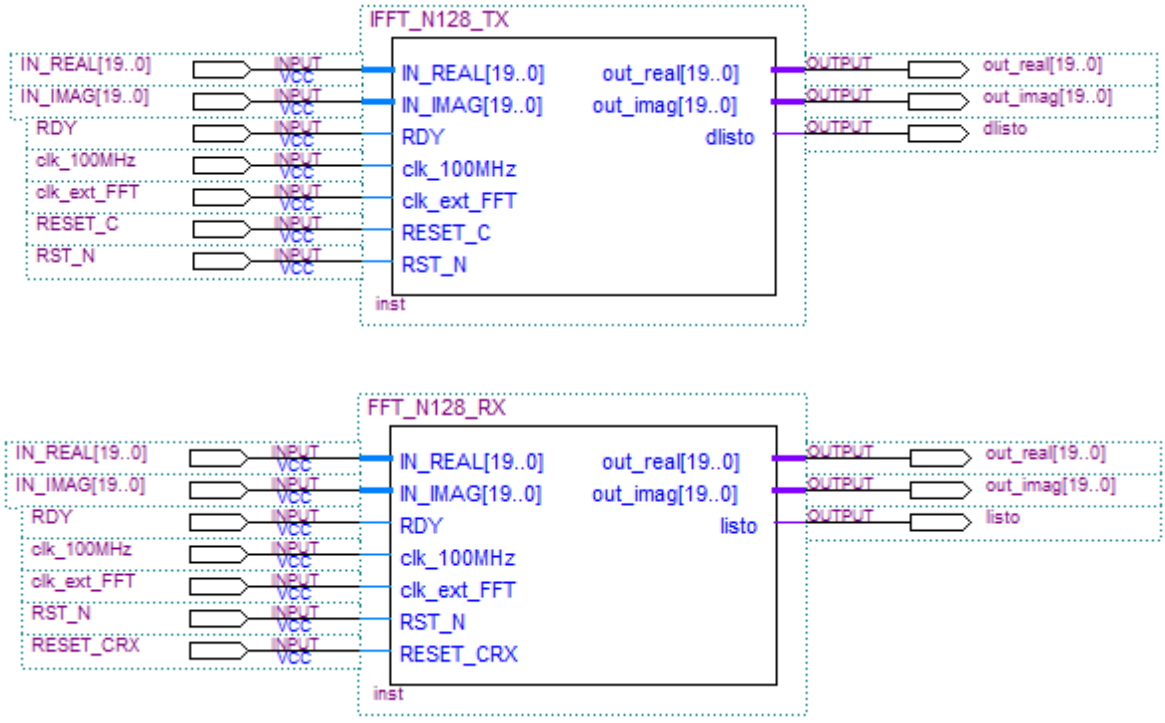


Figura 4.18: Bloques para el cálculo de la IFFT y FFT.

Las entradas y salidas en los bloques de la IFFT y FFT se describen a continuación:

IN_REAL	Recibe los 128 datos en serie (correspondientes a la parte real) a los que se les quiera aplicar la IFFT/FFT.
IN_IMAG	Recibe los 128 datos en serie (correspondientes a la parte imaginaria) a los que se les quiera aplicar la IFFT/FFT.
RDY	Esta entrada requiere un nivel en alto para almacenar los datos en IN_REAL y IN_IMAG para su posterior procesamiento.
CLK_100MHz	Entrada para la señal de reloj.
CLK_EXT_FFT	Permite leer o desplegar el resultado utilizando como referencia esta señal de reloj.
$RESET\_N$	Reset general.
$RESET\_C$	Reset integrado en el sistema OFDMA para propósitos de control, permite reiniciar bloques específicos.

OUT_REAL	Salida en serie del resultado de la IFFT/FFT correspondiente a la parte real.
OUT_IMAG	Salida en serie del resultado de la IFFT/FFT correspondiente a la parte imaginaria.
LISTO	Indica cuando el cálculo de la IFFT/FFT ha terminado, y permanece en alto mientras el resultado es leído.

4.2.7 Inserción y Supresión de Pilotos y Nulos

La inserción de pilotos y nulos se debe realizar antes del cálculo de la IFFT en el transmisor, en el esquema empleado en la simulación en MATLAB la inserción se realizaba con los datos en paralelo, y la entrada a la IFFT también era en paralelo, todos los datos eran introducidos a la vez, pero en la implementación la entrada a la IFFT se hace de forma serial por lo que la inserción se tiene que hacer de forma serial. La propuesta para realizar la inserción de Pilotos y Nulos, y poder crear la estructura del símbolo OFDM en el dominio de la frecuencia (ver Figura 4.19), se basó en el uso de los bloques ya existentes para el entrelazado y mapeo de símbolos. Una vez verificado el correcto funcionamiento de estos bloques, se realizó una modificación para poder integrar los símbolos de datos con las subportadoras nulas y piloto, en el entrelazador después de almacenar los datos estos son mostrados en un nuevo orden en serie, la modificación a este proceso fue integrar espacios o ciclos de reloj a donde se requería un piloto o nulo, después, con el mapeador de símbolos además de hacer su función, se asignaba el valor a las subportadoras piloto y nulas. La estructura del símbolo empleada es la misma que se presentó en la simulación en MATLAB. En el receptor la supresión de pilotos y nulos se realiza solo con el desentrelazador, en el cual solo se almacenan los bits correspondientes a los símbolos de datos.

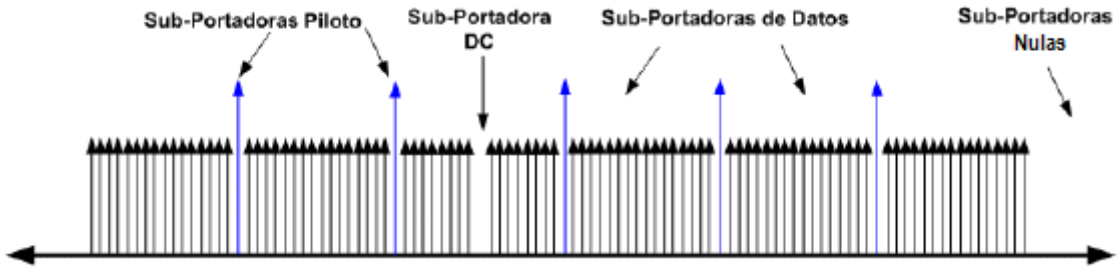


Figura 4.19: Estructura del símbolo OFDM.

4.2.8 Adición del CP

El bloque ilustrado en la Figura 4.20 fue integrado dentro del bloque de la IFFT, el cual es necesario para poder agregar el prefijo cíclico (CP). El prefijo o extensión cíclica consiste en copiar el resultado de las ultimas muestras de la IFFT al inicio de estas, para lograr lo anterior se necesita almacenar todo el resultado del cálculo de la IFFT, y después mostrarlo en el orden requerido, pero además se requiere cada resultado o muestra cada T_m para poder generar el símbolo OFDM en el dominio del tiempo, el bloque OUT_DATOS_CP permite realizar lo descrito anteriormente.

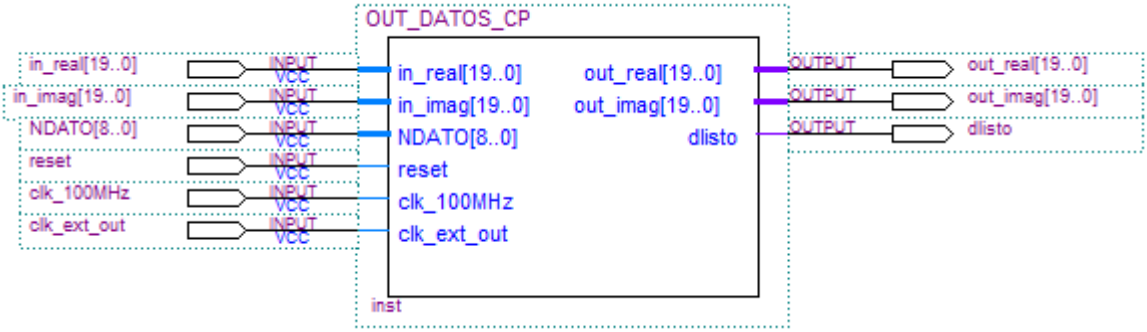


Figura 4.20: Bloque para la adición del prefijo cíclico.

