Universidad Nacional de Córdoba

Facultad de Ciencias Exactas, Físicas y Naturales

Carrera Ingeniería Electrónica

Proyecto integrador para la obtención del título de grado de ingeniero electrónico

---

SIMULACIÓN E IMPLEMENTACIÓN EN FPGA DE ECUALIZADOR FRACCIONALMENTE ESPACIADO EN EL DOMINIO DE LA FRECUENCIA CON ALGORITMO LMS PARA ADAPTACIÓN DE COEFICIENTES

Alumnos

Brignone, Matías Nicolás y Rodríguez, Lucía Fernanda

Director

Dr. Ing. Hueda, Mario Rafael

Co-Director

Dr. Ing. Pola, Ariel Luis

Córdoba, República Argentina

Abril / 2018

Universidad Nacional de Córdoba

Facultad de Ciencias Exactas, Físicas y Naturales

Escuela de Ingeniería Electrónica

El Tribunal Evaluador reunido en este acto y luego de haber aprobado la Solicitud de Aprobación de Tema y efectuado las distintas instancias de correcciones del Informe del Proyecto Integrador para la obtención del Título de Grado "Ingeniero Electrónico" y cumpliendo con el Reglamento correspondiente, declaran el Informe Final de los estudiantes **Brignone, Matías Nicolás** y **Rodríguez, Lucía Fernanda** como "aceptado sin correcciones" y la defensa oral Aprobada. Por lo tanto, luego de haber tenido en cuenta los aspectos de evaluación que indica el Reglamento, el Proyecto Integrador se considera Aprobado.

Se firma el Acta de Examen correspondiente y se distribuyen los ejemplares impresos.

Firma y aclaración del Tribunal Evaluador

Fecha:

Dedicatoria

Para nuestras familias...

Agradecimientos

A nuestros padres, madres y hermanos, por su incondicional apoyo a lo largo de toda la carrera.

A nuestros directores, Ariel Pola y Mario Hueda, por la excelente predisposición, la confianza y todo el soporte brindado que hizo posible este proyecto.

A nuestros amigos y futuros colegas, quienes hicieron de estos años de estudio una experiencia más placentera.

A la Fundación Fulgor y a la Fundación Tarpuy, junto a todo su personal, por las oportunidades y enseñanzas compartidas.

A la Facultad de Ciencias Exactas, Físicas y Naturales de la Universidad Nacional de Córdoba, por la oportunidad de realizar esta carrera de grado.

Resumen

Al transmitir una señal digital, los bits enviados sufren distorsiones debido a los efectos del canal, pudiendo provocar detecciones erróneas en el receptor. Uno de los principales problemas que surgen es la Interferencia Intersímbolo, en la que símbolos adyacentes se interfieren mutuamente debido a una dispersión en el tiempo de los mismos. Para mitigar estos efectos, se utilizan diferentes técnicas de ecualización que buscan contrarrestar los efectos del canal para reducir la probabilidad de error.

En este trabajo se presenta una implementación eficiente en FPGA de un Ecualizador Fraccionalmente Espaciado en el dominio de la frecuencia, utilizando el algoritmo LMS para la adaptación de coeficientes, la cual tiene un menor costo a nivel de hardware que su equivalente en el dominio del tiempo.

Se realizan estudios teóricos del comportamiento de los algoritmos a implementar, para luego efectuar simulaciones tanto en punto flotante como en punto fijo de los mismos. De esta forma se obtienen los parámetros óptimos para un correcto desempeño del sistema.

Además del ecualizador propiamente dicho, se implementan módulos adicionales que complementan su funcionamiento, como así también una plataforma de verificación que permite controlar y configurar el sistema.

Una vez implementado, se extraen los datos de interés de la FPGA y a partir de ellos se realizan las gráficas correspondientes, a los fines de verificar su apropiado comportamiento. Finalmente, se contrastan los resultados con los obtenidos en simulación y se trazan las respectivas curvas de BER.

Áreas Temáticas del Proyecto Integrador: comunicaciones, digitales.

Asignaturas: Teoría de Señales y Sistemas Lineales, Teoría de las Comunicaciones, Electrónica Digital II.

Palabras Claves: ecualización adaptiva, dominio de la frecuencia, algoritmo lms, fpga, fse.

Abstract

When a digital signal is transmitted, the bits get distorted due to the channel effects, which may introduce errors at the receiver. One of the main problems is the Intersymbol Interference, in which adjacent symbols interfere with each other because of the time spreading of the pulse. To mitigate this effects, different equalization techniques are used, attempting to undo the effect of the channel and reduce the error probability.

In this work we present an efficient FPGA implementation of a Fractionally Spaced Equalizer in the frequency domain, using the LMS algoritm for coefficient adaptation, which is less complex than its time domain implementation.

We perform a theoretical analysis of the algorithms to implement, in order to simulate their floating and fixed point behavior. Thus, it is possible to determine the optimum parameters for the correct performance of the system.

Besides the equalizer itself, we implement aditional modules that complement its operation. We also implement a verification platform that allows us to control and configure the whole system.

Once implemented, we extract data of interest from the FPGA and we plot the corresponding graphs, in order to verify its correct behavior. Finally, we compare these results with the ones obtained from the simulations and plot the BER curves.

Key Words: adaptive equalization, frequency domain, lms algorithm, fpga, fse.

Índice

Lista de Tablas

Lista de Figuras

Lista de Acrónimos

BER Bit Error Rate

BLMS Block Least Mean Square

BPSK Binary Phase Shift Keying

BRAM Block Random Access Memory

DFT Discrete Fourier Transform

DGD Differential Group Delay

DIT Decimation In Time

DQPSK Differential Quadrature Phase Shift Keying

DSP Digital Signal Processing

FFT Fast Fourier Transform

FIR Finite Impulse Response

FPGA Field Programmable Gate Array

FSE Fractionally Spaced Equalizer

FSK Frequency Shift Keying

GNG Gaussian Noise Generator

HDL Hardware Description Language

IFFT Inverse Fast Fourier Transform

ISI InterSymbol Interference

LFSR Linear Feedback Shift Register

LMS Least Mean Square

LUT Look Up Table

MSE Mean Squared Error

PAM Pulse Amplitude Modulation

PMD Polarization Mode Dispersion

PRBS Pseudo Random Binary Sequence

PSK Phase Shift Keying

PSP Principal State of Polarization

QAM Quadrature Amplitude Modulation

RC Raised Cosine

RF Register File

RRC Root Raised Cosine

RTL Register Transfer Level

SER Symbol Error Rate

SNR Signal to Noise Ratio

Capítulo 1

Introducción

El presente informe tiene por objetivo describir el trabajo realizado como Proyecto Integrador de la carrera de Ingeniería Electrónica de la Facultad de Ciencias Exactas, Físicas y Naturales (*FCEFyN*) de la Universidad Nacional de Córdoba (*UNC*).

En la primera parte se desarrolla la formulación del problema que el proyecto integrador pretende resolver, continuando luego con los objetivos planteados y la metodología de trabajo a seguir.

En la actualidad, existe una enorme demanda por anchos de banda y velocidades de transmisión cada vez mayores, con una fuerte tendencia creciente que se mantiene desde hace años y que se acentuará aún más en los años venideros. Esta demanda se debe principalmente a la proliferación de los servicios en la nube, acompañada por un gran incremento no sólo en el número de usuarios sino también en el consumo de cada uno de ellos, lo cual provoca que los centros de datos deban mover enormes volúmenes de datos a gran velocidad. Si bien la industria de las telecomunicaciones concentra la mayor parte de esta creciente demanda, otros sectores como el aeroespacial o militar también realizan su contribución.

Frente a las necesidades mencionadas, la fibra óptica es el medio de transmisión por excelencia que se presenta para soportar esa demanda, por lo que también hay un gran incremento en las transmisiones por fibra óptica. Su uso se debe a las grandes ventajas que presenta, principalmente su extremadamente alto ancho de banda (muchísimo mayor que el de un cable de cobre por ejemplo), pero también su capacidad para transmitir datos por distancias mayores con bajas pérdidas por atenuación, su inmunidad frente a interferencias electromagnéticas y su pequeño tamaño.

El crecimiento en el uso de la fibra óptica y los avances en su tecnología se ha visto acompañado y posibilitado por un incremento en la importancia de las comunicaciones digitales. Desde hace ya varias décadas, las

comunicaciones digitales han tomado una enorme importancia debido a las considerables ventajas que presentan frente a una comunicación analógica, aún cuando lo que se desea transmitir es una señal inherentemente analógica (en cuyo caso debe ser representada mediante una secuencia de bits con una determinada precisión).

El principal beneficio de las comunicaciones digitales radica en el hecho de poder aplicar algoritmos de Procesamiento Digital de Señales (*Digital Signal Processing - DSP*) y técnicas de codificación, permitiendo así incrementar drásticamente las velocidades de transmisión que un determinado medio puede soportar, como así también mejorar su desempeño ante diferentes interferencias. Además, los avances en el diseño y fabricación de circuitos integrados permiten que el procesamiento digital sea conveniente también desde un punto de vista económico.

1.1. Motivaciones

Teniendo en consideración la creciente demanda, en este proyecto el enfoque se orienta a potenciales aplicaciones en transmisiones de alta velocidad por fibra óptica, en las que los bits de datos, representados mediante pulsos de luz, se dispersan en el tiempo a medida que viajan a lo largo de la fibra óptica, pudiendo interferir unos con otros y causar lo que se denomina Interferencia Intersímbolo (*Intersymbol Interference - ISI*). Esto no suponía un problema cuando las transmisiones eran a velocidades más bajas, ya que había una mayor separación entre bits, pero a medida que las velocidades se fueron incrementando, la Interferencia Intersímbolo se convirtió en un factor crítico al momento de determinar el máximo ancho de banda admisible.

La dispersión de los pulsos de luz en la fibra óptica, y por lo tanto la Interferencia Intersímbolo, se debe principalmente a dos fenómenos: dispersión cromática y Dispersión por Modo de Polarización (*Polarization Mode Dispersion - PMD*).

La dispersión cromática se debe al hecho de que la luz viaja por la fibra a diferentes velocidades de acuerdo a su longitud de onda, lo cual resulta problemático debido a que la fuente generadora de pulsos de luz no suele contener una única longitud de onda, sino que está compuesta de varias longitudes de onda en torno a una central. Para reducir estos efectos, es posible operar en longitudes de onda en las que la dispersión cromática es menor, como así también utilizar una fuente de luz que esté compuesta por la menor cantidad de longitudes de onda posible.

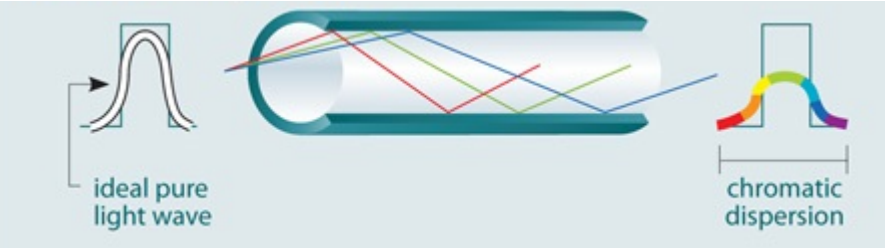


Figura 1.1: *Dispersión cromática en fibra óptica.*

La PMD es la que mayor dispersión causa en la fibra óptica y está asociada a la polarización de la luz, ya que la misma es una onda electromagnética cuyo campo eléctrico puede estar alineado con el eje horizontal (polarización horizontal), con el eje vertical (polarización vertical), denominándose estos modos Estado de Polarización Principal (*Principal State of Polarization - PSP*), o puede ser una composición de ambos en caso de que no esté alineado con ninguno de los ejes. A su vez, la velocidad con la que los pulsos se propagan a lo largo de la fibra óptica depende del índice de refracción de la misma.

Idealmente, en una fibra perfectamente simétrica, la velocidad de propagación no dependería de la polarización de la luz ya que el índice de refracción sería el mismo independientemente del modo de polarización. Sin embargo, esto no ocurre en la realidad y no se tienen los mismos índices de refracción en las direcciones horizontales y verticales, lo cual da como resultado que la luz con polarización horizontal se propague a una velocidad diferente que la que tiene una polarización vertical, introduciéndose así un retardo entre ambas polarizaciones denominado Retardo Diferencial de Grupo (*Differential Group Delay - DGD*), produciendo un estiramiento del pulso. Este fenómeno puede apreciarse en la Fig. 1.2, en la que se observa claramente cómo el pulso se ensancha en el tiempo debido al retardo existente entre ambas polarizaciones.

La PMD empeora a medida que las distancias recorridas son mayores, varía a lo largo de la fibra y ocurre por múltiples causas, como asimetrías en el núcleo de la fibra, fuerzas externas aplicadas sobre ella, curvas y torceduras en su disposición al instalarla, cambios de temperatura, o vibraciones mecánicas (debidas por ejemplo a un tren que pasa cerca). Por lo tanto, puede verse que la ISI causada por PMD tiene una naturaleza aleatoria y dependiente del tiempo.

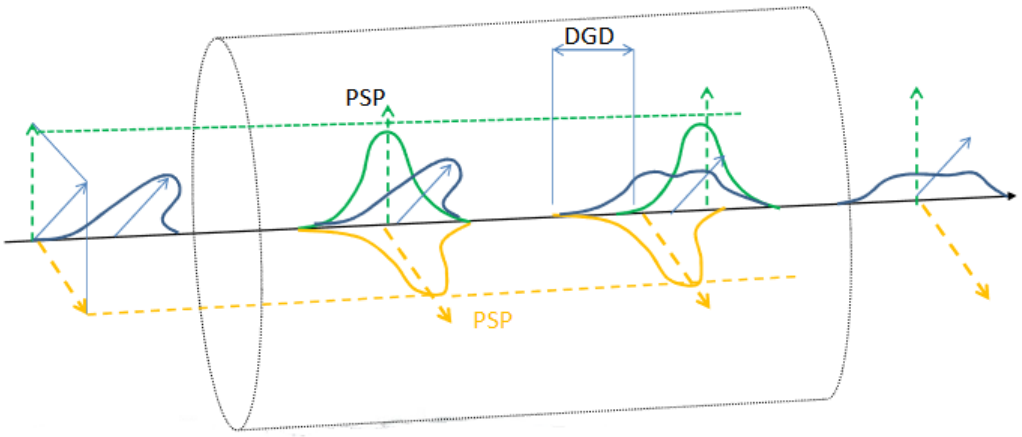


Figura 1.2: *Polarization Mode Dispersion (PMD) en fibra óptica.*

1.2. Formulación del Problema

A medida que se incrementa la velocidad de transmisión, se exacerban notablemente los efectos del canal sobre la señal. Por ello, para permitir transmisiones a tasas superiores a 100Gbps, es necesario corregir y mitigar estos efectos que el canal produce en las señales transmitidas, los cuales dependen del tipo de medio por el que se realice la transmisión (cable coaxial, fibra óptica, aire, etc.).

