UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

FACULTAD DE INGENIERÍA

DIVISIÓN DE INGENIERÍA ELÉCTRICA

ANÁLISIS, IMPLEMENTACIÓN Y EVALUACIÓN DE ALGORITMOS FFT PARA EL MULTIPLEXADO OFDM EN REDES DE COMUNICACIÓN INALÁMBRICA WiMAX

T E S I S

QUE PARA OBTENER EL TÍTULO DE :

INGENIERO ELÉCTRICO-ELECTRÓNICO

P R E S E N T A :

JOSÉ DOMINGO SANDOVAL GARCÍA

DIRECTOR DE TESIS:
DR. ROGEIO ALCÁNTARA SILVA

Noviembre 2010

Resumen.

Las necesidades de servicios de banda ancha personales están aumentando debido a que los usuarios requieren datos a altas velocidades, comunicaciones instantáneas, aplicaciones multimedia entre otras. La urgencia de tener estos servicios esta causando el desarrollo de nuevas tecnologías, como WiMAX, que proporcionen estos servicios en cualquier momento, en cualquier lugar de una zona metropolitana y aún cuando el usuario este en movimiento. La evolución de las tecnologías de comunicación de alta velocidad se debe en gran medida al desarrollo de sistemas de modulación como el Multiplexado con División en Frecuencia Ortogonal (OFDM), la cual es usada en los sistemas de comunicación de banda ancha comerciales. Una parte muy importante dentro de la cadena de OFDM es la utilización de la transformada de Fourier rápida (FFT), la cual permite el ahorro de ancho de banda para la transmisión de información así como el ahorro de energía y simplifica el diseño del sistema. La elección de un buen algoritmo FFT es indispensable para el funcionamiento eficiente de la modulación OFDM y así cubrir los requerimientos de velocidad de transmisión de datos. En este trabajo se presentan los resultados del análisis, implementación, evaluación y comparación de varios algoritmos FFT. La evaluación y la comparación fueron realizadas con respecto a la precisión, velocidad de cálculo, complejidad aritmética y dificultad de implementación de los algoritmos. Se determinó que el algoritmo FFT Radix-4 con Decimación en Frecuencia es el más indicado para implementarlo bajo el contexto analizado.

Dedicatorias.

A mis padres Alberto Sandoval Espinosa y Rosaura García Abarca sin los cuales no podría estar aquí en ningún sentido. Gracias por todo el apoyo y la paciencia que me han tenido, no hay palabras con las cuales poder expresar mi agradecimiento por todo lo que me han dado y han hecho de mi. Este pequeño triunfo también es de ustedes. Gracias de todo corazón, los quiero muchísimo.

A mi hermano Alberto Sandoval García. Gracias por tu compañía, consejos, pláticas, apoyo en todas las cosas que se me ocurren, por tu ejemplo y por ser el mejor hermano del mundo. Te quiero mucho.

A Erandi Bañuelos Soberanis, mi amiga, compañera, inspiración y cómplice en esta aventura llamada vida. Te amo, gracias por tratarme suavemente.

Gracias familia, sin ustedes no sería la persona que soy ...

A las familias Jiménez García, Martínez Jiménez, Sandoval Chávez y Sandoval García. Gracias por todo el apoyo y cariño que me tienen, sepan que yo también los quiero igual. Gracias tío Memo, tía Irma y mis hermanos Guillermo, Mariana y Gerardo; mis tíos Ignacio y Marta y primos Augusto y César; mis padrinos Francisco y Carmen y mis primos Delia, Francisco y Manuel.

A Mamá Yeyita, gracias por tu apoyo incondicional, tus pláticas con las que aprendo mucho y por el cariño que nos tienes a todos tus nietos. Te quiero mucho.

A mis amigos de la facultad: Anuar, Carlos, Celestino, Edgar, José Juan y Juan. Gracias por su amistad y ayuda brindada en todos estos años que estudiamos juntos y aún cuando ya no estamos en la facultad.

A mis amigos de toda la vida: Diana, Erika, Gaby, Josué, Julio, Maria Luisa, Mauricio, Omar, Raymundo y Valery. Gracias por su apoyo incondicional, a su lado he crecido y vivido cosas muy importantes. Los quiero mucho. Por favor hagan extensiva esta dedicatoria a sus familias las cuales me han brindado cariño y amistad, muchas gracias.

Gracias de todo corazón.

Agradecimientos.

A Dios por permitirme estar y llegar aquí, por permitirme cumplir una meta más.

Al Dr. Rogelio Alcántara Silva por permitirme trabajar junto a él y asesorarme para la realización de este trabajo. Gracias por transmitirme sus conocimientos y opiniones no sólo con respecto a la ingeniería, si no también de la vida.

Al Laboratorio de Sistemas de Procesamiento y Transmisión de la Información y todos sus integrantes por compartir sus conocimientos y su amistad conmigo. Gracias Moisés Alvarado Hermida por compartir tus conocimientos conmigo y brindarme tu amistad.

A la Universidad Nacional Autónoma de México y a su Facultad de Ingeniería por ser mi segunda casa y haberme permitido tener el privilegio de estudiar en ella. Siempre estarán en mi corazón.

A los sinodales por su tiempo y comentarios para mejorar este trabajo.

Tabla de Contenidos

Indice de tablas

Índice de figuras

Capítulo 1

Introducción

El crecimiento de los servicios de banda ancha personales es inevitable debido a que los usuarios requieren datos a altas velocidades, comunicaciones instantáneas, aplicaciones multimedia entre otras. La urgencia de tener estos servicios esta causando el desarrollo de nuevas tecnologías, como WiMAX, que proporcionen estos servicios en cualquier momento, en cualquier lugar de una zona metropolitana y aún cuando el usuario este en movimiento. WiMAX ha sido llamado el reemplazo inalámbrico de la tecnología DSL (*Digital Subscriber Line*), la cual proporciona internet de alta velocidad, pero WiMAX es mucho más que sólo una alternativa de DSL o del cable-modem. WiMAX esta diseñado para hacer por el Internet lo que el teléfono celular hizo por la telefonía, brindar acceso en cualquier lugar a cualquier momento.

WiMAX está basada en el estándar de la red inalámbrica de área metropolitana (*WMAN*) desarrollado por el grupo IEEE 802.16. Está diseñada con las ventajas de las redes de band ancha fijas y móviles. WiMAX móvil ofrece portabilidad y movilidad. En el estándar IEEE 802.16 se incluyen las definiciones de la capa física, la capa de acceso y algunas descripciones de características obligatorias y opcionales.

El desarrollo de tecnologías de comunicación tanto cableadas como inalámbricas de alta velocidad se debe en gran parte al desarrollo de sistemas de modulación, los cuales permiten enviar información a alta velocidad por anchos de banda menores. Tal es el caso de la modulación Multiplexado con División en Frecuencia Ortogonal (OFDM), la cual es usada en los sistemas de comunicación de banda ancha comerciales, como DSL, Wi-Fi, DVB-H y WiMAX.

OFDM es una técnica de modulación con multiples portadoras que divide un canal de comunicaciones en una serie de bandas de frecuencias equidistantes. Una subportadora que contiene una porción de la información es transmitida en cada banda. Cada subportadora es ortogonal a todas las demás subportadoras. La popularidad de OFDM por su alta velocidad de transferencia de datos se debe principalmente a su eficiencia y su manejo flexible de la interferencia intersímbolos.

Una parte muy importante dentro de la cadena de los sistemas OFDM es la aplicación de la transformada de Fourier rápida (*Fast Fourier Transform, FFT*). La FFT permite el ahorro de ancho de banda para la transmisión de información así como el ahorro de energía y simplifica el diseño del sistema. Aplicar la FFT y su inversa en el transmisor-receptor de un sistema de comunicaciones evita el uso de bancos de sistemas de Radio Frecuencia (RF) necesarios para poder alcanzar una buena velocidad en la transmisión de datos.

La FFT dentro de la cadena de OFDM se realiza en un procesador dedicado específicamente para esta tarea. Existen varios tipos de arquitecturas para estos procesadores. La mayoría de las arquitecturas toman como base una familia de algoritmos de la FFT, la familia Radix. Casi todos se basan en la estructura de mariposa del algoritmo Radix-2, que permite el cálculo de la transformada de Fourier de manera rápida. Los algoritmos FFT hacen posible la implementación de la transformada de Fourier discreta (*Discrete Fourier Transform, DFT*) para el procesamiento en tiempo real.

Existen varias metodologías de diseño de sistemas que permiten diseñar en menor tiempo y con mejores resultados sistemas complejos. Como la FFT está contenida en la capa física de WiMAX, que es un sistema de comunicaciones muy complejo, es necesaria una metodología que permita identificar con mayor claridad los requerimientos necesarios para una buena elección del algoritmo.

La elección del algoritmo FFT es indispensable para el correcto funcionamiento de la modulación OFDM y así cubrir los requerimientos de velocidad de transmisión de datos. La velocidad de cálculo de los algoritmos FFT es uno de los principales criterios que se deben tomar en cuenta para la elección del algoritmo pero no el único. También es muy importante tomar en cuenta la precisión, complejidad de la implementación del algoritmo y la complejidad aritmética.

El objetivo del presente trabajo de tesis es analizar, implementar y evaluar diferentes algoritmos de la transformada de Fourier rápida y así poder seleccionar el más apto para cumplir los requerimientos del multiplexado OFDM de la capa física de las redes de comunicación inalámbrica WiMAX.

A continuación se da un panorama general de esta tesis.

En el capítulo 2 se describe la metodología de diseño de sistemas de procesamiento digital de señales en la se basará para la realización de este trabajo.

En el capítulo 3 se da un panorama general de las redes de comunicación inalámbrica WiMAX, antecedentes, características de las redes WiMAX, de los estándares, la arquitectura de la red y la capa física.

En el capítulo 4 se describe el funcionamiento general de la modulación OFDM para WiMAX así como sus principales ventajas.

En el capítulo 5 se describen los diferentes algoritmos de la transformada de Fourier rápida, su implementación y validación.

En el capítulo 6 se describe la evaluación y las comparaciones realizadas de los algoritmos FFT así como el análisis de estas comparaciones.

En el capítulo 7 se analizan los resultados de las comparaciones realizadas a los algoritmos implementados para poder seleccionar cual de ellos puede ser implementado en la cadena de la modulación OFDM para WiMAX.

Capítulo **2**

Metodología de Diseño de Sistemas de Procesamiento Digital de Señales.

Existe una gran variedad de metodologías de diseño de sistemas complejos, las cuales permiten el diseño en un menor tiempo y con mejores resultados. Ayudan siempre y cuando se tenga un enfoque claro del problema a solucionar. En la referencia [1] se describen a detalle diferentes metodologías de diseño de sistemas.

En el caso de sistemas de Procesamiento Digital de Señales (PDS) se ha propuesto la siguiente metodología en la referencia [2].

2.1. Metodología de Diseño de Sistemas de PDS.

La metodología de diseño de sistemas de PDS, además de ser utilizada para el desarrollo de este tipo de sistemas puede ser adaptada a otras áreas donde se requiera tratar a los sistemas de una manera general.

Ésta metodología consta de 5 fases principales:

- Planteamiento del problema.
- Análisis del problema.
- Diseño de la solución y construcción del prototipo.
- Prueba y refinamiento del prototipo.
- Operación.

Es importante aclarar que los pasos indicados están basados en metodologías que contemplan tanto la solución software como la solución hardware de sistemas de PDS.

2.1.1. Planteamiento del Problema.

La primer etapa es la más importante de la metodología puesto que es donde se define el problema y las necesidades de su solución. Se identifican los requerimientos mínimos iniciales resultados de un primer análisis del problema. Se debe tener cuidado en la clara y correcta definición del problema para no resolver aspectos que no sean parte de él, lo cual conlleve al fracaso.

2.1.2. Análisis del Problema.

Una vez definido el problema se deben analizar las ventajas y desventajas de las posibles soluciones. Las fases del análisis del problema son:

Investigación. La actividad de mayor importancia en la cual se recopila la información del problema en cuestión.

Requerimientos. Debido a que se trata de sistemas de PDS, los requerimientos se relacionan con las señales de entrada y salida del sistema, el tipo y modo de procesamiento de dichas señales, el rendimiento del sistema como la capacidad de cálculo, la velocidad de cálculo, velocidad de transferencia de datos, memoria y ancho de banda entre otras. Otro requerimiento es el alcance del sistema, por lo tanto existen diferentes soluciones para los diferentes tipos de sistemas. Las soluciones pueden ser algorítmicas, de software, de microcódigo y de hardware.

Redacción de la especificación Es la producción de un documento con los requerimientos del sistema. Deberá incluir toda la información para el entendimiento, manejo y control del sistema con lo que se podrá resolver el problema en cuestión. Puede contener diagramas de flujo y/o diagramas de bloques para visualizar las fases del sistema.

2.1.3. Diseño de la Solución y Construcción del Prototipo.

La solución puede ser llevada hasta la fase planteada, pudiendo llegar desde una solución algorítmica hasta una solución de hardware.

Con base en el documento de especificación y al alcance es posible llegar a una solución de software, que puede ser una simulación o un sistema terminado en hardware. A continuación se describen algunos tipos de diseños de soluciones.

Algorítmica. La solución plasmada en el documento se implementa en algún programa en software de alto nivel. La herramienta Matlab permite la simulación de los algoritmos para su validación y así saber si puede llegar a ser la solución del problema específico.

Software. Consiste básicamente en la codificación o transformación de la algorítmica a un lenguaje de alto nivel más robusto.

Microcódigo. Está relacionada directamente con el hardware en el que se requiere construir el sistema. El lenguaje es de bajo nivel propios de los procesadores.

Hardware. Es el conjunto de las soluciones antes propuestas. Permite tanto implementar o adecuar sistemas con un procesador de propósito general o el diseño de una arquitectura específica para la resolución del problema planteado.

La implementación de cualquiera de las soluciones descritas es llamado prototipo. Una vez funcionando el prototipo son validados y verificados los resultados esperados con respecto a las especificaciones y requerimientos a través de entradas del sistema planteado en el problema.

2.1.4. Prueba y Refinamiento del Prototipo.

La prueba y validación del prototipo provee de seguridad al usuario, debido a que los resultados deberán ser reproducidos las veces que sea necesario obteniendo los mismos resultados. Cualquier error detectado en las pruebas será corregido de manera inmediata.

2.1.5. Operación.

La última fase de la metodología en la cual el prototipo deja de serlo para convertirse en un sistema de producción. Contempla la documentación y el mantenimiento del sistema con el objetivo de que el usuario final pueda utilizarlo.

Con base en ésta metodología se realizará un estudio comparativo de diferentes algoritmos de la transformada de Fourier rápida para la elección de uno de ellos e implementarlo en el multiplexado OFDM utilizado en la capa física de las redes de comunicación inalámbrica WiMAX.

Capítulo **3**

Arquitectura de las Redes de Comunicación Inalámbrica WiMAX.

La red inalámbrica de Interoperabilidad Mundial para Acceso por Microondas (*Worldwide Interoperability for Microwave Access, WiMAX*) está diseñada con las ventajas de las redes de banda ancha fija y móvil sin la necesidad de linea-de-vista directa (*direct line-of-sight LOS*) con la estación base. La banda ancha móvil ofrece portabilidad, sistema nómada y movilidad. Que un sistema sea nómada implica la habilidad de conectarse a la red de diferentes lugares via diferentes bases; movilidad implica la habilidad de seguir con la conexión activa mientras se desplaza a cierta velocidad en un vehículo.

WiMAX está basada en el estándar de la red inalámbrica de área metropolitana (*WMAN*) desarrollado por el grupo IEEE 802.16 y adoptado por el WiMAX Forum y el ETSI (*European Telecommunications Standards Institute*) HIPERMAN (*high-performance metropolitan area network*).

El IEEE es un organismo de normalización y opera en una función puramente técnica. El estándar IEEE 802.16 sólo define la Capa 1 -Capa Física- y la Capa 2 -Capa de Acceso MAC-. También incluye definiciones y descripciones de características obligatorias y opcionales.

WiMAX Forum es una organización sin fines de lucro dedicada a promover el uso del estándar IEEE 802.16. El WiMAX Forum es responsable de la certificación de interoperabilidad de los equipos de proveedores y laboratorios de prueba que operan en todo el mundo. También define los perfiles del sistema que definen el conjunto de características de los equipos WiMAX .

A continuación veremos un poco de los antecedentes de ésta tecnología.

3.1. Antecedentes.

El grupo de la IEEE 802.16 fue formado en 1998 para desarrollar un estándar de banda ancha inalámbrica. El estándar 802.16 original fue completado en diciembre del 2001 y estaba enfocado en un sistema inalámbrico, punto-a-multipunto con linea-de-vista (*LOS*) directa con la estación base con operación en la banda de 10GHz a 66GHz. Muchos de sus conceptos eran una adaptación a inalámbrico del estándar del modem cableado DOCSIS (*data over cable service interface specification*).

Posteriormente el grupo de la IEEE produjo el estándar 802.16a, el cual fue una corrección al anterior. Incluyó aplicaciones sin-linea-de-vista (*non-line-of-sight NLOS*) en la banda de 2GHz a 11GHz, usando una capa física basada en el multiplexado con división en frecuencia ortogonal (*OFDM*). También se incluyó en la capa de acceso soporte al acceso múltiple por división de frecuencia ortogonal (*OFDMA*). Nuevas revisiones dieron como resultado un nuevo estándar en el 2004, llamado 802.16-2004, el cual reemplazó todas las versiones anteriores y formó la base para WiMAX . Esta primera versión de WiMAX era sólo para aplicaciones fijas, se le conoce como WiMAX fijo.

En diciembre del 2005, la IEEE terminó y aprobó el estándar IEEE 802.16e-2005, el cual agrega la capacidad de movilidad. El IEEE 802.16e-2005 representa la base de WiMAX para aplicaciones móviles y nómadas, se le conoce como WiMAX móvil.

Con los estándares IEEE 802.16 se tiene una variedad de opciones de diseño. Esto se debe a que los estándares fueron desarrollados para resolver varias aplicaciones y varios escenarios, por lo que ofrece una gran variedad de diseños para ser escogidos por los diseñadores de los sistemas. Por razones de interoperabilidad entre los sistemas, el alcance del estándar necesita ser reducido y así tener un menor número de diseños a escoger para implementar. El WiMAX Forum realiza esto al definir un número limitado de perfiles del sistema y perfiles de certificación.

Un *perfil del sistema* define el subconjunto de características obligatorias y opcionales de las capas física y de acceso seleccionadas por el WiMAX Forum del estándar IEEE 802.16-2004 o del IEEE 802.16e-2005. Actualmente el WiMAX Forum cuenta con 2 perfiles de sistema: uno basado en el estándar IEEE 802.16-2004, con capa física basada en OFDM, llamado perfil del sistema fijo; el otro está basado en el estándar IEEE 802.16e-2005, con capa física basada en OFDMA escalable, llamado perfil del sistema móvil. Un *perfil de certificación* se define como una creación de instancias particulares de un perfil del sistema en donde la frecuencia de operación, el ancho de banda del canal, y el modo de duplexado se especifican.[3]

3.2. Características principales de WiMAX.

WiMAX es una alternativa de red inalámbrica de banda ancha que ofrece características muy flexibles en términos de opciones de desarrollo y posibles ofertas de servicios. Algunas de las características mas destacadas son:

Capa física basada en OFDM La capa física de WiMAX (*PHY*) está basada en el multiplexado con división en frecuencia ortogonal, un sistema que ofrece buena resistencia al Multipath y permite a WiMAX operar en condiciones de NLOS (*non-line-of-sight*). OFDM ahora es ampliamente reconocido como el método a elegir para la mitigación del multitrayecto de la banda ancha inalámbrica.

Picos de muy alta velocidad de transmisión de datos WiMAX es capaz de soportar picos muy altos de velocidad de transmisión de datos. De hecho el pico de velocidad de transmisión puede llegar a los 74Mbps cuando se opere mediante un espectro amplio de 20MHz. Operando con un espectro de 10MHz con sistema TDD (*Time Division Duplexing*) con relación 3:1 en downlink-to-uplink (*velocidad de carga y descarga*), el pico de velocidad de transmisión de datos es aproximadamente 25Mbps y 6.7Mbps para la velocidad de descarga y de carga respectivamente. Estos picos son alcanzados cuando se usa una modulación 64 QAM con tasa de 5/6 de corrección de errores de codificación.

Ancho de banda y velocidad de transmisión de datos escalables WiMAX cuenta con una arquitectura de la capa física escalable que permite para la velocidad de transmisión escalarse fácilmente con canales de ancho de banda disponibles. Esta escalabilidad es soportada en el modo OFDMA, donde el tamaño de la FFT (*Transformada de Fourier rápida*) puede ser escalado basándose en la disponibilidad de canales de banda ancha. Por ejemplo, un sistema WiMAX móvil puede usar FFTs de 128, 512 o 1024-bit basado en si el ancho de banda del canal es 1.25MHz, 5Mhz o 10MHz respectivamente. Este escalamiento se puede hacer dinámicamente para admitir el acceso a deferentes redes que puedan tener diferentes asignaciones de ancho de banda.

Retransmisiones en la capa de enlace Para conexiones que requieran mayor confiabilidad, WiMAX admite pedidos automáticos de retransmisión (*ARQ*) en la capa de enlace. Condiciones ARQ permitidas requieren que cada paquete transmitido sea conocido por el receptor; paquetes desconocidos se asumen perdidos y son retransmitidos.

Admisión de TDD y FDD Los estándares 802.16-2004 y 802.16-2005 de la IEEE admiten tanto división de tiempo dúplex (*TDD*) como división de frecuencia dúplex (*FDD*), así como FDD unidireccional (*half-duplex FDD*). TDD es favorecido por la mayoría de las implementaciones por sus ventajas: flexibilidad en escoger los coeficientes de velocidad de transmisión de subida y bajada, capacidad para explotar la

reciprocidad del canal, capacidad de ser implementado en nonpaired spectrum y un diseño menos complejo del transmisor-receptor.

Acceso múltiple por divisor de frecuencia ortogonal (*OFDMA*) WiMAX móvil usa OFDM como técnica de acceso múltiple, por lo que diferentes usuarios pueden ser asignados a diferentes subconjuntos de tonos de la OFDM. OFDMA facilita la explotación de la diversidad de frecuencias y la diversidad de multiusuario para mejorar significativamente la capacidad del sistema.

Asignación flexible y dinámica de los recursos por el usuario La asignación de la velocidad de descarga y de carga están controladas por un planificador en la estación base. La capacidad es compartida entre los multiples usuarios en base a la demanda, usando un sistema de modulado por división de tiempo (*TDM*). Cuando es usado el modo OFDMA-PHY, el multiplexado es realizado adicionalmente en el dominio de la frecuencia, asignando diferentes subconjuntos de subportadoras de OFDM a diferentes usuarios. Los recursos también pueden ser asignados en el dominio espacial cuando es usado el sistema avanzado de antenas (*advance antenna system, AAS*) opcional. El estándar permite que los recursos de banda ancha sean asignados en tiempo, frecuencia y espacio, y tiene un mecanismo flexible para transmitir la información de la asignación de los recursos.

Admisión de técnicas avanzadas de antenas WiMAX tiene un número de anzuelos construidos en el diseño de la capa física, que admiten el uso de técnicas de multiples antenas, como Beamforming, Multiplexado Espacial y codificación en espacio-tiempo. Estos sistemas pueden ser usados para mejorar la capacidad del sistema y la eficiencia espectral mediante el despliegue de múltiples antenas en el transmisor y/o el receptor.

Calidad de servicio (QoS) La capa MAC de WiMAX tiene una arquitectura orientada a la conexión diseñada para admitir una variedad de aplicaciones, incluyendo voz y servicios multimedia. El sistema ofrece soporte a la velocidad de bit constante, velocidad de bit variable, flujos de tráfico en tiempo real y no en tiempo real, además de el mejor esfuerzo en tráfico de datos. WiMAX MAC está diseñado para admitir un gran número de usuarios, con múltiples conexiones por terminal, cada una con sus requerimientos de Calidad de Servicio (*Quality of Service, QoS*).

Seguridad robusta WiMAX admite cifrado robusto, usando Estándar de cifrado avanzada (AES) y tiene una privacidad robusta y protocolo de manejador de llaves. El sistema también ofrece una arquitectura muy flexible de autenticación basada en el protocolo extensible de autenticación (*EAP*), el cual permite una variedad de credenciales de usuario, incluyendo nombre de usuaro/clave, certificados digitales y tarjetas inteligentes.

Admisión de movilidad La variante móvil del sistema WiMAX tiene un mecanismo que admite entregas seguras sin fisuras para aplicaciones tolerantes de retardos y

completa movilidad, como VoIP. El sistema cuenta también con soporte para mecanismos de ahorro de energía que extiende la vida de la batería de dispositivos de mano suscritos. Mejoras en la capa física, como estimaciones mas frecuentes de canal, subcanalización del enlace de subida y control de potencia están especificadas en el soporte a aplicaciones móviles.

Arquitectura basada en IP El WiMAX Forum ha definido una arquitectura de red de referencia que esta basada en una plataforma totalmente IP. Todos los servicios end-to-end son entregados sobre una arquitectura IP confiando en protocolos basados en IP para el transporte de servicios de extremo a extremo, Calidad de Servicio, administración de sesiones, seguridad y movilidad. La confianza en IP permite a WiMAX facilitar la convergencia con otras redes y explotar la gran variedad de aplicaciones desarrolladas para IP.