4.2.9 Almacenamiento de Muestras

En el sistema OFDMA es necesario almacenar los datos a la salida del transmisor y los datos o muestras a la entrada del receptor. En el transmisor esto permite procesar nueva información mientras todavía se está mostrando el resultado anterior, que es lo que sucede en el sistema real los símbolos OFDM tienen que generarse uno tras otro. Entonces en el receptor se estará recibiendo información o muestras continuamente, por lo que se requiere guardar la información mientras el receptor termina con el proceso anterior. El bloque ilustrado en la Figura 4.21 permite tomar muestras con un tiempo T_m del símbolo OFDM transmitido y almacenarlas, para después enviar las muestras al bloque de la FFT, tanto el bloque expuesto en el punto anterior como este permiten intercambiar información entre los bloques de la IFFT/FTT usando el reloj de 100MHz y después operar con el reloj de periodo T_m.

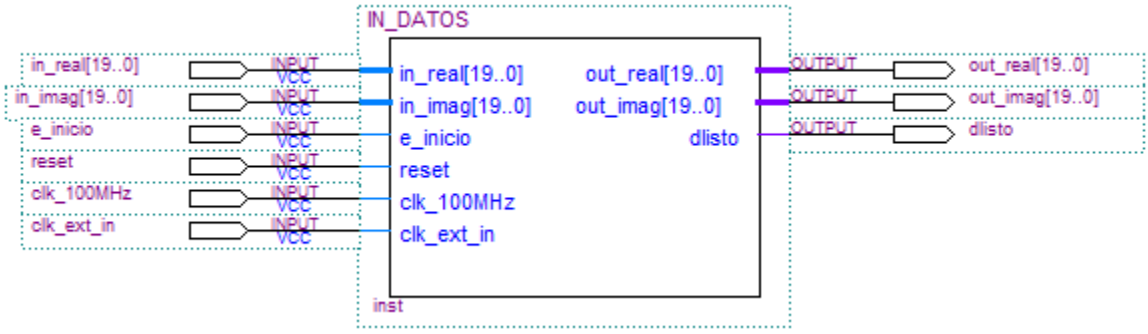


Figura 4.21: Bloque para almacenamiento de muestras.

4.3 Conclusión del capítulo

En este capítulo se describieron los bloques creados a nivel hardware, para la implementación de estos se tomó como base los algoritmos propuestos, diagramas de flujo y resultados expuestos en el capítulo anterior. Para el desarrollo de los bloques se utilizó el lenguaje de descripción de hardware de Altera (AHDL) y el software QUARTUS II. La plataforma de desarrollo empleada fue una tarjeta con el dispositivo FPGA Cyclone V 5CSEMA4U23C6N, que permitió alcanzar los requisitos exigidos por el diseño, en velocidad y tamaño. Lo que queda es demostrar el correcto funcionamiento de los bloques, esta comprobación se presenta en el siguiente capítulo.

Capítulo 5

Resultados

Para comprobar el correcto funcionamiento del sistema desarrollado se realizaron varias pruebas, tanto a nivel simulación con el software QUARTUS II, como con el dispositivo físico. En las simulaciones con QUARTUS II se busca exponer el correcto funcionamiento de los bloques del receptor, para esto se hace uso de los bloques del transmisor, donde se verificó que el receptor recuperara la información enviada del transmisor, también se hizo una comparación de los resultados con los obtenidos de la simulación en MATLAB y los reportados en el estándar IEEE 802.16. Después de comprobar el correcto funcionamiento a nivel simulación, se realizaron pruebas físicas, primero programando a una solo tarjeta todo el sistema, y después dividiendo el transmisor y el receptor en dos tarjetas. El presente capitulo tiene el objetivo de ilustrar y describir los resultados de dichas pruebas.

5.1 Datos de entrada

La simulación realizada con el software MATLAB está basada en el estándar IEEE 802.16, donde la secuencia de entrada para la simulación fue tomada de dicho estándar, para las simulaciones en QUARTUS II se empleó la misma secuencia de entrada, lo que permite hacer comparaciones entre simulaciones. Debido a que los bloques desarrollados todavía no incluyen la interfaz del aire, solo se tiene la parte del procesamiento digital, para poder probar la capacidad de recuperación del receptor se tomaron los resultados de las simulaciones en MATLAB, en donde se considera una transmisión con ruido. Para poder generar las secuencias de entrada fueron creados bloques adicionales a los ya implementados, los cuales se muestran en la Figura 5.1 y 5.2.

Figura 5.1: Bloque para introducir la secuencia de prueba.

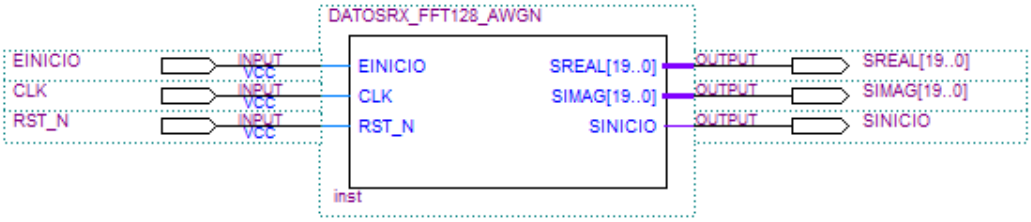


Figura 5.2: Bloque para introducir los datos a la FFT.

5.2 Resultados de la aleatorización

Para corroborar que el proceso de la aleatorización es correcto, se hizo uso del mismo bloque en el transmisor y en el receptor, la prueba consiste en introducir la entrada de prueba al primer bloque y la salida de este es introducida al segundo bloque.

En la Figura 5.3 se ilustran los resultados de la simulación con MATLAB en donde se muestran los valores obtenidos después de la aleatorización, estos se presentan en hexadecimal y en binario, la parte mostrada en binario tiene como fin poder comparar la simulación en MATLAB con la obtenida en QUARTUS II.

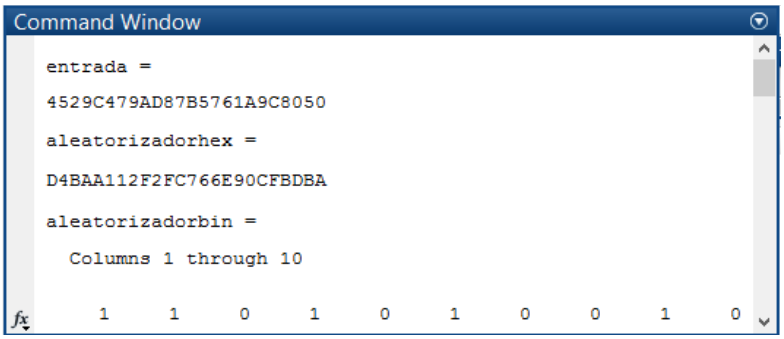


Figura 5.3: Resultado en MATLAB correspondiente a la aleatorización.

En la Figura 5.4 se presenta el resultado de la simulación en QUARTUS II en donde se tiene la señal *ENTRADA* correspondiente a la secuencia de prueba, la señal *ALEA_TX* que es la salida del aleatorizador, y por último la señal *ALEA_RX* que corresponde a la salida del desaleatorizador, esta última señal es igual que la señal de entrada, con lo que se verifica que se está realizando correctamente el proceso inverso a la aleatorización.

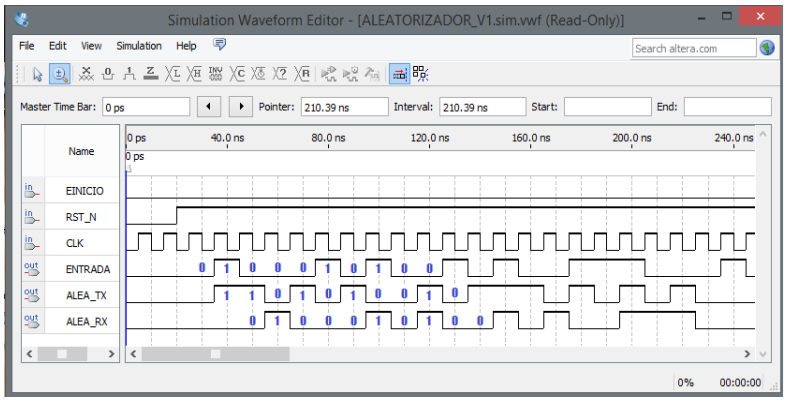


Figura 5.4: Simulación en QUARTUS II correspondiente a la aleatorización.