Uno de los principales problemas que surge es la ya mencionada Interferencia Intersímbolo, en la que el canal distorsiona la señal provocando un estiramiento en el tiempo de los símbolos transmitidos. Como consecuencia de esto, símbolos adyacentes al que se desea muestrear en un momento dado, contribuyen al valor medido pudiendo generar una detección errónea. Las causas de la ISI son muy diversas y dependen del medio de transmisión utilizado, explicándose el caso de la fibra óptica en la Sección 1.1.

Para mitigar los efectos de la Interferencia Intersímbolo, se utilizan diferentes técnicas de ecualización que buscan contrarrestar en el receptor los efectos del canal para reducir lo máximo posible la probabilidad de error.

En numerosas situaciones, no es posible solucionar el problema empleando un ecualizador con coeficientes conocidos y estáticos debido a la naturaleza aleatoria y dependiente del tiempo del canal, como el caso de la PMD en la fibra óptica. Por lo tanto, resulta necesario utilizar técnicas de adaptación que permitan minimizar el error entre la señal ecualizada y la salida del detector sin conocer el canal de transmisión. Dichos ecualizadores se denominan **Ecualizadores Adaptivos**.

De esta forma, se puede observar que el ecualizador realiza dos grandes

tareas. Por un lado lleva a cabo una operación de filtrado utilizando los valores actuales de sus coeficientes, contrarrestando así los efectos del canal, es decir, la ecualización propiamente dicha. Por otro lado, se realiza un proceso de adaptación de coeficientes mediante el cual se modifican los valores de los mismos de acuerdo al error estimado.

Estas operaciones de filtrado y adaptación de coeficientes pueden ser efectuadas en el dominio del tiempo o en el dominio de la frecuencia. Cuando la correcta compensación de la ISI requiere que el ecualizador tenga un elevado número de coeficientes, la complejidad de hardware en el dominio del tiempo aumenta rápidamente mientras mayor sea la cantidad de coeficientes, resultando poco práctica su implementación. Es por ello que en dichos casos se recurre a realizar una ecualización en el dominio de la frecuencia, lo cual permite reducir considerablemente la cantidad de operaciones necesarias para el procesamiento de las muestras.

Teniendo esto en consideración surge el proyecto propuesto, en el cual se diseña, simula e implementa en FPGA un Ecualizador Fraccionalmente Espaciado (*Fractionally Spaced Equalizer - FSE*) en el Dominio de la Frecuencia, utilizando el algoritmo LMS (*Least Mean Square*) para la adaptación de coeficientes, teniendo una potencial aplicación en la compensación de ISI debida a la PMD en fibras ópticas.

Debido a que el ecualizador formaría parte de un sistema de comunicaciones con otros módulos adicionales, para verificar su correcto funcionamiento es necesario probarlo en el marco de un sistema más completo. En virtud de esto, resulta necesario también llevar a cabo el diseño, simulación e implementación en FPGA de una plataforma de verificación que acompaña al ecualizador propiamente dicho.

1.3. Objetivos

1.3.1. Generales

El objetivo es, habiendo realizado los estudios teóricos necesarios, diseñar, simular e implementar en FPGA un Ecualizador Fraccionalmente Espaciado en el dominio de la frecuencia utilizando el algoritmo LMS para la adaptación de coeficientes, y realizar las pruebas de funcionamiento en un sistema de comunicaciones completo, agregando no sólo los módulos principales para realizar la transmisión (transmisor, modulador, contador de errores) sino también aquellos bloques que permiten simular el comportamiento del canal de comunicaciones para evaluar el funcionamiento del

ecualizador frente a los problemas que afectan a la transmisión de datos.

1.3.2. Particulares

- Estudiar el algoritmo BLMS (*Block Least Mean Square*) para la adaptación de coeficientes.
- Simular el funcionamiento de un ecualizador adaptivo en el dominio del tiempo utilizando el algoritmo BLMS para la adaptación de coeficientes.
- Estudiar el método Overlap&Save para realizar el filtrado en el dominio de la frecuencia.
- Simular un ecualizador adaptivo realizando el filtrado en el dominio de la frecuencia, utilizando el método de Overlap&Save.
- Simular en un lenguaje de alto nivel un ecualizador adaptivo realizando tanto el filtrado como la adaptación de coeficientes en el dominio de la frecuencia.
- Utilizar el paquete MyHDL de Python durante la verificación funcional de la implementación de cada bloque.
- Diseñar en RTL el módulo ecualizador en el dominio de la frecuencia verificando su correcto funcionamiento a nivel de síntesis y timing.
- Implementar en FPGA un ecualizador adaptivo en el dominio de la frecuencia, utilizando el algoritmo LMS para la adaptación de coeficientes.
- Desarrollar una plataforma de verificación que permita configurar y controlar el sistema.

1.4. Flujo de Trabajo

Para llevar a cabo el desarrollo del proyecto, se debe definir un flujo de trabajo tentativo a seguir durante su realización, el cual puede observarse en la Fig. 1.3.

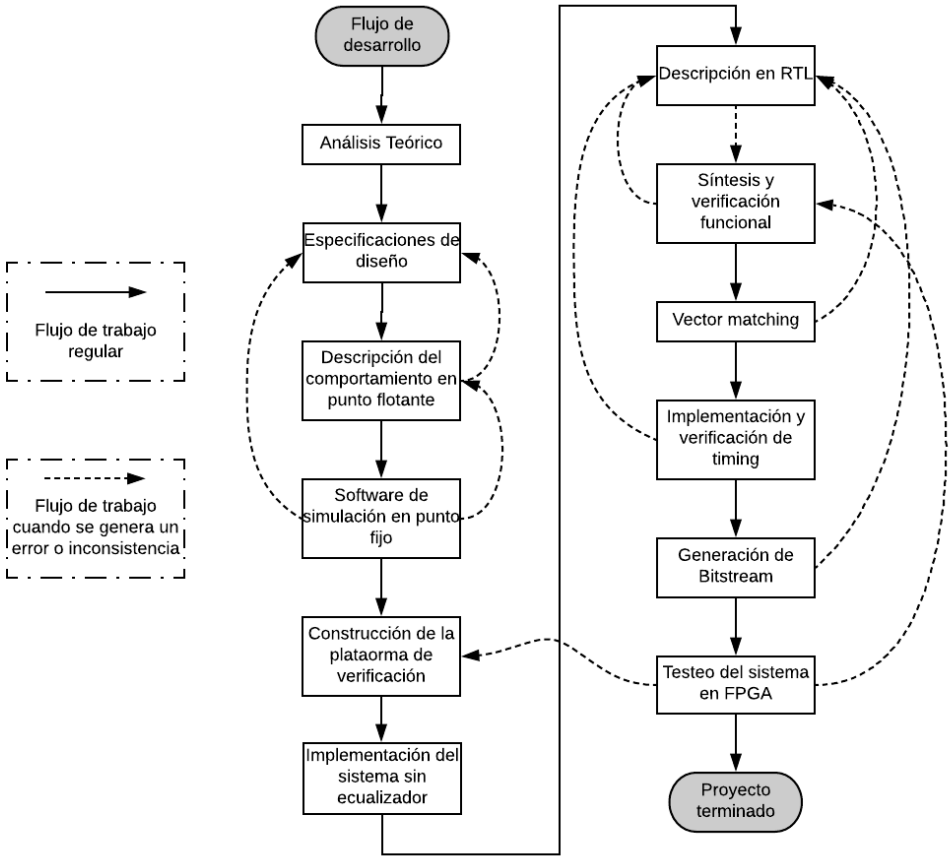


Figura 1.3: *Flujo de trabajo.*

Como se puede observar, el primer paso consiste en la realización de un análisis teórico exhaustivo del módulo a implementar y de los conceptos asociados al mismo, analizando los fundamentos y el comportamiento de las diferentes técnicas y algoritmos desde un punto de vista teórico, para garantizar así una correcta comprensión de lo que se desea implementar.

Luego se deben establecer las especificaciones que se pretenden lograr y los objetivos que se quieren alcanzar, debiendo los mismos ser realistas teniendo en cuenta los recursos y tiempos con los que se contará a la hora de llevar a cabo el proyecto.

Ya establecidas las especificaciones, se procede con la realización de una descripción del comportamiento del sistema a implementar en punto flotante, considerando tanto el ecualizador propiamente dicho como así también todos los módulos adicionales que son necesarios para probar su correcto funcionamiento. Para realizar este paso, se llevan a cabo simulaciones utilizando un lenguaje de alto nivel. En el caso de este proyecto, se utilizó Python y MATLAB para llevar a cabo estas simulaciones con todas las especificaciones planteadas, definiéndose además algunos de los parámetros propios del ecualizador.

Luego de que se ha verificado el correcto funcionamiento en punto flotante, se procede con la simulación del sistema en punto fijo, esta vez empleando solamente Python. Durante estas simulaciones se determinan los valores y las resoluciones óptimas de las diferentes variables y parámetros que se tendrán en el sistema. En caso de que no sea posible lograr un correcto funcionamiento en punto fijo, será necesario volver al paso anterior y ajustar el comportamiento del algoritmo en punto flotante.

Si por el contrario, se consigue un adecuado comportamiento, es posible proceder con la descripción del módulo a nivel RTL (*Register Transfer Level*), empleando un Lenguaje de Descripción de Hardware (*Hardware Description Language - HDL*), para comenzar a diseñar lo que se implementará en FPGA. Cuando se trabaja en RTL, se está trabajando en un nivel de abstracción en el que se modela un circuito digital síncrono en términos del flujo de señales entre registros y de las operaciones lógicas que se llevan a cabo sobre dichas señales. En el caso de este proyecto se utilizó el lenguaje Verilog.

Cabe mencionar que primero se lleva a cabo el desarrollo de la plataforma de verificación y de todos los módulos complementarios del sistema, y una vez que esa parte del proyecto funciona correctamente, se procede con la descripción en RTL del ecualizador propiamente dicho.

Antes de determinar si la descripción a nivel RTL funciona correctamente, o al menos en simultáneo, se debe verificar que el módulo descripto sea sintetizable. La síntesis consiste en la conversión de la descripción a nivel RTL, en la que se describe el comportamiento deseado del sistema, en una representación optimizada a nivel de compuertas lógicas (*gate-level*), la cual podría ser implementada en una FPGA (en caso de que se cumplan otros requerimientos adicionales).

El proceso de síntesis se lleva a cabo utilizando las denominadas *Herramientas de Síntesis*, que se encargan de llevar a cabo la conversión empleando también librerías de celdas, que incluyen desde simples compuertas lógicas (AND, OR, NOT, etc.), hasta celdas mas complejas como sumadores, multiplexores, flip-flops, y muchas otras.

Para comprobar si el diseño a nivel RTL se comporta de la manera esperada, se lleva a cabo un proceso que se denomina Vector Matching o Coverificación, el cual consiste en comparar las salidas del módulo obtenidas mediante el simulador en punto fijo con las obtenidas en el simulador a nivel RTL.

Tradicionalmente, para realizar este proceso se guardan en archivos los resultados de Python y luego desde el testbench realizado en Verilog se

levantan dichos archivos, pudiéndose comparar ambas salidas de manera directa. Sin embargo, en este flujo de trabajo se propone eliminar la necesidad de manejar archivos de datos al momento de llevar a cabo la tarea de Vector Matching y demás interacciones entre simulaciones en Python y descripción RTL. Por ello es que se decide utilizar el paquete MyHDL, el cual permite utilizar Python como un lenguaje de descripción y verificación de hardware, incluyendo características de cosimulación que permiten verificar el diseño RTL a través de Python.

Si al realizar el proceso de Vector Matching no se obtienen las coincidencias esperadas en los resultados, es necesario volver a una etapa anterior del flujo de trabajo, ya sea a la descripción en RTL o a la simulación en punto fijo. Si en cambio se llega a los resultados esperados, se procede con la *Implementación*.

Durante la implementación, el diseño sintetizado se mapea en recursos específicos de la FPGA que se utilizarán (LUTs, Flip-Flops, BRAMs, etc.) y luego tiene lugar lo que se denomina *Place and Route*, que suele ser la parte más problemática a la hora de la implementación, ya que se encarga de determinar la ubicación de los recursos a utilizar y de la interconexión de los mismos, y es en este punto donde se pueden presentar los problemas de *Timing* sobre los cuales se profundiza más adelante.

En caso de que a la hora de la implementación la herramienta reporte que hubo problemas de timing, se debe regresar a la etapa de descripción a nivel RTL para solucionar el problema. Si la implementación se lleva a cabo correctamente, se procede a la generación del bitstream, que es el archivo que se cargará en la FPGA para programarla.

Llegado este punto, ya es posible probar el funcionamiento real del sistema en la FPGA, para lo cual se construye una plataforma de verificación que permite configurar el sistema y extraer diferentes datos, a partir de los cuales se determina si está funcionando correctamente. Si las pruebas efectuadas en FPGA no son favorables, se deberán analizar los posibles módulos que podrían estar causando problemas y revisarlos en etapas anteriores del flujo de trabajo. Si por el contrario las pruebas arrojan los resultados esperados, finaliza el flujo de trabajo y el proyecto se da por terminado satisfactoriamente.

1.5. Organización del Informe

El desarrollo del informe continúa con el Marco Teórico, en el cual se exponen todos los conceptos y desarrollos matemáticos necesarios para

comprender y fundamentar las posteriores implementaciones prácticas.