3.3. Características de los estándares.

En la siguiente tabla se enlistan las principales características de los diferentes estándares publicados en la IEEE para el estándar IEEE 802.16 o WiMAX [1].

	802.16	802.16a	802.16e
Año de publicación	2001	2003	2005
Espectro	10 - 66 GHz	2 - 11 GHz	2 - 6 GHz
Funcionamiento	Sólo con vista directa (LOS)	Sin vista directa (NLOS)	Sin vista directa (NLOS)
Tasa de bits	32 - 134 Mb/s con canales de 28 MHz	75 Mb/s con canales de 20 MHz	15 Mb/s con canales de 5 MHz
Modulación	QPSK, 16QAM y 64QAM	OFDM con 256 subportadoras, QPSK, 16QAM y64QAM	OFDM con 256 subportadoras, QPSK, 16QAM y64QAM
Anchos de banda	20, 25 y 28 MHz	Seleccionables entre 1.25 y 20 MHz	Seleccionables entre 1.25 y 20 MHz
Radio de celda típico	2 - 5 km	5 - 10 km, alcance máximo 50 km	2 -5 km

3.4. Arquitectura de las Redes WiMAX.

En la figura 3.1 se muestra el modelo de referencia de las redes WiMAX desarrollada por el WiMAX Forum, que es una representación lógica de la arquitectura de la red.

[1] Todos los datos obtenidos de las referencias [4], [5] y [6]

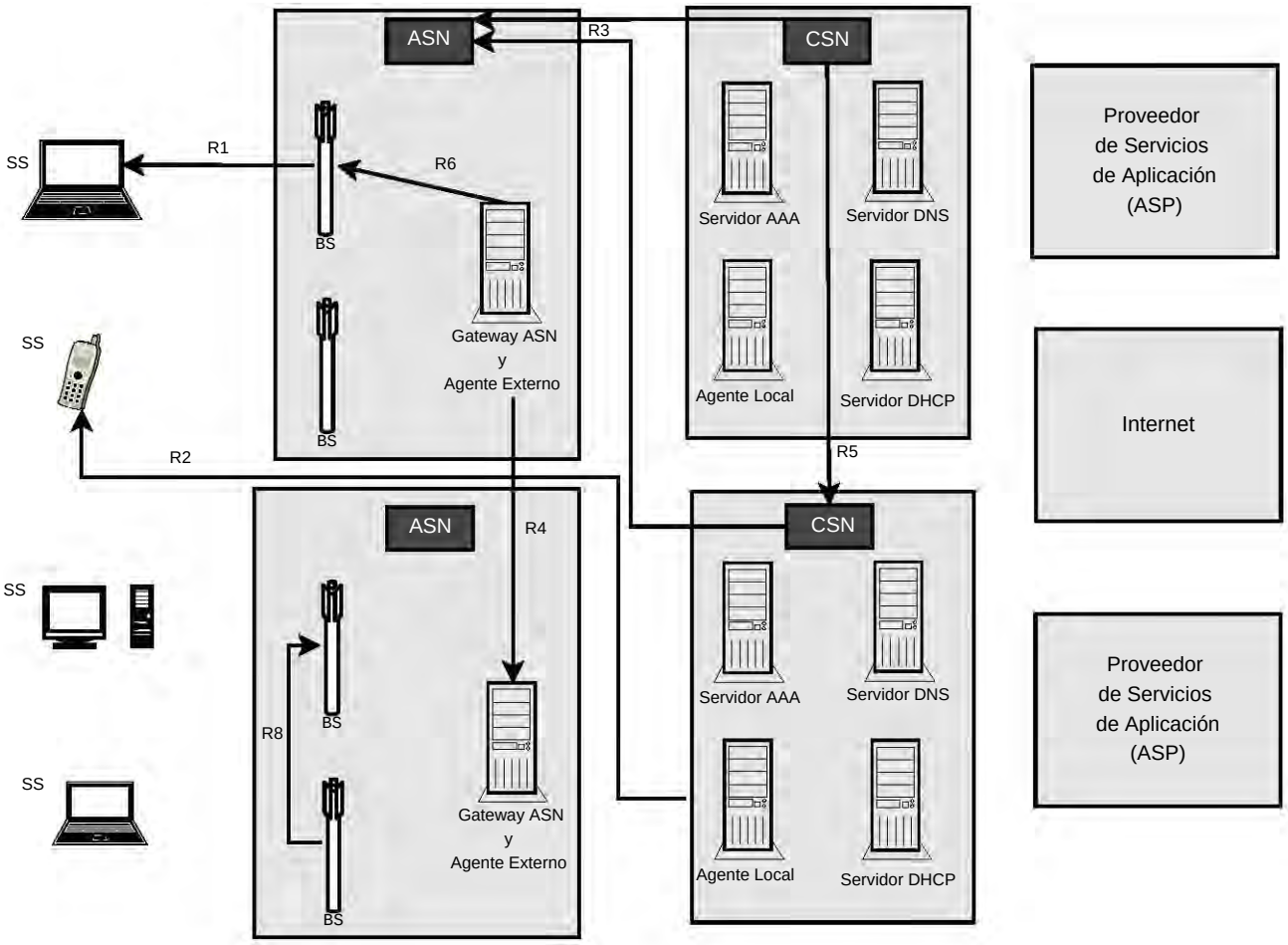


Figura 3.1: Modelo de referencia de las redes WiMAX.

La arquitectura de las redes WiMAX está basada en las redes IP, lo cual permite el uso de estándares existentes y se evita la creación de nuevos estándares.[7]

La arquitectura está compuesta principalmente por los siguientes elementos:

- Servicio de Acceso a la Red (*Access Service Network, ASN*)
  - Provee la interfaz inalámbrica que conecta la estación del suscriptor (*Suscriber Station, SS*) con la red.
  - Maneja la interfaz inalámbrica y abarca a las estaciones base (*Base Station, BS*).
  - Abarca al Gateway ASN el cual mantiene relación con la BS.
  - El Gateway ASN maneja la movilidad entre BS's.
  - La función del Agente Externo (*Foreign Agent, FA*) actúa en la autenticación y en las IP móviles.

- Servicio de Conectividad de la Red (Connectivity Service Network, CSN)
  - Provee de conectivadad entre las ASN e Internet o un proveedor de servicios de aplicación (Application Service Provider, ASP).
  - El Agente Locales (*Home Agent, HA*) y el Servidor AAA (Authentication, Authorization and Accounting, AAA) proveen la autenticación.
  - El Agente Local y el DHCP proveen el manejo de las direcciones IP.
  - El servidor AAA se encarga de los registros de facturación.
  - El Agente Local apoya en la movilidad.
- Interfaces Lógicas
  - Interfaces del estándar 802.16
    - Interfaz R1 - Conexión de SS a BS.
  - Interfaces del WiMAX Forum
    - Interfaz R2 - SS a HA, provee de roaming.
    - Interfaz R3 - ASN a CSN, provee autenticación, facturación y mensajes en IP Móvil (*Mobile IP, MIP*).
    - Interfaz R4 - ASN a ASN, define procedimientos de movilidad cuando un SS cruza de un ASN a otro.
    - Interfaz R5 - CSN a CSN, provee de roaming.
    - Interfaz R6 - BS a Gateway ASN, provee de mensajes de movilidad.
    - Interfaz R7 - Interno en el Gateway ASN.
    - Interfaz R8 - BS a BS, provee de handoff.

3.5. La Capa Física de WiMAX.

La capa física de WiMAX (*PHY*) esta basada en el estándar IEEE 802.16 y fue diseñada con mucha influencia de Wi-Fi. Al igual que Wi-Fi, WiMAX esta basada en los principios del multiplexado con división en frecuencia ortogonal (*OFDM*), la cual es una técnica de modulación y de acceso. El estándar IEEE 802.16 define cuatro diferentes tipos de capas físicas, de las cuales dos están basadas en OFDM. Éstas dos son:

- *WirelessMAN OFDM*, capa física basada en OFDM con una FFT de 256 puntos para operaciones punto-a-multipunto en condiciones NLOS en frecuencias entre 2GHz y 11GHz. Ésta PHY ha sido aceptada por WiMAX para operaciones en sistemas fijos.
- *WirelessMAN OFDMA*, PHY basada en SOFMDA (*OFDMA escalable*) con una FFT de 128, 512, 1024 ó 2048 puntos para operaciones punto-a-multipunto en condiciones NLOS en frecuencias entre 2GHz y 11GHz. El tamaño de la FFT variable

permite el óptimo desempeño del sistema en una amplia gama de anchos de banda de canal y condiciones del radio de cobertura. Ésta PHY fue aceptada por WiMAX para los sistemas móviles o portables.

A continuación se indicarán las diferentes etapas de la capa física de WiMAX . Las primeras etapas están relacionadas a la corrección de errores posteriores (*Forward Error correction, FEC*) lo cual incluye codificación de canal (*channel encoding*), rate matching, interleaving y mapeo de símbolos (*symbol mapping*). El siguiente conjunto de etapas está relacionado con la construcción del símbolo de OFDM en el dominio de la frecuencia. Durante esta etapa, los datos son mapeados a los subcanales y subportadoras apropiados. Símbolos pilotos son insertados en subportadoras piloto, lo que permite al receptor estimar y conocer la información del estadio del canal (*channel state information, CSI*). El último conjunto de funciones está relacionado a la conversión del símbolo de OFDM del dominio de la frecuencia al dominio del tiempo y eventualmente a una señal analógica que pueda ser transmitida por el aire. en la figura 3.2 se muestra un diagrama genérico de la capa física de un sistema WiMAX .

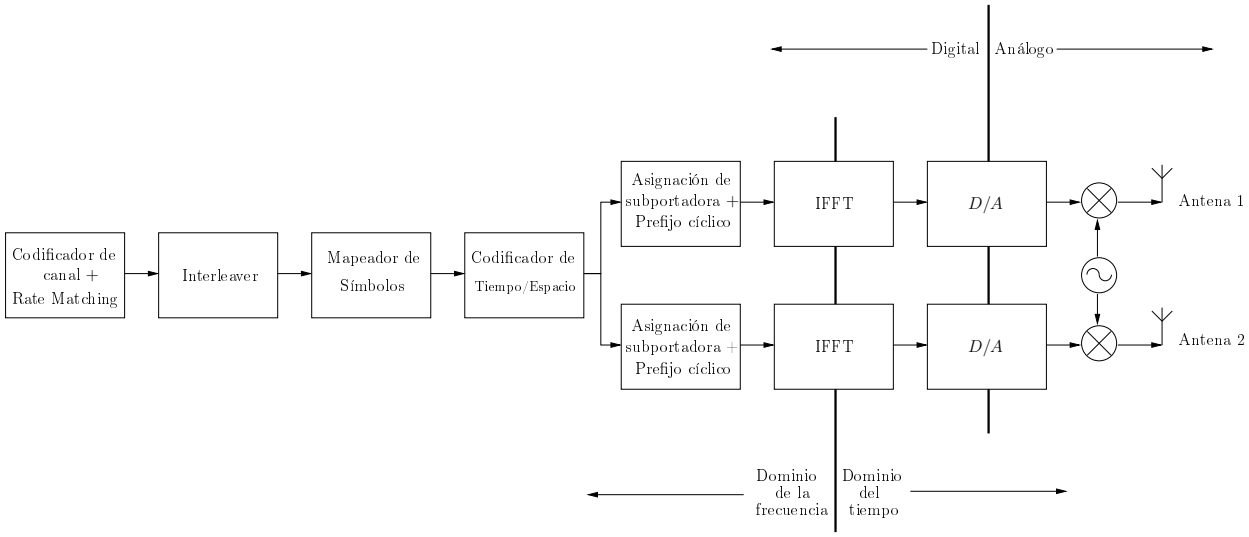


Figura 3.2: Diagrama de la Capa Física de WiMAX.

En el estándar IEEE 802.16 e-2005, la etapa de codificación de canal consiste en los siguientes pasos:

Aleatorización de datos. El propósito de esta etapa es de encriptar los datos y evitar que un receptor no autorizado decodifique los datos.

Codificación de canal. Se trata de una combinación de detección de errores y corrección de errores. Consiste en la división en un número entero de subcanales. Un subcanal

es la unidad básica de asignación de recursos en la capa física y comprende varios datos y subportadoras piloto.

Rate matching. Éste proceso se lleva a cabo para que el tamaño del bloque de datos concuerde con el establecido. Repetirá bits para incrementar la tasa o cortará los bits para disminuirla.

Hibrid Automatic Repeat Request (*HARQ*). Está diseñado para brindar corrección de errores y retransmisión de paquetes de datos a través de interfaces inalámbricas cuando el receptor lo requiera o detecte demasiados errores.

Interleaving (*intercalado*). Es el proceso mediante el cual una cadena de bits o símbolos de una Unidad de Datos de Protocolo (*Protocol Data Unit, PDU*) o de una Unidad de Datos de Servicio (*Service Data Unit, SDU*) puede ser propagada en un orden específico con bits o símbolos de otro PDU o SDU. Esta técnica es usada para reducir el efecto de errores en la transmisión.

En el interleaving los bits codificados son intercalados en un proceso de dos etapas. La primer etapa asegura que los bits adyacentes codificados son mapeados a subportadoras no adyacentes, lo que proporciona diversidad de frecuencias y mejora el rendimiento del decodificador. La segunda etapa asegura que los bits adyacentes son mepeados alternadamente a bits menos y más significativos de la constelación de la modulación.

En la siguiente etapa se realiza el mapeo de símbolos, la secuencia de bits binarios es convertida en símbolos de valores complejos. Las constelaciones de modulación definidas en el estándar son QPSK y 16QAM, con 64QAM opcional. Aunque la modulación 64QAM es opcional, la mayoría de los sistemas WiMAX prefieren implementarla, por lo menos para el downlink.

El siguiente bloque en la capa física de WiMAX es la codificación en tiempo y espacio. Existen varias opciones de codificación en espacio/tiempo, hay esquemas con dos, tres y cuatro antenas que pueden ser usados y están definidos en el estándar IEEE 802.16 e-2005. Los más comunes de implementar son los esquemas con dos antenas.

Los siguientes bloques en la capa física de WiMAX corresponden a la estructura de un sistema basado en OFDM .

3.6. OFDM.

La capa física de WiMAX está basada en el multiplexado con división en frecuencia ortogonal. OFDM es el esquema de transmisión elegido para habilitar las comunicaciones de datos a alta velocidad y multimedia. Es usado por una gran variedad de sistemas

de banda ancha comerciales, incluyendo DSL, Wi-Fi, DVB-H (*Digital Video Broadcast-Handheld*) al igual que WiMAX.

En un sistema OFDM una secuencia de símbolos a alta velocidad se divide en multiples secuencias de baja velocidad paralelas, cada una de ellas es usada para modular un tono o subportadora ortogonal. La señal de banda base transmitida, que es un ensamble de las señales en todas las subportadoras, puede ser representada como:

$$x(t) = \sum_{i=0}^{L-1} s[i] e^{-2\pi j(\Delta f + iB_c)t} \quad 0 \leq t \leq T,$$

donde $s[i]$ es el símbolo transportado en la i-esima subportadora; B_c es separación en frecuencia entre dos subportadoras adyacentes, también es conocido como el ancho de banda de las subportadoras; Δf es la frecuencia de la primer subportadora; y T es la duración total de los símbolos utilizables. En el receptor, un símbolo enviado en una subportadora especifica es recuperado mediante la integración de la señal recibida con un complejo conjugado de la señal tono durante todo T. Si la sincronización en tiempo y frecuencia entre el transmisor y el receptor es perfecta, la ortogonalidad entre las subportadoras se mantiene en el receptor.

El concepto de modulación de forma independiente de múltiples tonos de frecuencia ortogonal con símbolos de banda estrecha es equivalente a primero construir la señal completa de OFDM en el dominio de la frecuencia y después utilizar la transformada de Fourier rápida inversa (*IFFT*) para convertir la señal al dominio del tiempo. El método de la IFFT es mas fácil de implementar, como no requiere de multiples osciladores para transmitir y recibir la señal de OFDM . En el dominio de la frecuencia, cada símbolo OFDM es creado mapeando la secuencia de símbolos en las subportadoras. WiMAX tiene tres clases de subportadoras.

1. Subportadoras de datos, son usadas para transportar símbolos de datos.
2. Subportadoras piloto, son usadas para transportar símbolos piloto. Los símbolos piloto son conocidos a priori y pueden ser usados para estimación de canal y rastreo de canal.
3. Subportadoras nulas, no tienen potencia, incluye las subportadoras de DC y las subportadoras de guarda.

Para poder crear el símbolo de OFDM en el dominio de la frecuencia, los símbolos modulados son mapeados en los subcanales que han sido asignados para la transmisión del bloque de datos. Un subcanal, como esta definido en el estándar IEEE 802.16 e-2005, es un conjunto de subportadoras. Es importante darse cuenta que en WiMAX las subportadoras que constituyen un subcanal pueden ser adyacentes entre ellas o estar distribuidas en toda la banda de frecuencia.

Capítulo 4

Multiplexado con División en Frecuencia Ortogonal (OFDM).

El multiplexado con división en frecuencia ortogonal (*OFDM*) es una técnica de modulación que divide un canal de comunicaciones en una serie de bandas de frecuencias equidistantes. Una subportadora que contiene una porción de la información es transmitida en cada banda. Cada subportadora es ortogonal a todas las demás subportadoras, diferenciando a OFDM de los multiplexados con división en frecuencia más comunes.[8]

4.1. Definición y Estructura de OFDM.

OFDM es una técnica de modulación con multiples portadoras, la cual ha sido adoptada por sistemas de comunicación con alta velocidad de transferencia de datos. La popularidad de OFDM por su alta velocidad de transferencia de datos se debe principalmente a su eficiencia y su manejo flexible de la interferencia intersímbolos (*InterSymbol Interference, ISI*) en canales altamente dispersivos.[9]

Mientras la propagación del retraso del canal τ se convierte en un múltiplo cada vez mayor del tiempo T_s, la ISI se hace más severa. Por definición, un sistema con alta velocidad de transferencia de datos generalmente tiene $\tau \gg T_s$, debido a que el número de símbolos enviados por segundo es alto. En un sistema NLOS, como WiMAX , en el cual se va a transmitir en una distancia de moderada a larga, la propagación del retraso también frecuentemente será grande. Los sistemas inalámbricos de banda ancha de todos los tipos sufrirán de ISI severo y por lo tanto requerirán técnicas de transmisión y/o recepción que superen la ISI. Aunque el estándar IEEE 802.16 incluye técnicas de modulación de una portadora, la mayoría de los sistemas usan OFDM , el cual ha sido seleccionado como preferido por el WiMAX Forum.

4.1.1. Modulación con Multiples Portadoras.

Con el fin de tener un canal sin ISI, el símbolo del tiempo T debe ser mas grande que la propagación del retraso del canal τ. Los sistemas de comunicación digitales no pueden operar si ISI está presente; mientras T se aproxima a τ, la tasa de errores de bits se hace intolerable.

A fin de superar este problema, la modulación con multiples portadoras divide la alta tasa de bits a transmitir en L subcadenas de baja tasa, cada una de las cuales tiene $T_s/L \gg \tau$ y, por lo tanto, son libres de ISI. Estas subcadenas pueden ser enviadas sobre L subcanales paralelos, manteniendo el total de la tasa de datos deseada. Normalmente, los subcanales son ortogonales bajo la condición de propagación ideal, por lo que la modulación con multiples portadoras es referida como OFDM . La tasa de datos en cada uno de los subcanales es mucho menor que la tasa total, por lo que el ancho de banda del subcanal es mucho menor que el ancho de banda total del sistema. La ISI en cada subcanal es pequeña, por otra parte, en la implementación digital de OFDM la ISI puede ser completamente eliminada con el uso del prefijo cíclico.

La modulación con multiples portadoras divide la cadena de datos de banda ancha en L subcadenas de banda angosta, después cada una de ellas es transmitida sobre un diferente subcanal de frecuencia ortogonal. El número L de subcadenas se escoge para hacer que el símbolo del tiempo en cada subcadena sea mucho mayor que la propagación del retraso del canal o, lo que es equivalente, hacer el ancho de banda del subcanal menor que el ancho de banda del canal. Esto asegura que las subcadenas no experimentarán ISI significante.

Esta técnica de modulación tiene interpretaciones interesantes en los dominios del tiempo y de la frecuencia. En el dominio del tiempo, la duración del símbolo en cada subportadora se incrementa a $T = LT_s$, por lo que hacer que L sea grande asegura que la duración del símbolo excede el retraso en la propagación del canal, $T \gg \tau$, que es un requerimiento para una comunicación libre de ISI. En el dominio de la frecuencia, las subportadoras tiene un ancho de banda $B/L \ll B_c$, lo que asegura desvanecimiento plano (*flat fading*), el equivalente a la comunicación libre de ISI en el dominio de la frecuencia.

Aunque este simple tipo de modulación con multiples portadoras es fácil de entender, tiene varios defectos fundamentales. Primero, en una implementación realista, una penalización en el ancho de banda por que las subportadoras no pueden tener la forma de pulsos rectangulares perfectos y están limitadas en tiempo. Además filtros paso bajas de muy alta calidad serán requeridos para mantener la ortogonalidad de las subportadoras en el receptor. Y más importante, este esquema requiere L unidades de RF independientes y trayectorias de demodulación. A continuación se presentará como OFDM supera estos defectos.[10]

4.1.2. Las bases de OFDM.

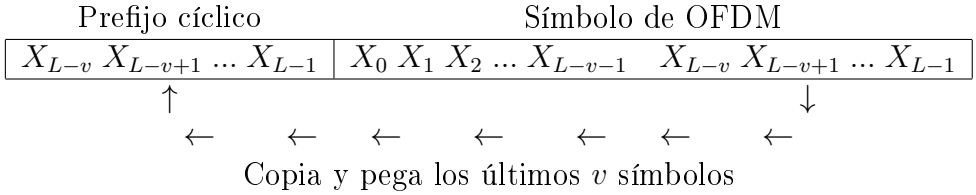
Para superar la necesidad de L unidades de RF en el transmisor y en el receptor, OFDM usa una técnica computacional efectiva, la transformada de Fourier rápida (FFT). La FFT y su inversa, la IFFT, pueden crear múltiples subportadoras ortogonales utilizando una sola unidad de RF.

Se inicia agrupando L símbolos de datos en un bloque conocido como el símbolo de OFDM (*OFDM symbol*). Un símbolo de OFDM tiene una duración de T segundos, donde $T = LT_s$. Para mantener cada símbolo de OFDM independiente de los otros que irán a través del canal inalámbrico, es necesario introducir un tiempo de guarda entre los símbolos de OFDM :

Símbolo de OFDM	Guarda	Símbolo de OFDM	Guarda	Símbolo de OFDM

De esta manera, después de recibir una serie de símbolos de OFDM , mientras el tiempo de guarda T_g sea más largo que la propagación del retraso del canal τ, cada símbolo de OFDM sólo interferirá con él mismo.

La clave para hacer OFDM realizable en la practica es el uso del algoritmo FFT. Para que la IFFT/FFT puedan crear un canal libre de ISI, el canal debe proveer una convolución circular. Al añadir un prefijo cíclico a la señal transmitida, como se muestra a continuación, se crea una señal $x[n]_L$, y por lo tanto $y[n] = x[n] \circledast h[n]$.



Esto es, si la propagación del retraso del canal máxima tiene una duración de $v + 1$ muestras, añadir una banda de guarda de al menos v muestras entre símbolos de OFDM hace cada símbolo de OFDM independiente de los que vienen después y antes de él y así sólo un símbolo de OFDM puede ser considerado.

El prefijo cíclico viene con penalizaciones tanto en ancho de banda como en potencia. Puesto que v símbolos redundantes son enviados, los requerimientos de banda ancha se incrementan. De la misma manera, v símbolos adicionales deben ser contabilizados para su transmisión y así consumir más potencia.

Para poder estimar los símbolos recibidos, la ganancia compleja del canal de cada subportadora debe ser conocida, se debe conocer la amplitud y la fase de la subportadora.

Para técnicas de modulación, como QPSK, que no usan la amplitud para transmitir información, sólo la información de la fase será suficiente.

Después de que se realiza la FFT, los símbolos de datos son estimados usando un ecualizador en el dominio de la frecuencia (*Frequency Domain Equalizer ,FEQ*) de la siguiente manera:

$$\hat{X}_1 = \frac{Y_1}{H_1} ,$$

donde H_1 es la respuesta compleja del canal a una determinada frecuencia y por lo tanto corrige la fase y ecualiza la amplitud.[3]

En la figura 4.1 se muestra un diagrama de bloques de un sistema de comunicaciones basado en OFDM .