5.3 Resultados de la decodificación

En el receptor el decodificador se encarga de la búsqueda de errores y la corrección de estos. Para comprobar que el decodificador implementado cumpliera con su objetivo, se consideraron dos casos, en el primero las salidas del codificador son introducidas a las entradas del decodificador, en el segundo caso las salidas del codificador son modificadas, esto equivale a tener errores en la transmisión, después estos datos son introducidos al decodificador.

En la Figura 5.5 se muestran los resultados obtenidos con MATLAB, en la parte superior se tiene la secuencia de entrada y después las salidas del codificador pero con errores, con el programa implementado se contabilizan los errores encontrados y seguido se desplegaban los datos decodificados, en la simulación se puede observar que se corrigieron los 23 errores encontrados, ya que la salida del decodificador corresponde con los introducidos a la entrada del codificador.

Figura 5.5: Decodificación en MATLAB.

En la Figura 5.6 se muestran los resultados obtenidos en la simulación con QUARTUS II, en donde se tiene la señal SX_TX y SY_TX que son las salidas del codificador, seguido se encuentran las señales SX_RX y SY_RX generadas con un bloque de prueba, muy semejantes a las salidas del codificador pero con algunas cambios que se interpretan como errores, las señales SX_RX y SY_RX sirvieron para evaluar la capacidad para corregir los errores del decodificador desarrollado, la señal $ERROR$ se muestra en alto cuando un error es detectado por el decodificador, por último se tiene las señales $SDECO$ y $SDECO\_SE$, donde $SDECO\_SE$ es la señal obtenida del decodificador sin introducir errores, en la simulación se observa que esta corresponde con la señal de entrada ($SERIAL$), con lo que se verifica que se está realizando el proceso inverso a la codificación, pero además la señal $SDECO$ corresponde a la decodificación introduciendo errores, es importante destacar que el decodificador se probó con error, obteniéndose la secuencia original de entrada y demostrándose que el decodificador corrigió los errores.

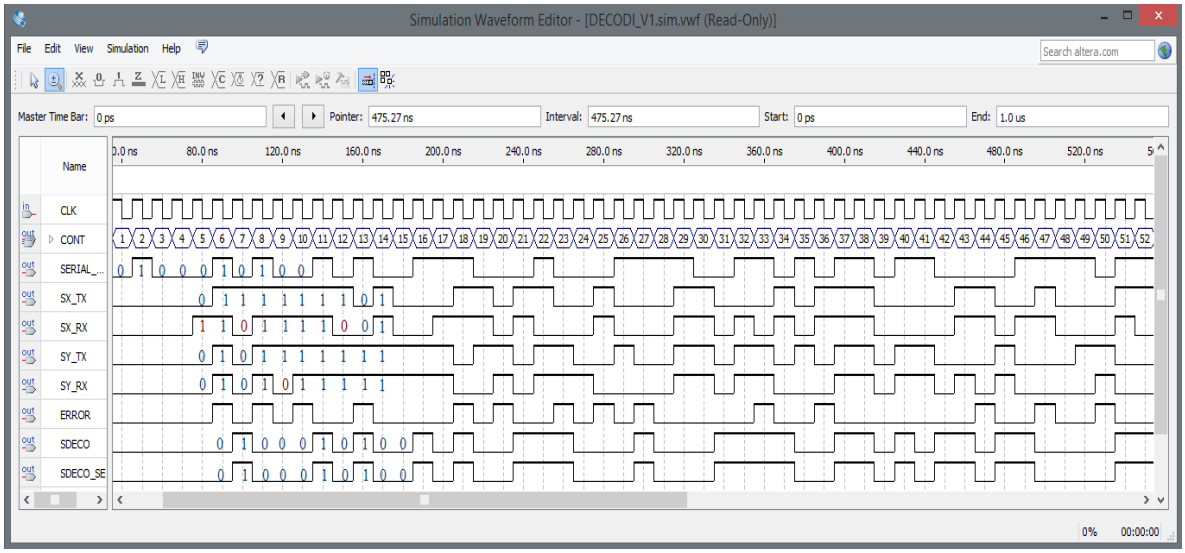


Figura 5.6: Simulación de la decodificación en QUARTUS II.

5.4 Entrelazador/Desentrelazador

En la Figura 5.7 se presentan los resultados obtenidos en la simulación con QUARTUS II del proceso de entrelazado, para el diseño implementado se tienen dos entradas al bloque de entrelazado, ya que el bloque anterior es el codificador y este tiene dos salidas, entonces las señales a entrelazar son ENTRADA1 y ENTRADA2, las señales ENTB1_TX y ENTB2_TX son las salidas de este bloque, el tiempo requerido para realizar el entrelazado es de 1.92µs, donde la primera mitad de este tiempo es necesario para almacenar la secuencia de bits y posteriormente los datos son mostrados en el orden calculado con las expresiones tomadas del estándar IEEE 802.16. En el receptor se realiza un proceso similar con el bloque de desentrelazado pero las entradas a este bloque provienen del desmapeador de símbolos. En la prueba realizada la salida del entrelazador es introducida al bloque del desentrelazador. En la simulación se observa que se obtiene nuevamente la secuencia original de entrada al entrelazador, con lo que se prueba que el proceso de entrelazo se realiza correctamente.

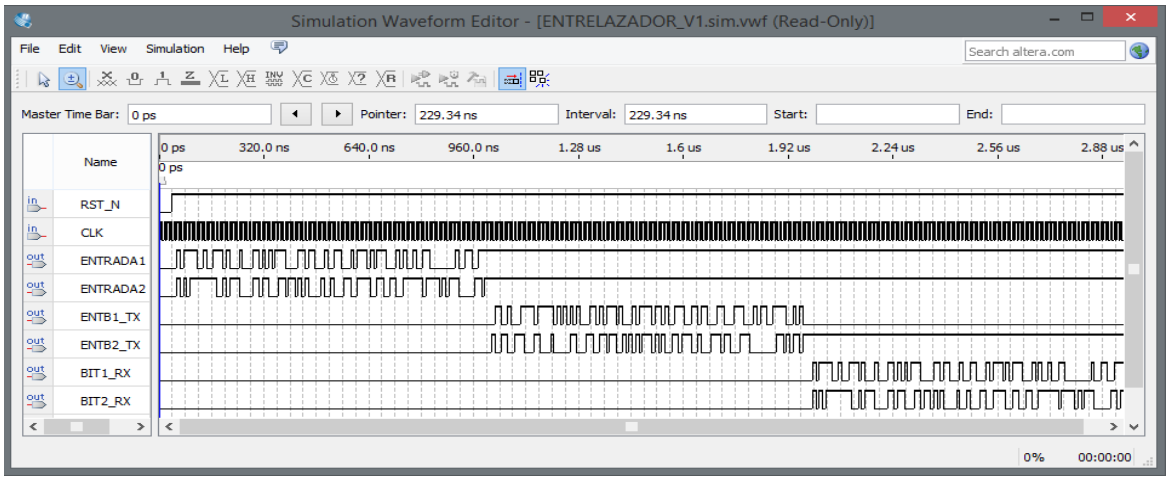


Figura 5.7: Simulación del proceso de entrelazado en QUARTUS II.

5.5 Resultados del mapeo y desmapeo de símbolos.

En la Figura 5.8 se presentan los resultados obtenidos en la simulación con QUARTUS II correspondiente al mapeo de símbolos, a la entrada del mapeador de símbolos se tienen las señales $ENT1$ y $ENT2$, y a la salida de este bloque se tiene $SREAL$ y $SIMAG$, formados por 20 bits, estos dos números sirven para representar los valores de los ejes de la constelación QPSK, en el receptor se realiza el proceso inverso, según sean los valores de $SREAL$ y $SIMAG$ se obtienen un par de bits. En la simulación se recupera la secuencie de entrada al mapeador.