El marco teórico comienza con el Capítulo 2, en el que partiendo de los bloques básicos que integran un sistema de comunicaciones digitales tradicional, se introduce el problema de la Interferencia Intersímbolo y cómo solucionarla mediante diferentes técnicas de Ecualización.

En el Capítulo 3 se presenta un algoritmo de adaptación alternativo llamado BLMS, el cual favorece el procesamiento en paralelo de las muestras de entrada y permite una eficiente implementación.

Luego, en el Capítulo 4 se introduce el concepto de procesamiento en el dominio de la frecuencia para implementar un algoritmo equivalente al LMS/BLMS en el dominio del tiempo.

Ya cubiertos todos los conceptos teóricos necesarios, se da comienzo al Marco Metodológico, en el cual se desarrolla concretamente qué se realizó en el Proyecto Integrador y cómo se lo llevó a cabo, incluyendo simulaciones, diseño e implementación.

Se inicia en el Capítulo 5, en el cual se llevan a cabo simulaciones de los algoritmos de adaptación de coeficientes analizados desde un punto de vista puramente teórico en los capítulos anteriores, variando sus diferentes parámetros para analizar su comportamiento.

A continuación, se comienza con la etapa de diseño en el Capítulo 6, en el cual se plantea la arquitectura del proyecto y nuevamente se realizan simulaciones, pero esta vez de todo el sistema propuesto, a los fines de determinar los valores óptimos de todos los parámetros y resoluciones que se utilizan.

En el capítulo 7 se detalla el modo en que se lleva a cabo la implementación en FPGA de la arquitectura propuesta, precisando las herramientas utilizadas y analizando el desarrollo de cada módulo.

Finalmente, en los Capítulos 8 y 9 respectivamente se presentan los resultados y las conclusiones del Proyecto Integrador.

Parte I

Marco Teórico

Capítulo 2

Conceptos Preliminares

Resumen

En este capítulo se analizan los diferentes bloques que forman parte de un sistema de comunicaciones digitales, para luego introducir el problema de la Interferencia Intersímbolo y cómo solucionarla mediante diferentes técnicas de Ecualización.

Se presenta posteriormente el concepto de Ecualización Adaptiva, explicando por qué es necesaria y detallando cómo se lleva a cabo la adaptación de los coeficientes del ecualizador mediante el algoritmo LMS, precisando cómo varía su comportamiento al modificar sus diferentes parámetros.

2.1. Sistema Básico de Comunicaciones Digitales

Para poder comprender los desarrollos que se ven en los próximos capítulos es necesario analizar una serie de conceptos previos asociados a las comunicaciones digitales.

Un sistema de comunicaciones digitales es aquel que permite realizar el procesamiento y transmisión de datos de un lugar a otro en forma de bits modulados, a través de un cierto canal de transmisión, para que luego los mismos sean recibidos de forma correcta y con la menor pérdida de información posible.

En la Fig. 2.1, se puede ver un diagrama general de un sistema de comunicaciones, con los bloques básicos que se describen en esta sección.

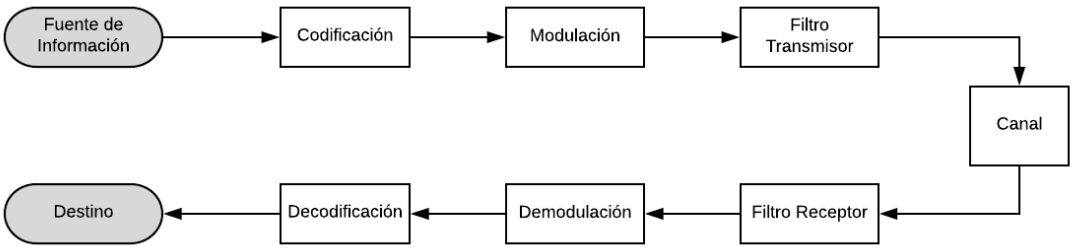


Figura 2.1: *Sistema de Comunicaciones Digitales Básico.*

2.1.1. Generación de Símbolos

En una comunicación digital, lo que se desea transmitir es una secuencia discreta de bits que representan la información a enviar. Antes de ser transmitidos, los bits pasan por lo que se conoce como *mapper*, el cual se encarga de transformarlos en los símbolos que finalmente se enviarán.

Dichos símbolos pueden estar formados por varios bits, dependiendo de la técnica de modulación utilizada, por lo que es necesario hacer una distinción entre la velocidad de transmisión en bits/seg (v), y la velocidad de transmisión en símbolos/seg, baudios o baud rate (B). Siendo la relación entre ambos la que se muestra en la Ec. (2.1), donde N es la cantidad de bits por símbolo transmitido.

$$B = \frac{v}{log_2 N} \tag{2.1}$$

Como se puede apreciar, al aumentar el valor de N se logra reducir el baud rate y por lo tanto los requerimientos de ancho de banda a la hora de transmitir, manteniendo la velocidad de transmisión de bits. Sin embargo, al aumentar dicho valor la separación entre símbolos se reduce (a igualdad de potencia transmitida) y por lo tanto surgen mayores problemas de ISI, sobre la cual se profundiza más adelante.

2.1.2. Modulación

Modular consiste en utilizar la señal que se desea transmitir para modificar alguna de las propiedades (amplitud, frecuencia, fase) de una señal distinta a la que contiene la información, llamada *portadora o carrier*, que es la que se transmite físicamente. De esta forma, luego de la modulación, en la portadora queda contenida la información a transmitir. Modular la señal contribuye a incrementar la inmunidad al ruido y mejora la eficiencia de la transmisión.

Existen diferentes técnicas de modulación digital, pudiéndose dividir en 3 grandes grupos: Modulación por Amplitud de Pulso (*Pulse Amplitude Modulation - PAM*), en el que se modifica la amplitud de la portadora, Modulación por Desplazamiento de Frecuencia (*Frequency Shift Keying - FSK*), en el que se modifica la frecuencia, y Modulación por Desplazamiento de Fase (*Phase Shift Keying - PSK*), en el que se modifica la fase. Cada una posee distintas variantes de acuerdo a la cantidad total de símbolos diferentes que se puedan transmitir, dando origen a nombres como BPSK (*Binary PSK*), 4PAM o 16PSK. Incluso se suelen combinar estas técnicas, modulando la señal tanto en amplitud como en fase, dando origen a la Modulación de Amplitud en Cuadratura (*Quadrature Amplitude Modulation - QAM*).

En la Fig. 2.2 se muestran las diferentes constelaciones que se pueden lograr con una modulación QAM, entendiendo por constelación a una representación en el plano complejo de todos los posibles símbolos de salida que puede generar un modulador.

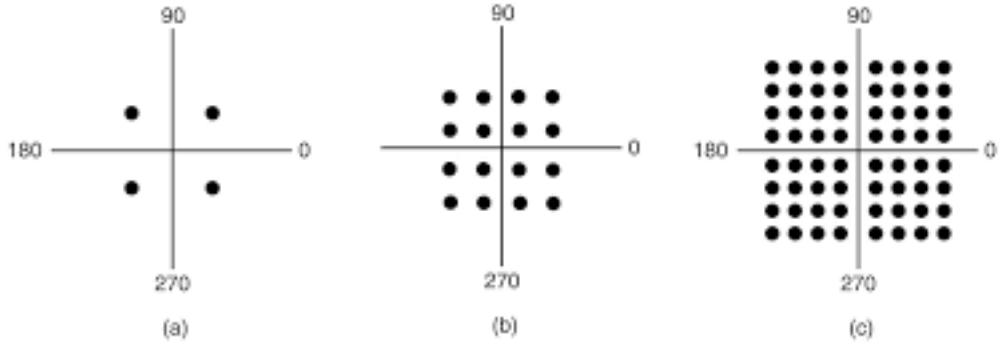


Figura 2.2: *Constelaciones resultantes para diferentes formatos de modulación QAM. (a) 4QAM o QPSK. (b) 16QAM. (c) 64QAM.*

En la práctica, a la hora de trabajar con una modulación QAM lo que se hace es separar la señal en 2 componentes independientes, una en fase (*In phase - I*) y otra en cuadratura (*Quadrature - Q*), habiendo un desfasaje de 90° entre ellas lo que las hace componentes ortogonales.

El problema de la modulación QPSK es la posible ambigüedad de fase. Si el canal por el cual pasa la señal introduce un cambio de fase arbitrario de un determinado valor, la constelación llegaría rotada y el demodulador no sería capaz de determinar cuál es la posición correcta de cada uno de los puntos de la constelación, resultando así una detección errónea. Para solventar este problema, surge la modulación diferencial, en la cual no se emplea la fase absoluta, sino que la información a transmitir se codifica como el cambio de fase del símbolo a transmitir con respecto al símbolo

transmitido anterior. El demodulador también detecta los símbolos mediante los cambios de fase, evitando así los problemas de ambigüedad de fase, debido a que en caso de que el canal introduzca una rotación, la misma afectaría a todos los símbolos por igual y por lo tanto no tiene influencia al computar la diferencia de fase entre símbolos consecutivos (ya que ambos fueron afectados por esa rotación).

Aplicar una codificación diferencial a una modulación QPSK, da como resultado una modulación que recibe el nombre de Modulación por Desplazamiento Diferencial de Fase en Cuadratura (*Differential Quadrature Phase Shift Keying - DQPSK*) y es la que se utiliza en este proyecto.

Se debe mencionar que el empleo de modulación DQPSK tiene el inconveniente de aumentar la tasa de errores, debido a que en este caso la detección de un símbolo depende del símbolo anterior, por lo que si uno es detectado erróneamente, lo mismo ocurrirá con el símbolo inmediatamente posterior. Esto se traduce en que el error es aproximadamente el doble que en el caso QPSK para un mismo nivel de Relación Señal-Ruido (*Signal to Noise Ratio - SNR*).

2.1.3. Filtro Transmisor y Receptor

Dado que lo que se desea transmitir consiste básicamente en pulsos rectangulares que representan los "1" y "0" sucesivos, con tiempos de subida y bajada sumamente rápidos y por lo tanto con componentes de muy alta frecuencia, los requerimientos de ancho de banda del canal son demasiado elevados, imposibles de alcanzar en el mundo físico.

Para solucionar este problema, lo que se hace es pasar los símbolos a transmitir por un filtro que limita el ancho de banda de la señal, posibilitando su envío por el canal deseado. Sin embargo, al limitar la señal en frecuencia, se produce una distorsión de su forma en el tiempo, volviéndose menos "rectangular", y pudiendo hacer que los símbolos transmitidos se interfieran entre sí en caso de que no se tomen los recaudos necesarios, ya que en el instante de muestreo habrá contribuciones de más de un símbolo. Esto es lo que se conoce como Interferencia Intersímbolo.

Existen diferentes filtros que pueden emplearse como filtro transmisor, como por ejemplo el Raíz Cuadrada de Coseno Realzado (*Root Raised Cosine - RRC*) o Coseno Realzado (*Raised Cosine - RC*).

El filtro receptor cumple diferentes funciones, siendo un caso típico aquel en el que se lo utiliza para conformar en conjunto con el filtro transmisor un filtro de Coseno Realzado, siendo cada uno de ellos de tipo Raíz Cuadrada

de Coseno Realzado, por lo que al realizar el producto en frecuencia de los filtros (el equivalente a la convolución en el tiempo, ya que la señal pasaría por ambos filtros) se obtiene un Coseno Realzado. Sin embargo, en este proyecto se utiliza únicamente un filtro transmisor de tipo Coseno Realzado, eliminando el filtro receptor.

2.1.4. Canal

El canal de comunicaciones utilizado introduce diferentes efectos en la señal a transmitir, debiéndoselos evitar o corregir cada uno de ellos al llegar la señal al receptor.

Por un lado, el canal tiene su propia respuesta al impulso y por lo tanto genera una distorsión en la forma de onda transmitida, pudiendo provocar ISI.

Además, el canal agrega ruido a la señal transmitida. El ruido es una señal no deseada que se acopla a la señal que se pretende transmitir, modificándola de manera tal que se puede perder la información original que se envió y pudiendo llevar a una falsa interpretación de los resultados en el receptor. Esta señal se suma en el canal de transmisión por el cual viajan los datos y es necesario disminuirla todo lo posible a fin de evitar sus efectos en la recepción.

El ruido tiene una distribución continua y aleatoria, imposible de predecir con precisión, sólo se puede conocer su distribución de probabilidad. De esta forma, es posible definir la probabilidad de error en la detección de símbolos y a partir de ello determinar un método para corregir dichos errores. A lo largo de este trabajo se considera que el ruido se comporta de forma similar a una campana de Gauss, para poder modelarlo en las simulaciones y trabajar para corregirlo.

Por último, el canal introduce en la señal transmitida una rotación de fase, la cual puede ser constante (en cuyo caso la constelación llegaría rotada pero estática) o variante en el tiempo (por lo que la constelación llegaría girando constantemente).

2.1.5. Bit Error Rate

El principal parámetro que se emplea para medir el desempeño de un sistema de comunicación digital es lo que se denomina Tasa de Error Binaria (*Bit Error Rate - BER*), definida como el cociente entre la cantidad de bits recibidos erróneamente y la cantidad total de bits que se transmitieron. Análogamente, es posible definir la Tasa de Error en Símbolos (*Symbol*

Error Rate - SER) si se trabaja con los símbolos recibidos en lugar de los bits.

Para evaluar el desempeño de una técnica de modulación (y de todo el sistema de comunicación), es común graficar el Bit Error Rate en función de la SNR presente en el sistema, obteniéndose diferentes curvas para cada modulación.

Se realizó un análisis teórico de la probabilidad de error de algunas técnicas de modulación y luego se llevaron a cabo simulaciones en Python para comprobar que los resultados teóricos efectivamente se cumplen, asumiendo siempre que el sistema está en presencia de ruido gaussiano.

Finalmente se realizó una comparación de las curvas obtenidas con cada técnica de modulación, obteniendo la gráfica de la Fig. 2.3.