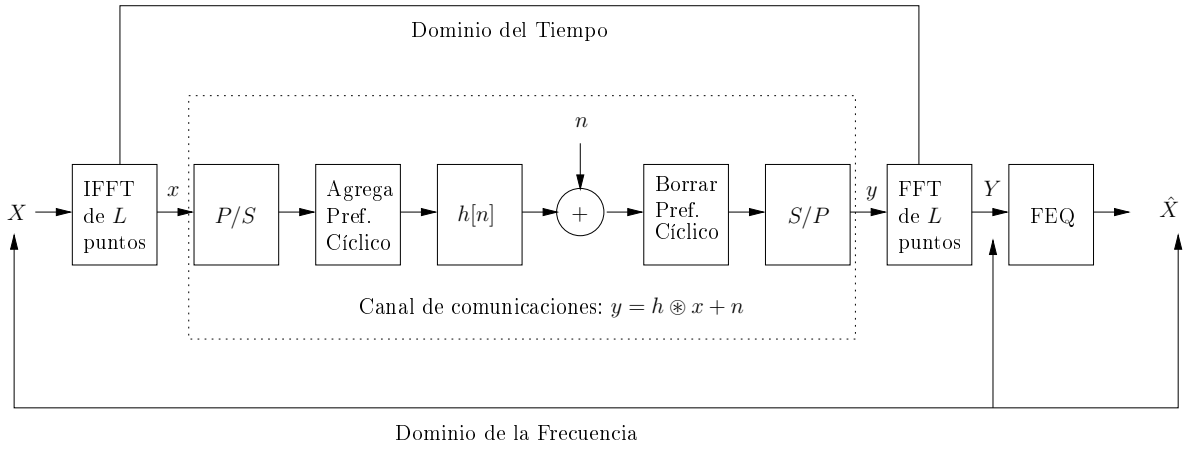


Figura 4.1: Diagrama de bloques de un sistema OFDM

En OFDM la codificación y decodificación de la señal son realizadas en el dominio de la frecuencia, donde X, Y y $\hat{X}$ contienen los L símbolos de datos transmitidos, recibidos y estimados. A continuación se recapitularán los pasos clave de OFDM .

El primer paso es dividir la señal de banda ancha con ancho de banda B en L señales de banda angosta -subportadoras-, cada una de las cuales tiene un ancho de banda B/L. De esta manera la tasa de símbolos se mantiene pero cada subportadora experimenta una comunicación libre de ISI, mientras que se use un prefijo cíclico que exceda la propagación del retraso. Las L subportadoras en un símbolo de OFDM son representadas por un vector X.

Para poder usar sólo un dispositivo de RF de banda ancha en lugar de L de banda angosta, las subportadoras son moduladas usando la IFFT.

Para que la IFFT/FFT pueda separar un canal con ISI en subportadoras ortogonales es necesario agregar un prefijo cíclico de longitud v después de la IFFT. Los resultantes $L + v$ símbolos son enviados en serie a través de un canal de banda ancha.

En el receptor, el prefijo cíclico es descartado y los L símbolos recibidos son demodulados, usando una FFT, los cual resulta en L símbolos de datos con la forma $Y_l = H_l X_l + N_l$ provenientes de la subportadora l.

Ahora cada subportadora puede ser ecualizada por un FEQ simplemente dividiendo entre la ganancia compleja del canal $H[i]$ para esa subportadora. Esto da como resultado $\hat{X}_l = X_l + N_l/H_l$.

Para esta descripción se ha asumido que el transmisor y el receptor se encuentran perfectamente sincronizados y que el receptor conoce perfectamente el canal para poder realizar el FEQ.

En la tabla 4.1 se enlistan los principales parámetros de OFDM y los valores más ocupados para los sistemas WiMAX móviles.

Símbolo	Descripción	Relación	Valor en WiMAX
B^*	Ancho de banda nominal	$B = 1/T_s$	$10\,MHz$
L^*	Número de subportadoras	Tamaño de la IFFT/FFT	1024
G^*	Fracción de guarda	% de L para el prefijo cíclico	1/8
L_d^*	Subportadoras de datos	L subportadoras piloto/nulas	768
T_s	Tiempo de muestreo	$T_s = 1/B$	$1\,\mu seg.$
N_g	Símbolos de guarda	$N_g = GL$	128
T_g	Tiempo de guarda	$T_g = T_s N_g$	$12.8\,\mu seg.$
T	Tiempo del símbolo de OFDM	$T = T_s(L + N_g)$	$115.2\,\mu seg.$
B_{sc}	Ancho de banda de subportadora	$B_{sc} = B/L$	$9.76\,KHz$

* Son parámetros para WiMAX establecidos; los demás parámetros de OFDM pueden ser calculados con estos valores.

Tabla 4.1: Principales parámetros de OFDM para WiMAX.

4.1.3. Espectro del Multiplexado con División en Frecuencia.

En el Multiplexado con División en Frecuencia (*Frequency Division Multiplexing, FDM*) convencional, cada frecuencia de portadora esta separada por una banda de guarda para evitar interferencia. Las frecuencias en la banda de guarda no pueden ser utilizadas

para mandar información. En la figura 4.2 se muestra el espectro de una modulación FDM.

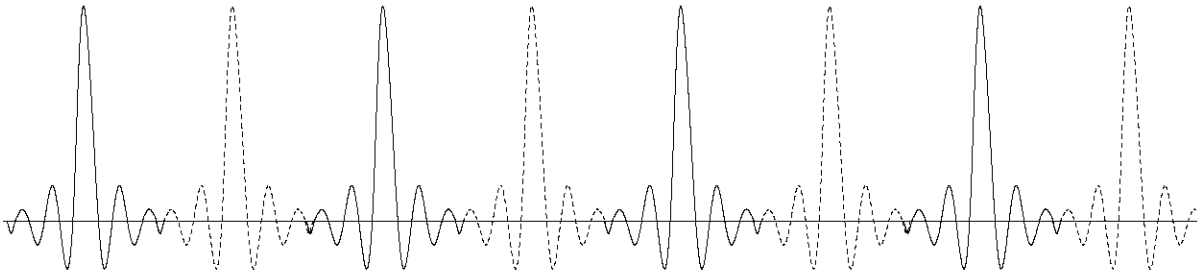


Figura 4.2: Espectro de la modulación FDM.

En OFDM las subportadoras tienen una respuesta en frecuencia con forma de sinc $(sen(x)/x)$ resultando en un traslape en el dominio de la frecuencia. Este traslape no causa interferencia por la ortogonalidad de las subportadoras. En la figura 4.3 se muestra el espectro de OFDM.

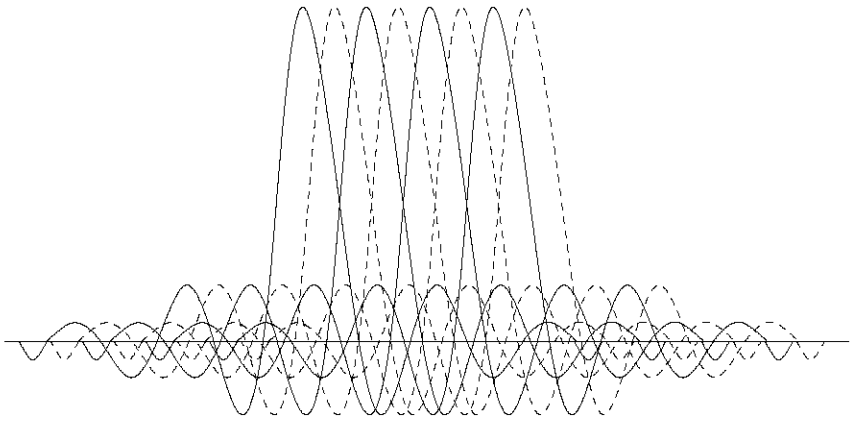


Figura 4.3: Espectro de la modulación OFDM.

El receptor de OFDM usa una FFT sincronizada en tiempo y frecuencia para transformar el símbolo de OFDM del dominio del tiempo al de la frecuencia. En este proceso la FFT toma muestras de las frecuencias, que corresponden a los picos de las subportadoras. A cada una de estas frecuencias, las demás subportadoras pasan por cero eliminando la interferencia entre ellas.

Se puede observar que uno de los pasos más importantes para la realización de la modulación OFDM es la aplicación de la IFFT/FFT, puesto que simplifica de gran manera su implementación.

4.2. La Transformada de Fourier Rápida en OFDM.

La implementación de un sistema OFDM es posible cuando la complejidad del equipo y el consumo de energía son reducidos enormemente al utilizar la FFT en tiempo real para reemplazar el banco de (de)moduladores necesario para cada una de las subportadoras.

Para procesamiento en tiempo real, la FFT requiere de $F(log_2 N)$ operaciones aritméticas por ciclo de muestreo, donde N es la longitud de la transformada. Se puede lograr el procesamiento en tiempo real de alta velocidad de dos diferentes maneras. De manera convencional, mediante un procesador de propósito general, un solo procesador que realice la transformada con una frecuencia de reloj muy alta, la cual es $F(log_2 N)$ veces la frecuencia de muestreo. La otra manera es utilizar una aplicación específica, mediante procesadores en paralelo o pipeline, con frecuencias de reloj cercanas o equivalentes a la frecuencia de muestreo. Ésta solución ha resultado ser la más preferida cuando el consumo de energía está limitado por el ambiente en el que se encuentra la aplicación, como son las comunicaciones móviles.

Existen varias arquitecturas que han sido propuestas con diferentes enfoques. Las arquitecturas de pipeline tienen las ventajas del paralelismo, por lo que usualmente son muy rápidas pero no son tan flexibles y requieren de mayor complejidad en hardware.[11] Las arquitecturas de estos procesadores requieren unidades en donde se realizan las estructuras de mariposa de la transformada rápida de Fourier.

A continuación se mencionan algunas arquitecturas de pipeline:

Radix-2 Multi-path Delay Commutator (R2MDC). Es una implementación del algoritmo FFT Radix-2. La secuencia de entrada se divide en 2 cadenas de datos paralelas, las cuales, con la distancia correcta entre elementos, entran a las mariposas. Se requieren $log_2 N - 2$ multiplicadores, $log_2 N$ mariposas Radix-2 y $3/2N - 2$ registros.

Radix-2 Single-path Delay Feedback (R2SDF). Usa el registro más eficientemente al guardar la salida de la mariposa en el registro de realimentación. Una sola cadena de datos pasan por los elementos en cada etapa. Tiene el mismo número de elementos que la arquitectura R2MDC pero con menos registros, $N - 1$

Radix-4 Single-path Delay Feedback (R4SDF). Una versión del R2SDF con el algoritmo FFT Radix-4. Guarda 3 de los 4 valores resultantes par la realimentación. Requiere $log_4 N - 1$ multiplicadores, $los_4 N$ mariposas completas de Radix-4 y $N - 1$ registros.

Radix-4 Multi-path Delay Commutator (R4MDF). Es la versión Radix-4 del R2MDF. Se pueden procesar 4 FFTs simultáneamente. Requiere de $3log_4 N$ multiplicadores, log_4 mariposas Radix-4 y $5/2N - 4$ registros.

Radix-4 Single-path Delay Commutator (R4SDC). Utiliza 1/4 de la mariposa Radix-4, un multiplexor en el conmutador de retardos que reduce la memoria requerida a

$2N - 2$, manteniendo el número de componentes igual que el R4MDF.

Las arquitecturas con delay-feedback son más eficientes que las que utilizan delay-commutator con respecto a la utilización de la memoria. Las arquitecturas basadas en el algoritmo Radix-4 utilizan más el multiplicador, las basadas en el Radix-2 tienen mariposas más simples.[12]

Los principales algoritmos de la FFT ocupados en estas arquitecturas, como se puede observar, son el Radix-2 y el Radix-4, así como derivados de estos mismos. En el siguiente capitulo se describen estos dos algoritmos de la FFT.

Capítulo 5

Adecuación e Implementación de los Algoritmos de la Transformada de Fourier Rápida.

La esencia de la transformada de Fourier de una señal con forma de onda es descomponerla o separarla en una suma de sinusoidales de diferentes frecuencias. Si la suma de estas sinusoidales da como resultado la forma de onda original, se ha determinado la Transformada de Fourier de la onda. La representación gráfica de la Transformada de Fourier es un diagrama en el que se muestra la amplitud y la frecuencia de cada una de las sinusoidales.[13]

La Transformada de Fourier Discreta (*Discrete Fourier Transform, DFT*) puede ser empleada para obtener resultados esencialmente equivalentes a los de la transformada de Fourier continua. La DFT es una de las operaciones más importantes en el procesamiento digital de señales. Su implementación usando el algoritmo de la Transformada de Fourier Rápida (*Fast Fourier Transform, FFT*) ha hecho posible su uso en aplicaciones para procesamiento de señales en tiempo real.

La DFT de una señal $x(k)$ en tiempo discreto se define como

$$X(n) = \sum_{k=0}^{N-1} x(k) W_N^{nk} \qquad n = 0, 1, ..., N-1 \tag{5.1}$$

donde $W_N^{nk} = e^{-j2\pi nk/N}$ constituye las funciones complejas base o factores de fase de la DFT.

Para definir el factor de fase se puede tomar una secuencia de $N = 8$ muestras y así representarlo como un vector en el círculo unitario.

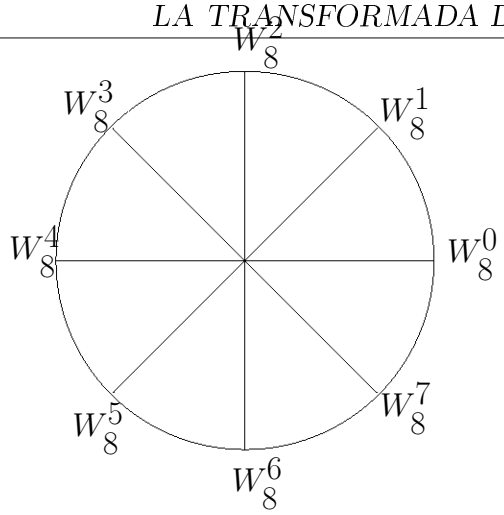


Figura 5.1: Representación en círculo unitario del factor de fase.

Se puede apreciar en la figura 5.1 que los vectores son:

1. Periódicos
2. Simétricos
3. Igualmente espaciados al rededor del círculo con espacios iguales a ΔW

Donde

$$\Delta W = \frac{2\pi}{N} = \frac{\Omega}{N}$$

Ω Es la frecuencia de muestreo, N es el número de muestras y ΔW es llamado frecuencia de resolución o espacios de salida de la DFT.

El mapeo del factor de fase en el círculo unitario es periódico. La frecuencia normalizada de muestreo es 2π, entonces la DFT resulta periódica con respecto a la frecuencia de muestreo.

El factor de fase es inversamente simétrico con respecto al origen.

$$W_8^1 = -W_8^5 \quad ; \quad W_8^2 = -W_8^6$$

Esto quiere decir que solamente la primera mitad del factor de fase contiene toda la información necesaria.[14]

Dado $X(n)$ que es una secuencia de valores frecuenciales, la Transformada de Fourier Discreta Inversa (*Inverse Discrete Fourier Transform, IDFT*) da la secuencia temporal

$$x(k) = \frac{1}{N} \sum_{n=0}^{N-1} X(n) W_N^{-nk} \tag{5.2}$$

La forma de la ecuación de la IDFT es idéntica a la forma de la DFT salvo el factor de normalización $1/N$ y el signo del exponente del factor de fase.

5.1. La Transformada de Fourier Rápida.

Casi siempre aplicaciones de análisis espectral requieren de DFTs en tiempo real para muestras de entrada continuas. Al ver la importancia de la DFT en varias aplicaciones del procesamiento digital de señales, como filtrado lineal, análisis de correlación y análisis espectral, su calculo eficiente es un tema que ha recibido considerable atención por matemáticos, ingenieros y científicos especializados.

La FFT es un algoritmo rápido para la implementación eficiente de la DFT, en donde un número N de muestras temporales se transforman en N muestras frecuenciales.

El calculo de la DFT para N muestras de entrada requiere N^2 multiplicaciones complejas y $N^2 - N$ sumas complejas. Lo cual significa un número elevado de operaciones, así como de datos almacenados. Los algoritmos de la FFT factoriza una DFT con gran cantidad de puntos en varias DFTs con pocos puntos para si poder reducir el número de operaciones.

Existen dos familias principales:

- Radix
- Algoritmos de Factor Primo (*Prime Factor Algorithm, PFA*)

Por ser los más utilizados en los sistemas de modulación OFDM para redes de comunicación inalámbrica WiMAX se revisarán los algoritmos de la familia Radix.

5.2. Radix-2.

5.2.1. Decimación en el tiempo (DIT).

El algoritmo Radix-2 con decimación en el tiempo (*Decimation In Time, DIT*) es el más simple y la forma más común del algoritmo, también es llamado algoritmo Cooley-Tukey. El algoritmo Radix-2 DIT divide la DFT de longitud N en dos DFTs de longitud $N/2$.

El algoritmo Radix-2 DIT primero calcula la transformada de Fourier de las componentes pares $x(2k)$ y de las componentes impares $x(2k+1)$, después combina los dos resultados para producir la transformada de Fourier de la secuencia completa. Al dividir

la ecuación (5.1) en componentes pares e impares se obtiene:

$$X(n) = \sum_{k=0}^{(N/2)-1} x(2k)W_N^{2kn} + \sum_{k=0}^{(N/2)-1} x(2k+1)W_N^{(2k+1)n}$$

Si $x_1(k) = x(2k)$ y $x_2(k) = x(2k+1)$ donde $k = 0, 1, ..., (N/2)-1$ se puede reescribir la ecuación anterior como:

$$X(n) = \sum_{k=0}^{(N/2)-1} x_1(k)W_{N/2}^{kn} + W_N^n \sum_{k=0}^{(N/2)-1} x_2(k)W_{N/2}^{kn}$$

donde $W_N^{2k} = W_{N/2}^k$. En su forma general quedaría de la siguiente manera:

$$\begin{aligned} X(n) &= X_1(n) + W_N^n X_2(n) \\ X(n+N/2) &= X_1(n) - W_N^n X_2(n) \end{aligned} \tag{5.3}$$

La ecuación (5.3) se conoce como la mariposa de la FFT Radix-2 DIT[15]. En la figura 5.2 se muestra gráficamente la mariposa.

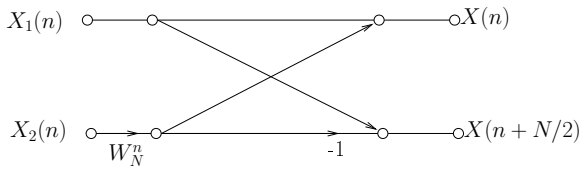


Figura 5.2: Representación de la mariposa del algoritmo de la FFT Radix-2 DIT.

De esta primera descomposición se obtienen 2 DFTs de longitud $N/2$. Los datos de entrada a la mariposa se encuentran separados por $N/2$ muestras y los exponentes de los factores de fase son consecutivos. Si a cada una de las DFTs de longitud $N/2$ se le aplica la misma descomposición obtenemos de cada una de ellas 2 DFTs de longitud $N/4$. Los datos de entrada ahora se encuentran separados por $N/4$ muestras y los exponentes de los factores de fase se separan por un factor de 2.

El proceso de descomposición se repite hasta generar DFTs de 2 puntos. Cada descomposición es conocida como etapa y el número total de etapas está dado por

$$M = log_2 N$$

De esta manera, una DFT de 16 muestras requiere de cuatro etapas. En la figura 5.3 se muestra la gráfica de flujo del algoritmo FFT Radix-2 para una señal de 16 muestras.

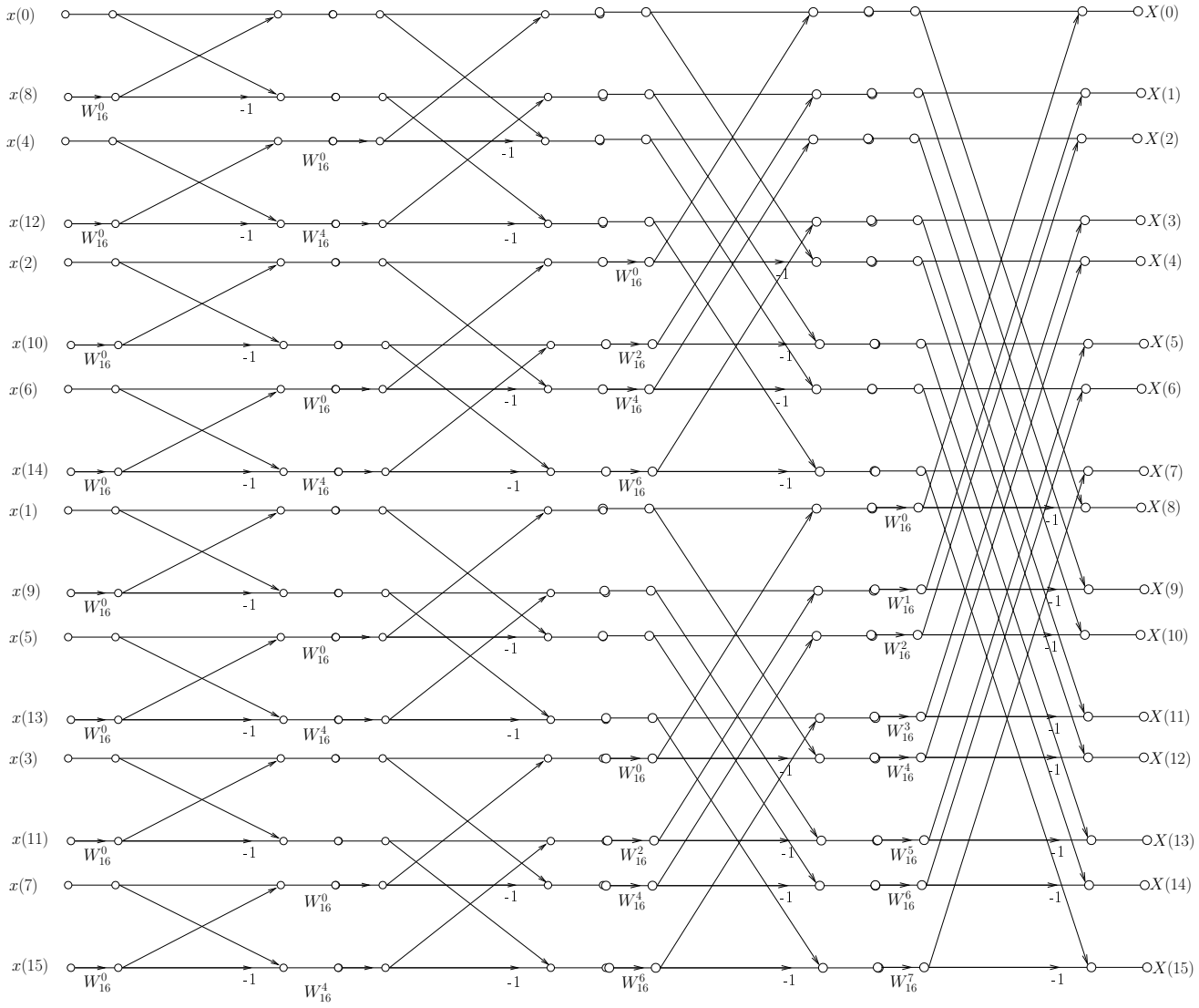


Figura 5.3: Gráfica de flujo del algoritmo FFT Radix-2 DIT para una señal con 16 muestras.

Se puede observar en la figura 5.3 que el algoritmo requiere que la señal de entrada este guardada en orden de bit inverso (*bit reversal order*) en localidades de memoria continuos. Para lograr éste orden en la señal de entrada cada uno de los indices decimales de las muestras temporales son convertidos a su representación binaria. La cadena binaria se invierte y convirtiendo la nueva cadena binaria a decimal da el orden de bit inverso en

los indices de las muestras temporales. En la tabla 5.1 se muestra este proceso para una señal temporal con 16 muestras.

Orden original		Orden de bit inverso	
Decimal	Binario	Binario	Decimal
0	0000	0000	0
1	0001	1000	8
2	0010	0100	4
3	0011	1100	12
4	0100	0010	2
5	0101	1010	10
6	0110	0110	6
7	0111	1110	14
8	1000	0001	1
9	1001	1001	9
10	1010	0101	5
11	1011	1101	13
12	1100	0011	3
13	1101	0111	11
14	1110	0111	7
15	1111	1111	15

Tabla 5.1: Proceso del ordenamiento de una señal con 16 muestras por bit inverso.

Para el cálculo del algoritmo FFT Radix-2 DIT se requieren $\frac{N}{2}log_2N$ multiplicaciones complejas y $Nlog_2N$ sumas complejas, lo cual se puede observar que es mucho menor que los cálculos requeridos para la DFT.

5.2.2. Decimación en Frecuencia (DIF).

El desarrollo del algoritmo de la FFT Radix-2 DIF es muy similar al desarrollo del algoritmo FFT Radix-2 DIT. De la misma manera, los datos se separan en grupos de $N/2$ muestras de datos continuos. $X(n)$ se puede expresar como:

$$X(n) = \sum_{k=0}^{(N/2)-1} x(k)W_N^{nk} + \sum_{k=0}^{(N/2)-1} x(k+(N/2))W_N^{n(k+N/2)}$$

A continuación se factoriza el término W_N^{nk} y se separan $X(n)$ en muestras pares e impares quedando de la siguiente manera:

$$
\begin{aligned}
X(2n) &= \sum_{k=0}^{(N/2)-1} [x(k) + x(k + (N/2))] W_{N/2}^{nk} \\
X(2n+1) &= \sum_{k=0}^{(N/2)-1} [x(k) - x(k + (N/2))] W_N^k W_{N/2}^{nk}
\end{aligned} \tag{5.4}
$$

La ecuación 5.4 es la mariposa Radix-2 DIF y en la figura 5.4 se muestra gráficamente.

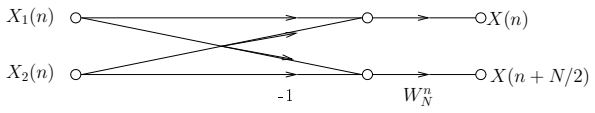


Figura 5.4: Representación de la mariposa del algoritmo FFT Radix-2 DIF.