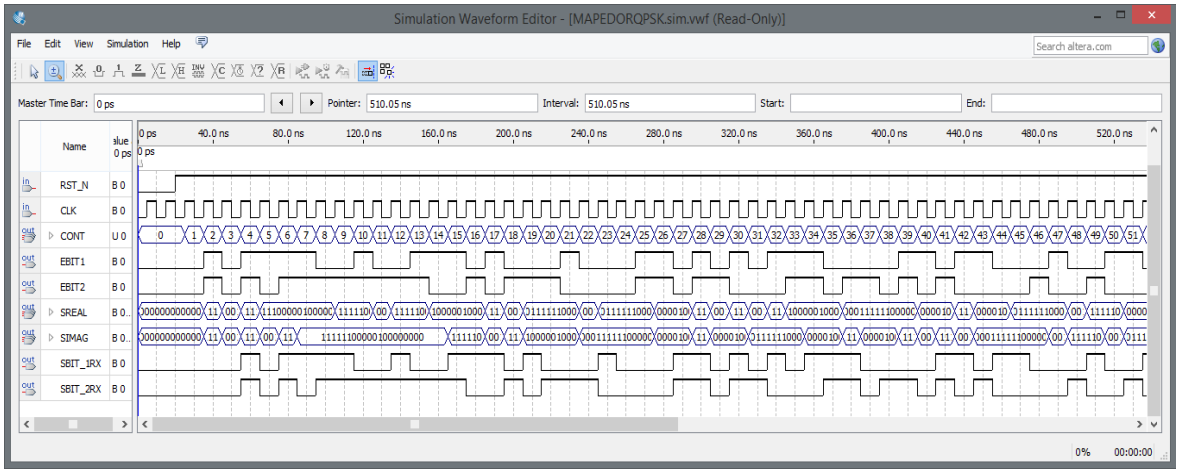


Figura 5.8: Simulación del mapeo de símbolos en QUARTUS II.

5.6 Resultados del transmisor

A continuación se presentan los resultados del procesamiento que se realiza en el transmisor. Los bloques que conforman al transmisor son: Aleatorizador, Codificador, Entrelazador, Mapeador, Inserción de Pilotos y Nulos, IFFT, Adición del CP, y CLKS. La Figura 5.9 tiene como objetivo mostrar las señales involucradas en el proceso (es decir, las salidas de los bloques que conforman al transmisor) e indicar el tiempo necesario para realizar el procesamiento en el transmisor, que es de 36 μs, después de este tiempo ya se tienen los datos para generar el símbolo OFDM en el dominio del tiempo. La Figura 5.10 es parte de la misma simulación, pero permite observar algunos resultados de la IFFT y la adición del prefijo. En el sistema se requiere que los datos se muestren en base a CLK_FFT.

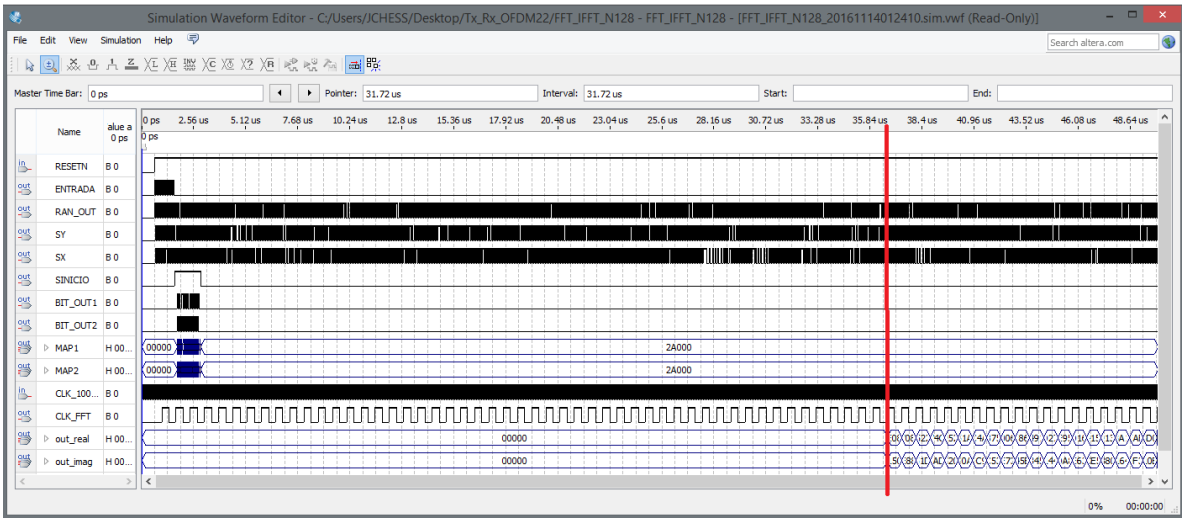


Figura 5.9: Simulación del transmisor.

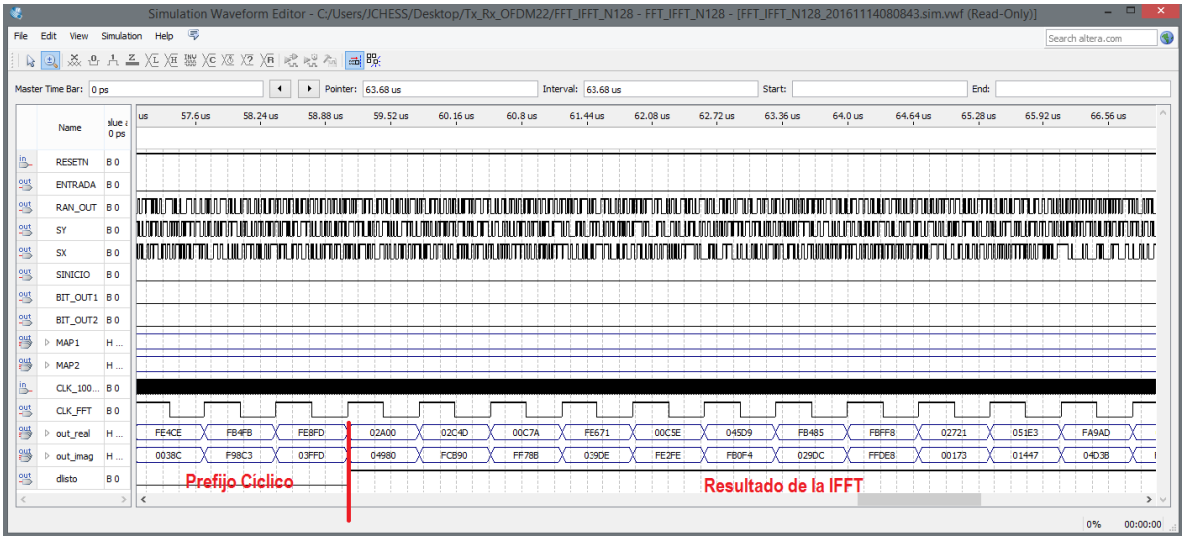


Figura 5.10: Resultado de la IFFT en el transmisor.

5.7 Resultados del receptor

El receptor está conformado por los siguientes bloques: Supresión del CP, Almacenamiento de Muestras, FFT, Supresión de Pilotos y Nulos, Desmapeador, Desentrelazador, Decodificador, Aleatorizador y CLKS. En la Figura 5.11 se muestran las señales involucradas en el proceso (que son las salidas de los bloques que conforman al receptor), en el receptor después de tener los datos de entrada para la FFT, se requiere de 37.6 μs para recuperar los bits enviados.

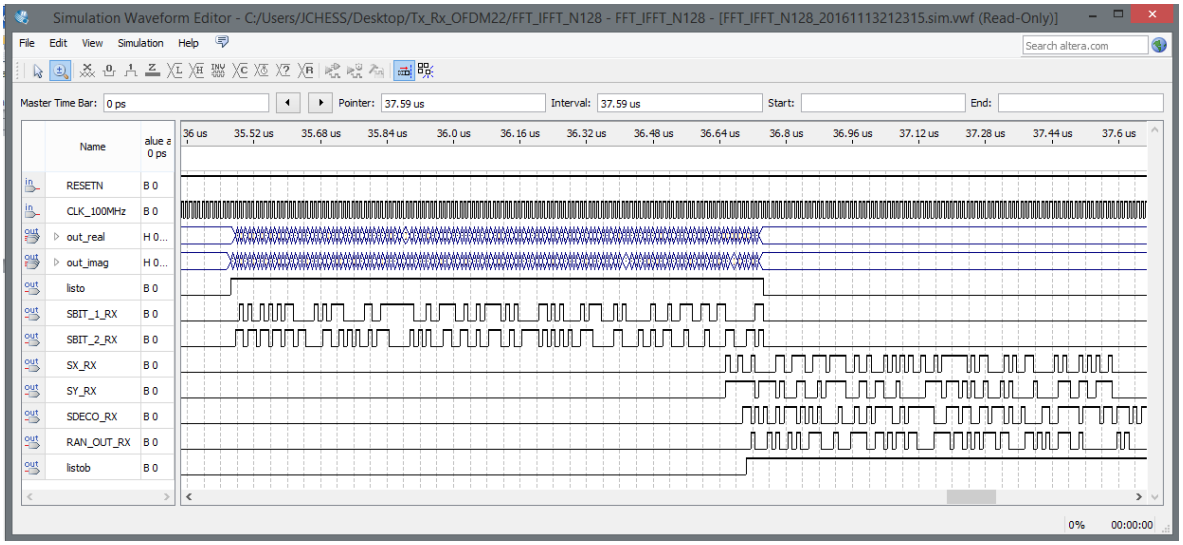


Figura 5.11: Simulación del receptor.