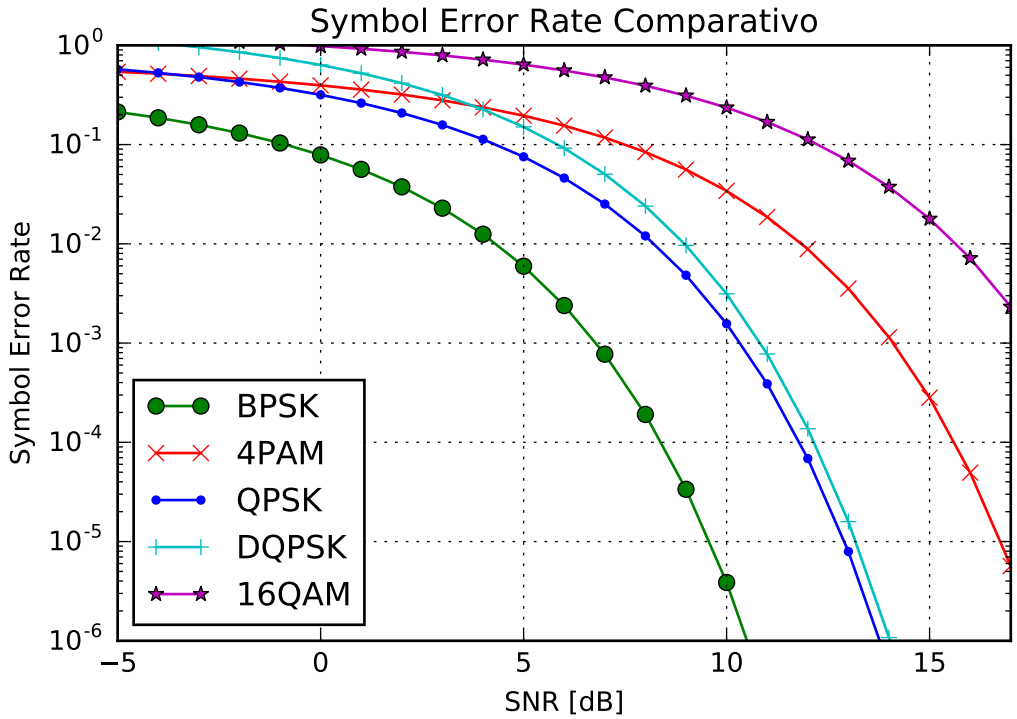


Figura 2.3: *Comparación curvas SER-SNR para diferentes modulaciones.*

Como se puede observar, a medida que disminuye la distancia entre símbolos (ya sea por haber un mayor número total de símbolos en la constelación o directamente por ser una técnica de modulación diferente), aumenta la tasa de errores a igualdad de SNR, ya que se requiere un nivel de ruido menor para que el símbolo sea confundido por alguno de sus símbolos adyacentes.

También se puede apreciar que hay una relación de compromiso entre la tasa de errores y los requerimientos de ancho de banda del canal, ya

que por ejemplo, si bien BPSK es la técnica menos propensa a errores, es también la que requiere un mayor ancho de banda para transmitir a una determinada velocidad en bits/s, ya que cada símbolo representa un sólo bit. Por otra parte, 16QAM es la que tiene un mayor error pero es también la más eficiente en cuanto ancho de banda, requiriendo 4 veces menos ancho de banda que una modulación BPSK para transmitir la misma cantidad de bits por segundo (ya que agrupa 4 bits por símbolo).

2.2. Ecualización

En esta sección se describen los principales problemas que generan los canales dispersivos en la transmisión de datos y cómo solventar los mismos utilizando métodos de ecualización.

2.2.1. Interferencia Intersímbolo y Criterio de Nyquist

La Interferencia Intersímbolo surge debido a la distorsión (tanto de amplitud como de fase) que genera el canal en los pulsos transmitidos, provocando un solapamiento entre símbolos adyacentes y haciendo que se interfieran mutuamente, pudiendo causar una detección errónea del mismo, debido a que al momento de detectar un símbolo se estará midiendo la contribución del símbolo correcto pero también la contribución de otros símbolos contiguos.

Siguiendo el análisis efectuado en el Capítulo 5 de [1] (el mismo se toma como referencia a lo largo de todo el desarrollo teórico), para determinar las condiciones que se deben cumplir para que no haya ISI, se considerará una señal $s(t)$ a la salida del filtro transmisor de la forma de la Ec. (2.2).

$$s(t) = \sum_{m=-\infty}^{\infty} a_m \, g(t - mT) \tag{2.2}$$

Siendo a_m el símbolo transmitido, $g(t)$ el filtro transmisor y T el período de símbolos ($1/T$ es la velocidad de transmisión).

Muestreando la salida del filtro $s(t)$ en múltiplos de T a los fines de recuperar los símbolos a_m, se obtiene la Ec. (2.3) para una determinada muestra k.

$$s(kT) = \sum_{m=-\infty}^{\infty} a_m \, g(kT - mT) = a_k \circledast g(kT) \tag{2.3}$$

Como se puede apreciar, la sumatoria resultante puede interpretarse como una suma de convolución entre los símbolos transmitidos y el filtro transmisor muestreado. Descomponiendo la sumatoria es posible obtener la expresión de la Ec. (2.4).

$$s(kT) = g(0)\, a_k + \sum_{m \neq k} a_m\, g(kT - mT) \tag{2.4}$$

Donde el primer término es el símbolo transmitido y el segundo término representa la interferencia de los demás símbolos adyacentes, es decir la ISI.

Para que no haya ISI, la contribución del segundo término debe ser nula, lo cual sólo ocurre si el pulso del filtro transmisor es un impulso unitario. Esto se traduce en lo que muestra la Ec. (2.5).

$$g(kT) = \delta_k \qquad \xrightarrow{\mathcal{F}} \qquad \frac{1}{T} \sum_{m=-\infty}^{\infty} G(f - \frac{m}{T}) = 1 \tag{2.5}$$

Esto se conoce como **Criterio de Nyquist** y establece que para que no haya ISI, el espectro plegado del canal debe ser una constante, entendiendo por espectro plegado a las repeticiones de la respuesta en frecuencia cada $1/T$.

Así, el criterio de Nyquist establece un ancho de banda mínimo para que sea posible transmitir por un canal a una determinada velocidad sin que haya ISI, concretamente el ancho de banda del canal debe ser al menos la mitad de la velocidad de transmisión (en baudios).

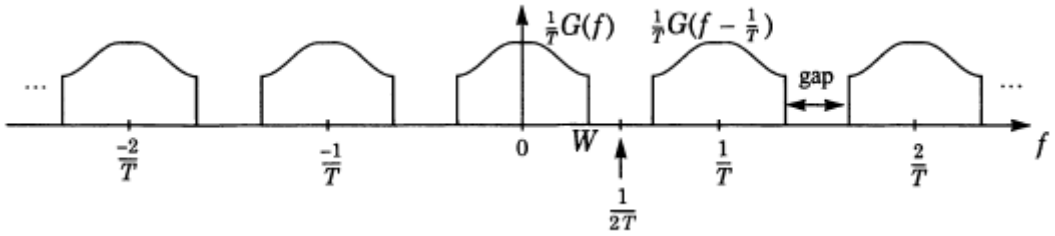


Figura 2.4: *Espectro plegado de un pulso g(t) que no cumple el Criterio de Nyquist.*

Esto se puede visualizar en la Fig. 2.4, en la que se puede apreciar cómo independientemente de la forma de la respuesta en frecuencia $G(f)$, debido a que su ancho de banda W es menor a $1/2T$, siempre habrá una separación entre cada repetición de la respuesta y por lo tanto la suma nunca podrá ser una constante, siendo imposible que se cumpla el Criterio de Nyquist.

También es posible notar, que aunque el ancho de banda del canal cumpla el criterio de Nyquist, no cualquier pulso generará una respuesta cons-

tante. Un ejemplo de señal que cumple con este criterio es aquella cuya respuesta en frecuencia sea un pulso rectangular, y es el que utiliza de manera más eficiente el ancho de banda disponible.

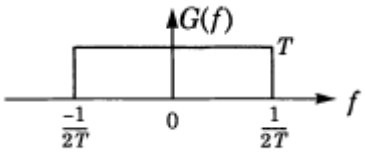


Figura 2.5: *Rectángulo en frecuencia que ocupa el mínimo ancho de banda cumpliendo el Criterio de Nyquist.*

Tomando la Transformada de Fourier Inversa del pulso de la Fig. 2.5, se obtiene en el tiempo una función seno cociente que responde a la Ec. (2.6).

$$g(t) = \frac{sin(\pi t/T)}{\pi t/T} \tag{2.6}$$

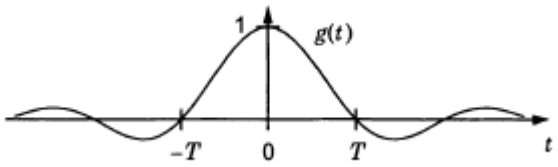


Figura 2.6: *Forma de onda de una función Seno Cociente.*

Se debe observar que la forma de onda de la Fig. 2.6 toma valores nulos en todos los múltiplos de T excepto en $t = 0$, que es lo que se deseaba obtener.

Lamentablemente, este pulso ideal resulta poco práctico a la hora de implementarlo ya que requiere un número de coeficientes demasiado elevado, por lo que se debe recurrir a un pulso diferente que se asemeje todo lo posible al ideal. Esta nueva señal tendrá un ancho de banda $1 + \alpha$ veces mayor al del pulso rectangular, como se muestra en la Ec. (2.7).

$$W = \frac{1 + \alpha}{2T} \tag{2.7}$$

Donde α es el factor de *roll-off* definido como un parámetro de exceso de ancho de banda que toma valores entre 0 y 1. A medida que este valor disminuye, el pulso tiende al ideal. Se debe notar que para $\alpha = 0$, se está en presencia del caso ideal, ya que no hay exceso de ancho de banda. En

la Fig. 2.7 se muestra como varía la forma de onda al variar el factor de roll-off, para el caso de un pulso de tipo Coseno Realzado.

El análisis previo se hizo considerando sólo el filtro transmisor, pero en la práctica el pulso que debe cumplir el criterio de Nyquist es la respuesta en cascada del filtro transmisor, del canal y del filtro receptor.

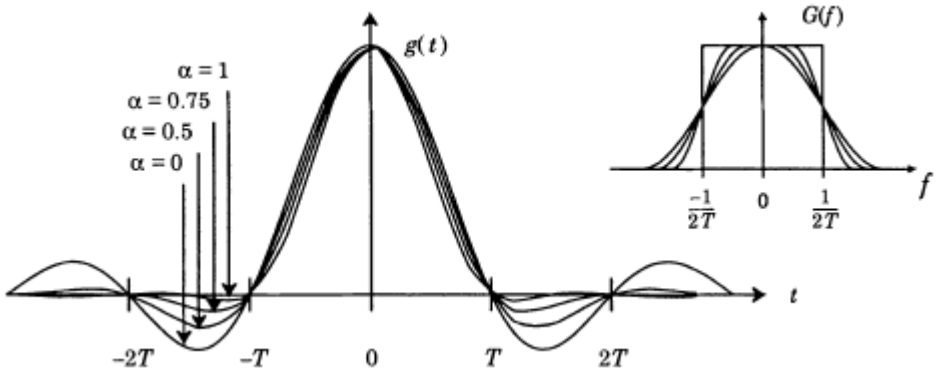


Figura 2.7: *Formas de onda en tiempo y frecuencia para Cosenos Realzados con diferentes valores de roll-off.*

Las condiciones del Criterio de Nyquist no se cumplen habitualmente al incorporar un canal de transmisión, y de allí surge la necesidad de realizar un proceso de ecualización.

2.2.2. Ecualizador

Dado que el criterio de Nyquist no suele cumplirse, es necesario recurrir a un módulo denominado ecualizador, el cual busca lograr que la respuesta total del sistema sí lo cumpla.

Existen diferentes criterios a la hora de determinar la forma de ecualizar. El más simple de ellos y el que describe mejor el objetivo de un ecualizador, se conoce como Ecualizador Zero Forcing (Forzado a Cero) y busca que la cancelación de la Interferencia Intersímbolo sea total. Para lograr esto, la conexión en cascada del canal, con su respuesta $S(Z)$, y el ecualizador, con su respuesta $C(Z)$, debe ser una constante, lo cual implica que la relación entre ellas debe ser la que se muestra en la Ec. (2.8).

$$C(z) = \frac{1}{S(z)} \tag{2.8}$$

El resultado de esta relación puede verse en la Fig. 2.8, donde se aprecia que la inversa de la función intenta compensar los cambios de $C(z)$, a fin de convertirla en una constante.

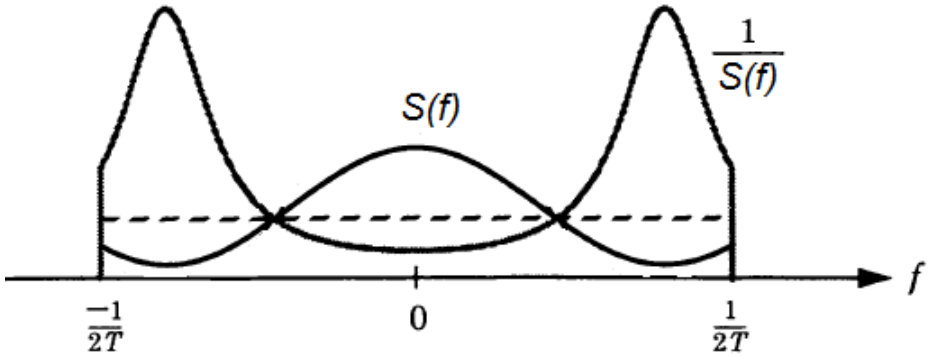


Figura 2.8: *Respuesta del canal (S), respuesta del ecualizador Zero Forcing($\frac{1}{S}$), y respuesta combinada de ambos (línea de puntos).*

Este tipo de ecualizador es ideal en ausencia de ruido, pero este no suele ser el caso en la realidad y por lo tanto no resulta del todo práctico. El problema radica en el hecho de que si bien este criterio corrige la ISI, también es susceptible de amplificar notablemente el ruido que se agrega después del canal, especialmente cuando su respuesta $S(z)$ toma valores bajos (es decir, en aquellas frecuencias donde el canal produce una gran atenuación), provocando que su inversa $C(z)$ tome valores elevados y por lo tanto amplifique significativamente lo que no haya sido afectado por la respuesta del canal. Este problema es aún peor cuando la respuesta del canal se hace nula en algún punto.