De nuevo, la descomposición se realiza hasta que en la última etapa tenemos DFTs de dos puntos. En la figura 5.5 se tiene una gráfica de flujo del algoritmo FFT Radix-2 DIF para una señal con 16 muestras.

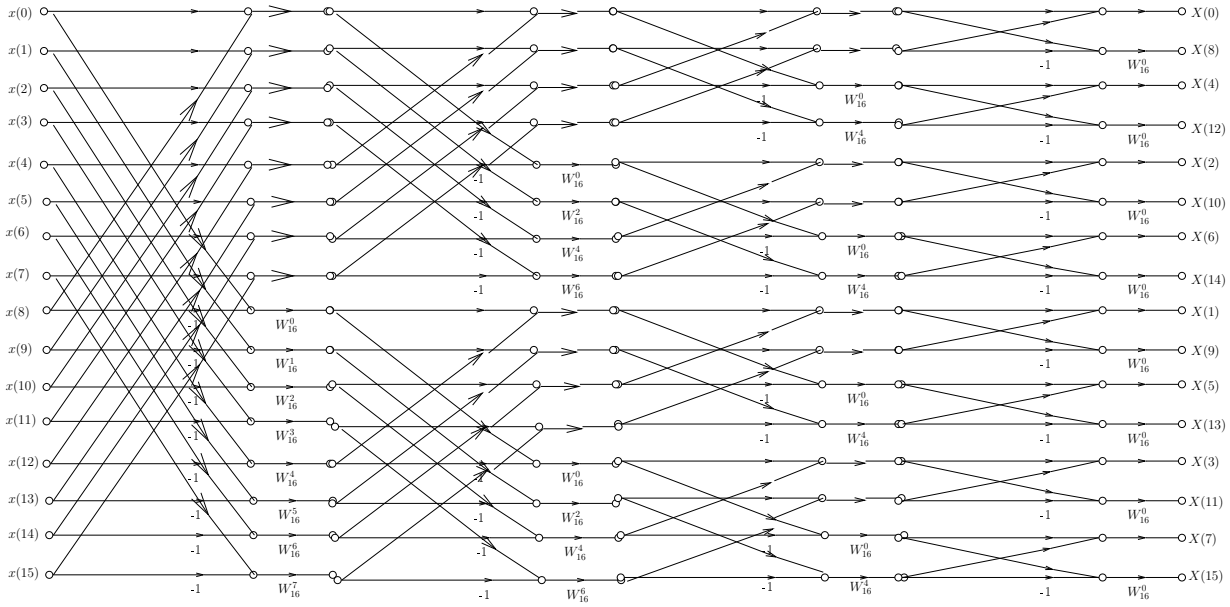


Figura 5.5: Gráfica de flujo del algoritmo FFT Radix-2 DIF para una señal con 16 muestras.

Se puede observar que la descomposición es de izquierda a derecha y las relaciones de simetría se invierten con respecto al algoritmo DIT, por lo que el reordenamiento de la

señal se realiza a la salida, en la frecuencia en lugar de en el tiempo.

Los cálculos requeridos en este algoritmo son los mismos que en el algoritmo DIT.

5.3. Radix-4.

5.3.1. DIT y DIF.

Los algoritmos FFT Radix-4 con decimación en el tiempo y decimación en frecuencia son más rápidos que los radix-2 puesto que reutilizan los resultados intermedios para el cálculo de la DFT. Esto se logra al descomponer la DFT en cuatro partes de longitud $N/4$. La ecuación (5.1) queda expresada de la siguiente manera:

$$X(n) = \sum_{k=0}^{(N/4)-1} x(k)W_N^{kn} + \sum_{k=N/4}^{(N/2)-1} x(k)W_N^{kn} + \sum_{k=N/2}^{(3N/4)-1} x(k)W_N^{kn} + \sum_{k=3N/4}^{N-1} x(k)W_N^{kn}$$

Se puede reescribir de la siguiente manera:

$$\begin{aligned} X(n) = & \sum_{k=0}^{(N/4)-1} x(k)W_N^{kn} + W_N^{Nn/4} \sum_{k=0}^{(N/4)-1} x\left(k+\frac{N}{4}\right) W_N^{kn} \\ & + W_N^{Nn/2} \sum_{k=0}^{(N/4)-1} x\left(k+\frac{N}{2}\right) W_N^{kn} + W_N^{3Nn/4} \sum_{k=0}^{(N/4)-1} x\left(k+\frac{3N}{4}\right) W_N^{kn} \end{aligned}$$

De la definición del factor de fase tenemos

$$W_N^{Nn/4} = (-j)^n \qquad W_N^{Nn/2} = (-1)^n \qquad W_N^{3Nn/4} = (j)^n$$

Por lo que

$$X(n) = \sum_{k=0}^{(N/4)-1} \left[x(k) + (-j)^n x\left(k+\frac{N}{4}\right) + (-1)^n x\left(k+\frac{N}{2}\right) + (j)^n x\left(k+\frac{3N}{4}\right) \right] W_N^{nk}$$

Si esta expresión se divide en 4 secuencias de $N/4$ puntos se obtiene la ecuación de la mariposa del algoritmo FFT Radix-4.

$$
\begin{aligned}
X(4h) &= \sum_{k=0}^{(N/4)-1} \left[x(k) + nx\left(k+\frac{N}{4}\right) + x\left(k+\frac{N}{2}\right) + x\left(k+\frac{3N}{4}\right) \right] W_{N/4}^{nk} \quad (5.5) \\
X(4n+1) &= \sum_{k=0}^{(N/4)-1} \left[x(k) - jx\left(k+\frac{N}{4}\right) - x\left(k+\frac{N}{2}\right) + jx\left(k+\frac{3N}{4}\right) \right] W_N^k W_{N/4}^{nk} \\
X(4n+2) &= \sum_{k=0}^{(N/4)-1} \left[x(k) - x\left(k+\frac{N}{4}\right) + x\left(k+\frac{N}{2}\right) - x\left(k+\frac{3N}{4}\right) \right] W_N^{2k} W_{N/4}^{nk} \\
X(4n+3) &= \sum_{k=0}^{(N/4)-1} \left[x(k) + jx\left(k+\frac{N}{4}\right) - x\left(k+\frac{N}{2}\right) - jx\left(k+\frac{3N}{4}\right) \right] W_N^{3k} W_{N/4}^{nk}
\end{aligned}
$$

En la figura 5.6 se muestra la estructura de mariposa del algoritmo FFT Radix-4 DIT [16].

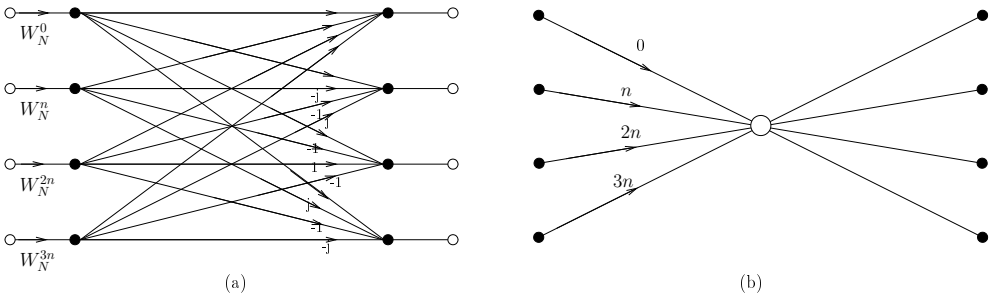


Figura 5.6: Representación de la mariposa del algoritmo FFT Radix-4. (a) Forma completa (b) Forma simplificada

Esta descomposición se realiza hasta tener DFTs de 4 puntos. El número de etapas para este algoritmo esta dado por:

$$M = \frac{log(N)}{log(4)}$$

donde N es la longitud de la señal de entrada. Para poder aplicar el algoritmo Radix-4 N debe ser múltiplo de 4.

En la figura 5.7 se muestra la gráfica de flujo del algoritmo FFT Radix-4 DIT para una señal con $N = 16$ muestras.

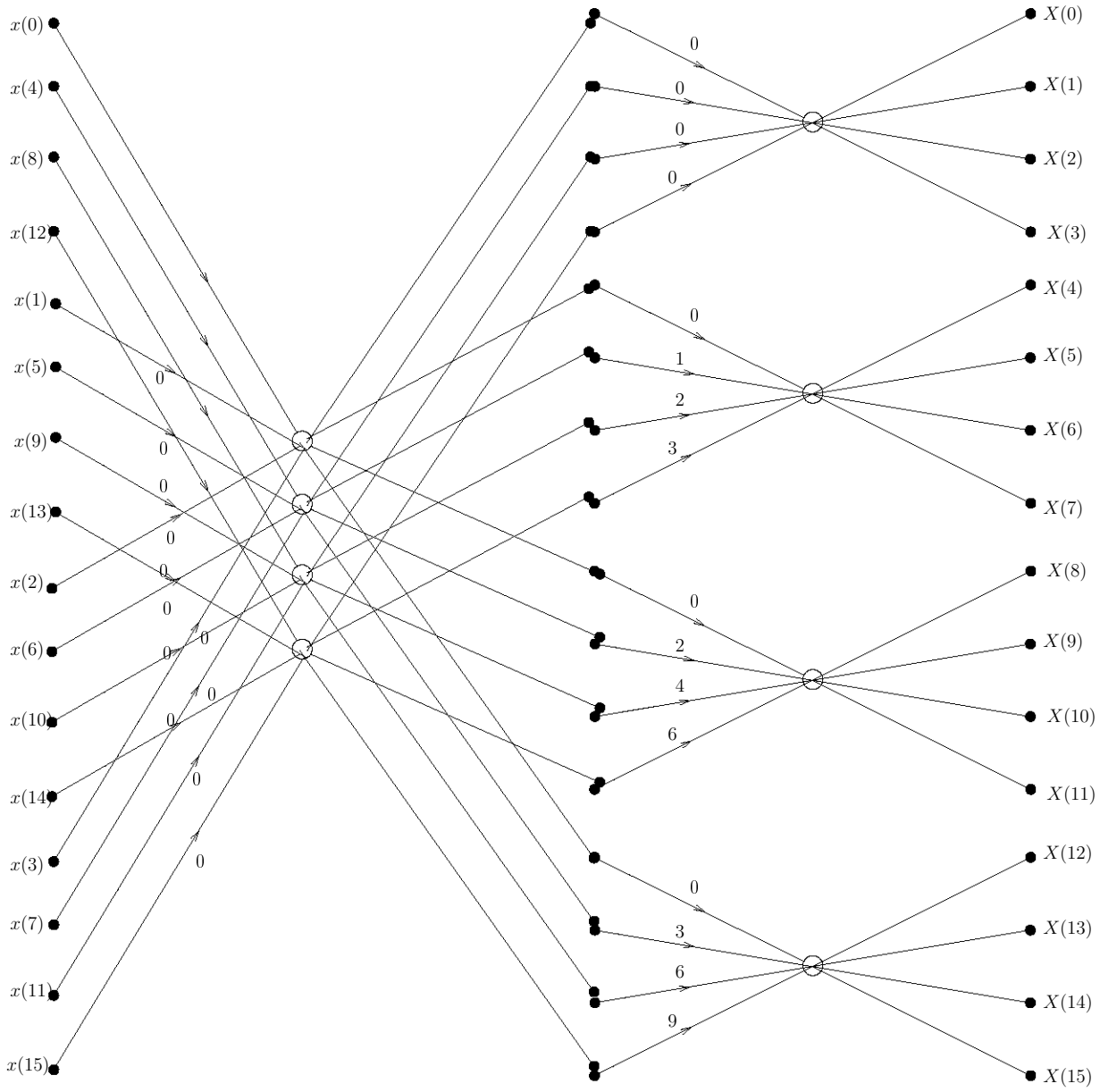


Figura 5.7: Gráfica de flujo del algoritmo FFT Radix-4 DIT.

El orden de los datos de entrada están reordenados por medio del orden de dígito inverso (*reverse digit order*), el cual consiste en convertir el índice temporal de forma decimal a su forma en base 4 y, al igual que en el orden de bit inverso, invertir la cadena y así obtener el nuevo orden de la señal. En la tabla 5.2 se muestra este proceso para una señal temporal con 16 muestras.

Orden original		Orden de dígito inverso	
Decimal	Base 4	Base 4	Decimal
0	00	00	0
1	01	10	4
2	02	20	8
3	03	30	12
4	10	01	1
5	11	11	5
6	12	21	9
7	13	31	13
8	20	02	2
9	21	12	6
10	22	22	10
11	23	32	14
12	30	03	3
13	31	13	7
14	32	23	11
15	33	33	15

Tabla 5.2: Proceso del ordenamiento de una señal con 16 muestras por dígito inverso base 4.

El número de operaciones para estos algoritmos son:
$\frac{3}{8}Nlog_2N$ multiplicaciones complejas (% 25 menos que en la FFT Radix-2)
$Nlog_2N$ sumas complejas (igual que en la FFT Radix-2)

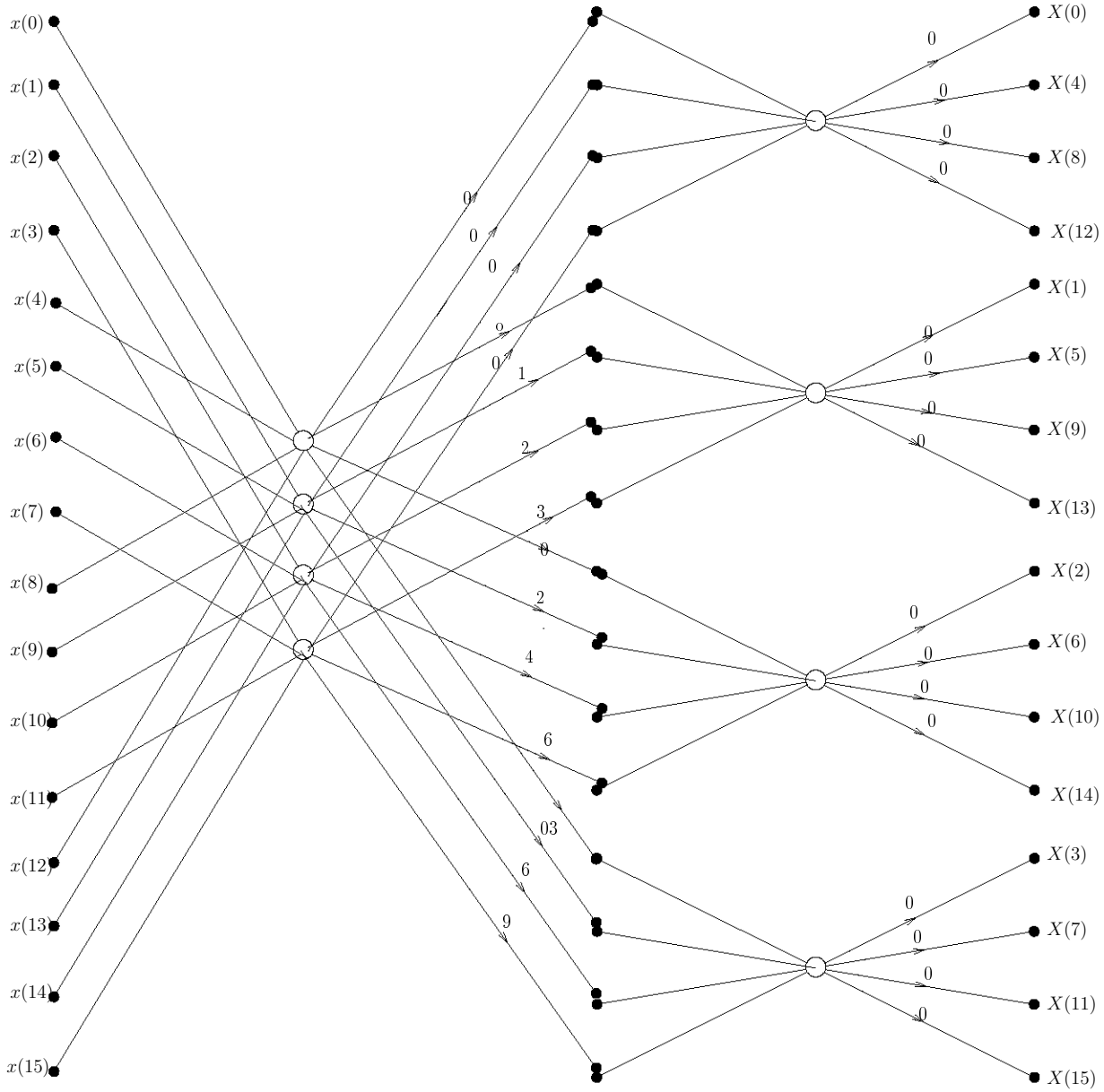


Figura 5.8: Gráfica de flujo del algoritmo FFT Radix-4 DIF.

En la figura 5.8 se muestra la gráfica de flujo del algoritmo FFT Radix-4 DIF para una señal con 16 muestras. Al igual que en el algoritmo FFT Radix-2 DIF, la diferencia con el DIT es que la entrada se realiza de manera ordenada y la salida se tiene que reordenar por dígito inverso base 4.

5.4. Complejidad Aritmética de los Algoritmos FFT en Tiempo Real.

Las aplicaciones de procesamiento digital de señales requieren que se realice la FFT en tiempo real para una cadena de datos N con una tasa de entrada de datos F [15]. En la tabla 5.3 se muestra un comparativo de los requerimientos en cuanto a multiplicaciones complejas por segundo (*complex multiplies per second, CMPS*) y sumas complejas por segundo (*complex additions per second, CAPS*) requieren tanto la DFT así como los algoritmos FFT Radix-2 y FFT Radix-4.

	DFT	FFT Radix-2	FFT Radix-4
CMPS	NF	$\frac{F}{2}log_2N$	$\frac{3F}{8}log_2N$
CAPS	$(N-1)F$	$Flog_2N$	$Flog_2N$

Tabla 5.3: Número de operaciones necesarias para el cálculo de la DFT y los diferentes algoritmos para ser implementadas en tiempo real.

Para WiMAX $F = floor(n \cdot BW/8000) * 8000$
Donde $n = 8/7$;
$BW = 10\ MHZ$
$\therefore F = 11'424,000[s/s]$

En la tabla 5.4 se muestran el número de cálculos aproximados necesarios para los requerimientos de WiMAX en tiempo real y en la tabla 5.5 se muestran el porcentaje de ahorro en el número de cálculos dentro de las FFTs.

	DFT		Radix-2		Radix-4	
N	CMPS	CAPS	CMPS	CAPS	CMPS	CAPS
128	1462272000	1450848000	39984000	79968000	-	-
256	2924544000	2913120000	45696000	91392000	34272000	91392000
512	5849088000	5837664000	51408000	102816000	-	-
1024	11698176000	11686752000	57120000	114240000	42840000	114240000
2048	23396352000	23384928000	62832000	125664000	-	-

Tabla 5.4: Número de cálculos necesarios para la FFT en OFDM WiMAX .

	Radix-2		Radix-4	
N	% CMPS	% CAPS	% CMPS	% CAPS
128	97.27	94.49	-	-
256	98.44	96.86	98.83	96.86
512	99.12	98.24	-	-
1024	99.51	99.02	99.63	99.02
2048	99.73	99.46	-	-

Tabla 5.5: Porcentaje de ahorro en el número de cálculos necesarios para la FFT en OFDM WiMAX .

Se puede observar que el número de cálculos requerido por los algoritmos de la FFT son mucho menores que los requeridos por la DFT. También se puede observar que el algoritmo FFT Radix-4 requiere menor número de multiplicaciones complejas para el cálculo de la transformada.

Una vez introducidos los cuatro algoritmos de la FFT de nuestro interés, serán implementados en el programa Matlab para así poder compararlos y posteriormente concluir cuál de ellos es el más apto para su implementación en un coprocesador en la cadena de la modulación OFDM de la capa física de WiMAX.

5.5. Programación de los algoritmos FFT.

5.5.1. FFT Radix-2 DIT.

En la siguiente sección se describe la programación del algoritmo Radix-2 DIT para el cálculo de la transformada de Fourier rápida.

5.5.1.1. Programación.

Para la programación del algoritmo Radix-2 fue necesario programar la estructura de la mariposa. Se debe tomar en cuenta cuales son los datos que van a entrar a la mariposa en cada etapa. El número de etapas se relaciona directamente con el número de datos que componen la señal de entrada, representados por N. La mariposa se realizara $N/2$ veces en cada una de las v etapas, en donde $v = log_2(N)$. N debe ser potencia de 2.

Los datos que se utilizan en la primera etapa deben pasar por el proceso de decimación del algoritmo, el cual consiste en separar la señal en 2 conjuntos de datos, los pares y los impares de acuerdo con su posición en el vector de datos. Este proceso se repite con cada conjunto hasta que se tengan N vectores con un dato cada uno.

En cada una de las etapas se utilizan diferentes factores de fase, los cuales se pueden volver a utilizar, por lo que se calculan todos los necesarios antes de entrar a la mariposa y al ser utilizados solamente serán llamados.

Los datos a la salida de la última etapa están en orden, por lo que no se debe realizar ningún acomodo. Estos datos representan la transformada de Fourier rápida de la señal de entrada.

En la figura 5.9 se presenta el diagrama de flujo ocupado para la programación del algoritmo Radix-2 DIT.

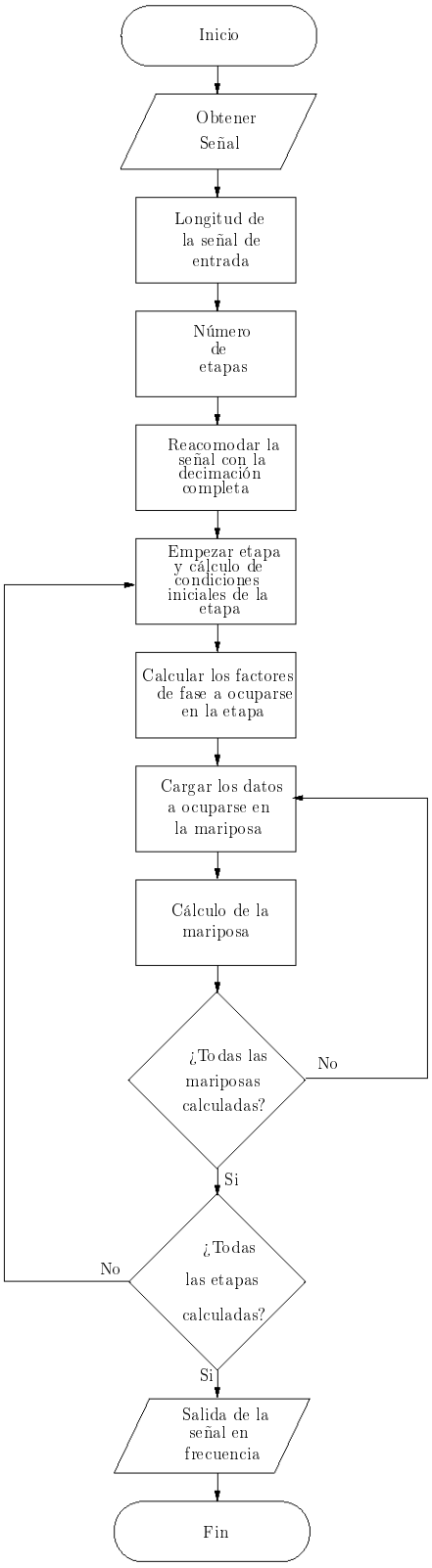


Figura 5.9: Diagrama de Flujo del Radix-2 DIT FFT.

Con base en este diagrama de flujo del algoritmo se programará la FFT Radix-2 DIT.

5.5.1.2. Programa.

El siguiente es el programa en Matlab de la FFT Radix-2 DIT.

```
    % Algoritmo Radix-2 DIT
function [X] = fftradix2dit(x)
N=length(x); % Long. de la señal de entrada.
v=log2(N); % Núm. de etapas.
N2=N/2;
X=bitrevorder(x); % Decimación de la señal.
for e=1:v, % Inicio de etapa.
  L=2^e; % Condiciones iniciales.
  n1=(2^(e-1))-1;
    r=2^(v-e);
    W=exp((-2*pi*i)*[0:n1]*r/N); % Cálculo de factores de fase.
    n2=1;
    for k=0:L:(N-L), % Inicio del cálculo de las mariposas.
        for n=0:(L/2)-1,
            a1=n+k+1;
            b1=(L/2)+a1;
            a=X(a1); % Carga valor 1.
            b=X(b1)*W(n2); % Carga valor 2 * factor de fase.
            X(a1) = a + b; % Salida 1 de la mariposa.
            X(b1) = a - b; % Salida 2 de la mariposa.
            n2=n2+1; % Proceso para el siguiente factor de fase.
            if n2>n1+1,
                n2=1;
            end
        end
    end % Fin de mariposas.
end % Fin de etapa.
end % Fin de función.
```

Lo primero que se hace es obtener la longitud de la señal, recordando que la única restricción para este tipo de algoritmos es que sea potencia de 2. Con la longitud se obtiene el número de etapas necesarias para la realización del algoritmo.

La función *bitrevorder(x)* de Matlab realiza la decimación de la señal. Ésta consiste en reacomodar la señal original de acuerdo con su posición en el vector de la señal original, se realiza al leer el número de la posición en binario de derecha a izquierda, por ejemplo $001 = 1$ se leerá $100 = 4$.