En la Figura 5.12 se muestran algunos resultados de la FFT en el receptor, el procesamiento se realiza en base al reloj de 100 MHz.

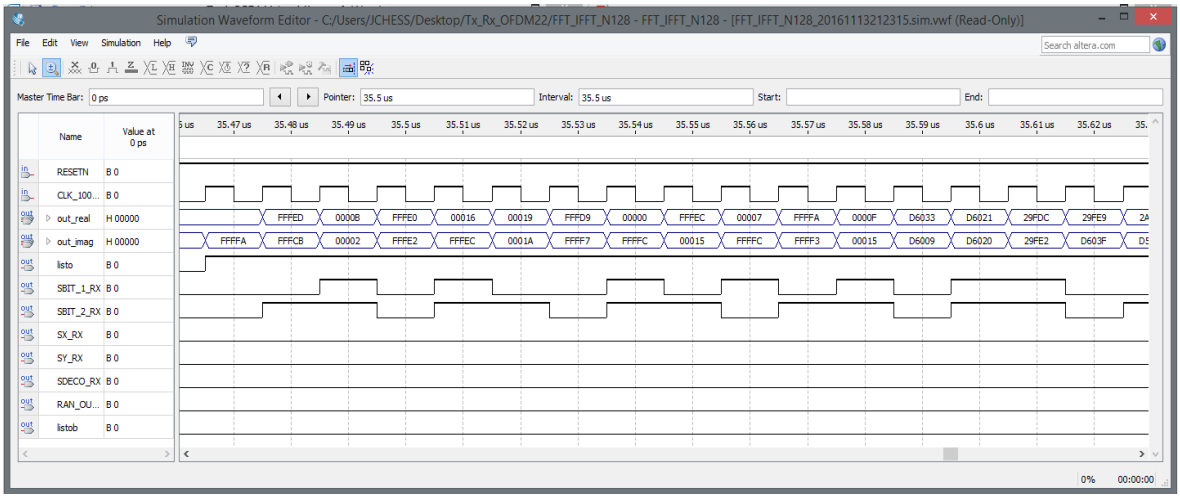


Figura 5.12: Resultados de la FFT en el receptor.

En la Figura 5.13 se muestran los resultados finales obtenidos en el receptor, después de realizar todo el proceso inverso del transmisor, se recupera la información enviada, que es la salida del aleatorizador en el receptor, esta salida es representada por la señal RAN_OUT_RX, el resultado final de esta simulación es que se logra recuperar la secuencia de prueba, en hexadecimal: 4529C479AD87B5761A9C8050.

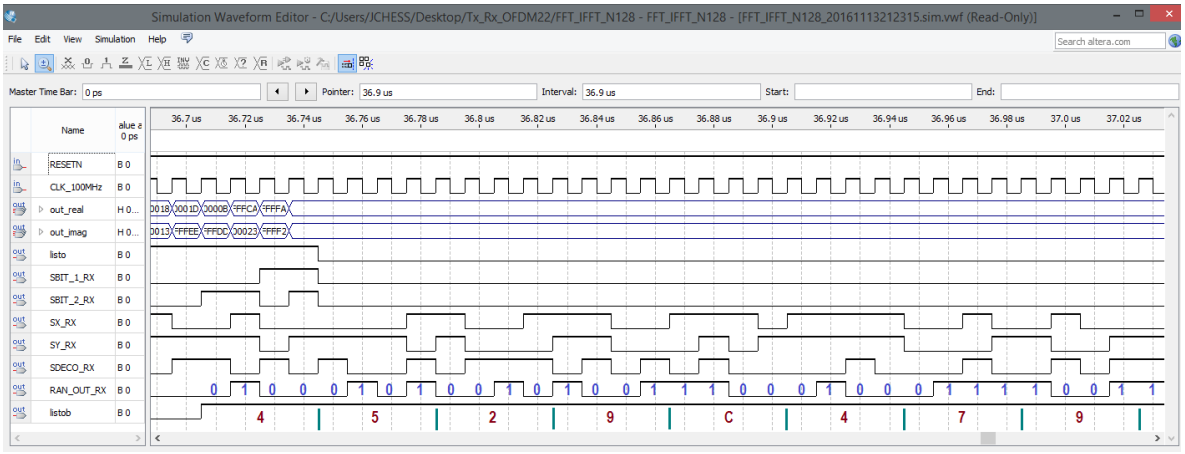


Figura 5.13: Bits recuperados.

5.8 Pruebas experimentales

Después de verificar el correcto funcionamiento de los bloques desarrollados a nivel simulación se realizaron pruebas con el dispositivo físico. La plataforma de desarrollo empleada fue una tarjeta DEO-Nano-SoC con el dispositivo FPGA Cyclone V 5CSEMA4U23C6N, la cual permitió incluir el trasmisor y receptor en una sola tarjeta. Las pruebas realizadas se hicieron primero con el uso de una sola tarjeta y después se dividió el transmisor y el receptor usando dos tarjetas.

En el sistema implementado se requieren enviar 96 bits cada 112.36 μs. Para las pruebas se usó un convertidor analógico-digital de 8 bits, el cual se colocó a la entrada del FPGA. Con el FPGA se realizaron 12 lecturas del convertidor en 112.36 μs, con lo que se obtenían los 96 bits de entrada al sistema, la salida del transmisor se mandó a la entrada del receptor, y finalmente el receptor después de recuperar los 96 bits, envió los bits en paquetes de 8 a un convertidor digital-analógico para reconstruir la señal muestreada al comienzo. En la Figura 5.14 se ilustra el esquema para la prueba.

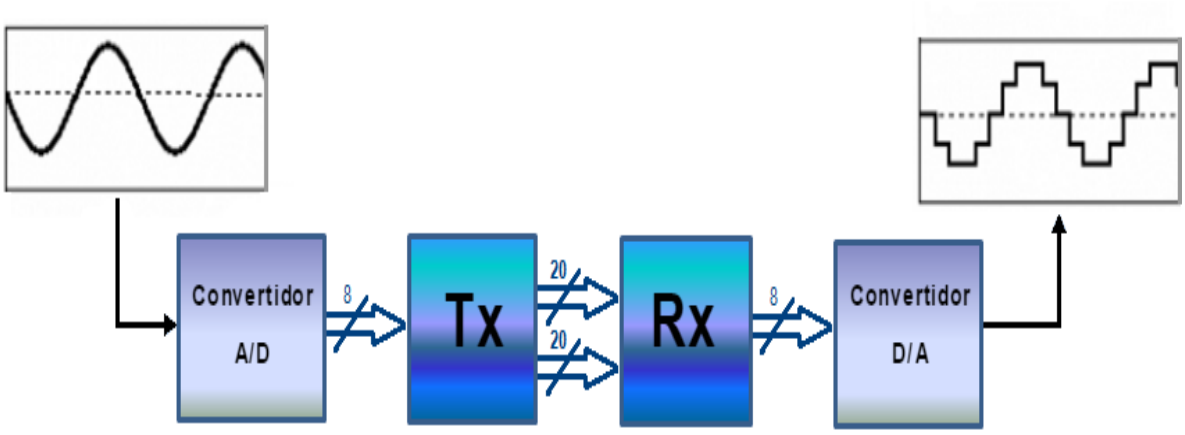


Figura 5.14: Diagrama a bloques de la prueba con el dispositivo real.

Para la conversión analógica-digital se empleó un PIC 16F877A y una etapa de acondicionamiento de la señal, para el proceso inverso se usó un arreglo de resistencias para la conversión digital-analógica y otra etapa para acondicionar la señal. El acondicionamiento de la señal se hizo para poder transmitir señales de audio. En la Figura 5.15 y 5.16 se muestran los elementos empleados en la prueba y la conexión entre ellos.