Ante esto, surge el **Ecualizador de Mínimo Error Cuadrático Medio**, en el que se busca minimizar la probabilidad de error en la detección del símbolo, debido tanto al ruido como a la ISI. Este ecualizador tiene una forma similar al del Zero Forcing pero con un término adicional en el denominador, tal como puede verse en la Ec. (2.9), donde k es un valor que depende del ruido y de la varianza de los símbolos transmitidos.

$$C(z) = \frac{1}{S(z) + k} \tag{2.9}$$

El error entre el símbolo detectado y el símbolo deseado tiene una componente debida al ruido y otra debido a la ISI, existiendo una relación de compromiso entre ambas en la que minimizar la ISI empeora el ruido, por lo que si no se elimina completamente, es posible llegar a un punto en el que se minimiza el error total del sistema. Es decir, se permite que quede una cierta ISI residual pero se obtiene una probabilidad de error menor que con el criterio de Zero Forcing, que es lo que en última instancia se desea lograr en un sistema de comunicaciones.

2.2.3. Ecualizador Adaptivo

En el análisis que se ha hecho hasta el momento, se asumió que el pulso que llega al ecualizador tiene una forma conocida, lo cual implica que se conoce la respuesta del canal, lo cual no suele ser el caso. Aún cuando se conozca el canal, este puede tener ciertas variaciones a lo largo del tiempo, por lo que tampoco se sabría con certeza la forma exacta del pulso que llega al ecualizador.

Ante este problema surge el concepto de **Ecualización Adaptiva**, en la cual se busca que, sin conocer la respuesta del canal, los coeficientes del ecualizador se adapten para minimizar tanto el ruido como la Interferencia Intersímbolo a la salida del mismo. Para determinar cómo se realiza la adaptación, se utiliza una señal de error que resulta de la diferencia entre la entrada y la salida del *slicer* (encargado de la detección del símbolo a partir de la salida del ecualizador) y es la que indica en qué dirección deben moverse los coeficientes para lograr una ecualización más precisa. Esto puede apreciarse en el esquema de la Fig. 2.9.

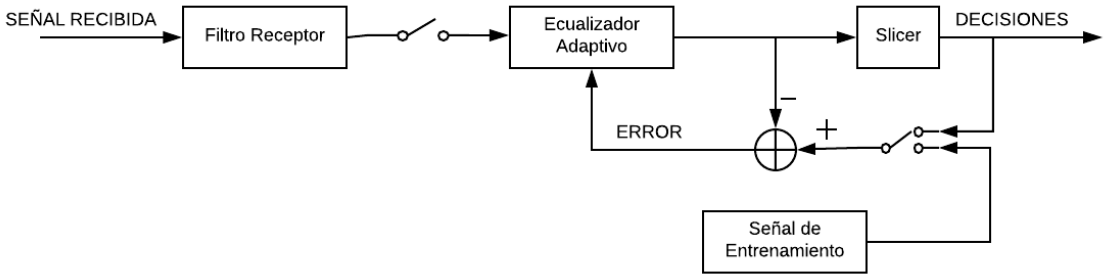


Figura 2.9: *Diagrama de bloques de un Ecualizador Adaptivo.*

Una vez que el ecualizador ya se encuentra en estado estable, las decisiones tomadas por el *slicer* son las que se utilizan para computar la señal de error, siendo posible utilizarlas debido a que constituyen una representación de los símbolos transmitidos libre de ISI y de ruido. Si bien es cierto que se cometen algunos errores en la detección de los símbolos, estos no tienen efectos significativos (mientras no sean excesivos) ya que no afectan en el largo plazo a la adaptación de los coeficientes.

Debido a que inicialmente, antes de que los coeficientes hayan podido adaptarse, la tasa de error puede llegar a ser considerablemente alta, provocando que muchas de las decisiones tomadas por el detector sean erróneas y por lo tanto el filtro no se adapte correctamente, suele utilizarse una señal de entrenamiento. El transmisor genera una secuencia conocida y en el ecualizador se utiliza dicha secuencia para calcular el error, evitando así

los efectos de las detecciones erróneas. Pasada una determinada cantidad de tiempo, cuando los coeficientes han sido aceptablemente adaptados, se comienzan a utilizar las decisiones del slicer, dado que ya se tiene una tasa de error muchísimo menor.

Cabe mencionar también que si el error es nulo, y por lo tanto no hay ruido ni ISI, la señal de error e_k tendrá un valor nulo, indicándole al ecualizador que no es necesario realizar un ajuste en los coeficientes, por lo que permanecerán inalterados. Por otra parte, si hay ruido pero no hay ISI, el error e_k tendrá un valor distinto de 0, pero su valor promedio sí será 0 y nuevamente los coeficientes permanecerán constantes. Así, puede verse que el ecualizador sólo se adaptará ante la presencia de ISI.

2.2.4. Estructura del Ecualizador

El ecualizador adaptivo está constituido por un filtro que tiene la estructura básica que se puede apreciar en el diagrama de bloques de la Fig. 2.10, que es la misma que la de un filtro FIR (*Finite Impulse Response*) con M coeficientes.

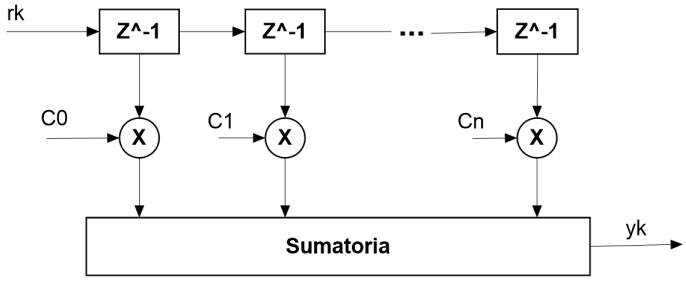


Figura 2.10: *Diagrama de bloques de filtro FIR del Ecualizador Adaptivo.*

Es decir que el ecualizador será un filtro de la forma que se muestra en la Ec. (2.10), siendo M el número total de coeficientes (considerado siempre impar) y $L = (M-1)/2$.

$$C(z) = \sum_{m=-L}^{L} w_m \, z^{-m} \tag{2.10}$$

Los coeficientes se adaptan a medida que se comienzan a recibir datos, utilizando para ello un algoritmo de adaptación que depende del error e_k entre la salida del ecualizador y_k y el símbolo detectado $\hat{a}_k$. Dicho error se calcula como se muestra en la Fig. 2.11.

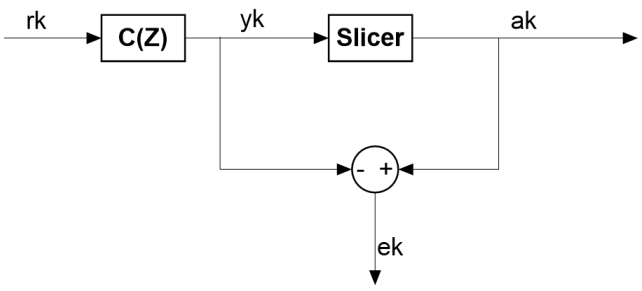


Figura 2.11: *Diagrama de bloques para cálculo de error e_k en Ecualizador Adaptivo.*

2.2.5. Adaptación de Coeficientes

Para determinar el algoritmo de adaptación de coeficientes, se trabaja con el criterio de minimización del error cuadrático medio, ya mencionado en la Sección 2.2.2, para lo cual se emplea el **Algoritmo LMS**.

A partir del diagrama de la Fig. 2.10, y considerando que el ecualizador $C(z)$ tiene la forma de la Ec. (2.10), es posible escribir la salida y_k como se muestra en la Ec. (2.11), en la que se efectúa un cambio de notación que se mantendrá a lo largo de todo el desarrollo del marco teórico, haciendo $R_k = x(n)$ y $y_k = y(n)$, simbolizando así las muestras de entrada y de salida del ecualizador respectivamente.

$$y(n) = w_{-L}\, x(n-L) + \cdots + w_0\, x(n) + \cdots + w_L\, x(n+L) = \sum_{j=-L}^{L} w_j\, x(n+j) \tag{2.11}$$

Se debe notar que es posible expresar la Ec. (2.11) en forma matricial, tal como se muestra en la Ec. (2.12).

$$y(n) = \overline{w}^T\, \overline{x}(n) \tag{2.12}$$

Siendo:

$$\overline{w}^T = \begin{bmatrix} w_{-L} & \cdots & w_0 & \cdots & w_L \end{bmatrix} \tag{2.13}$$

$$\overline{x}(n) = \begin{bmatrix} x(n-L) \\ \vdots \\ x(n) \\ \vdots \\ x(n+L) \end{bmatrix} \tag{2.14}$$

Teniendo en cuenta que $M = 2L + 1$, se puede ver que $\overline{w}^T$ es un vector fila con dimensión $1 \times M$, mientras que $\overline{x}(n)$ es un vector columna con dimensión $M \times 1$, por lo que el producto matricial tendrá dimensión 1×1, es decir que será un escalar igual a la sumatoria de la Ec. (2.11), por lo que ambas expresiones son equivalentes.

A su vez, se define el error $e(n)$ en un instante dado de la forma en que se muestra en la Ec. (2.15), en la que nuevamente se introduce el cambio de notación $a_k = a(n)$.

$$e(n) = a(n) - y(n) = a(n) - \overline{w}^T \, \overline{x}(n) \tag{2.15}$$

Siguiendo una vez más los desarrollos que se efectúan en [1], el objetivo del algoritmo LMS es encontrar el conjunto de coeficientes w_i que minimice un funcional de costo J, definido como se muestra en la Ec. (2.16), siendo $E\{\}$ la esperanza matemática.

$$J(n) = E\{ \, |e(n)|^2 \, \} = E\{|a(n) - y(n)|^2\} \tag{2.16}$$

Es decir que se intenta minimizar el Error Cuadrático Medio (*Mean Squared Error - MSE*), de allí el nombre del algoritmo. El MSE depende de los coeficientes del ecualizador adaptivo, y para el caso tridimensional en el que se tienen sólo 2 coeficientes w_1 y w_2, el error tendrá la forma de la Fig. 2.12, en la que los ejes x e y se corresponden con los coeficientes y el eje z al error cuadrático medio.

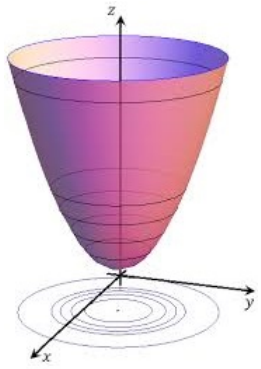


Figura 2.12: *Gráfica del MSE con ecualizador de 2 coeficientes*

Un método para determinar el módulo al cuadrado de un número complejo consiste en multiplicarlo por su conjugado.

$$|e(n)|^2 = [a(n) - \overline{w}^T \, \overline{x}(n)] \times [a(n) - \overline{w}^T \, \overline{x}(n)]^*$$

$$|e(n)|^2 = [a(n) - \overline{w}^T \, \overline{x}(n)] \times [a^*(n) - \overline{w}^H \, \overline{x}^*(n)] \tag{2.17}$$

Siendo $\overline{w}^H$ la conjugada transpuesta de la matriz $\overline{w}$, y teniendo en cuenta que el conjugado de la resta es igual a la resta de los conjugados.

Operando en la expresión de la Ec. (2.17) se llega a la Ec. (2.18).

$$|e(n)|^2 = |a(n)|^2 - \left[a(n)\overline{w}^H\overline{x}^*(n) + a^*(n)\overline{w}^T\overline{x}(n)\right] + \overline{w}^H\overline{x}^*(n)\overline{x}^T(n)\overline{w}$$

$$|e(n)|^2 = |a(n)|^2 - 2\Re\{a(n)\overline{w}^H\overline{x}^*(n)\} + \overline{w}^H\overline{x}^*(n)\overline{x}^T(n)\overline{w} \tag{2.18}$$

Una vez obtenida la expresión para el error cuadrático, se procede a calcular su esperanza, teniendo en cuenta que los coeficientes permanecen constantes en un instante dado, por lo que su esperanza es igual a su propio valor.

$$E\{\,|e(n)|^2\,\} = E\{\,|x(n)|^2\,\} - 2\Re\{\overline{w}^H\alpha\} + \overline{w}^H\phi\,\overline{w} \tag{2.19}$$

Siendo:

$$\alpha = E\{a_k R_x^*\} \tag{2.20}$$

Y ϕ la *matriz de autocorrelación* de los datos que ingresan al ecualizador, definida como se muestra en la Ec. (2.21).

$$\phi = E\{\overline{x}^*(n)\overline{x}^T(n)\} \tag{2.21}$$

En la mayoría de las aplicaciones, la matriz ϕ será definida positiva y por lo tanto no singular, es decir que tendrá una matriz inversa.

La expresión de la Ec. (2.19) es la que se busca minimizar en función de los coeficientes $\overline{w}$, para lo cual se recurre al cálculo del gradiente de dicha ecuación con respecto al vector de coeficientes (∇_w), y luego se lo iguala a 0, encontrando así el vector $\overline{w}_{opt}$ que minimiza el MSE.

$$\nabla_w E\{\,|e(n)|^2\,\} = 2\phi\overline{w} - 2\alpha = 0 \quad \rightarrow \quad \overline{w}_{opt} = \phi^{-1}\alpha \tag{2.22}$$

2.2.6. Algoritmo del Gradiente

Para determinar el valor de los coeficientes óptimos, se recurre al algoritmo del gradiente para el MSE, el cual mediante un proceso iterativo define una secuencia de vectores de coeficientes que convergerá hacia el valor de $\overline{w}_{opt}$.

El MSE tiene una naturaleza cuadrática, pudiéndoselo interpretar como una superficie en un espacio de $M + 1$ dimensiones, siendo M el número de coeficientes del ecualizador. El caso tridimensional, con $M = 2$, ya se ejemplificó gráficamente en la Fig. 2.12, pudiéndose apreciar que tiene un mínimo global único, hecho que se cumple independientemente de la dimensión de la superficie. Debido justamente a esta naturaleza cuadrática que posee el Error Cuadrático Medio, el ajuste de los coeficientes puede realizarse simplemente modificando sus valores iterativamente descendiendo a lo largo de la superficie del MSE.