Las estructuras de mariposa se repiten $\frac{N}{2}$ veces cada una de las v etapas, por lo que se inicia el ciclo de cada una de las etapas. Dentro del ciclo se deben calcular las condiciones iniciales de cada etapa así como los factores de fase a utilizarse. También incluye otros

dos ciclos con los cuales se llama a los valores a utilizar en cada mariposa. Así como la variable que llama al factor de fase a utilizarse en cada mariposa.

Puesto que la salida de la etapa es la entrada de la siguiente, el resultado se guarda en una sola variable reemplazando los valores pasados por lo que se deben cargar los valores a ocuparse antes de entrar a la mariposa, asegurando así que utilizamos los valores correctos.

5.5.2. FFT Radix-2 DIF.

A continuación se describe la programación para el algoritmo Radix-2 DIF para el cálculo de la FFT.

5.5.2.1. Programación.

Una vez programado el algoritmo Radix-2 DIT es mucho mas sencillo programar el algoritmo Radix-2 DIF. La primera diferencia es el reacomodo de la señal de entrada, en la Decimación en Frecuencia el reacomodo se realiza al final del algoritmo. Como a la entrada del algoritmo se tiene la señal completa, los datos que entran a las mariposas son n y $n + N/2^e$ donde e es la etapa del algoritmo. La estructura de la mariposa es la misma que el Radix-2 DIT, la diferencia radica en los datos que entran a las mariposas en cada etapa y en donde se ocupa el factor de fase.

En la figura 5.10 se presenta el diagrama de flujo ocupado para la programación del algoritmo Radix-2 DIF.

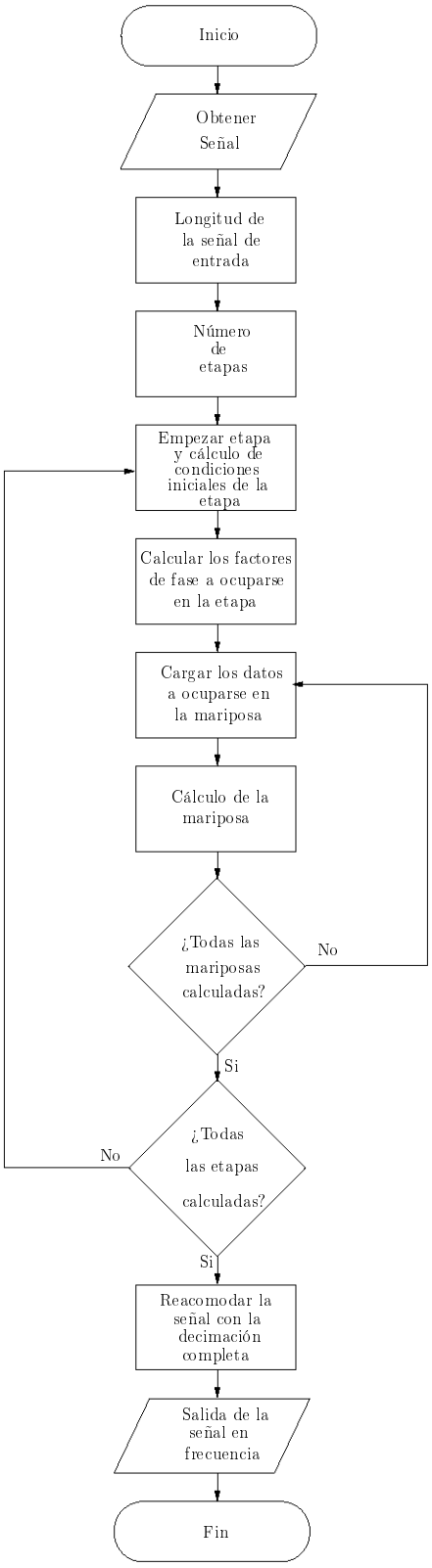


Figura 5.10: Diagrama de Flujo del Radix-2 DIF FFT.

5.5.2.2. Programa.

El siguiente es el programa en Matlab de la FFT Radix-2 DIF.

```
    % Algoritmo Radix-2 DIF
function [X] = fftradix2dif(x)
N=length(x); % Long. de la señal de entrada.
v=log2(N); % Núm. de etapas.
N2=N/2;
X=x; % Reasignación de la señal original.
for e=v:-1:1, % Inicio de etapa.
  L=2^e; % Condiciones iniciales.
  n1=(2^(e-1))-1;
  r=2^(v-e);
  W=exp((-2*pi*i)*[0:n1]*r/N); % Cálculo de factores de fase.
  n2=1;
  for k=0:L:(N-L), % Inicio del cálculo de las mariposas.
      for n=0:(L/2)-1,
          a1=n+k+1;
          b1=(L/2)+a1;
          a=X(a1); % Carga valor 1.
          b=X(b1); % Carga valor 2.
          X(a1) = a + b; % Salida 1 de la mariposa.
          X(b1) = (a - b) * W(n2); % Salida 2 de la mariposa.
          n2=n2+1; % Proceso para el sig. factor de fase.
          if n2>n1+1,
              n2=1;
          end
      end
  end % Fin de mariposas.
end % Fin de etapa.
X=bitrevorder(x); % Reacomodo de la señal de salida.
end % Fin de función.
```

En general, el programa es muy parecido al del Radix-2 cambiando en detalles mínimos. El primer cambio es el reacomodo de la señal de entrada, en la decimación en frecuencia se realiza al final de las etapas.

Una de las diferencias significativas de este algoritmo con el del Radix-2 DIT radica en cuales son los datos que entran a cada mariposa en las diferentes etapas. como ya se había mencionado, los datos que entran a cada mariposa son n y $n + N/2^e$ donde e es la etapa en la que se encuentra el algoritmo. Como tenemos la señal completa al principio del algoritmo, el contador e de las etapas irá desde v decrementándose hasta 1 en donde $v = log_2(N)$.

Otra diferencia esta en la multiplicación por el factor de fase, en este algoritmo se multiplica la segunda salida de la mariposa por el factor de fase, $X(b1) = (a-b)*W(n2)$, a diferencia del algoritmo Radix-2 DIT que la multiplicación se realiza por el segundo dato que entra a la mariposa antes de la suma o la resta respectiva de la mariposa.

5.5.3. FFT Radix-4 DIT.

A continuación se describirá la programación del algoritmo Radix-4 DIT para el cálculo de la transformada de Fourier rápida.

5.5.3.1. Programación.

Con la experiencia obtenida al programar el algoritmo Radix-2, la programación del Radix-4 fue un poco mas sencilla. La estructura de mariposa de este algoritmo es un poco diferente a la mariposa del Radix-2 puesto que entran y salen 4 datos de ella. Al igual que en el Radix-2, se deben realizar $N/4$ veces la mariposa en v etapas en donde, en este caso, $v = log_4(N)$. Debido a esto, la longitud N del vector de entrada debe ser potencia de 4.

La decimación en el tiempo del Radix-4 es diferente a la realizada en el Radix-2. Ésta divide en 4 conjuntos de datos dependiendo de su posición en el vector de datos. Al igual que en la decimación en el Radix-2 esta división se realiza repetidamente hasta tener N vectores con un dato cada uno.

En el Radix-4 se calculan todos los factores de fase antes de entrar a las etapas del algoritmo y son llamados de memoria cuando son utilizados.

Por ser caso de decimación en el tiempo, la salida se encuentra en orden y representa la transformada de Fourier rápida de la señal de entrada.

En la figura 5.11 se presenta el diagrama de flujo ocupado para la programación del algoritmo Radix-4 DIT.

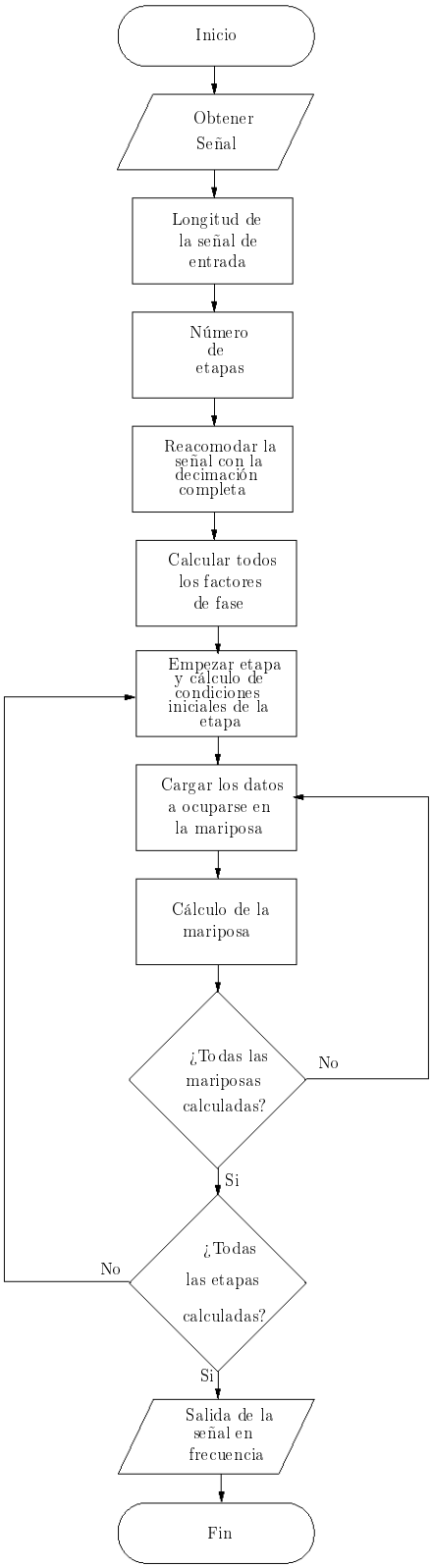


Figura 5.11: Diagrama de Flujo del Radix-4 DIT FFT.

5.5.3.2. Programa.

A continuación se muestra el código del programa del algoritmo Radix-4 DIT.

```
    %Algoritmo Radix-4 DIT
function [X] = fftradix4dit(x)
N=length(x); % Long. de la señal de entrada.
v=log(N)/log(4); % Núm. de etapas.
N4=N/4;
X=x; % Reasignación de la entrada.
W=exp((-2*pi*i)*[0:N-1]/N); % Cálculo de los factores de fase.
X=digitrevorder(X,4); % Reacomodo de la señal de salida.
for e=1:v, % Inicio de etapa.
    L=4^e; % Condiciones iniciales.
    n2=4^(v-e);
    if e==1, n2=0; end
    for k=0:L:(N-L), % Inicio del cálculo de las mariposas.
        n1=0;
        for n=0:(L/4)-1,
            a=n+k+1;
            b=(L/4)+a;
            c=(L/4)+b;
            d=(L/4)+c;
            A=X(a); % Carga valor 1.
            B=X(b)*W((1*n1*n2)+1); % Carga valor 2 * Factor de fase.
            C=X(c)*W((2*n1*n2)+1); % Carga valor 3* Factor de fase.
            D=X(d)*W((3*n1*n2)+1); % Carga valor 4* Factor de fase.
            X(a)=A+B+C+D; % Salida 1 de la mariposa.
            X(b)=(A-(j*B)-C+(j*D)); % Salida 2 de la mariposa.
            X(c)=(A-B+C-D); % Salida 3 de la mariposa.
            X(d)=(A+(j*B)-C-(j*D)); % Salida 4 de la mariposa.
            n1=n1+1; % Sig. factor de fase
        end
    end % Fin de mariposa.
end % Fin de etapa.
end % Fin de función.
```

La estructura el programa del algoritmo Radix-4 DIT es muy parecida a la del Radix-2 DIT. Lo primero que se realiza es obtener la longitud del vector de datos de entrada, recordando que debe ser potencia de 4. Una vez obtenida, se calcula el número de etapas que deberá realizar el programa. Al no poderse calcular directamente el $log_4(N)$ en Matlab se calcula de la siguiente manera: $v = \frac{log(N)}{log(4)}$.

Se calculan todos los factores de fase para luego ser llamados de memoria al utilizarlos. También se realiza la decimación con ayuda del comando $digitrevorder(X, 4)$ al cual se le debe indicar en vector a ser decimado, en este caso X, y el número de dígitos sobre el cual se basará para realizarla, en este caso es 4.

Se realizarán $N/4$ mariposas por cada etapa. En cada mariposa entran 4 datos, los cuales se combinan para obtener 4 datos de salida. Los datos 2,3 y 4 se multiplican por

un correspondiente factor de fase. La salida se guarda en la misma variable puesto que será la entrada en la siguiente etapa del programa.

Al terminar las v etapas obtenemos la transformada de Fourier rápida de los datos que entraron en el programa.

5.5.4. FFT Radix-4 DIF.

En la siguiente sección se describe la programación del algoritmo Radix-4 DIF para el cálculo de la transformada de Fourier rápida.

5.5.4.1. Programación.

Al igual que en el caso del algoritmo Radix-2, la programación de la variante con decimación en frecuencia después de haber programado la decimación en el tiempo es muy sencilla. Los cambios son en la decimación o reacomodo de la señal de entrada, en éste caso se realiza a la salida del algoritmo al terminar todas las etapas. Otra diferencia se encuentra el la multiplicación por los factores de fase, se realizan a la salida de la mariposa y no a la entrada como en la decimación en el tiempo.

En la figura 5.12 se presenta el diagrama de flujo ocupado para la programación del algoritmo Radix-4 DIF.

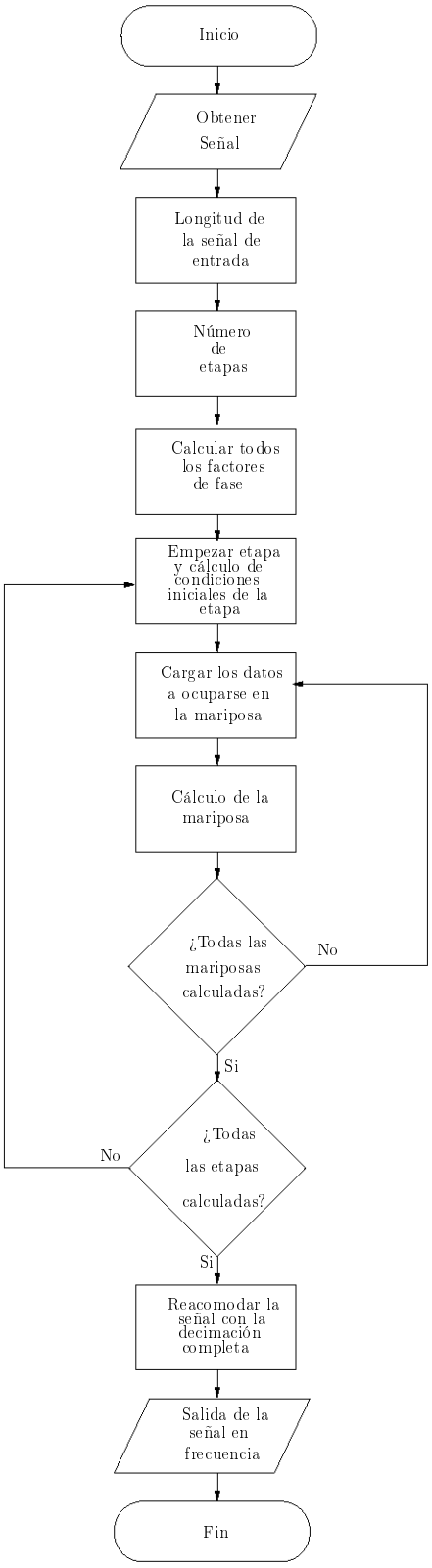


Figura 5.12: Diagrama de Flujo del Radix-4 DIF FFT.

5.5.4.2. Programa.

El siguiente es el programa en Matlab de la FFT Radix-4 DIF.

```
 %Algoritmo Radix-4 DIF
function [X] = fftradix4dif(x)
N=length(x); % Long. de la señal de entrada.
v=log(N)/log(4); % Núm. de etapas.
N4=N/4;
X=x;
W=exp((-2*pi*i)*[0:N-1]/N); % Cálculo de los factores de fase.
for e=v:-1:1, % Inicio de etapa.
    L=4^e; % Condiciones iniciales.
    n2=4^(v-e);
    if e==1, n2=0; end
    for k=0:L:(N-L), % Inicio del cálculo de las mariposas.
        n1=0;
        for n=0:(L/4)-1,
            a=n+k+1;
            b=(L/4)+a;
            c=(L/4)+b;
            d=(L/4)+c;
            A=X(a); % Carga valor 1.
            B=X(b); % Carga valor 2.
            C=X(c); % Carga valor 3.
            D=X(d); % Carga valor 4.
            X(a)=A+B+C+D; % Salida 1 de la mariposa.
            X(b)=(A-(j*B)-C+(j*D))*W((1*n1*n2)+1); % Salida 2 de la mariposa.
            X(c)=(A-B+C-D)*W((2*n1*n2)+1); % Salida 3 de la mariposa.
            X(d)=(A+(j*B)-C-(j*D))*W((3*n1*n2)+1); % Salida 4 de la mariposa.
            n1=n1+1; % Sig. factor de fase
        end
    end % Fin de mariposa.
end % Fin de etapa.
X=digitrevorder(X,4); % Reacomodo de la señal de salida.
end % Fin de función.
```

La estructura del programa es muy similar a la estructura del programa del Radix-4 DIT. El primer cambio se encuentra el las condiciones iniciales de los contadores que nos permiten apuntar a los datos que entraran en cada una de las etapas a las diferentes mariposas.

Los factores de fase son llamados de memoria a la salida de las mariposas, multiplicando las combinaciones de los datos de entrada en las salidas 2, 3 y 4.

Como ya se había mencionado, la decimación se realiza al final de las etapas antes de que salga la señal y así obtenemos la transformada de Fourier rápida de los datos de entrada.

Una vez programados los algoritmos de nuestro interés se realizó la validación de estos.

5.6. Prueba y Validación de los Algoritmos FFT.

La validación de los algoritmos programados se realizó con la función de Matlab, se probó con diferentes tipos de señales, la respuesta al impulso de un sistema con un polo y la respuesta al impulso de tres sistemas que representan tres canales de comunicación de fase mínima. Las funciones de transferencia de estos sistemas se muestran en la tabla 5.6.

Señal	Función de Transferencia
yt	$H_{yt} = \frac{1}{1-1.357z^{-1}+0.9216z^{-2}}$
$yc1$	$H_{c1} = 1 + 0.2z^{-1} - 0.6z^{-2}$
$yc2$	$H_{c2} = 1 + 0.51z^{-1} + 0.1997z^{-2}$
$yc3$	$H_{c3} = 1 + 0.536z^{-1} + 0.0718z^{-2}$

Tabla 5.6: Señales empleadas en la simulación.

La longitud de estas señales fue de 256 puntos, el cual es la longitud en el estándar de IEEE para WiMAX fijo.

En las figuras 5.13, 5.14, 5.15 y 5.16 se muestran las gráficas comparativas entre la FFT de Matlab y las FFTs programadas para las señales de respuesta al impulso de los sistemas de un solo polo y los canales de comunicación.

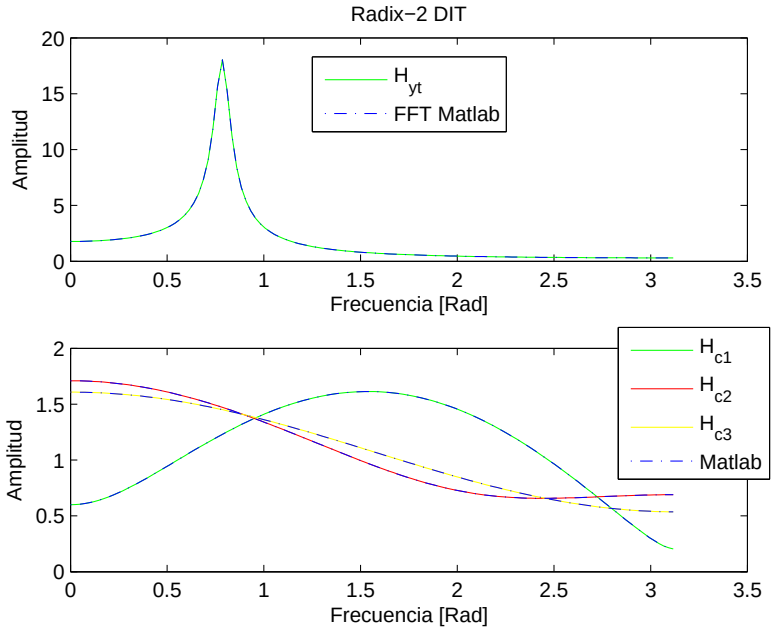


Figura 5.13: Representación en frecuencia de las señales. Comparación entre FFT de Matlab y FFT **Radix-2 DIT** programada.

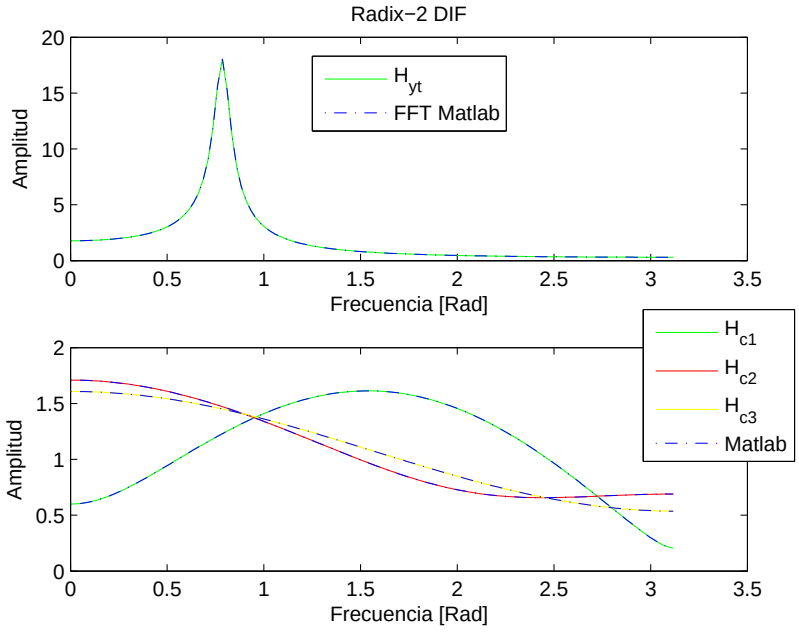


Figura 5.14: Representación en frecuencia de las señales. Comparación entre FFT de Matlab y FFT **Radix-2 DIF** programada.

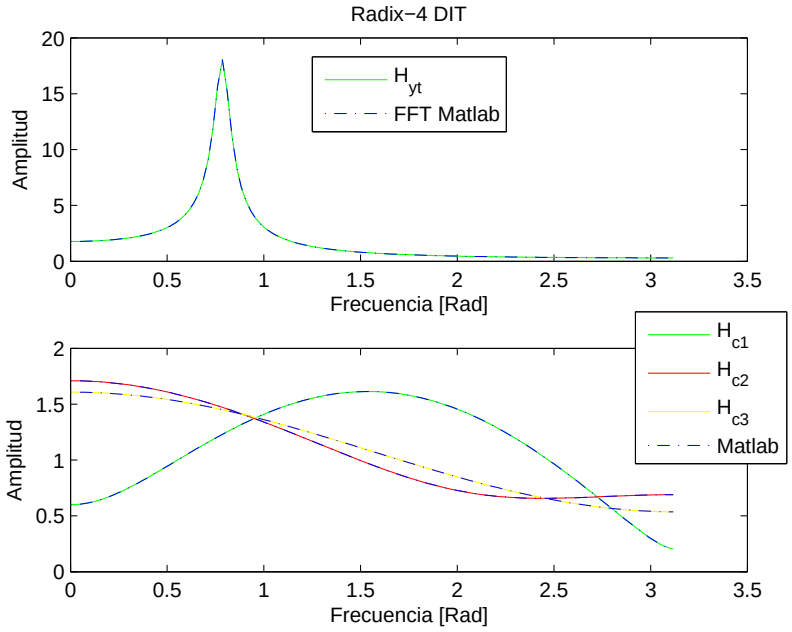


Figura 5.15: Representación en frecuencia de las señales. Comparación entre FFT de Matlab y FFT **Radix-4 DIT** programada.

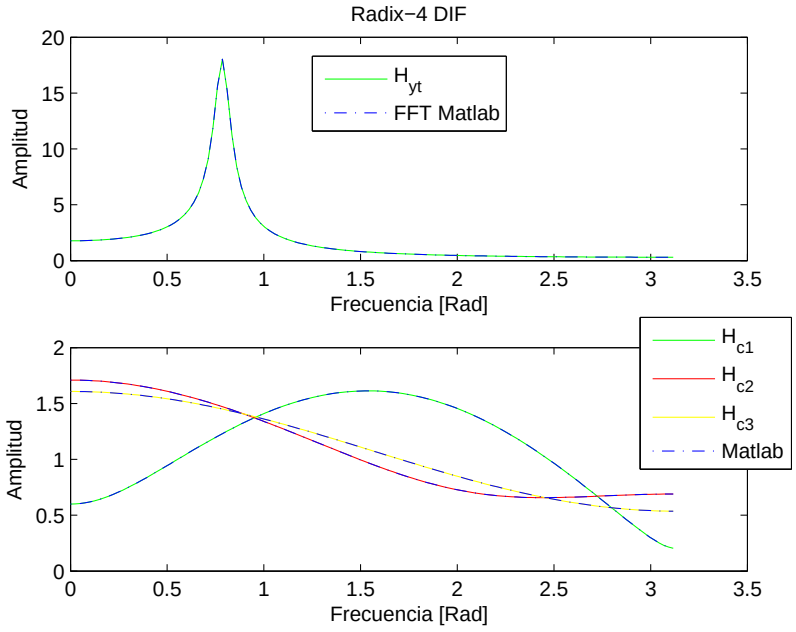


Figura 5.16: Representación en frecuencia de las señales. Comparación entre FFT de Matlab y FFT **Radix-4 DIF** programada.