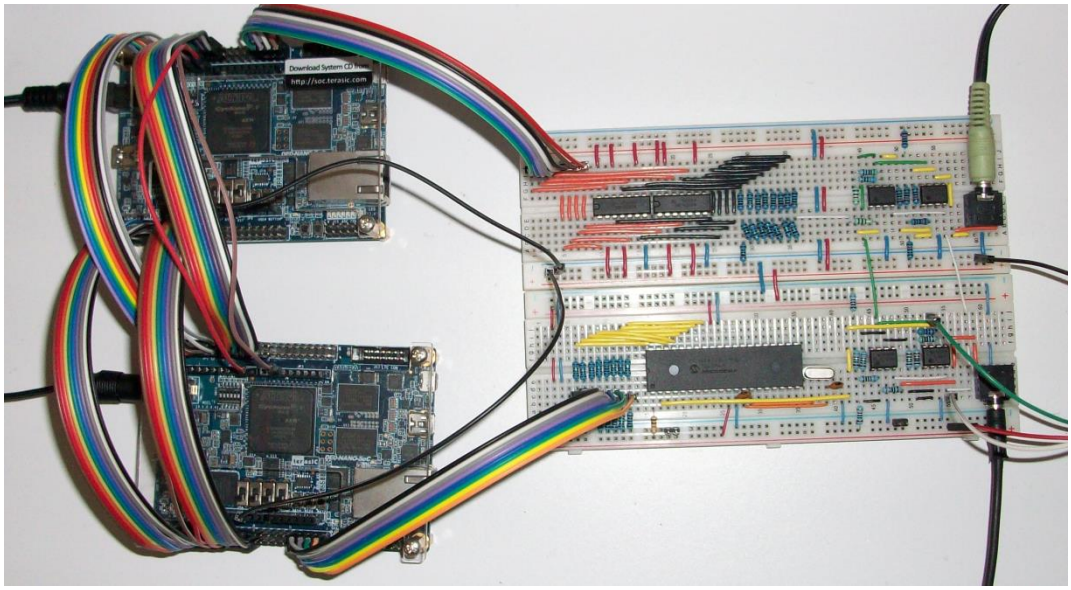


Figura 5.15: Conexión entre las tarjetas y la etapa de entrada-salida al sistema.

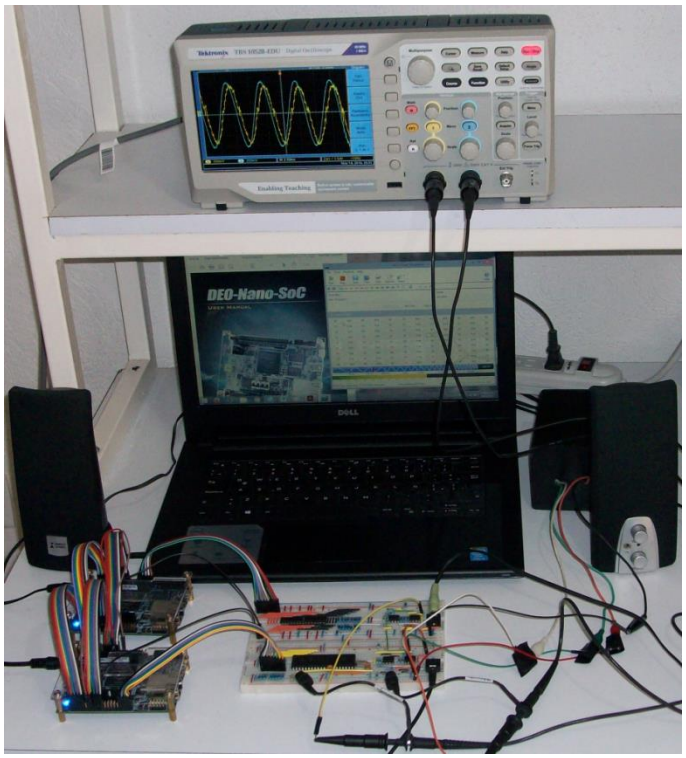


Figura 5.16: Prueba experimental del transmisor y receptor.

En la Figura 5.17 se muestran las señales obtenidas en la prueba experimental. La señal en color amarillo corresponde a la entrada de la prueba, y después de haber sida procesado por el transmisor y el receptor se obtuvo la señal en color azul. En la prueba se verificó el correcto funcionamiento del sistema, el receptor logró recuperar la información. La diferencia entre las señales es por las conversiones digital-analógico, y el desfase por el tiempo requerido para el procesamiento.

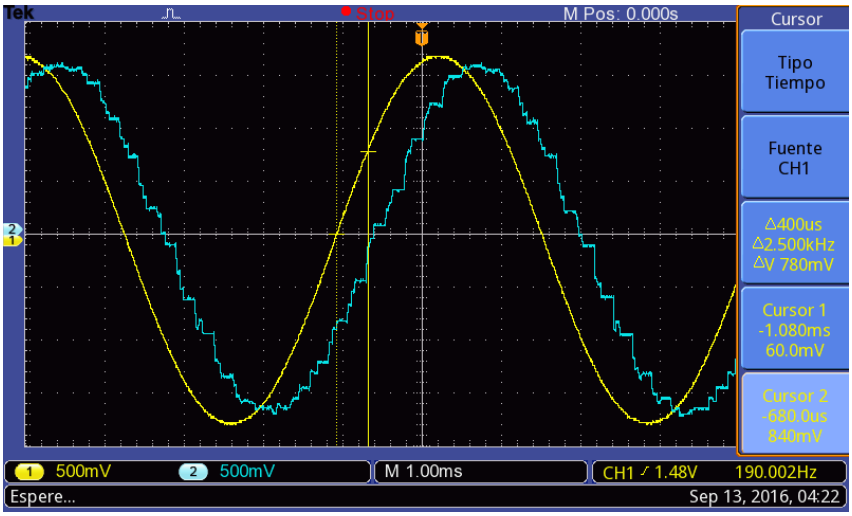
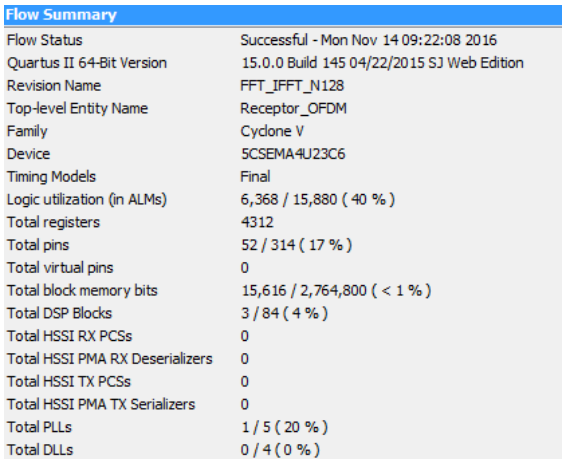


Figura 5.17: Señales obtenidas en la prueba experimental.

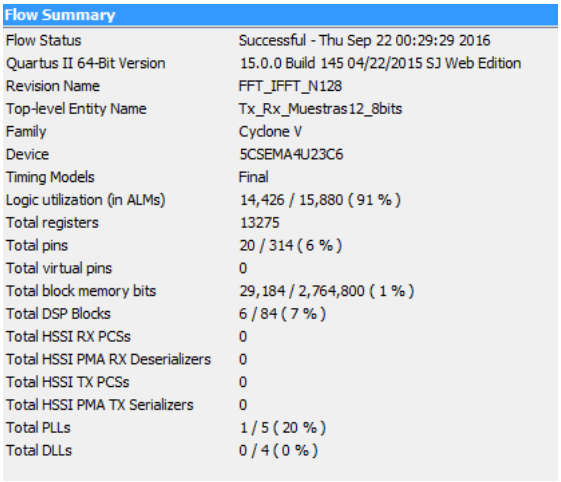
5.9 Reporte de compilación

Otro resultado importante de este proyecto fue la optimización de los bloques diseñados para la creación del transmisor y receptor OFDMA en plataforma FPGA. Esta optimización se puede observar en los reportes de compilación. En la Figura 5.18 se observa el reporte de compilación del receptor, en el se muestra que se ocupa menos de la mitad de los elementos disponibles en la tarjeta de desarrollo seleccionada. La correcta optimización y las características de la tarjeta permiten programar el transmisor y receptor juntos. Este resultado es de suma importancia para trabajos posteriores. En la Figura 5.19 se muestra el reporte del sistema completo. Nótese que para el sistema completo se requiere de un 92% de la capacidad del FPGA lo que deja espacio para futuras implementaciones.