Para modificar los coeficientes, al valor actual de los mismos se le resta un término proporcional al gradiente. Dado que el gradiente representa la dirección en la que la velocidad de crecimiento es máxima, moverse en la dirección contraria debería entonces reducir el error. Teniendo esto en cuenta, es posible escribir el algoritmo del gradiente explícitamente como se muestra en la Ec. (2.23).

$$\overline{w}(n+1) = \overline{w}(n) - \frac{\mu}{2}\nabla_w E\{\, |e(n)|^2 \,\} \tag{2.23}$$

Siendo $\overline{w}(n)$ el valor actual del vector de coeficientes, $\overline{w}(n+1)$ el valor en la próxima iteración y μ el paso de adaptación. Reemplazando la expresión del gradiente de la Ec. (2.22) en la Ec. (2.23), es posible escribir la Ec. (2.24), donde I es la matriz identidad.

$$\overline{w}(n+1) = \overline{w}(n) + \mu[\alpha - \phi\,\overline{w}(n)] = (I - \mu\phi)\overline{w}(n) + \mu\alpha \tag{2.24}$$

El valor de μ de las ecuaciones anteriores es sumamente crítico, siendo el principal parámetro que determina si el algoritmo converge o no, y qué tan rápido lo hace (velocidad de convergencia).

Un valor elevado de μ hace que el algoritmo converja más rápidamente, pero deja un error residual mayor y se corre el riesgo de que el algoritmo se vuelva inestable si el paso es demasiado grande. Por otra parte, un paso de adaptación pequeño favorece la convergencia del algoritmo y permite alcanzar un error menor, pero si el valor es muy pequeño, el algoritmo se vuelve demasiado lento, siendo necesario procesar demasiadas muestras para llegar al valor de $\overline{w}_{opt}$. Este comportamiento se ilustra en la Fig. 2.13, en la que se puede apreciar cómo partiendo de una posición inicial, se avanza un determinado paso μ en la dirección opuesta a la del gradiente.

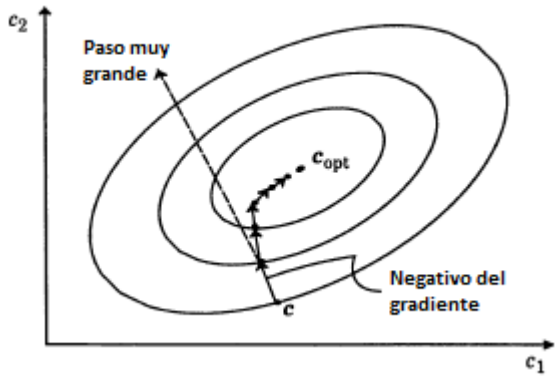


Figura 2.13: *Método del Gradiente para diferentes pasos.*

2.2.7. Algoritmo del Gradiente Estocástico

El método iterativo al que finalmente se llega con la Ec. (2.24) requiere poder determinar la matriz de autocorrelación ϕ, lo cual implica ser capaz de calcular la esperanza matemática involucrada en su expresión, y este no siempre es el caso, ya que para ello debe conocerse el canal. Ante este problema surge una modificación del algoritmo anterior, dando lugar al algoritmo del gradiente estocástico, en el cual se trabaja directamente con el gradiente del error definido por la Ec. (2.18).

$$\nabla_w |e_(n)|^2 = -2\,\overline{x}^*(n)[a(n) - \overline{x}^T(n)\overline{w}] = -2\,e(n)\overline{x}^*(n) \tag{2.25}$$

La expresión de la Ec. (2.25) se utiliza como un *estimador del gradiente* de la Ec. (2.23), por lo que realizando los reemplazos correspondientes se obtiene la Ec. (2.26), que es la que finalmente se utiliza para actualizar el valor de los coeficientes.

$$\boxed{\overline{w}(n+1) = \overline{w}(n) + \mu e(n)\overline{x}^*(n)} \tag{2.26}$$

Se debe notar que el nuevo valor de los coeficientes en el instante $n+1$, depende del valor anterior de los mismos, del paso de adaptación μ, del error $e(n)$ en el instante n y de las muestras recibidas que se procesaron en el instante n.

Al ingresar una nueva muestra, se modifica el vector $\overline{x}(n)$, lo que provocará una modificación de la salida del ecualizador $y(n)$ y del error $e(n)$, y por lo tanto también habrá cambios en los coeficientes, por lo que se puede apreciar cómo los coeficientes del filtro se actualizarán cada vez que ingrese una nueva muestra al módulo.

Debe notarse también que en la Ec. (2.26), tanto $e(n)$ como μ son escalares, por lo que se está llevando a cabo una suma elemento a elemento, en la que cada coeficiente será afectado por su correspondiente muestra retenida en el vector $\overline{x}(n)$.

Es posible realizar un análisis para determinar el paso de adaptación óptimo y las condiciones para su convergencia, llegándose a que la máxima velocidad de convergencia se obtiene con el valor de la Ec. (2.27), tal como se muestra en el Capítulo 9 de [1].

$$\mu_{opt} = \frac{1}{M\phi_0} \tag{2.27}$$

Siendo M la cantidad de coeficientes del filtro y ϕ_0 la energía de las muestras de entrada, la cual puede ser interpretada como la esperanza de las muestras recibidas al cuadrado, como se ve en la Ec. (2.28).

$$\phi_0 = E\{\ |\overline{x}(n)|^2\ \} \tag{2.28}$$

Si se continúa aumentando el paso más allá de ese valor, la velocidad de convergencia comienza a disminuir hasta que eventualmente el sistema se vuelve inestable. Específicamente, la condición de convergencia es la que se muestra en la Ec. (2.29), en la que puede apreciarse cómo una vez que el valor de μ es el doble del óptimo, el sistema ya se vuelve inestable y el algoritmo no converge.

$$0 < \mu < \frac{2}{M\phi_0} \tag{2.29}$$

Otro hecho destacable es que el paso óptimo depende fuertemente de la energía de la señal de entrada (ϕ_0), lo cual puede provocar que ante un incremento demasiado grande de la misma, el sistema se vuelva inestable. Para evitar este inconveniente, es usual normalizar el paso μ utilizando una estimación de la energía de la señal en un instante dado y algunas constantes para evitar que el paso se vuelva demasiado grande cuando la energía de entrada disminuye considerablemente.

2.2.8. Ecualizador Fraccionalmente Espaciado

En el análisis realizado hasta el momento, se ha considerado que la velocidad de las muestras de entrada del ecualizador es la misma que la velocidad de salida, concretamente tanto la entrada como la salida tienen

una frecuencia igual a la tasa de símbolos (*symbol rate*). Este tipo de ecualizadores se denominan *Symbol-Spaced Equalizer*, ya que tanto la entrada como la salida trabajan a la tasa de símbolos.

Pero esto no necesariamente tiene que ser así. En ocasiones, el ecualizador recibe K muestras de entrada (siendo K un número entero) antes de generar una nueva salida y actualizar los coeficientes. Es decir que la frecuencia de entrada es K/T mientras que la de salida es $1/T$, que es la tasa de símbolos. Este tipo de ecualizadores recibe el nombre de Ecualizador Fraccionalmente Espaciado (*Fractionally-Spaced Equalizer - FSE*), ya que la frecuencia de salida es una fracción de la de entrada. Para que la entrada trabaje a una velocidad mayor que la tasa de símbolos, es necesario haber realizado previamente un proceso de sobremuestreo (*oversampling*), transmitiendo OS cantidad de muestras por cada símbolo transmitido, siendo OS el Factor de Sobremuestreo.

Utilizar un FSE tiene considerables ventajas frente a la alternativa de utilizar un ecualizador convencional. Por un lado, es capaz de detectar correctamente la fase de muestreo adecuada (la cantidad de fases posibles es igual al factor de sobremuestreo utilizado), por lo que este tipo de ecualizador es considerablemente menos sensible a los problemas asociados al *offset* de fase en el receptor. Por otra parte, también exhibe un comportamiento similar al de un filtro receptor óptimo en presencia de ruido gaussiano, tendiendo a optimizar la SNR del sistema.

2.2.9. Influencia de los Parámetros

Resulta interesante considerar los diferentes efectos que tienen en el comportamiento del ecualizador sus dos principales parámetros: el paso de adaptación μ y el número de coeficientes M (también llamado número de *taps*).

Como ya se ha mencionado anteriormente, el paso de adaptación μ tiene incidencia directa en la velocidad de convergencia del algoritmo y en el MSE resultante una vez que el sistema se encuentra en estado estable. Un valor elevado de μ permite alcanzar una elevada velocidad de convergencia, como así también brinda la posibilidad de seguir cambios rápidos en la respuesta del canal. En contrapartida, se tiene como resultado un mayor MSE en estado estacionario, teniendo el valor de los coeficientes fluctuaciones mayores. En el peor de los casos, un valor demasiado elevado del paso provoca que el algoritmo se vuelva inestable.

Por otra parte, trabajar con un paso de adaptación que tenga un valor

pequeño favorece la convergencia del sistema y disminuye el MSE remanente a la salida. Sin embargo, también provoca una disminución en la velocidad de convergencia del algoritmo y reduce la capacidad de seguimiento a los cambios del canal.

Como se puede apreciar, al variar el paso hay una clara relación de compromiso entre velocidad de convergencia (directamente asociada con la capacidad de seguimiento) y el MSE en estado estable. Por ejemplo, si se tiene un canal que no posee variaciones rápidas en el tiempo, se puede dejar de lado la velocidad de convergencia para priorizar un menor error en estado estable.

En muchas aplicaciones, resulta sumamente útil tener una rápida velocidad de convergencia al inicializar el sistema, para que el mismo se adapte rápidamente, pero una vez que los coeficientes del filtro ya están aceptablemente adaptados, se desea que el sistema tenga un MSE pequeño en estado estable. Este comportamiento se puede lograr utilizando lo que se denomina Cambio de Marcha (*Gear-Shift*), comenzando con un paso de adaptación alto y luego de que haya transcurrido una cierta cantidad de tiempo (suficiente para que el sistema ya haya convergido), se comienza a trabajar con un paso más pequeño.

Con respecto al número de coeficientes del filtro, es evidente que mientras mayor sea este número, mejor será el comportamiento del ecualizador, pudiendo compensar mejor los efectos del canal. Sin embargo, aumentar el número de taps trae aparejado un aumento directo en la complejidad del ecualizador a la hora de implementarlo, siendo necesarias una mayor cantidad de operaciones por cada muestra procesada. No sólo se deben retener una mayor cantidad de muestras de entrada (tantas como cantidad de coeficientes se tengan), sino que también se deben actualizar un mayor número de coeficientes. Pero el principal incremento en la complejidad se debe a la convolución que se debe realizar entre las muestras de entrada y los coeficientes del ecualizador, aumentando notablemente el número de sumas y de multiplicaciones que se deben llevar a cabo, siendo las multiplicaciones las operaciones que tienen un mayor costo de hardware.

Capítulo 3

Algoritmo de Adaptación BLMS

Resumen

En este capítulo se presenta una alternativa al algoritmo LMS, llamada Block LMS, la cual favorece el procesamiento en paralelo de las muestras de entrada y permite una eficiente implementación. Además se enfatizan las equivalencias y diferencias con respecto al algoritmo LMS.

3.1. Algoritmo LMS

Utilizando un algoritmo LMS tradicional en un ecualizador adaptivo, la adaptación de los coeficientes se realiza una vez por cada muestra que ingresa al ecualizador, es decir que se tiene la ecuación de adaptación que se muestra en la Ec. (3.1).

$$\overline{w}(n+1) = \overline{w}(n) + \mu\overline{x}(n)e(n) \tag{3.1}$$

Siendo $\overline{w}(n)$ el vector de coeficientes de largo M en el instante n, $\overline{x}(n)$ el vector de entrada, $e(n)$ un escalar que representa el error entre la salida deseada $d(n)$ o la muestra detectada $a(n)$, y la salida obtenida $y(n)$, y μ el paso de adaptación. Estas definiciones pueden verse en las Ec. (3.2), (3.3) y (3.4).

$$\overline{w}(n) = [w_0(n)\ w_1(n)\ \cdots\ w_{M-1}(n)]^T \tag{3.2}$$

$$\overline{x}(n) = [x(n)\ x(n-1)\ \cdots\ x(n-M+1)]^T \tag{3.3}$$

$$e(n) = d(n) - y(n) \tag{3.4}$$

Por otra parte, la salida del ecualizador se obtiene a partir de las muestras del vector de entrada y de los coeficientes del ecualizador, lo cual puede expresarse matricialmente como se muestra en la Ec. (3.5).

$$y(n) = \overline{x}^T(n)\overline{w}(n) \tag{3.5}$$

3.2. Algoritmo Block LMS

Una alternativa a la adaptación de la Ec. (3.1) consiste en mantener fijos los coeficientes $\overline{w}(n)$ hasta que se hayan recibido una cantidad L de muestras, y luego se actualizan los coeficientes una única vez teniendo en cuenta la información de todas las muestras procesadas durante ese tiempo. Es decir que el concepto básico, el cual es analizado exhaustivamente en [4] y [16], consiste en procesar un bloque de muestras, en el que cada elemento del bloque consiste en un vector de muestras de tamaño M, obteniendo un bloque de salidas, cada una con su respectivo error, y a partir de ello realizar la adaptación de coeficientes.