Se observa en las cuatro gráficas que los espectros en frecuencia obtenidos por los algoritmos programados son idénticos a los obtenidos con la función de Matlab. Para saber que tanto se asemejan se realizó la prueba de precisión de cálculo que se muestra en el siguiente capítulo junto con la prueba de velocidad de cálculo.

Capítulo **6**

Evaluación y Comparación de los Algoritmos FFT.

Una vez programados y validados los algoritmos se realizaron pruebas con ellos para poderlos evaluar y comparar. A continuación se presentan dichas pruebas y sus resultados.

6.1. Evaluación del Desempeño de los Algoritmos.

Las pruebas realizadas a los algoritmos programados fueron 2:

1. Precisión de cálculo, determinada con el error promedio cuadrático y la desviación estándar del error.
2. Velocidad de cálculo en segundos y en millones de operaciones por segundo.

Todas las pruebas fueron realizadas en una computadora Dell Inspiron 1420 con procesador Intel Core 2 Duo T7250 @ 2.00 GHz. y 2 Gb. en RAM y el software Matlab 7.9.

6.1.1. Precisión de Cálculo.

Para poder calcular la precisión de los algoritmos programados se tomó como valor exacto la FFT de Matlab, para el error hacia adelante, y la señal de entrada, para el error hacia atrás. Teniendo estos valores se compararon con los resultados de cada algoritmo de dos maneras. Se obtuvo el *Error Promedio Cuadrático hacia adelante* y el *Error Promedio Cuadrático hacia atrás*. El error promedio cuadrático se calcula al comparar un valor que se considera exacto con el valor obtenido de la siguiente manera:

$$EPC = \frac{1}{N}\sum_{n=0}^{N-1}(x_r[n] - x_p[n])^2$$

En donde:

$x_r[n]$	es el valor de la FFT de Matlab para el error hacia adelante y la señal de entrada para el error hacia atrás.
$x_p[n]$	es el valor de la FFT de los algoritmos programados para el error hacia adelante y los algoritmos inversos programados aplicados a los resultados de los algoritmos para el error hacia atrás.
N	es el número de datos que tiene la señal de entrada.

Esta prueba se realizó tanto al módulo de la señal de respuesta como al ángulo para determinar que tan exactos son los algoritmos programados.

La señal de entrada, al igual que en la velocidad de cálculo, se trata de una señal de ruido blanco con longitud N de **2** a **4096** puntos. En la figura 6.1 se muestra un ejemplo de la señal con longitud $N = 1024$ puntos.

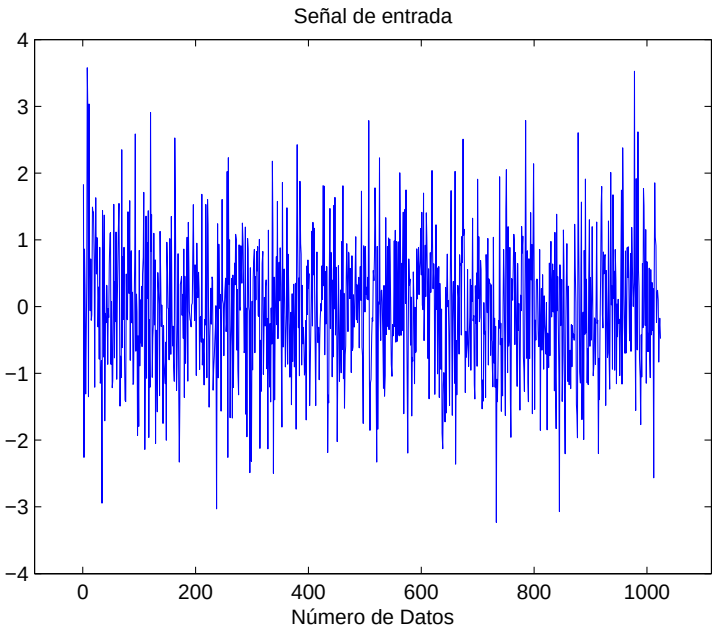


Figura 6.1: Señal de entrada de $N = 1024$ puntos.

En las gráficas 6.2 y 6.3 se presentan los resultados de los Errores Promedio Cuadráticos hacia adelante y hacia atrás del módulo de la señal respectivamente. En las tablas 6.1 y 6.2 se muestran los valores de los errores graficados.

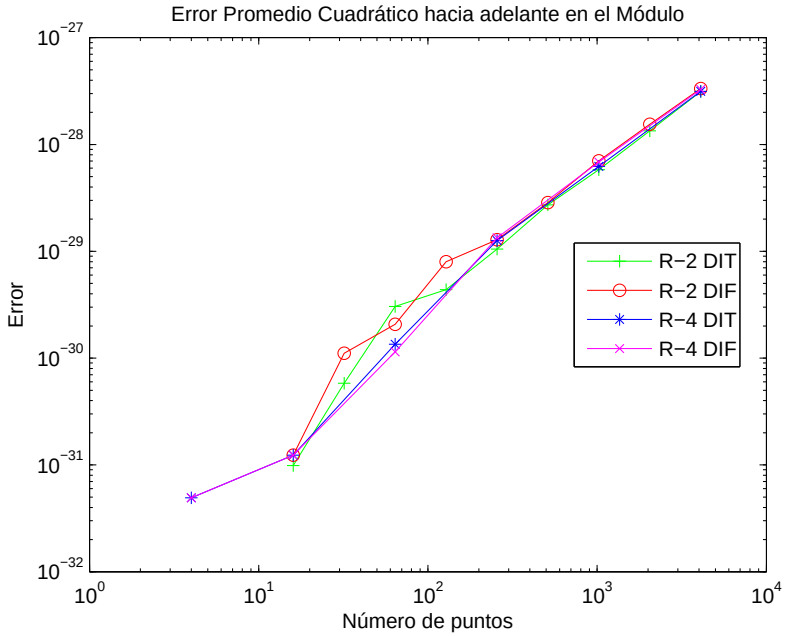


Figura 6.2: Error promedio cuadrático hacia adelante en el módulo.

N	R-2 DIT adelante	R-2 DIF adelante	R-4 DIT adelante	R-4 DIF adelante
2	0	0	-	-
4	0	0	4.9304e-032	4.9304e-032
8	0	0	-	-
16	9.8608e-032	1.2326e-031	1.2326e-031	1.2326e-031
32	5.8086e-031	1.1124e-030	-	-
64	3.0514e-030	2.0723e-030	1.347e-030	1.1442e-030
128	4.3708e-030	7.9801e-030	-	-
256	1.0498e-029	1.2858e-029	1.2622e-029	1.3381e-029
512	2.7185e-029	2.8562e-029	-	-
1024	5.8148e-029	7.0336e-029	6.2393e-029	6.8429e-029
2048	1.3398e-028	1.5467e-028	-	-
4096	3.1332e-028	3.3418e-028	3.1427e-028	3.2821e-028

Tabla 6.1: Error promedio cuadrático de los algoritmos hacia adelante en el módulo.

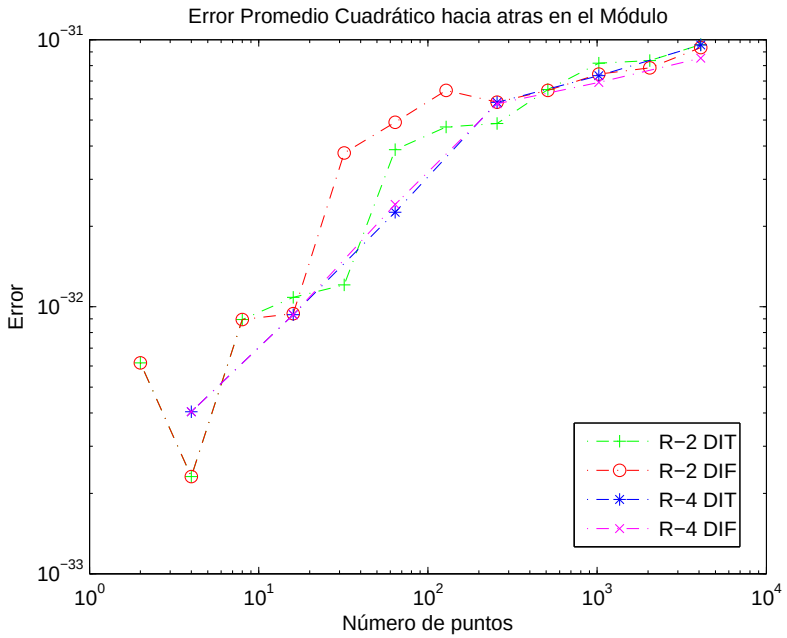


Figura 6.3: Error promedio cuadrático hacia atrás en el módulo.

N	R-2 DIT atrás	R-2 DIF atrás	R-4 DIT atrás	R-4 DIF atrás
2	6.163e-033	6.163e-033	-	-
4	2.3111e-033	2.3111e-033	4.0445e-033	4.0445e-033
8	8.9556e-033	8.9556e-033	-	-
16	1.0845e-032	9.413e-033	9.3408e-033	9.3408e-033
32	1.2075e-032	3.7665e-032	-	-
64	3.8735e-032	4.9087e-032	2.2581e-032	2.4167e-032
128	4.71e-032	6.4552e-032	-	-
256	4.8565e-032	5.8356e-032	5.8292e-032	5.757e-032
512	6.51e-032	6.4575e-032	-	-
1024	8.1676e-032	7.4304e-032	7.3299e-032	6.9194e-032
2048	8.3565e-032	7.8386e-032	-	-
4096	9.5837e-032	9.3396e-032	9.5604e-032	8.528e-032

Tabla 6.2: Error promedio cuadrático de los algoritmos hacia atrás en el módulo.

En las tablas 6.3 y 6.4 se muestran los valores de la varianza de los errores hacia adelante y hacia atrás en el módulo para cada señal de entrada con diferentes longitudes para los algoritmos programados.

N	R-2 DIT adelante	R-2 DIF adelante	R-4 DIT adelante	R-4 DIF adelante
2	0	0	-	-
4	0	0	4.9304e-032	4.9304e-032
8	0	0	-	-
16	9.8608e-032	1.2326e-031	1.2326e-031	1.2326e-031
32	5.8086e-031	1.1124e-030	-	-
64	3.0514e-030	2.0723e-030	1.347e-030	1.1442e-030
128	4.3708e-030	7.9801e-030	-	-
256	1.0498e-029	1.2858e-029	1.2622e-029	1.3381e-029
512	2.7185e-029	2.8562e-029	-	-
1024	5.8148e-029	7.0336e-029	6.2393e-029	6.8429e-029
2048	1.3398e-028	1.5467e-028	-	-
4096	3.1332e-028	3.3418e-028	3.1427e-028	3.2821e-028

Tabla 6.3: Varianza del error para los algoritmos hacia adelante en el módulo.

N	R-2 DIT atrás	R-2 DIF atrás	R-4 DIT atrás	R-4 DIF atrás
2	6.163e-033	6.163e-033	-	-
4	2.3111e-033	2.3111e-033	4.0445e-033	4.0445e-033
8	8.9556e-033	8.9556e-033	-	-
16	1.0845e-032	9.413e-033	9.3408e-033	9.3408e-033
32	1.2075e-032	3.7665e-032	-	-
64	3.8735e-032	4.9087e-032	2.2581e-032	2.4167e-032
128	4.71e-032	6.4552e-032	-	-
256	4.8565e-032	5.8356e-032	5.8292e-032	5.757e-032
512	6.51e-032	6.4575e-032	-	-
1024	8.1676e-032	7.4304e-032	7.3299e-032	6.9194e-032
2048	8.3565e-032	7.8386e-032	-	-
4096	9.5837e-032	9.3396e-032	9.5604e-032	8.528e-032

Tabla 6.4: Varianza del error para los algoritmos en el módulo hacia atrás.

De la misma manera se realizó el cálculo del error promedio cuadrático y la varianza de éste del ángulo de fase para los algoritmos programados. El código del programa es igual al del módulo, con la diferencia del uso del comando *angle* para obtener el ángulo de fase de la señal de respuesta de los algoritmos programados.

En las figuras 6.4 y 6.5 se presentan las gráficas de los errores promedio cuadráticos hacia adelante y hacia atrás para el ángulo de fase. Y en las tablas 6.5 y 6.6 se muestran los valores respectivos de cada gráfica.

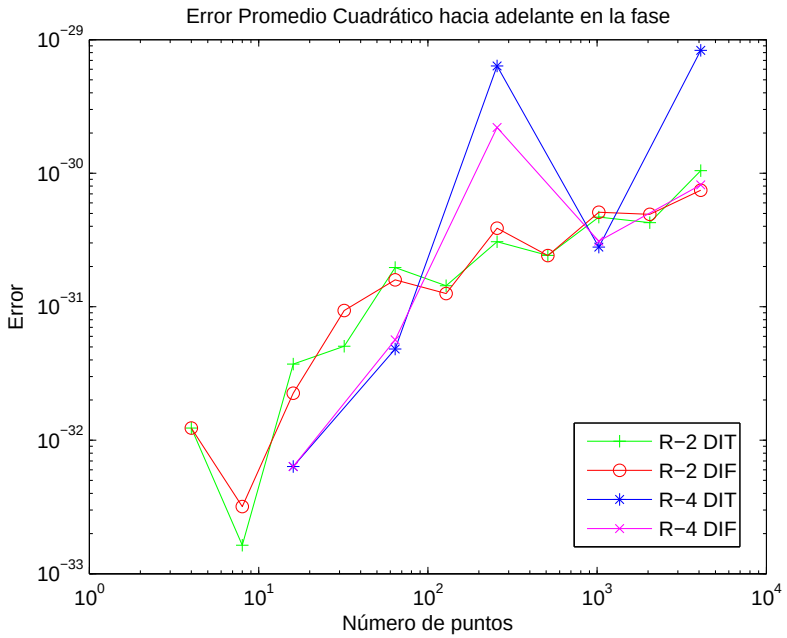


Figura 6.4: Error promedio cuadrático hacia adelante del ángulo de fase.

N	R-2 DIT adelante	R-2 DIF adelante	R-4 DIT adelante	R-4 DIF adelante
2	0	0	-	-
4	1.2326e-032	1.2326e-032	0	0
8	1.637e-033	3.1838e-033	-	-
16	3.7232e-032	2.248e-032	6.3556e-033	6.3556e-033
32	5.0556e-032	9.3696e-032	-	-
64	1.9636e-031	1.5905e-031	4.8219e-032	5.6434e-032
128	1.4361e-031	1.2561e-031	-	-
256	3.0721e-031	3.8792e-031	6.3494e-030	2.2013e-030
512	2.4198e-031	2.4181e-031	-	-
1024	4.6867e-031	5.0934e-031	2.796e-031	3.0968e-031
2048	4.2635e-031	4.9235e-031	-	-
4096	1.0462e-030	7.4557e-031	8.3298e-030	8.1742e-031

Tabla 6.5: Error cuadrático promedio de los algoritmos hacia adelante en el ángulo de fase.

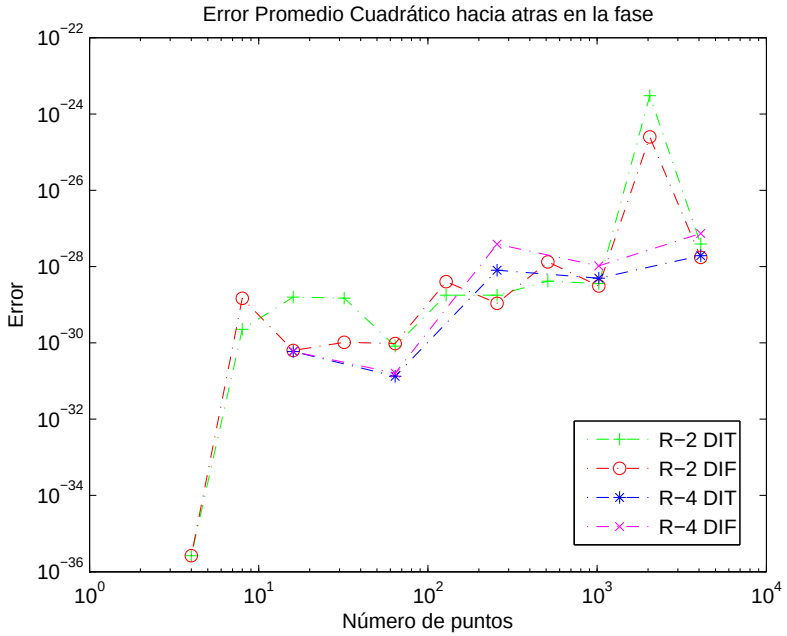


Figura 6.5: Error promedio cuadrático hacia atrás del ángulo de fase.

N	R-2 DIT atrás	R-2 DIF atrás	R-4 DIT atrás	R-4 DIF atrás
2	0	0	-	-
4	2.6267e-036	2.6267e-036	0	0
8	2.2655e-030	1.4774e-029	-	-
16	1.5921e-029	6.2911e-031	5.9211e-031	5.9211e-031
32	1.4822e-029	1.0327e-030	-	-
64	8.2379e-031	9.645e-031	1.3243e-031	1.5978e-031
128	1.7832e-029	4.0388e-029	-	-
256	1.7812e-029	1.0843e-029	8.0322e-029	3.866e-028
512	4.1467e-029	1.3197e-028	-	-
1024	3.5726e-029	3.0925e-029	4.9677e-029	1.0449e-028
2048	3.0261e-024	2.5158e-025	-	-
4096	3.9022e-028	1.7447e-028	1.934e-028	7.3825e-028

Tabla 6.6: Error promedio cuadrático de los algoritmos hacia atrás en el ángulo de fase.

En las tablas 6.7 y 6.8 se muestran los valores de la varianza de los errores hacia adelante y hacia atrás para el ángulo de fase para cada señal de entrada procesada con cada uno de los algoritmos programados.

N	R-2 DIT adelante	R-2 DIF adelante	R-4 DIT adelante	R-4 DIF adelante
2	0	0	-	-
4	1.2326e-032	1.2326e-032	0	0
8	1.637e-033	3.1838e-033	-	-
16	3.7232e-032	2.248e-032	6.3556e-033	6.3556e-033
32	5.0556e-032	9.3696e-032	-	-
64	1.9636e-031	1.5905e-031	4.8219e-032	5.6434e-032
128	1.4361e-031	1.2561e-031	-	-
256	3.0721e-031	3.8792e-031	6.3494e-030	2.2013e-030
512	2.4198e-031	2.4181e-031	-	-
1024	4.6867e-031	5.0934e-031	2.796e-031	3.0968e-031
2048	4.2635e-031	4.9235e-031	-	-
4096	1.0462e-030	7.4557e-031	8.3298e-030	8.1742e-031

Tabla 6.7: Varianza del error para los algoritmos hacia adelante en el ángulo de fase.

N	R-2 DIT atrás	R-2 DIF atrás	R-4 DIT atrás	R-4 DIF atrás
2	0	0	-	-
4	2.6267e-036	2.6267e-036	0	0
8	2.2655e-030	1.4774e-029	-	-
16	1.5921e-029	6.2911e-031	5.9211e-031	5.9211e-031
32	1.4822e-029	1.0327e-030	-	-
64	8.2379e-031	9.645e-031	1.3243e-031	1.5978e-031
128	1.7832e-029	4.0388e-029	-	-
256	1.7812e-029	1.0843e-029	8.0322e-029	3.866e-028
512	4.1467e-029	1.3197e-028	-	-
1024	3.5726e-029	3.0925e-029	4.9677e-029	1.0449e-028
2048	3.0261e-024	2.5158e-025	-	-
4096	3.9022e-028	1.7447e-028	1.934e-028	7.3825e-028

Tabla 6.8: Varianza del error para los algoritmos hacia atrás en el ángulo de fase.

6.1.2. Velocidad de Cálculo.

La velocidad del cálculo de los algoritmos programados fue medida de 2 diferentes formas. Primero se contó el tiempo en segundos que tardan en calcular la FFT para diferentes señales de entrada con diferentes longitudes. Después se tomo ese tiempo para calcular las millones de operaciones por segundo -llamadas "mflops"- que realiza aproximadamente en el cálculo de las FFTs.

Las señales de entrada, como en el caso de la precisión de cálculo, fueron señales de ruido blanco. La longitud de la señal se fue incrementando en cada iteración en potencias de 2 para los algoritmos Radix-2 y en potencias de 4 para los algoritmos Radix-4. La longitud de la señal de entrada va desde 2 hasta 4096 puntos.

El tiempo de cálculo se promedio para 10 iteraciones para cada longitud. Este promedio se realizó con la función *tic toc*, la cual mide el tiempo transcurrido entre *tic* y *toc*.

El tiempo transcurrido se divide entre 10 para obtener el promedio de cada FFT para las diferentes longitudes, en la gráfica 6.6 y en la table 6.9 se muestra éste tiempo en segundos.

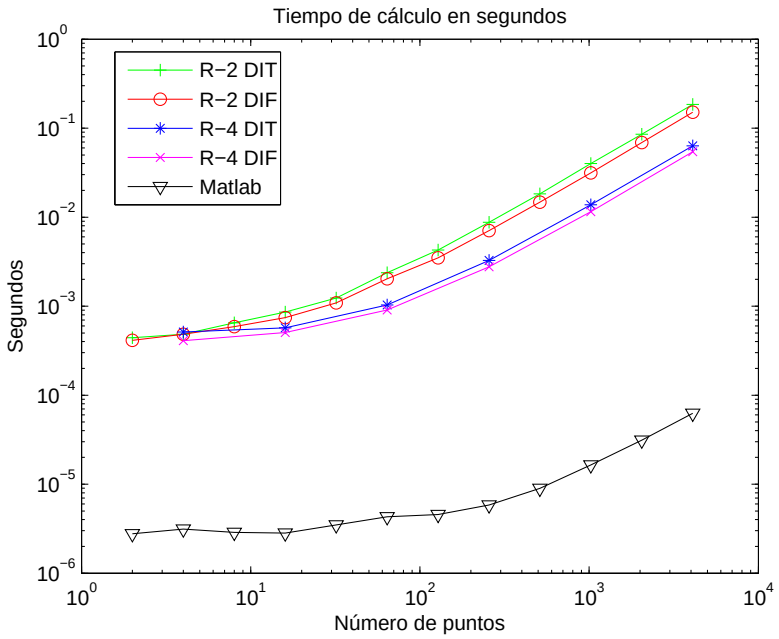


Figura 6.6: Comparación del tiempo de cálculo en segundos para los algoritmos programados y la FFT de Matlab.

N	R-2 DIT	R-2 DIF	R-4 DIT	R-4 DIF	Matlab
2	4.4274e-04	4.1333e-04	-	-	2.7716e-06
4	4.8288e-04	4.8391e-04	5.1229e-04	4.0959e-04	3.1309e-06
8	6.4995e-04	5.881e-04	-	-	2.8743e-06
16	8.6034e-04	7.4249e-04	5.7039e-04	5.0526e-04	2.823e-06
32	1.2329e-03	1.0886e-03	-	-	3.4902e-06
64	2.3736e-03	2.0325e-03	1.0333e-03	9.054e-04	4.3114e-06
128	4.282e-03	3.4899e-03	-	-	4.5681e-06
256	8.7629e-03	7.08e-03	3.2676e-03	2.7631e-03	5.8512e-06
512	1.8314e-02	1.4761e-02	-	-	8.9821e-06
1024	4.0101e-02	3.1449e-02	1.3768e-02	1.1523e-02	1.6424e-05
2048	8.5367e-02	6.8878e-02	-	-	3.1207e-05
4096	0.18451	0.15123	6.3285e-02	5.4398e-02	6.2567e-05

Tabla 6.9: Tiempo de cálculo de los algoritmos en segundos.

El cálculo de los "mflop" se realiza con la siguiente ecuación

$$mflops = \frac{5\ N\ log_2(N)}{tiempo\ para\ una\ FFT\ en\ microsegundos}$$

Donde N es el número de datos que entran en la FFT. Esta no es una cuenta real de "mflops", es para tener una escala conveniente, basada en el hecho que el algoritmo FFT Radix-2 requiere $Nlog_2N$ operaciones de punto flotante. Esto permite comparar los diferentes algoritmos en una gráfica y proporciona una medida aproximada de la eficiencia en relación con la velocidad de reloj del sistema en el que se probó [17].

En las figuras 6.7 y 6.8 se muestran las gráficas comparativas de las velocidades de los algoritmos programados y de la FFT de Matlab. En la tabla 6.10 se muestra el número de "mflops" para cada algoritmo.