Flow Summary	
Flow Status	Successful - Mon Nov 14 09:22:08 2016
Quartus II 64-Bit Version	15.0.0 Build 145 04/22/2015 SJ Web Edition
Revision Name	FFT_IFFT_N128
Top-level Entity Name	Receptor_OFDM
Family	Cyclone V
Device	5CSEMA4U23C6
Timing Models	Final
Logic utilization (in ALMs)	6,368 / 15,880 (40 %)
Total registers	4312
Total pins	52 / 314 (17 %)
Total virtual pins	0
Total block memory bits	15,616 / 2,764,800 (< 1 %)
Total DSP Blocks	3 / 84 (4 %)
Total HSSI RX PCSs	0
Total HSSI PMA RX Deserializers	0
Total HSSI TX PCSs	0
Total HSSI PMA TX Serializers	0
Total PLLs	1 / 5 (20 %)
Total DLLs	0 / 4 (0 %)

Figura 5.18: Reporte de compilación del receptor.



Flow Summary	
Flow Status	Successful - Thu Sep 22 00:29:29 2016
Quartus II 64-Bit Version	15.0.0 Build 145 04/22/2015 SJ Web Edition
Revision Name	FFT_IFFT_N128
Top-level Entity Name	Tx_Rx_Muestras12_8bits
Family	Cyclone V
Device	5CSEMA4U23C6
Timing Models	Final
Logic utilization (in ALMs)	14,426 / 15,880 (91 %)
Total registers	13275
Total pins	20 / 314 (6 %)
Total virtual pins	0
Total block memory bits	29,184 / 2,764,800 (1 %)
Total DSP Blocks	6 / 84 (7 %)
Total HSSI RX PCSs	0
Total HSSI PMA RX Deserializers	0
Total HSSI TX PCSs	0
Total HSSI PMA TX Serializers	0
Total PLLs	1 / 5 (20 %)
Total DLLs	0 / 4 (0 %)

Figura 5.19: Reporte de compilación del sistema completo.

Conclusiones

A partir de las pruebas y los resultados alcanzados se obtuvieron las siguientes conclusiones:

Fue posible la comprensión del funcionamiento del transmisor y receptor OFDMA basado en el estándar IEEE 802.16, identificando sus características, sus principios de operación, parámetros, especificaciones y las etapas que lo conforman.

Se desarrolló el sistema OFDMA a nivel simulación con el software MATLAB. La simulación previa a la implementación real sirvió para evaluar el correcto funcionamiento de los algoritmos propuestos, permitiendo la visualización de la respuesta e información esperada en cada etapa del sistema. Las simulaciones sirvieron como base para la implementación en el FPGA.

Se crearon los bloques que conforman al sistema en hardware programable (AHDL) para su implementación en plataforma FPGA. También en esta plataforma se lograron integrar los bloques de la IFFT y FFT al sistema, ya que estos fueron desarrollados en otro trabajo de tesis. La integración requirió el análisis y optimización de la implementación de la IFFT y FFT para adaptar los bloques a nuestro sistema.

La plataforma de desarrollo empleada fue una tarjeta DEO-Nano-SoC con el dispositivo FPGA Cyclone V 5CSEMA4U23C6N, que permitió alcanzar los requisitos exigidos por el diseño, en velocidad y tamaño. En tamaño permitió incluir el trasmisor y receptor en una sola tarjeta, y en velocidad se logró realizar el procesamiento en 38 μs de los 112.36 μs requeridos para un ancho de banda de 1.25 MHz. Para mayores anchos de banda se requiere la optimización en tiempo de la FFT. De hecho, en este trabajo nuestro tope de operación es de 2.5 MHz.

Se comprobó el correcto funcionamiento de los bloques del transmisor y el receptor, primero mediante simulaciones en QUARTUS II y luego con el dispositivo real. En los dos casos el receptor recupero la información enviada por el transmisor.

Los resultados en general fueron comprobados con los datos que proporciona el estándar IEEE 802.16-2009. La implementación del sistema bajo las normas y recomendaciones del

estándar IEEE 802.16 permite la compatibilidad con otros productos de banda ancha basados en el mismo estándar.

La gran ventaja de haber desarrollado los bloques del sistema con diseños propios es que se puede realizar modificaciones y utilizarlos sin ningún problema de licencia. En la primera versión del transmisor se usaba un IP Core de Altera para la IFFT, lo que elevaba en gran medida los costos del sistema de comunicación ya que se requería la compra de una licencia.

Una de las mayores aportaciones de nuestro trabajo es el desarrollo del sistema de transmisión y recepción completo e integrado en un solo dispositivo FPGA.

A futuro faltaría desarrollar la interfaz de aire, después de haber logrado el enlace se pueden realizar cambios a los bloques desarrollados, en donde se puede modificar el tipo de modulación digital empleada por una del tipo 16-QAM o 64-QAM que permitiría una mayor tasa de transferencia.

Apéndice A

Publicación

Diseño e instrumentación de un sistema de recepción OFDMA en plataforma FPGA.

J.C. Gutiérrez Ortega, J. Castañeda Camacho, S. Vergara Limon, J. E. M. Gutiérrez Arias.
Facultad de Ciencias de la Electrónica (FCE), Maestría en Ciencias de la Electrónica Opción en Automatización,
Benemérita Universidad Autónoma de Puebla (BUAP), Av. San Claudio y 18 Sur S/N. C.U.,
Edificio 109A C.P. 72570 Puebla, Pue., México.

Resumen: *El Multiplexaje por División Ortogonal de Frecuencia (OFDM) es una técnica de modulación popular y ampliamente aceptada en el área de comunicaciones inalámbricas. La versión multiusuario del sistema OFDM es llamada Acceso Múltiple por División Ortogonal de Frecuencia (OFDMA). La principal ventaja de OFDMA es su robustez al desvanecimiento del canal en el entorno inalámbrico. En un sistema OFDMA la Transformada Rápida de Fourier (FFT) y su contraparte la transformada rápida de Fourier inversa (IFFT) se utilizan para modular y demodular los símbolos de datos sobre las subportadoras. En OFDMA la idea es utilizar un número de subportadoras esparcidas uniformemente sobre una banda de frecuencia, de tal manera que el ancho de banda disponible se utilice para una máxima eficiencia. OFDM y OFDMA tienen muchas ventajas que contribuyen a su popularidad; por lo tanto, ha sido usado en muchos sistemas inalámbricos y adoptado por diversas normas. El estándar IEEE 802.16e se basa en el empleo de OFDMA y ha surgido como un fuerte candidato para futuros sistemas inalámbricos. El objetivo de este trabajo es implementar un sistema OFDMA basado en el estándar IEEE 802.16e usando un FPGA de Altera de la familia Cyclone III y mostrar los resultados de las simulaciones de los bloques utilizados en el sistema propuesto usando Matlab y Quartus II.*

Abstract: *Orthogonal Frequency Division Multiplexing (OFDM) is a popular and widely accepted modulation technique in the area of wireless communication. Multi-user version of the OFDM system is called Orthogonal Frequency Division Multiple Access (OFDMA). The main advantage of OFDMA is its robustness to channel fading in wireless environment. In the OFDMA systems the Fast Fourier Transform (FFT) and its counterpart the Fast Fourier Transform Inverse (IFFT) are used to modulate and demodulate the data symbol on the subcarriers. In OFDMA the idea is to utilize a number of subcarriers, spread regularly over a frequency band, in such a way so that the available bandwidth is utilized to maximal efficiency. OFDM and OFDMA have many benefits contributing to its popularity; therefore, it has been used in many wireless systems and adopted by various standards. The standard IEEE 802.16e uses an OFDMA access technique and it has emerged as a strong candidate for future wireless systems. The objective of this paper is the implementation of the OFDMA system based on the standard IEEE 802.16e using a FPGA Altera Cyclone III family and show results of simulations of the blocks used in the proposed system by using Matlab and QuartusII.*

Keywords: *FFT, IFFT, FPGA, OFDMA, OFDM.*

I. INTRODUCCÓN

Hoy en día la cantidad de usuarios de los sistemas inalámbricos ha ido en aumento y no solo eso, el incremento del requerimiento de mayores tasas de datos ha crecido drásticamente haciendo de gran importancia la renovación de los sistemas de comunicaciones para poder satisfacer dichas demandas. Una de las propuestas tecnológicas que ha tenido gran aceptación está basada en el uso del Multiplexaje por División Ortogonal de Frecuencia que permite la transmisión de grandes cantidades de datos digitales, a través de una onda de radio. Este esquema ha sido usado en Europa para la transmisión de televisión y radio digital, pero a nivel mundial la atención hacia OFDM y su versión multiusuario OFDMA ha ido en aumento haciendo de estos claramente la tendencia a seguir para los próximos años [1].