Teniendo esto en cuenta, la ecuación para la adaptación de coeficientes puede escribirse como se ve en la Ec. (3.6), siendo L el tamaño del bloque.

$$\overline{w}(n+L) = \overline{w}(n) + \mu\frac{\sum_{i=0}^{L-1}\overline{x}(n+i)e(n+i)}{L} \tag{3.6}$$

Esta forma de adaptación es lo que se conoce como **Block LMS**. Se puede ver que todas las adaptaciones dentro de un mismo bloque se llevan a cabo a partir del valor de los coeficientes del bloque anterior, es decir que los mismos se actualizan cada L muestras procesadas. Con respecto a las salidas, las mismas están determinadas por la Ec. (3.7), pudiéndose ver que dentro de un mismo bloque, todas se calculan a partir de los mismos coeficientes.

$$y(n+m) = \overline{x}^T(n+m)\overline{w}(n) \tag{3.7}$$

Como la adaptación se lleva a cabo a una tasa menor que la tasa de las muestras de entrada, es posible definir un nuevo índice de tiempos k para los bloques, de manera tal que un incremento en k represente L incrementos en el índice de tiempo original n, es decir que $n = kL$. De esta forma, es posible reescribir la Ec. (3.6) utilizando el nuevo índice k, como se muestra en la Ec. (3.8), en la que $e(kL+i) = d(kL+i) - y(kL+i)$, es decir que se computa el error de cada una de las salidas del bloque y se lo utiliza para estimar el gradiente.

$$\boxed{\overline{w}(k+1) = \overline{w}(k) + \mu \frac{1}{L} \sum_{i=0}^{L-1} \overline{x}(kL+i)e(kL+i)} \tag{3.8}$$

Con respecto a los elementos del bloque, cada uno de ellos está formado por un vector con muestras de entrada desplazado una posición con respecto al elemento anterior. A continuación se ejemplifica el contenido de un bloque en 3 instantes de tiempo k consecutivos, considerando $M = L = 3$.

$$k-1 \left\{ \begin{bmatrix} x(3k-3) & x(3k-4) & x(3k-5) \\ x(3k-2) & x(3k-3) & x(3k-4) \\ x(3k-1) & x(3k-2) & x(3k-3) \end{bmatrix} \begin{bmatrix} w_0(k-1) \\ w_1(k-1) \\ w_2(k-1) \end{bmatrix} = \begin{bmatrix} y(3k-3) \\ y(3k-2) \\ y(3k-1) \end{bmatrix} \right. \tag{3.9}$$

$$k \left\{ \begin{bmatrix} x(3k) & x(3k-1) & x(3k-2) \\ x(3k+1) & x(3k) & x(3k-1) \\ x(3k+2) & x(3k+1) & x(3k) \end{bmatrix} \begin{bmatrix} w_0(k) \\ w_1(k) \\ w_2(k) \end{bmatrix} = \begin{bmatrix} y(3k) \\ y(3k+1) \\ y(3k+2) \end{bmatrix} \right. \tag{3.10}$$

$$k+1 \left\{ \begin{bmatrix} x(3k+3) & x(3k+2) & x(3k+1) \\ x(3k+4) & x(3k+3) & x(3k+2) \\ x(3k+5) & x(3k+4) & x(3k+3) \end{bmatrix} \begin{bmatrix} w_0(k+1) \\ w_1(k+1) \\ w_2(k+1) \end{bmatrix} = \begin{bmatrix} y(3k+3) \\ y(3k+4) \\ y(3k+5) \end{bmatrix} \right. \tag{3.11}$$

En este punto cabe mencionar que si bien cada vector del bloque es diferente al resto de los vectores del bloque, varias muestras entre ellos coinciden y resultaría redundante almacenarlas en cada uno de los vectores. Por esto es que en una eventual implementación se trabajaría con un único vector de entrada, de longitud $M + L - 1$, en el que se encuentran todas las muestras que se necesitarán para procesar el bloque, y luego se toman diferentes secciones de ese vector de entrada de mayor tamaño para conformar cada uno de los L vectores del bloque (con M muestras cada uno). De esta forma, cada muestra necesaria se almacena una única vez. Cuando se termina de procesar el bloque, se incorporan L nuevas muestras de entrada, haciendo un desplazamiento en el vector de entrada, para así contar con las muestras necesarias para procesar el siguiente bloque.

Las ecuaciones vistas anteriormente pueden expresarse matricialmente, pero para ello es necesario hacer algunas definiciones previas. La matriz $X(k)$ que representa cada bloque está dada por los diferentes vectores filas con las muestras de entrada desplazadas una unidad de tiempo con respecto a las del elemento anterior.

$$X(k) = \begin{bmatrix} x(kL) & x(kL-1) & \cdots & x(kL-M-1) \\ x(kL+1) & x(kL) & \cdots & x(kL-M) \\ \vdots & \vdots & \vdots & \vdots \\ x(kL+L-1) & x(kL+L-2) & \cdots & x(kL-M+L) \end{bmatrix} \tag{3.12}$$

Representando cada vector de entrada, es decir cada fila de la matriz, como $\overline{x}(kL)$, es posible reescribir la Ec. (3.12) de una forma más compacta.

$$X(k) = [\ \overline{x}(kL)\ \overline{x}(kL+1)\ \cdots\ \overline{x}(kL+L-1)\ ]^T \tag{3.13}$$

Se define también un vector de errores $\overline{e}(k)$, que representa los errores de cada una de los vectores del bloque, y un vector de salida $\overline{y}(k)$, que representa las salidas correspondientes a cada uno de los vectores del bloque.

$$\overline{e}(k) = [e(kL)\ e(kL+1)\ \cdots\ e(kL+L-1)]^T \tag{3.14}$$

$$\overline{y}(k) = [y(kL)\ y(kL+1)\ \cdots\ y(kL+L-1)]^T \tag{3.15}$$

A partir de estas nuevas matrices, es posible reescribir la Ec. (3.8), que representa la adaptación de los coeficientes.

$$\boxed{\overline{w}(k+1) = \overline{w}(k) + \frac{\mu}{L} X^T(k)\ \overline{e}(k)} \tag{3.16}$$

Haciendo un análisis dimensional del producto matricial, puede verse que $X(k)$ tiene dimensión $L \times M$, por lo que su transpuesta tendrá dimensión $M \times L$, y es multiplicada por $\overline{e}(k)$, que tiene dimensión $L \times 1$. Por lo tanto el producto matricial es realizable y tendrá dimensión $M \times 1$, coincidiendo con la dimensión del vector de coeficientes $\overline{w}(k)$.

Así, mediante la Ec. (3.8) o la Ec. (3.16) queda determinado el algoritmo de adaptación de coeficientes, ya sea expresado como sumatoria o matricialmente.

Un aspecto que hasta el momento no se tuvo en consideración es el tamaño que debe tener el bloque, es decir la determinación del valor de L. Si se trabaja con $L > M$, la estimación del gradiente utiliza más información que el filtro adaptivo en sí, teniendo como resultado la realización de operaciones redundantes. Si en cambio se emplea un valor $L < M$, el filtro ecualizador sería más largo que el bloque de entrada siendo procesado, resultando en un desperdicio de coeficientes del filtro. Teniendo en cuenta

las consideraciones anteriores, se suele utilizar en la práctica un tamaño de bloque igual a la cantidad de coeficientes del ecualizador, es decir $L = M$.

3.3. Comparación entre BLMS - LMS

Primeramente, cabe mencionar que el algoritmo LMS tradicional es simplemente un caso particular del algoritmo BLMS en el que la longitud del bloque es 1, es decir $L = 1$. Por lo tanto, es esperable que el algoritmo BLMS no tenga un comportamiento sustancialmente diferente al caso tradicional.

Debido a que la estimación del gradiente se lleva a cabo a través de un promedio de los datos recibidos, al trabajar con bloques de datos en simultáneo se logra un promedio más realista que en el caso tradicional, por lo que la estimación del gradiente será más precisa al trabajar con BLMS, y será más precisa mientras mayor sea el tamaño del bloque. Esto se traduce en que el error en estado estacionario resultante tendrá un valor menor que en el caso tradicional.

Por otra parte, al llevarse a cabo la actualización de los coeficientes cada L muestras de entrada, la incidencia del paso es relativamente menor que en el caso LMS tradicional. Como consecuencia, puede ocurrir que para un mismo paso de adaptación, el algoritmo tradicional no converja pero el BLMS sí lo haga.

Por la misma razón que ocurre lo expresado anteriormente, el algoritmo BLMS necesita procesar una mayor cantidad de muestras para llegar a su estado estable, y además se tiene una menor capacidad de seguimiento a los cambios en el canal.

Por último, el algoritmo BLMS permite una eficiente implementación en el dominio de la frecuencia utilizando la Transformada Rápida de Fourier (*Fast Fourier Transform - FFT*). Este aspecto es analizado en profundidad en el Capítulo 4.

Se puede ver entonces que a la hora de implementarlo, el algoritmo BLMS resulta más eficiente que el algoritmo LMS tradicional, debido al procesamiento en paralelo de las muestras de entrada que se lleva a cabo y a la eficiente implementación en el dominio de la frecuencia.

Capítulo 4

Ecualizador Adaptivo en el Dominio de la Frecuencia

Resumen

En este capítulo se introduce el concepto de procesamiento en el dominio de la frecuencia, evidenciando cómo es posible trabajar en dicho dominio, mediante el uso de la Transformada de Fourier, para implementar un algoritmo equivalente al LMS/BLMS en el dominio del tiempo.

Se demuestra también cómo se reduce considerablemente la complejidad computacional al trabajar en el dominio de la frecuencia a medida que se incrementa la cantidad de coeficientes del ecualizador.

4.1. Filtrado en el Dominio de la Frecuencia

A lo largo de esta sección se explica la forma en que se calcula la Transformada de Fourier para pasar del dominio del tiempo al dominio de la frecuencia y cómo sacarle provecho a sus propiedades para realizar el filtrado de una señal. Esto resulta de interés considerando que el Ecualizador es un filtro por el cual debe pasar la señal a transmitir, por lo tanto es de gran importancia que este proceso se realice de la forma más eficiente posible. Para esto, es necesario partir de la definición de la Transformada Discreta de Fourier (*Discrete Fourier Transform - DFT*), tomando como referencia para las siguientes secciones los desarrollos de [11].

4.1.1. Transformada Discreta de Fourier

La Transformada Discreta de Fourier permite representar una secuencia discreta finita $x(n)$ utilizando muestras de su Transformada de Fourier $X(\omega)$. Considerando una secuencia discreta $x(n)$ con longitud finita de N

muestras, es decir que $x(n) = 0$ fuera de un determinado intervalo de N muestras, es posible asociarla a la secuencia periódica $\widetilde{x}(n)$ de la Ec. (4.1).

$$\widetilde{x}(n) = \sum_{r=-\infty}^{\infty} x(n - rN) = x(n \; modulo \; N) = x(n_N) \tag{4.1}$$

Siendo posible recuperar la señal de longitud finita a partir de su equivalente periódica, simplemente extrayendo un período de la misma, como se ve en la Ec. (4.2).

$$x(n) = \begin{cases} \widetilde{x}(n) & \text{si } 0 \leqslant n \leqslant N-1 \\ 0 & \text{en el resto} \end{cases} \tag{4.2}$$

Es posible demostrar que los coeficientes del desarrollo en serie de Fourier $\widetilde{X}(k)$ de una señal periódica, como $\widetilde{x}(n)$, son muestras de la transformada de Fourier de $x(n)$ espaciadas en $2\pi/N$, y también constituyen una señal periódica de período N. Es decir que ambas señales periódicas se encuentran relacionadas a través de las Ec. (4.3) y (4.4).

$$\widetilde{X}(k) = \sum_{n=0}^{N-1} \widetilde{x}(n) e^{-j\frac{2\pi}{N}kn} \tag{4.3}$$

$$\widetilde{x}(n) = \frac{1}{N} \sum_{k=0}^{N-1} \widetilde{X}(k) e^{j\frac{2\pi}{N}kn} \tag{4.4}$$

A los fines de mantener una dualidad entre el dominio del tiempo y el dominio de la frecuencia, se definen los coeficientes de Fourier asociados a la señal de duración finita $x(n)$ como una secuencia de duración finita correspondiente a un período de $\widetilde{X}(k)$.

$$X(k) = \begin{cases} \widetilde{X}(k) & \text{si } 0 \leqslant k \leqslant N-1 \\ 0 & \text{en el resto} \end{cases} \tag{4.5}$$

La secuencia de duración finita $X(k)$ es lo que se conoce como **Transformada Discreta de Fourier**.

Como $\widetilde{X}(k)$ trabaja con $\widetilde{x}(n)$ solamente en un intervalo que va de 0 a $N-1$, es posible reemplazar dicho valor por la señal finita $x(n)$, ya que ambas son iguales en dicho intervalo. Considerando esto y expresando implícitamente que las ecuaciones toman valores nulos fuera del intervalo indicado, es posible expresar las ecuaciones de análisis y de síntesis de la DFT a través de las Ec. (4.6) y (4.7).

$$\textit{Ecuación de análisis - } X(k) = \sum_{n=0}^{N-1} x(n) e^{-j\frac{2\pi}{N}kn} \quad 0 \leqslant k \leqslant N-1 \tag{4.6}$$

$$\textit{Ecuación de síntesis - } x(n) = \frac{1}{N} \sum_{k=0}^{N-1} X(k) e^{j\frac{2\pi}{N}kn} \quad 0 \leqslant n \leqslant N-1 \tag{4.7}$$

Para relacionar estas ecuaciones se utiliza la notación de la Ec. (4.8).

$$x(n) \xrightarrow{\text{DFT}} X(k) \tag{4.8}$$

4.1.2. Convolución Periódica

Partiendo de dos señales $\widetilde{x_1}(n)$ y $\widetilde{x_2}(n)$, periódicas de periodo N, si se realiza entre ambas una suma de convolución y dicha suma sólo se aplica en un periodo de las señales, se obtiene como resultado una *Convolución Periódica* de periodo N como se muestra en la Ec. (4.9).

$$\widetilde{x_3}(n) = \sum_{m=0}^{N-1} \widetilde{x_1}(m) \widetilde{x_2}(n-m) \tag{4.9}$$

Esta expresión es similar a la convolución no periódica pero se realiza en el intervalo $0 \leqslant m \leqslant N-1$.

Esta expresión tiene su correspondiente secuencia en desarrollo de Fourier y es la que se muestra en la Ec. (4.10).

$$\sum_{m=0}^{N-1} \widetilde{x_1}(m) \widetilde{x_2}(n-m) \longleftrightarrow \widetilde{X_1}(k) \widetilde{X_2}(k) \tag{4.10}$$

Por lo tanto, la convolución periódica de secuencias periódicas corresponde a la multiplicación de las correspondientes secuencias periódicas de los coeficientes de sus desarrollos en serie de Fourier.