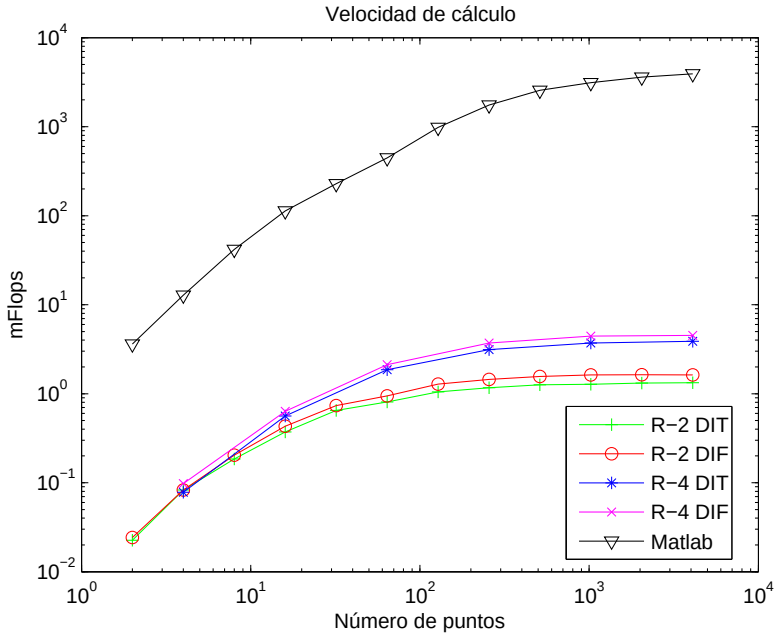


Figura 6.7: Comparación de la velocidad de cálculo en "mflop" para los algoritmos programados y la FFT de Matlab.

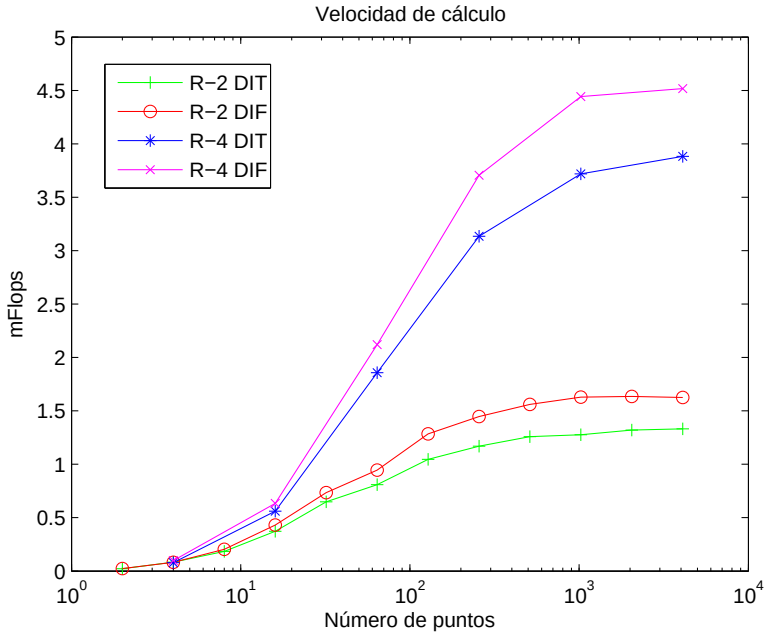


Figura 6.8: Comparación del la velocidad de cálculo en "mflops" para los algoritmos programados.

N	R-2 DIT	R-2 DIF	R-4 DIT	R-4 DIF	FFT Matlab
2	0.023043	0.024759	-	-	3.5424
4	0.084305	0.082511	0.089556	0.098523	12.5697
8	0.20281	0.21524	-	-	44.9608
16	0.39404	0.43878	0.57824	0.63997	115.4548
32	0.58763	0.67128	-	-	205.0842
64	0.88165	1.054	1.8788	2.123	456.1873
128	1.0314	1.2813	-	-	938.5359
256	1.1864	1.3606	3.1371	3.5792	1608.9187
512	1.2026	1.5781	-	-	2565.0761
1024	1.3197	1.6487	3.5049	4.2962	2977.7003
2048	1.3306	1.6518	-	-	3429.008
4096	1.3764	1.6481	3.8626	4.712	4023.6488

Tabla 6.10: Velocidad de cálculo para los algoritmos en "mflops".

Si se comparan los algoritmos con la función de Matlab se puede observar que existe una gran diferencia entre ellos, la función fft de Matlab es mucho más rápida que los algoritmos programados. Esto se debe a que la función esta basada en una librería llamada FFTW. La librería FFTW realiza la operación con una combinación de los algoritmos Radix-2, Radix-4, PFA y Split-Radix[18], el cual es una mezcla de los algoritmos Radix-2 y Radix-4[19].

La librería FFTW no usa un algoritmo en especifico, adapta el algoritmo de la DFT a los requerimientos del usuario. El cálculo de la transformada se realiza en dos fases. Primero el planeador de la FFTW "aprende" la manera más rápida de calcular la transformada en la máquina en la que se esta ejecutando. El planeador produce una estructura de datos llamada plan que contiene la información que "aprendió" de la máquina. A continuación se ejecuta el plan para realizar la transformada[17]. La FFTW puede guardar planes de transformadas previamente calculadas y de esta manera realizar los cálculos cada vez más rápido.

En el apéndice A se muestran los códigos de los programas realizados en Matlab para el cálculo de las FFTs así como los códigos de las pruebas realizadas para su comparación.

6.2. Análisis de Resultados.

De la prueba de comparación de los algoritmos programados con la función de Matlab se puede observar que prácticamente se obtienen los mismos resultados, por lo que es necesario observar los resultados obtenidos en las pruebas de precisión de cálculo.

Debido a que en la modulación OFDM para WiMAX la longitud de la FFT puede ser 128, 256, 512, 1024 y 2048, serán los valores que más se tomarán en cuenta para la toma de decisión. Se debe tomar en cuenta que el algoritmo FFT Radix-4 sólo puede ser ocupado directamente para las longitudes 256 y 1024, puesto que estos valores son potencias de 4, para los demás valores se puede realizar un ventaneo de la mitad de la longitud y así, por ejemplo, para calcular una FFT de 512 puntos se pueden calcular dos FFTs de 256 y obtener el mismo resultado. El algoritmo FFT Radix-2 puede ser utilizado sin ventaneo para las 5 longitudes pues trabaja con todas las potencias de 2.

De la figura 6.2 y la tabla 6.1 se puede determinar que el algoritmo FFT Radix-2 DIT es el que mantiene el Error Promedio Cuadrático hacia adelante con respecto al módulo más bajo de los 4 algoritmos.

De la figura 6.3 y la tabla 6.2 se puede determinar que al algoritmo FFT Radix-4 DIF cuenta con el error promedio cuadrático hacia atrás en el módulo menor para las longitudes 256 y 1024, para las longitudes restantes el algoritmo FFT Radix-2 DIF es el de mejor desempeño.

En el error promedio cuadrático hacia adelante del ángulo de fase podemos observar en la figura 6.4 y tabla 6.5 que el que demuestra mejor desempeño es el algoritmo FFT Radix-2 DIT, no siempre es el que tiene el error más bajo, pero se mantiene muy bajo sin brincos como los Radix-4.

En el error promedio cuadrático hacia atrás del ángulo de fase, en la figura 6.5 y la tabla 6.6 se puede observar que los cuatro algoritmos tiene valores muy cercanos sin mostrar que alguno tenga un mejor desempeño que los demás.

Las siguientes pruebas realizadas a los algoritmos fueron las de velocidad de cálculo. Para los valores de interés se puede observar que, para los calculables directamente por el algoritmo, el FFT Radix-4 DIF fue el más rápido. Al ser el algoritmo más rápido en tiempo de cálculo, es el que realiza más operaciones por segundo. Se puede observar que realizando 2 FFTs de la mitad de la longitud para los valores 128, 512 y 2048 sigue siendo más rápido que los algoritmos Radix-2.

6.3. Selección del Algoritmo.

Para la selección del algoritmo FFT se deben tener en cuenta los siguientes criterios: Complejidad de implementación, complejidad aritmética, precisión y velocidad de cálculo.

En cuanto a la complejidad de implementación de los algoritmos se puede decir que los Radix-4 tienen una estructura más compleja que los Radix-2. La mariposa del Radix-4

es un poco más complicada de implementar pero una vez comprendida la estructura de la mariposa y con la experiencia obtenida con la implementación del Radix-2 se simplifica.

La complejidad aritmética de los algoritmos FFT Radix-4 es menor en cuanto al número de multiplicaciones complejas realizadas para la obtención de la transformada de Fourier. Por ser menor el número de operaciones se tiene un menor tiempo de cálculo.

De las pruebas de precisión aplicadas a los algoritmos no es posible seleccionar uno de ellos para su posible implementación. Los errores mostrados en general son muy bajos, entre $1e^{-36}$ a $1e^{-28}$ para las diferentes longitudes y sin variar mucho cuando se analiza las mismas longitudes.

En cuanto a la velocidad de cálculo si existe una diferencia considerable entre los algoritmos Radix-2 y los algoritmos Radix-4, siendo el FFT Radix-4 DIF el más rápido de todos.

A partir del análisis estas pruebas se podría determinar que para WiMAX el algoritmo FFT Radix-4 DIF podría ser el más indicado para ser implementado en un procesador especifico.

Capítulo 7

Conclusiones Generales y Futuros Trabajos.

7.1. Conclusiones.

Para este trabajo se realizó un estudio de análisis, implementación y evaluación de cuatro algoritmos FFT con el fin de encontrar el que cumpla con los requerimientos del multiplexado OFDM de la capa física de las redes de comunicación inalámbrica WiMAX. Para esto fueron implementados los algoritmos y las pruebas de precisión y velocidad de cálculo en el software Matlab®. Para la selección del algoritmo FFT se tomaron en cuenta los siguientes criterios: Complejidad de implementación, complejidad aritmética, precisión y velocidad de cálculo.

La implementación no es mucho más compleja en los algoritmos Radix-4 y si se tiene una gran ventaja en cuanto a tiempos de cálculo. La precisión en los cálculos es muy parecida entre los algoritmos implementados y el valor de referencia, que fue el resultado de la función fft de Matlab. En los algoritmos Radix-4 tenemos un porcentaje de ahorro mayor en el número de multiplicaciones complejas con respecto al ahorro de los algoritmos Radix-2. Para el algoritmo Radix-4 DIF se tiene un menor tiempo de cálculo en comparación con el Radix-4 DIT dándole una ligera ventaja. Si bien el algoritmo Radix-4 no puede ser aplicado directamente a todas las longitudes de la FFT utilizadas en WiMAX se pueden hacer ventaneos de longitudes en las que se pueda aplicar directamente el algoritmo, por ejemplo realizar 2 FFT's de 256 puntos para cumplir la longitud de 512 puntos, y aún así realizar el proceso de la FFT en menor tiempo que con el algoritmo Radix-2.

Una vez realizadas las pruebas implementadas para la evaluación de los algoritmos, analizados los resultados y tomando en cuenta los criterios establecidos para la selección del algoritmo, se determinó que el algoritmo FFT Radix-4 DIF es el que cumple con las características necesarias para ser implementado dentro de OFDM de la capa física de WiMAX.

7.2. Futuros Trabajos.

Con las pruebas realizadas a los algoritmos programados se ha demostrado que las variantes con decimación en la frecuencia son un poco más rápidos que los que tienen decimación en el tiempo. La velocidad en el cálculo de las transformadas sería el factor para la elección de uno de los algoritmos puesto que la precisión de los cuatro algoritmos fue muy buena sin muchas diferencias entre ellos, la complejidad aritmética, aunque es mayor en los Radix-4 no es muy diferente entre ellos al igual que la complejidad de implementación.

Para poder asegurar que tendrán buen desempeño ya siendo implementados se deben hacer estas mismas pruebas a los algoritmos pero ahora en una implementación con longitud de palabra finita, las cuales ayudarán en la elección del mejor algoritmo para el multiplexado OFDM dentro de las redes de comunicación inalámbrica WiMAX. Después de haber validado la implementación en punto fijo se podrá implementar en alguna arquitectura DSP.

Apéndices

Apéndice **A**

Códigos programados para Matlab.

En esta sección se muestran los códigos de los algoritmos de la FFT programados en Matlab y los códigos de las pruebas para la evaluación de los algoritmos.

A.1. Algoritmos de la FFT.

A.1.1. Radix-2 DIT.

```
    % Algoritmo Radix-2 DIT
function [X] = fftradix2dit(x)
N=length(x); % Long. de la señal de entrada.
v=log2(N); % Núm. de etapas.
N2=N/2;
X=bitrevorder(x); % Decimación de la señal.
for e=1:v, % Inicio de etapa.
  L=2^e; % Condiciones iniciales.
  n1=(2^(e-1))-1;
    r=2^(v-e);
    W=exp((-2*pi*i)*[0:n1]*r/N); % Cálculo de factores de fase.
    n2=1;
    for k=0:L:(N-L), % Inicio del cálculo de las mariposas.
        for n=0:(L/2)-1,
            a1=n+k+1;
            b1=(L/2)+a1;
            a=X(a1); % Carga valor 1.
            b=X(b1)*W(n2); % Carga valor 2 * factor de fase.
            X(a1) = a + b; % Salida 1 de la mariposa.
            X(b1) = a - b; % Salida 2 de la mariposa.
            n2=n2+1; % Proceso para el siguiente factor de fase.
            if n2>n1+1,
                n2=1;
            end
        end
    end % Fin de mariposas.
end % Fin de etapa.
end % Fin de función.
```

A.1.2. Radix-2 DIF.

```
    % Algoritmo Radix-2 DIF
function [X] = fftradix2dif(x)
```

```
N=length(x); % Long. de la señal de entrada.
v=log2(N); % Núm. de etapas.
N2=N/2;
X=x; % Reasignación de la señal original.
for e=v:-1:1, % Inicio de etapa.
  L=2^e; % Condiciones iniciales.
  n1=(2^(e-1))-1;
  r=2^(v-e);
  W=exp((-2*pi*i)*[0:n1]*r/N); % Cálculo de factores de fase.
  n2=1;
  for k=0:L:(N-L), % Inicio del cálculo de las mariposas.
      for n=0:(L/2)-1,
          a1=n+k+1;
          b1=(L/2)+a1;
          a=X(a1); % Carga valor 1.
          b=X(b1); % Carga valor 2.
          X(a1) = a + b; % Salida 1 de la mariposa.
          X(b1) = (a - b) * W(n2); % Salida 2 de la mariposa.
          n2=n2+1; % Proceso para el sig. factor de fase.
          if n2>n1+1,
              n2=1;
          end
      end
  end % Fin de mariposas.
end % Fin de etapa.
X=bitrevorder(x); % Reacomodo de la señal de salida.
end % Fin de función.
```

A.1.3. Radix-4 DIT.

```
    %Algoritmo Radix-4 DIT
function [X] = fftradix4dit(x)
N=length(x); % Long. de la señal de entrada.
v=log(N)/log(4); % Núm. de etapas.
N4=N/4;
X=x; % Reasignación de la entrada.
W=exp((-2*pi*i)*[0:N-1]/N); % Cálculo de los factores de fase.
X=digitrevorder(X,4); % Reacomodo de la señal de salida.
for e=1:v, % Inicio de etapa.
    L=4^e; % Condiciones iniciales.
    n2=4^(v-e);
    if e==1, n2=0; end
    for k=0:L:(N-L), % Inicio del cálculo de las mariposas.
        n1=0;
        for n=0:(L/4)-1,
            a=n+k+1;
            b=(L/4)+a;
            c=(L/4)+b;
            d=(L/4)+c;
            A=X(a); % Carga valor 1.
            B=X(b)*W((1*n1*n2)+1); % Carga valor 2 * Factor de fase.
            C=X(c)*W((2*n1*n2)+1); % Carga valor 3* Factor de fase.
            D=X(d)*W((3*n1*n2)+1); % Carga valor 4* Factor de fase.
            X(a)=A+B+C+D; % Salida 1 de la mariposa.
            X(b)=(A-(j*B)-C+(j*D)); % Salida 2 de la mariposa.
            X(c)=(A-B+C-D); % Salida 3 de la mariposa.
            X(d)=(A+(j*B)-C-(j*D)); % Salida 4 de la mariposa.
            n1=n1+1; % Sig. factor de fase
        end
    end % Fin de mariposa.
end % Fin de etapa.
end % Fin de función.
```

A.1.4. Radix-4 DIF.

```
 %Algoritmo Radix-4 DIF
function [X] = fftradix4dif(x)
N=length(x); % Long. de la señal de entrada.
v=log(N)/log(4); % Núm. de etapas.
N4=N/4;
X=x;
W=exp((-2*pi*i)*[0:N-1]/N); % Cálculo de los factores de fase.
for e=v:-1:1, % Inicio de etapa.
    L=4^e; % Condiciones iniciales.
    n2=4^(v-e);
    if e==1, n2=0; end
    for k=0:L:(N-L), % Inicio del cálculo de las mariposas.
        n1=0;
        for n=0:(L/4)-1,
            a=n+k+1;
            b=(L/4)+a;
            c=(L/4)+b;
            d=(L/4)+c;
            A=X(a); % Carga valor 1.
            B=X(b); % Carga valor 2.
            C=X(c); % Carga valor 3.
            D=X(d); % Carga valor 4.
            X(a)=A+B+C+D; % Salida 1 de la mariposa.
            X(b)=(A-(j*B)-C+(j*D))*W((1*n1*n2)+1); % Salida 2 de la mariposa.
            X(c)=(A-B+C-D)*W((2*n1*n2)+1); % Salida 3 de la mariposa.
            X(d)=(A+(j*B)-C-(j*D))*W((3*n1*n2)+1); % Salida 4 de la mariposa.
            n1=n1+1; % Sig. factor de fase
        end
    end % Fin de mariposa.
end % Fin de etapa.
X=digitrevorder(X,4); % Reacomodo de la señal de salida.
end % Fin de función.
```

A.2. Algoritmos inversos de la FFT.

A.2.1. Radix-2 DIT inverso.

```
    % Algoritmo Radix-2 DIT inverso
function [X] = fftradix2dit(x)
N=length(x); % Long. de la señal de entrada.
v=log2(N); % Núm. de etapas.
N2=N/2;
X=bitrevorder(x); % Decimación de la señal.
for e=1:v, % Inicio de etapa.
  L=2^e; % Condiciones iniciales.
  n1=(2^(e-1))-1;
    r=2^(v-e);
    W=exp((2*pi*i)*[0:n1]*r/N); % Cálculo de factores de fase.
    n2=1;
    for k=0:L:(N-L), % Inicio del cálculo de las mariposas.
        for n=0:(L/2)-1,
            a1=n+k+1;
            b1=(L/2)+a1;
            a=X(a1); % Carga valor 1.
            b=X(b1)*W(n2); % Carga valor 2 * factor de fase.
            X(a1) = a + b; % Salida 1 de la mariposa.
            X(b1) = a - b; % Salida 2 de la mariposa.
            n2=n2+1; % Proceso para el siguiente factor de fase.
            if n2>n1+1,
                n2=1;
```

```
            end
        end
    end % Fin de mariposas.
end % Fin de etapa.
X=X/N;
end % Fin de función.
```

A.2.2. Radix-2 DIF inverso.

```
    % Algoritmo Radix-2 DIF inverso
function [X] = fftradix2dif(x)
N=length(x); % Long. de la señal de entrada.
v=log2(N); % Núm. de etapas.
N2=N/2;
X=x; % Reasignación de la señal original.
for e=v:-1:1, % Inicio de etapa.
  L=2^e; % Condiciones iniciales.
  n1=(2^(e-1))-1;
  r=2^(v-e);
  W=exp((2*pi*i)*[0:n1]*r/N); % Cálculo de factores de fase.
  n2=1;
  for k=0:L:(N-L), % Inicio del cálculo de las mariposas.
      for n=0:(L/2)-1,
          a1=n+k+1;
          b1=(L/2)+a1;
          a=X(a1); % Carga valor 1.
          b=X(b1); % Carga valor 2.
          X(a1) = a + b; % Salida 1 de la mariposa.
          X(b1) = (a - b) * W(n2); % Salida 2 de la mariposa.
          n2=n2+1; % Proceso para el sig. factor de fase.
          if n2>n1+1,
              n2=1;
          end
      end
  end % Fin de mariposas.
end % Fin de etapa.
X=bitrevorder(x)/N; % Reacomodo de la señal de salida.
end % Fin de función.
```

A.2.3. Radix-4 DIT inverso.

```
    %Algoritmo Radix-4 DIT inverso
function [X] = fftradix4dit(x)
N=length(x); % Long. de la señal de entrada.
v=log(N)/log(4); % Núm. de etapas.
N4=N/4;
X=x; % Reasignación de la entrada.
W=exp((2*pi*i)*[0:N-1]/N); % Cálculo de los factores de fase.
X=digitrevorder(X,4); % Reacomodo de la señal de salida.
for e=1:v, % Inicio de etapa.
    L=4^e; % Condiciones iniciales.
    n2=4^(v-e);
    if e==1, n2=0; end
    for k=0:L:(N-L), % Inicio del cálculo de las mariposas.
        n1=0;
        for n=0:(L/4)-1,
            a=n+k+1;
            b=(L/4)+a;
            c=(L/4)+b;
            d=(L/4)+c;
            A=X(a); % Carga valor 1.
            B=X(b)*W((1*n1*n2)+1); % Carga valor 2 * Factor de fase.
```

```
            C=X(c)*W((2*n1*n2)+1); % Carga valor 3* Factor de fase.
            D=X(d)*W((3*n1*n2)+1); % Carga valor 4* Factor de fase.
            X(a)=A+B+C+D; % Salida 1 de la mariposa.
            X(b)=(A-(j*B)-C+(j*D)); % Salida 2 de la mariposa.
            X(c)=(A-B+C-D); % Salida 3 de la mariposa.
            X(d)=(A+(j*B)-C-(j*D)); % Salida 4 de la mariposa.
            n1=n1+1; % Sig. factor de fase
        end
    end % Fin de mariposa.
end % Fin de etapa.
X=X/N;
end % Fin de función.
```

A.2.4. Radix-4 DIF inverso.

```
 %Algoritmo Radix-4 DIF inverso
function [X] = fftradix4dif(x)
N=length(x); % Long. de la señal de entrada.
v=log(N)/log(4); % Núm. de etapas.
N4=N/4;
X=x;
W=exp((2*pi*i)*[0:N-1]/N); % Cálculo de los factores de fase.
for e=v:-1:1, % Inicio de etapa.
    L=4^e; % Condiciones iniciales.
    n2=4^(v-e);
    if e==1, n2=0; end
    for k=0:L:(N-L), % Inicio del cálculo de las mariposas.
        n1=0;
        for n=0:(L/4)-1,
            a=n+k+1;
            b=(L/4)+a;
            c=(L/4)+b;
            d=(L/4)+c;
            A=X(a); % Carga valor 1.
            B=X(b); % Carga valor 2.
            C=X(c); % Carga valor 3.
            D=X(d); % Carga valor 4.
            X(a)=A+B+C+D; % Salida 1 de la mariposa.
            X(b)=(A-(j*B)-C+(j*D))*W((1*n1*n2)+1); % Salida 2 de la mariposa.
            X(c)=(A-B+C-D)*W((2*n1*n2)+1); % Salida 3 de la mariposa.
            X(d)=(A+(j*B)-C-(j*D))*W((3*n1*n2)+1); % Salida 4 de la mariposa.
            n1=n1+1; % Sig. factor de fase
        end
    end % Fin de mariposa.
end % Fin de etapa.
X=digitrevorder(X,4)/N; % Reacomodo de la señal de salida.
end % Fin de función.
```

A.3. Pruebas a los algoritmos.