De manera conceptual OFDM ha existido durante décadas, pero su implementación real y con costos aceptables ha sido posible desde la llegada y propagación de tecnologías como los microprocesadores de alta velocidad y los dispositivos lógico programables para poder hacer fiable el procesamiento digital requerido [2].

La implementación de este esquema de acceso requiere de la IFFT para lograr la transmisión por medio de subportadoras ortogonales, por eso el número de subportadoras está ligado al número de muestras de la transformada. De manera general, OFDM se refiere a la transmisión de una trama digital que requiere una gran tasa de transferencia mediante N líneas paralelas más lentas en subportadoras contiguas y ortogonales que transportan símbolos independientes que son producto de algún tipo de modulación digital como QPSK, 16-QAM, 64-QAM [3]. La Figura 1 muestra un ejemplo con el módulo de los espectros correspondientes a un conjunto de 6 subportadoras. La diferencia entre OFDM y OFDMA es que en la última cada subportadora puede pertenecer a usuarios distintos [4].

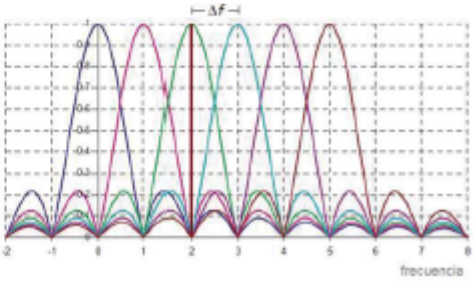


Figura 1. Espectro correspondiente a 6 subportadoras OFDM.

ISBN: 978-607-97128-2-2

SEP

SECRETARÍA DE EDUCACIÓN PÚBLICA

EL INSTITUTO TECNOLÓGICO DE LA PAZ

OTORGA EL PRESENTE

RECONOCIMIENTO

A

JUAN CARLOS GUTIÉRREZ ORTEGA, JOSEFINA CASTAÑEDA CAMACHO, SERGIO VERGARA LIMON Y JOSE EMILIO MOISES GUTIÉRREZ ARIAS

POR SU ARTÍCULO: DISEÑO E INSTRUMENTACIÓN DE UN SISTEMA DE RECEPCIÓN OFDMA EN PLATAFORMA FPGA

PRESENTADO EN EL TERCER CONGRESO INTERNACIONAL DE ROBÓTICA Y COMPUTACIÓN

CELEBRADO DEL 2 AL 4 DE MAYO DEL AÑO EN CURSO.

LA PAZ, B.C.S., 4 DE MAYO DE 2016.

SECRETARÍA DE EDUCACIÓN PÚBLICA
TECNOLÓGICO NACIONAL DE MÉXICO
INSTITUTO TECNOLÓGICO DE LA PAZ
DIRECCIÓN

ING. JESÚS DAVID ESTRADA RUIZ
DIRECTOR

Referencias

[1] N. LaSorte, J Barnes, H. Refai, The History of Orthogonal Frequency Division Multiplexing, Articulo IEEE, 2008.

[2] L. Hanzo, J. Akhtman, L. Wang, M. Jiang, MIMO-OFDM for LTE, Wi-Fi and WIMAX: coherent versus non-coherent and cooperative turbo-transceivers, John Wiley & Sons Ltd Publication, 2011.

[3] M. Schwartz, Introduction to "The History of OFDM", IEEE Communications Magazine, 2009

[4] H. Liu and G. Li, OFDM-Based Broadband Wireless Networks: Design and Optimization. John Wiley and Sons Ltd, 2005.

[5] A. Gutiérrez, Diseño del nivel físico de un radio OFDM para comunicación digital con una plataforma SDR, Universidad Autónoma Metropolitana, tesis de Maestría, 2014.

[6] J. Tao, S. Lingyang, Z. Yan, Orthogonal Frequency Division Multiple Access Fundamentals and Aplications, CRC Press, USA, 2010.

[7] Samuel C. Yang, OFDMA System Analysis and Design, Artech House, 2010.

[8] Molisch, Andreas F., Wireless Communications, John Wiley & Sons, Ltd, USA, 2011.

[9] O.M. Sabino Moxo, J. Castañeda Camacho, M.A.D. Vargas Treviño, S. Vergara Limon, J.A. Ortega-Redondo, R.L Maya Ramírez, G. Mino Aguilar, Evaluación del Desempeño de un Sistema de Comunicaciones OFDMA, CIINDET, 2013.

[10] F. Mora Hernández, J. Castañeda Camacho, S. Vergara Limon, E. Ríos Silva, J. E. M. Gutiérrez Arias, Implementación de la Transformada Discreta de Fourier IFFT/FFT en un sistema embebido para automatizar un sistema de comunicación OFDMA, CIINDET, 2015.

[11] N. Weste and D. Skellern, VLSI for OFDM, IEEE Communications Magazine, 1998.

[12] D. Torres, Programación y evaluación en un DSP del nivel físico de un modem OFDM para comunicación digital por la red eléctrica doméstica, tesis de Maestría, 2012.

[13] G. Zhang, M. Leenheer, A. Morea and B. Mukherjee, A Survey on OFDM-Based Elastic Core Optical Networking, IEEE Communications Surveys and Tutorials, 2013.

[14] R. Agusti, F. Bernardo, F. Casadevall, R. Ferrús, J. Pérez y O. Sallent, LTE: Nuevas tendencias en comunicaciones móviles, Fundación Vodafone España, 2010.

[15] Yong Soo, Jaekwon Kim, Won Young, Chung Kang, MIMO-OFDM Wireless Communications with MATLAB, John Wiley & Sons Ltd, 2010.

[16] J. Vergara González, Simulación de un Esquema de Modulación/Demodulación OFDM Utilizando un Modelo de Canal Multitrayectoria. Guayaquil, Ecuador, 2008.

[17] Tzi-Dar Chiueh, Pei-Yun Tsai, OFDM Baseband Receiver Design for Wireless Communications, John Wiley & Sons Ltd, 2007.

[18] K.C. Chen, J.R.B De Marca, Mobile WiMAX, John Wiley & Sons Ltd, 2008.

[19] J. Andrews, A. Ghosh, y R. Muhamed, Fundamentals of WiMAX: understanding broadBand wireless networking, Prentice Hall, USA, 2007.

[20] IEEE Standard 802.16-2009, IEEE standard for Local and metropolitan area networks, Part 16: Air Interface for Broadband Wireless Access Systems. IEEE Microwave Theory and Techniques Society, 2009.

[21] J.M. Vergara, Análisis del desempeño de la capa física basada en OFDM, tesis de Maestría, UNAM, 2009.

[22] R. S. Bahai, R. Saltzberg, M. Ergen, Multi-Carrier Digital Communications: Theory and Applications of OFDM, Springer, 2004.

[23] H. Díaz, O. Hurtado, Implementación en software de un decodificador de Viterbi para aplicaciones DVB-S, Instituto Politécnico Nacional, 2013.

[24] Fernando Reyes Cortés, MATLAB Aplicado a Robótica y Mecatrónica, Editorial Alfaomega, 2012.

[25] Tzi-Dar Chiueh, Pei-Yun Tsai, I-Wei Lai, Baseband Receiver Design for Wireless MIMO-OFDM Communications, John Wiley & Sons Singapore , segunda edición, 2012

[26] MAX PLUS II AHDL, Languaje Reference, Altera, 1995.

[27] ALTERA. Cyclone Series FPGAs & SoCs [Online]. Available: http://www.altera.com/products/fpga/cyclone-series.html, 2016.

[28] TERASIC, DEO-Nano-SoC: USER MANUAL, ALTERA UNIVERSITY PROGRAM, 2015.