A.3.1. Comparación con FFT de Matlab.

```
  %Pruebas para los algoritmos Radix
load yt; %Carga señal 1
load yes; %Carga canales de comunicación

alg='Radix-2 DIT';
YT=fftradix2dit(yt);
figure
subplot(2,1,1), graf2(yt,YT,'H_{yt}'); %Llama programa de graficación 2
```

```
title(alg)
YT1=fftradix2dit(yc1);
YT2=fftradix2dit(yc2);
YT3=fftradix2dit(yc3);
subplot(2,1,2), graf3(yc1,yc2,yc3,YT1,YT2,YT3); %Llama programa de graficación 3

alg='Radix-2 DIF';
YT=fftradix2dif(yt);
figure
subplot(2,1,1), graf2(yt,YT,'H_{yt}');
title(alg)
YT1=fftradix2dif(yc1);
YT2=fftradix2dif(yc2);
YT3=fftradix2dif(yc3);
subplot(2,1,2), graf3(yc1,yc2,yc3,YT1,YT2,YT3);

alg='Radix-4 DIT';
YT=fftradix4dit(yt);
figure
subplot(2,1,1), graf2(yt,YT,'H_{yt}');
title(alg)
YT1=fftradix4dit(yc1);
YT2=fftradix4dit(yc2);
YT3=fftradix4dit(yc3);
subplot(2,1,2), graf3(yc1,yc2,yc3,YT1,YT2,YT3);

alg='Radix-4 DIF';
YT=fftradix4dif(yt);
figure
subplot(2,1,1), graf2(yt,YT,'H_{yt}');
title(alg)
YT1=fftradix4dif(yc1);
YT2=fftradix4dif(yc2);
YT3=fftradix4dif(yc3);
subplot(2,1,2), graf3(yc1,yc2,yc3,YT1,YT2,YT3);

  %Graficación 2
function graf2(yt,YT,alg)
%alg = nombre del algoritmo
N2 = length(yt)/2;
a=pi/N2;
t=0:a:pi-a;
plot(t,abs(YT(1:N2)),'g')
hold on
YTM=abs(fft(yt));
plot(t,YTM(1:N2),'b-.')
legend(alg,'FFT Matlab','Location','Best')
ylabel('Amplitud')
xlabel('Frecuencia [Rad]')
end

  %Graficación 3
function graf3(y1,y2,y3,YT1,YT2,YT3)
%alg = nombre del algoritmo
N2 = length(y1)/2;
a=pi/N2;
t=0:a:pi-a;
plot(t,abs(YT1(1:N2)),'g')
hold on
plot(t,abs(YT2(1:N2)),'r')
plot(t,abs(YT3(1:N2)),'y')
YTM1=abs(fft(y1));
plot(t,YTM1(1:N2),'b-.')
YTM2=abs(fft(y2));
plot(t,YTM2(1:N2),'b-.')
```

```
YTM3=abs(fft(y3));
plot(t,YTM3(1:N2),'b-.')
legend('H_{c1}','H_{c2}','H_{c3}','Matlab','Location','Best')
ylabel('Amplitud')
xlabel('Frecuencia [Rad]')
end
```

A.3.2. Precisión del módulo.

```
  %Precisión Módulo
N=N+1;
yt=randn(1,2^N); % Generando la señal de entrada.
v=fft(yt); % Cálculo de FFT Matlab.

%Radix-2 DIT
YT=fftradix2dit(yt); % Cálculo del algoritmo.
iv=ifftradix2dit(YT); % Cálculo del algoritmo inverso.
efr2t=abs(v)-abs(YT); % Diferencia hacia adelante.
ebr2t=abs(yt)-abs(iv); % Diferencia hacia atrás.
eer2tf=0;
eer2tb=0;
for n=1:2^N,
    eer2tf=(efr2t(n)^2) + eer2tf; % Cálculo error cuadrático promedio hacia adelante.
    eer2tb=(ebr2t(n)^2)+ eer2tb; % Cálculo error cuadrático promedio hacia atrás.
end
epcr2tf(N)=(eer2tf/2^N);
epcr2tb(N)=(eer2tb/2^N);
dsfr2tp=0;
dsbr2tp=0;
for n=1:2^N,
    dsfr2tp=(efr2t(n)-epcr2tf(N))^2 + dsfr2tp; % Cálculo varianza adelante.
    dsbr2tp=(ebr2t(n)-epcr2tb(N))^2 + dsbr2tp; % Cálculo varianza atrás.
end
dsfr2t(N)=dsfr2tp/2^N;
dsbr2t(N)=dsbr2tp/2^N;

%Radix-2 DIF
YT=fftradix2dif(yt); % Cálculo del algoritmo.
iv=ifftradix2dif(YT); % Cálculo del algoritmo inverso.
efr2f=abs(v)-abs(YT);  % Diferencia hacia adelante.
ebr2f=abs(yt)-abs(iv); % Diferencia hacia atrás.
eer2ff=0;
eer2fb=0;
for n=1:2^N,
    eer2ff=(efr2f(n)^2) + eer2ff; % Cálculo error cuadrático promedio adelante.
    eer2fb=(ebr2f(n)^2)+ eer2fb; % Cálculo error cuadrático promedio atrás.
end
epcr2ff(N)=(eer2ff/2^N);
epcr2fb(N)=(eer2fb/2^N);
dsfr2fp=0;
dsbr2fp=0;
for n=1:2^N,
    dsfr2fp=(efr2f(n)-epcr2ff(N))^2 + dsfr2fp; % Cálculo varianza adelante.
    dsbr2fp=(ebr2f(n)-epcr2fb(N))^2 + dsbr2fp; % Cálculo varianza atrás.
end
dsfr2f(N)=dsfr2fp/2^N;
dsbr2f(N)=dsbr2fp/2^N;

if N<=6,
yt=randn(1,4^N);
v=fft(yt);

%Radix-4 DIT
YT=fftradix4dit(yt); % Cálculo del algoritmo.
```

```
iv=ifftradix4dit(YT); % Cálculo del algoritmo inverso.
efr4t=abs(v)-abs(YT);  % Diferencia hacia adelante.
ebr4t=abs(yt)-abs(iv); % Diferencia hacia atrás.
eer4tf=0;
eer4tb=0;
for n=1:4^N,
    eer4tf=(efr4t(n)^2) + eer4tf; % Cálculo error cuadrático promedio adelante.
    eer4tb=(ebr4t(n)^2) + eer4tb; % Cálculo error cuadrático promedio atrás.
end
epcr4tf(N)=(eer4tf/4^N); % Cálculo varianza hacia adelante.
epcr4tb(N)=(eer4tb/4^N); % Cálculo varianza hacia atrás.
dsfr4tp=0;
dsbr4tp=0;
for n=1:4^N,
    dsfr4tp=(efr4t(n)-epcr4tf(N))^2 + dsfr4tp;
    dsbr4tp=(ebr4t(n)-epcr4tb(N))^2 + dsbr4tp;
end
dsfr4t(N)=dsfr4tp/4^N;
dsbr4t(N)=dsbr4tp/4^N;

%Radix-4 DIF
YT=fftradix4dif(yt); % Cálculo del algoritmo.
iv=ifftradix4dif(YT); % Cálculo del algoritmo inverso.
efr4f=abs(v)-abs(YT);  % Diferencia hacia adelante.
ebr4f=abs(yt)-abs(iv); % Diferencia hacia atrás.
eer4ff=0;
eer4fb=0;
for n=1:4^N,
    eer4ff=(efr4f(n)^2) + eer4ff; % Cálculo error cuadrático promedio adelante.
    eer4fb=(ebr4f(n)^2) + eer4fb; % Cálculo error cuadrático promedio atrás.
end
epcr4ff(N)=(eer4ff/4^N);
epcr4fb(N)=(eer4fb/4^N);
dsfr4fp=0;
dsbr4fp=0;
for n=1:4^N,
    dsfr4fp=(efr4f(n)-epcr4ff(N))^2 + dsfr4fp; % Cálculo varianza adelante.
    dsbr4fp=(ebr4f(n)-epcr4fb(N))^2 + dsbr4fp; % Cálculo varianza atrás.
end
dsfr4f(N)=dsfr4fp/4^N;
dsbr4f(N)=dsbr4fp/4^N;
end

if N~=12, % Siguiente longitud de señal de entrada.
    precimodulo;
end

if N==12, % Graficación
    for n=1:12, t(n)=2^n; end
    for n=1:6, t1(n)=4^n; end

    figure
    loglog(t,epcr2tf,'g-+')
    hold on
    loglog(t,epcr2ff,'r-o')
    loglog(t1,epcr4tf,'b-*')
    loglog(t1,epcr4ff,'m-x')
    legend('R-2 DIT','R-2 DIF','R-4 DIT','R-4 DIF','Location','Best')
    title('Error Promedio Cuadrático hacia adelante en el Módulo')
    xlabel('Número de puntos')
    ylabel('Error')

    figure
    loglog(t,epcr2tb,'g-.+')
    hold on
```

```
    loglog(t,epcr2fb,'r-.o')
    loglog(t1,epcr4tb,'b-.*')
    loglog(t1,epcr4fb,'m-.x')
    legend('R-2 DIT','R-2 DIF','R-4 DIT','R-4 DIF','Location','Best')
    title('Error Promedio Cuadrático hacia atrás en el Módulo')
    xlabel('Número de puntos')
    ylabel('Error')
```

A.3.3. Precisión del ángulo de fase.

```
  %Precisión Ángulo de fase
N=N+1;
yt=randn(1,2^N); % Generando la señal de entrada.
v=fft(yt); % Cálculo de FFT Matlab.

%Radix-2 DIT
YT=fftradix2dit(yt); % Cálculo del algoritmo.
iv=ifftradix2dit(YT); % Cálculo del algoritmo inverso.
efr2t=angle(v)-angle(YT); % Diferencia hacia adelante.
ebr2t=angle(yt)-angle(iv); % Diferencia hacia atrás.
eer2tf=0;
eer2tb=0;
for n=1:2^N,
    eer2tf=(efr2t(n)^2) + eer2tf; % Cálculo error cuadrático promedio hacia adelante.
    eer2tb=(ebr2t(n)^2)+ eer2tb; % Cálculo error cuadrático promedio hacia atrás.
end
epcr2tf(N)=(eer2tf/2^N);
epcr2tb(N)=(eer2tb/2^N);
dsfr2tp=0;
dsbr2tp=0;
for n=1:2^N,
    dsfr2tp=(efr2t(n)-epcr2tf(N))^2 + dsfr2tp; % Cálculo varianza adelante.
    dsbr2tp=(ebr2t(n)-epcr2tb(N))^2 + dsbr2tp; % Cálculo varianza atrás.
end
dsfr2t(N)=dsfr2tp/2^N;
dsbr2t(N)=dsbr2tp/2^N;

%Radix-2 DIF
YT=fftradix2dif(yt); % Cálculo del algoritmo.
iv=ifftradix2dif(YT); % Cálculo del algoritmo inverso.
efr2f=angle(v)-angle(YT);  % Diferencia hacia adelante.
ebr2f=angle(yt)-angle(iv); % Diferencia hacia atrás.
eer2ff=0;
eer2fb=0;
for n=1:2^N,
    eer2ff=(efr2f(n)^2) + eer2ff; % Cálculo error cuadrático promedio adelante.
    eer2fb=(ebr2f(n)^2)+ eer2fb; % Cálculo error cuadrático promedio atrás.
end
epcr2ff(N)=(eer2ff/2^N);
epcr2fb(N)=(eer2fb/2^N);
dsfr2fp=0;
dsbr2fp=0;
for n=1:2^N,
    dsfr2fp=(efr2f(n)-epcr2ff(N))^2 + dsfr2fp; % Cálculo varianza adelante.
    dsbr2fp=(ebr2f(n)-epcr2fb(N))^2 + dsbr2fp; % Cálculo varianza atrás.
end
dsfr2f(N)=dsfr2fp/2^N;
dsbr2f(N)=dsbr2fp/2^N;

if N<=6,
yt=randn(1,4^N);
v=fft(yt);

%Radix-4 DIT
```

```
YT=fftradix4dit(yt); % Cálculo del algoritmo.
iv=ifftradix4dit(YT); % Cálculo del algoritmo inverso.
efr4t=angle(v)-angle(YT);  % Diferencia hacia adelante.
ebr4t=angle(yt)-angle(iv); % Diferencia hacia atrás.
eer4tf=0;
eer4tb=0;
for n=1:4^N,
    eer4tf=(efr4t(n)^2) + eer4tf; % Cálculo error cuadrático promedio adelante.
    eer4tb=(ebr4t(n)^2) + eer4tb; % Cálculo error cuadrático promedio atrás.
end
epcr4tf(N)=(eer4tf/4^N); % Cálculo varianza hacia adelante.
epcr4tb(N)=(eer4tb/4^N); % Cálculo varianza hacia atrás.
dsfr4tp=0;
dsbr4tp=0;
for n=1:4^N,
    dsfr4tp=(efr4t(n)-epcr4tf(N))^2 + dsfr4tp;
    dsbr4tp=(ebr4t(n)-epcr4tb(N))^2 + dsbr4tp;
end
dsfr4t(N)=dsfr4tp/4^N;
dsbr4t(N)=dsbr4tp/4^N;

%Radix-4 DIF
YT=fftradix4dif(yt); % Cálculo del algoritmo.
iv=ifftradix4dif(YT); % Cálculo del algoritmo inverso.
efr4f=angle(v)-angle(YT);  % Diferencia hacia adelante.
ebr4f=angle(yt)-angle(iv); % Diferencia hacia atrás.
eer4ff=0;
eer4fb=0;
for n=1:4^N,
    eer4ff=(efr4f(n)^2) + eer4ff; % Cálculo error cuadrático promedio adelante.
    eer4fb=(ebr4f(n)^2) + eer4fb; % Cálculo error cuadrático promedio atrás.
end
epcr4ff(N)=(eer4ff/4^N);
epcr4fb(N)=(eer4fb/4^N);
dsfr4fp=0;
dsbr4fp=0;
for n=1:4^N,
    dsfr4fp=(efr4f(n)-epcr4ff(N))^2 + dsfr4fp; % Cálculo varianza adelante.
    dsbr4fp=(ebr4f(n)-epcr4fb(N))^2 + dsbr4fp; % Cálculo varianza atrás.
end
dsfr4f(N)=dsfr4fp/4^N;
dsbr4f(N)=dsbr4fp/4^N;
end

if N~=12, % Siguiente longitud de señal de entrada.
    precifase;
end

if N==12, % Graficación
    for n=1:12, t(n)=2^n; end
    for n=1:6, t1(n)=4^n; end

    figure
    loglog(t,fepcr2tf,'g-+')
    hold on
    loglog(t,fepcr2ff,'r-o')
    loglog(t1,fepcr4tf,'b-*')
    loglog(t1,fepcr4ff,'m-x')
    legend('R-2 DIT','R-2 DIF','R-4 DIT','R-4 DIF','Location','Best')
    title('Error Promedio Cuadrático hacia adelante en la fase')
    xlabel('Número de puntos')
    ylabel('Error')

    figure
    loglog(t,fepcr2tb,'g-.+')
```

```
hold on
loglog(t,fepcr2fb,'r-.o')
loglog(t1,fepcr4tb,'b-.*')
loglog(t1,fepcr4fb,'m-.x')
legend('R-2 DIT','R-2 DIF','R-4 DIT','R-4 DIF','Location','Best')
title('Error Promedio Cuadrático hacia atrás en la fase')
xlabel('Número de puntos')
ylabel('Error')
```

A.3.4. Tiempo de cálculo en segundos.

```
    %Tiempo de cálculo
N=N+1;
yt=randn(1,2^N); % Generando la señal de entrada
%disp('Radix-2 DIT')
tic, % Inicia el cronómetro. R2-DIT
for k=1:10, % Número de iteraciones.
    YT=fftradix2dit(yt); % Función programada.
end % Fin de iteraciones.
tr2dit(N)=toc/10; % Para el cronómetro.
%err(yt,YT)

%disp('Radix-2 DIF')
tic, % Inicia el cronómetro. R2-DIF
for k=1:10, % Número de iteraciones.
    YT=fftradix2dif(yt); % Función programada.
end % Fin de iteraciones.
tr2dif(N)=toc/10; % Para el cronómetro.
%err(yt,YT)

%disp('FFT Matlab')
tic, % Inicia el cronómetro. FFT-Matlab.
for k=1:10, % Número de iteraciones.
    YT=fft(yt); % Función programada.
end % Fin de iteraciones.
tfft(N)=toc/10; % Para el cronómetro.

if N<=6,
yt=randn(1,4^N);
% disp('Radix-4 DIT')
tic, % Inicia el cronómetro. R4-DIT
for k=1:10, % Número de iteraciones.
    YT=fftradix4dit(yt); % Función programada.
end % Fin de iteraciones
tr4dit(N)=toc/10; % Para el cronómetro.
%err(yt,YT)

% disp('Radix-4 DIF')
tic, % Inicia el cronómetro. R4-DIF
for k=1:10, % Número de iteraciones.
    YT=fftradix4dif(yt); % Función programada.
end % Fin de iteraciones
tr4dif(N)=toc/10; % Para el cronómetro.
%err(yt,YT)
end
if N~=12,
    tical;
end

if N==12, % Graficación
figure
loglog(t,tr2dit,'g+-')
hold on
loglog(t,tr2dif,'ro-')
```

```
loglog(t1,tr4dit,'b*-')
loglog(t1,tr4dif,'mx-')
loglog(t,tfft,'kv-')
legend('R-2 DIT','R-2 DIF','R-4 DIT','R-4 DIF','Matlab','Location','Best')
title('Tiempo de cálculo en segundos')
xlabel('Número de puntos')
ylabel('Segundos')
```

A.3.5. Tiempo de cálculo en "mflops´´.

```
  %mflops
for N=1:12, % Cálculo de mflops para Radix-2
    mfr2f(N)=((5*(2^N)*log2(2^N))./(tr2dif(N)*10^6));
    mfr2t(N)=((5*(2^N)*log2(2^N))./(tr2dit(N)*10^6));
    mffft(N)=((5*(2^N)*log2(2^N))./(tfft(N)*10^6));
end
for N=1:6, % Cálculo de mflops para Radix-4
    mfr4t(N)=((5*(4^N)*log2(4^N))./(tr4dit(N)*10^6));
    mfr4f(N)=((5*(4^N)*log2(4^N))./(tr4dif(N)*10^6));
end

figure % Graficiación
loglog(t,mfr2t,'g+-')
hold on
loglog(t,mfr2f,'ro-')
loglog(t1,mfr4t,'b*-')
loglog(t1,mfr4f,'mx-')
loglog(t,mffft,'kv-')
legend('R-2 DIT','R-2 DIF','R-4 DIT','R-4 DIF','Matlab','Location','Best')
title('Velocidad de cálculo')
xlabel('Número de puntos')
ylabel('mFlops')

figure
semilogx(t,mfr2t,'g+-')
hold on
semilogx(t,mfr2f,'ro-')
semilogx(t1,mfr4t,'b*-')
semilogx(t1,mfr4f,'mx-')
%semilogx(t,mffft,'ko-')
legend('R-2 DIT','R-2 DIF','R-4 DIT','R-4 DIF','Location','Best')
title('Velocidad de cálculo')
xlabel('Número de puntos')
ylabel('mFlops')
```

Glosario

A

aliasing Es el efecto que causa que señales continuas distintas se tornen indistinguibles cuando son muestreadas digitalmente.

ARQ Automatic Retransmission Requests, Retransmisión Automática por Pedido., p. 9.

B

Beamforming Consiste en la formación de una onda de señal reforzada mediante es desfase en distintas antenas., p. 10.

D

DFT Discrete Fourier Transform, Transformada de Fourier Discreta.

downlink La dirección de la estación base a la estación del suscriptor., p. 9.

F

FDD Frequency Division Duplexing. Sistema dúplex en el cual la carga y descarga de datos usan diferentes frecuencias, pero normalmente son simultáneas., p. 9.

FEC Forward Error Correction, es un tipo de mecanismo de corrección de errores que permite su corrección en el receptor sin retransmisión de la información original., p. 14.

FFT Fast Fourier Transform, Transformada de Fourier Rápida., p. I.

H

handoff Transferir el servicio de una estación base a otra cuando la calidad del enlace es insuficiente., p. 13.

L

LOS line-of-sight, linea-de-vista., p. 7.

M

Mbps Mega bits por segundo., p. 9.

mflops Millones de operaciones de punto flotante por segundo., p. 62.

Multipath Multitrayecto., p. 9.

Multiplexado Espacial Consiste en el multiplexado de una señal de mayor ancho de banda en señales de menor ancho de banda iguales transmitidas desde diferentes antenas., p. 10.

N

NLOS non-line-of-sight, sin-linea-de-vista., p. 8.

O

OFDM Orthogonal Frequency Division Multiplexing, Multiplexado por División de Frecuencia Ortogonal., p. 1.

OFDMA Orthogonal Frequency Division Multiple Access, Acceso Múltiple por División de Frecuencia Ortogonal., p. 10.

Q

QAM Quadrature Amplitude Modulation, Modulación de Amplitud en Cuadratura., p. 9.

QoS Quality-of-service, son las tecnologías que garantizan la transmisión de cierta cantidad de datos en un tiempo dado., p. 10.

QPSK Quadrature Phase-Shift Keying, Modulación por Desplazamiento de Fase., p. 15.

T

TDD Time Division Duplexing. Un sistema dúplex en el cual la carga y descarga de datos ocurre en diferentes tiempos,pero pueden compartir la misma frecuencia., p. 9.

TDM Time Division Multiplexing, Multiplexado por División de Tiempo., p. 10.

U

uplink La dirección de la estación del suscriptor a la estación base., p. 9.

W

WiMAX Worldwide Interoperability For Microwave Access, Red Inalámbrica de Interoperabilidad Mundial para Acceso por Microondas., p. I.

WMAN Wireless Metropolitan Area Network, Red Inalámbrica de Área Metropolitana., p. 1.

Bibliografía

[1] M. R. Arroyo. Metodología de Desarrollo de sistemas de Procesamiento Digital de Señales. Tesis, U.N.A.M., Facultad de Ingeniería, 1996. 4

[2] Vianney Muñoz Jiménez Jorge Arturo Palacios Castillo. Implantación en Aritmética entera de algoritmos DSP: Reconocedor de Palabras Ailsadas. Tesis, U.N.A.M., Facultad de Ingeniería. 4

[3] Jeffrey G. Andrews Arunabha Ghosh Rias Muhamed. *Fundamentals of WiMAX: understanding broadband wireless networking*. Prentice Hall, 2007. 8, 20

[4] IEEE computer Society, the IEEE Microwave Theory, and Tenchniques Society. IEEE Standard for local and metropolitan area networks. Part 16: Air Interface for Fixed Broadband Wireless Access Systems. 2001. 11

[5] IEEE computer Society, the IEEE Microwave Theory, and Tenchniques Society. IEEE Std 802.16a^{TM}-2003. 2003. 11

[6] IEEE computer Society, the IEEE Microwave Theory, and Tenchniques Society. IEEE Std 802.16e^{TM}-2005. 2005. 11

[7] Anritsu Company. WiMAX Wall Poster (Instruction Sheet), Septiembre 2008. 12

[8] The International Engineering Consortium. OFDM for mobile data communications. *WEB ProForum Tutorial*, pages 1–18. 17

[9] Ove Edfors Magnus Sandell Jan-Jaap van de Beek Daniel Landström Frank Sjöberg. An Introduction to Orthogonal Frequency Division Multiplexing. 1996. 17

[10] Ramjee Prasad. *OFDM for Wireless Communications Systems*. Artech House, 2004. 18

[11] Pei-Yun Tsai Tsung-Hsueh Lee and Tzi-Dar Chiueh. Power-Efficient Continous-Flow Memory-Based FFT Processor for WiMAX OFDM Mode. In *2006 International Symposium on Intelligent Signal Processing and Communication System*, pages 622–625, 2006. 23

[12] Shousheng He and Mats Torkelson. Design and Implementationof a 1024-point Pipeline FFT Processor for OFDM. In *IEEE 1998 Costum Integrated Circuits Conference.*, pages 131–134, S-22100 Lund, Sweden, 1998. Department of Applied Electronics, Lund University. 24

[13] E. Oran Brigham. *The Fast Fourier Transform.* Prentice-Hall, Inc., 1974. 25

[14] José Francisco Peña Verduzco. Generación de código para el análisis de señales analógicas y digitales mediante el TMS320C2X DSP. Master's thesis, Universidad de Colima, Facultad de Ingeniería Mecánica y Eléctrica, Agosto 1999. 26

[15] David J. DeFatta Joseph G. Lucas and William S. Hodgkiss. *Digital Signal Processing: A System Design Approach.* John Wiley & Sons, 1988. 28, 37

[16] John G. Proakis and Dimitris G. Manolakis. *Digital Signal Processing: Principles, Algorithms & Applications.* . Prentice-Hall, 3^{ra} edition, 1996. 33

[17] Matteo Frigo and Steven G. Johnson. The design and implementation of FFTW3. *Proceedings of the IEEE*, 93(2):216–231, 2005. Special issue on "Program Generation, Optimization, and Platform Adaptation". 64, 66

[18] Matworks Inc. *User's Guide.* 3 Apple Hill Drive, Natick, MA, 2004-2006. 66

[19] P. Duhamel and H. Hollmann. 'Split-Radix' FFT algorithm. *ELECTRONICS LETTERS*, 20(1):14–16, Enero 1984. 66

[20] J. W. Cooley and J.W. Tukey. An algorithm for machine computation of complex fourier series. *Math. Comput.*, (19):297 – 301, 1965.

[21] Young jin Moon and Young il Kim. A mixed-radix 4-2 butterfly with simple bit revering for ordering the output sequences. In *Advanced Communication Technology*, volume 3, pages 4pp.–1774, Daejeon, Korea, Febrero 2006.

[22] Yunho Jung Yonji Tak Jaeseok Kim Junhyun Park Dongkyu Kim Hyuncheol Park. Efficient FFT algorithm for OFDM modulation. In *Electrical and Electronic Technology, TENCON, IEEE Region 10 Internacional Conference*, volume 2, pages 676–678 vol.2, Seoul, 120-749, Korea, 2001.

[23] Shousheng He and Mats Torkelson. A new approach to pipeline FFT processor. In *The 10th International Parallel Processing Symposium, 1996., Proceedings of IPPS '96*, pages 766–770, S-22100 Lund, Sweden, Abril 1996. Department of Applied Electronics, Lund University.

[24] Shousheng He and Mats Torkelson. Designing pipeline FFT processor for OFDM (de)modulation. In *URSI International Symposium on Signals, Systems, and Electronics, 1998. ISSSE 98.*, pages 257–262, S-22100 Lund, Sweden, Octubre 1998. Department of Applied Electronics, Lund University.

[25] Muhammad Nadeem Khan Sabir Ghauri. The WiMAX 802.16e physical layer model. In *IET International Conference on Wireless, Mobile and Multimedia Networks, 2008*, pages 117 – 120, 2008.

[26] Deepak Pareek. *Wimax: Taking Wireless to the Max*. Auerbach Publications, 2006.

[27] IEEE computer Society, the IEEE Microwave Theory, and Tenchniques Society. IEEE Std 802.16j^{TM}-2009. 2009.