4.1.3. Convolución Circular

Considerando dos secuencias de duración finita $x_1(n)$ y $x_2(n)$, ambas de longitud N, que son equivalentes a las secuencias periódicas $\widetilde{x_1}(n)$ y $\widetilde{x_2}(n)$ tomadas en un solo período, tal como se expresa en la Ec. (4.2), es posible obtener la secuencia $x_3(n)$ como se muestra en la Ec. (4.11).

$$x_3(n) = \sum_{m=0}^{N-1} \widetilde{x}_1(m)\widetilde{x}_2(n-m) \qquad 0 \leqslant n \leqslant N-1 \tag{4.11}$$

Como en el intervalo de la sumatoria se cumple $\widetilde{x}_1(m) = x_1(m)$, y utilizando la notación de la Ec. (4.1) para la equivalencia entre secuencias periódicas y secuencias finitas, es posible reescribir la Ec. (4.11) como se muestra en la Ec. (4.12).

$$x_3(n) = \sum_{m=0}^{N-1} x_1(m)x_2(n-m)_N \tag{4.12}$$

La convolución definida en la Ec. (4.12) es diferente a la convolución lineal utilizada normalmente. En este caso la segunda señal se invierte en el tiempo y se desplaza de manera circular sobre la primera, dando lugar a la *Convolución Circular*. Dependiendo de la cantidad de puntos que tengan las secuencias $x_1(n)$ y $x_2(n)$, se expresa la convolución circular de N puntos como indica la Ec. (4.13).

$$x_3(n) = x_1(n)\textcircled{N}x_2(n) \qquad 0 \leqslant n \leqslant N-1 \tag{4.13}$$

Para comprender mejor el modo de resolución de la convolución circular, se muestra en la Fig. 4.1 un ejemplo gráfico de cómo se computan los valores de la misma.

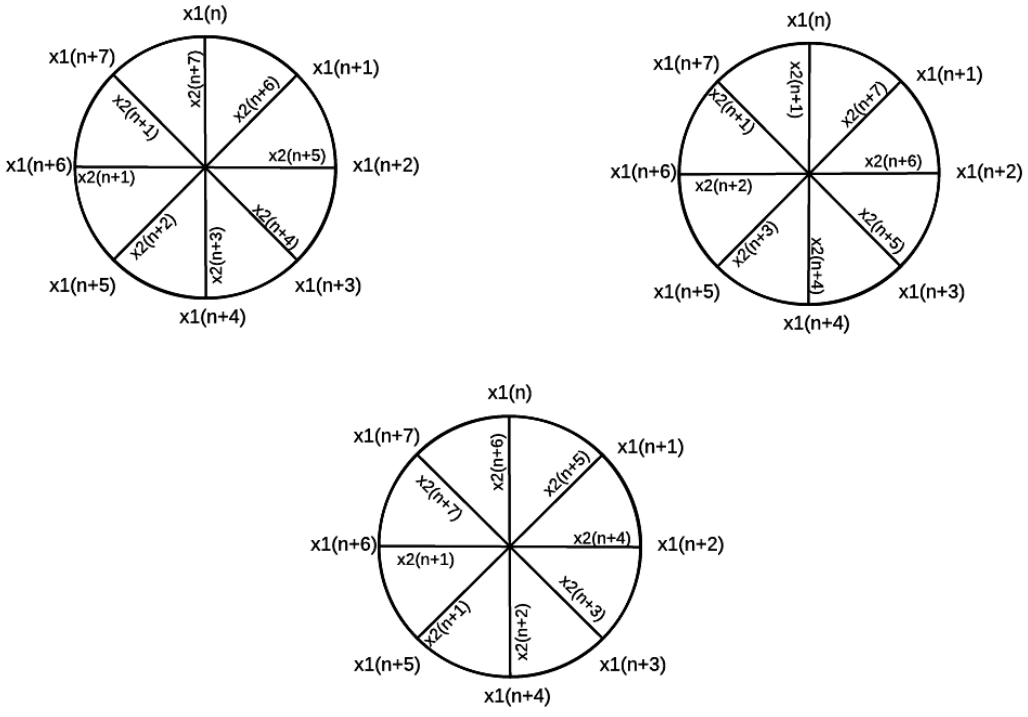


Figura 4.1: *Ejemplo de convolución circular.*

En cada una de las iteraciones, los valores van rotando circularmente y se calcula para cada posición un valor de la salida $x_3(n)$. De esta forma, se tendrán 8 resultados de la salida diferentes (N en el caso general), y luego los mismos se repetirán de manera periódica.

4.1.4. Producto de dos DFT

Partiendo de 2 secuencias discretas $x_1(n)$ y $x_2(n)$, de duración finita N, con sus respectivas DFT de N puntos.

$$X_1(k) = \sum_{n=0}^{N-1} x_1(n) e^{-j\frac{2\pi}{N}kn} \quad 0 \leqslant k \leqslant N-1 \tag{4.14}$$

$$X_2(k) = \sum_{n=0}^{N-1} x_2(n) e^{-j\frac{2\pi}{N}kn} \quad 0 \leqslant k \leqslant N-1 \tag{4.15}$$

El producto de ambas transformadas entre sí será también una DFT de N puntos, correspondiente a una secuencia $x_3(n)$, cuya relación con las secuencias originales $x_1(n)$ y $x_2(n)$ se desea obtener.

Es posible obtener $x_3(m)$ tomando la DFT inversa de $X_3(k) = X_1(k)X_2(k)$.

$$x_3(m) = \frac{1}{N} \sum_{k=0}^{N-1} X_3(k) e^{j\frac{2\pi}{N}kn} = \frac{1}{N} \sum_{k=0}^{N-1} X_1(k)X_2(k) e^{j\frac{2\pi}{N}kn} \tag{4.16}$$

Reemplazando en la Ec. (4.16) los valores de las DFT de las Ec. (4.14) y (4.15), y reordenando luego las sumatorias.

$$x_3(m) = \frac{1}{N} \sum_{k=0}^{N-1} \left[\sum_{n=0}^{N-1} x_1(n) e^{-j\frac{2\pi}{N}kn} \right] \left[\sum_{l=0}^{N-1} x_2(l) e^{-j\frac{2\pi}{N}kl} \right] e^{j\frac{2\pi}{N}km}$$

$$x_3(m) = \frac{1}{N} \sum_{n=0}^{N-1} x_1(n) \sum_{l=0}^{N-1} x_2(l) \left[\sum_{k=0}^{N-1} e^{j\frac{2\pi}{N}k(m-n-l)} \right] \tag{4.17}$$

Analizando la sumatoria en el interior de los corchetes de la Ec. (4.17), considerando $a = e^{j\frac{2\pi}{N}(m-n-l)}$, se puede ver que tiene la forma de la Ec. (4.18).

$$\sum_{k=0}^{N-1} a^k = \begin{cases} N & \text{si } a = 1 \\ \frac{1-a^N}{1-a} & \text{si } a \neq 1 \end{cases} \tag{4.18}$$

Teniendo en cuenta que $a = 1$ ocurre solamente cuando $m - n - l$ da como resultado un múltiplo de N, y que $a^N = 1$, por lo que $1 - a^N = 0$, la Ec. (4.18) puede reducirse a la Ec. (4.19), siendo p un entero.

$$\sum_{k=0}^{N-1} a^k = \begin{cases} N & \text{si } l = m - n - pN \\ 0 & \text{en el resto} \end{cases} \tag{4.19}$$

A partir del resultado de la sumatoria de la Ec. (4.19), es posible expresar $x_3(m)$ considerando solamente los términos no nulos, es decir sólo para $l = m - n - pN$, en cuyo caso el resultado de la sumatoria será N y se cancelará con el término $1/N$, resultando así la Ec. (4.20), en la que se emplea la notación de módulo introducida en la Ec. (4.1).

$$x_3(m) = \sum_{n=0}^{N-1} x_1(n)x_2(m - n - pN) = \sum_{n=0}^{N-1} x_1(n)x_2((m - n)_N) \tag{4.20}$$

La Ec. (4.20) tiene la forma de una convolución circular, por lo tanto se deduce que a partir del producto de las DFT se obtiene la convolución circular de las secuencias finitas representadas por dichas DFT.

4.1.5. Convolución Lineal mediante DFT

Debido a la existencia de algoritmos eficientes para computar la DFT de una secuencia, conocidos como Transformada Rápida de Fourier, resultaría sumamente útil aplicarla para realizar operaciones de filtrado en comunicaciones digitales. Es decir que se computaría la DFT de las secuencia de entrada y del filtro, se haría el producto en frecuencia y luego se haría la transformada inversa para obtener finalmente la secuencia de salida del filtro. Sin embargo, estas operaciones de filtrado requieren llevar a cabo convoluciones lineales, mientras que a través del producto de DFT se obtienen convoluciones circulares, por lo que es necesario encontrar una equivalencia entre ambas.

Dada una secuencia $x(n)$ de longitud L que pasa por un filtro $h(n)$ de longitud M, la salida $y(n)$ tendrá una longitud $L + M - 1$, ya que el producto de la Ec. (4.21) será nulo para cualquier valor de m si $n < 0$ o si $n > L + M - 2$.

$$y(n) = \sum_{m=-\infty}^{\infty} h(m)\, x(n - m) = \sum_{m=0}^{M-1} h(m)\, x(n - m) \tag{4.21}$$

Considerando $L \geqslant M$, si se computan DFT de P puntos para ambas secuencias y luego se hace la DFT Inversa, el resultado será una secuencia de longitud P que coincidirá parcialmente con el resultado de la convolución lineal pero tendrá un solapamiento temporal en las primeras $M-1$ muestras.

Este inconveniente se puede solucionar llevando a cabo DFT de N puntos, siendo $N \geqslant L+M-1$, ya que de esta manera la convolución circular resultante del producto de las DFT será igual a la convolución lineal que se busca obtener. Para lograr esto, se deben rellenar con ceros (colocándolos al final) las secuencias $x(n)$ y $h(n)$ hasta que ambas queden de longitud N, computando luego las DFT de N puntos de estas secuencias. Debe mencionarse que el solapamiento temporal sigue ocurriendo, sólo que ahora ocurre con valores que son nulos y por lo tanto no afecta el resultado.

De esta forma, puede verse que es posible calcular la convolución lineal de dos secuencias finitas a partir de sus DFT.

4.1.6. Método Overlap-Save

En la gran mayoría de las aplicaciones prácticas, se tiene una secuencia de entrada $x(n)$ con una gran longitud, mucho mayor a la del filtro $h(n)$. Si bien teóricamente podría almacenarse la totalidad de la señal de entrada y luego realizar el filtrado utilizando una DFT con una gran cantidad de puntos, esto resultaría muy poco práctico debido a los grandes requerimientos de memoria (para almacenar toda la señal) y a la demora que se introduciría, ya que no se puede realizar el filtrado antes de haber almacenado toda la secuencia de entrada.

Por ello, como se detalla en [14], lo que se hace es dividir la secuencia de entrada en segmentos de longitud L, procesando un bloque a la vez mediante el cómputo de su DFT y produciendo sucesivos bloques de salida que se combinan de forma apropiada para generar una secuencia de salida idéntica a la que se obtendría trabajando directamente con la totalidad de la señal de entrada. Existen 2 métodos principales para llevar a cabo esta tarea, conocidos como Overlap-Save y Overlap-Add. En este caso sólo se considerará el método Overlap-Save.

Con este procedimiento, el tamaño de los bloques que se procesan es $N = L+M-1$, y están constituidos por las $M-1$ últimas muestras del bloque anterior seguidas por L nuevas muestras de la secuencia de entrada. Se computa la DFT de N puntos de cada bloque ($X_m(k)$), como así también la DFT de N puntos del filtro $h(n)$ ($H(k)$), para lo cual se lo rellena con

$L-1$ ceros (debido a que es de tamaño M).

Multiplicando ambas DFT se obtiene $\hat{Y}_m(k)$, con su respectiva Transformada Discreta de Fourier Inversa (*Inverse Discrete Fourier Transform - IDFT*) $\hat{y}_m(n)$, también de N puntos.

$$\hat{Y}_m(k) = H(k)\, X_m(k) \tag{4.22}$$

$$\hat{y}_m(n) = \{\, \hat{y}_m(0)\, \hat{y}_m(1)\, \cdots\, \hat{y}_m(M-1)\, \hat{y}_m(M)\, \cdots\, \hat{y}_m(N-1)\,\} \tag{4.23}$$

La salida obtenida tiene tamaño N pero solamente se han procesado L nuevas muestras, por lo que las primeras $M-1$ muestras de $\hat{y}_m(n)$ están afectadas por *aliasing* y deben descartarse, mientras que las últimas L muestras son exactamente las mismas que se obtendrían a partir de una convolución lineal, por lo que con ellas se arma el bloque de salida $y_m(n)$.

$$y_m(n) = \hat{y}_m(n) \qquad n = M, M+1, \ldots, N-1 \tag{4.24}$$

Para evitar pérdida de información, se almacenan las últimas $M-1$ muestras del bloque procesado y se las utiliza como las primeras $M-1$ muestras del siguiente, tal como puede verse en las ecuaciones siguientes.

$$x_1(n) = \{\, \underbrace{0, 0, \ldots, 0}_{\text{M-1 ceros}}, x(0), x(1), \ldots, x(L-1)\,\} \tag{4.25}$$

$$x_2(n) = \{\, \underbrace{x(L-M+1), \ldots, x(L-1)}_{\text{últimos M-1 datos de } x_1(n)}, \underbrace{x(L), \ldots, x(2L-1)}_{\text{L nuevos datos}}\,\} \tag{4.26}$$

$$x_3(n) = \{\, \underbrace{x(2L-M+1), \ldots, x(2L-1)}_{\text{últimos M-1 datos de } x_2(n)}, \underbrace{x(2L), \ldots, x(3L-1)}_{\text{L nuevos datos}}\,\} \tag{4.27}$$

De esta forma puede apreciarse cómo a partir de los diferentes bloques que se arman, se obtienen los distintos bloques de salida que reagrupados de la manera descripta forman la secuencia de salida deseada.

El procedimiento aplicado puede visualizarse en la Fig. 4.2.

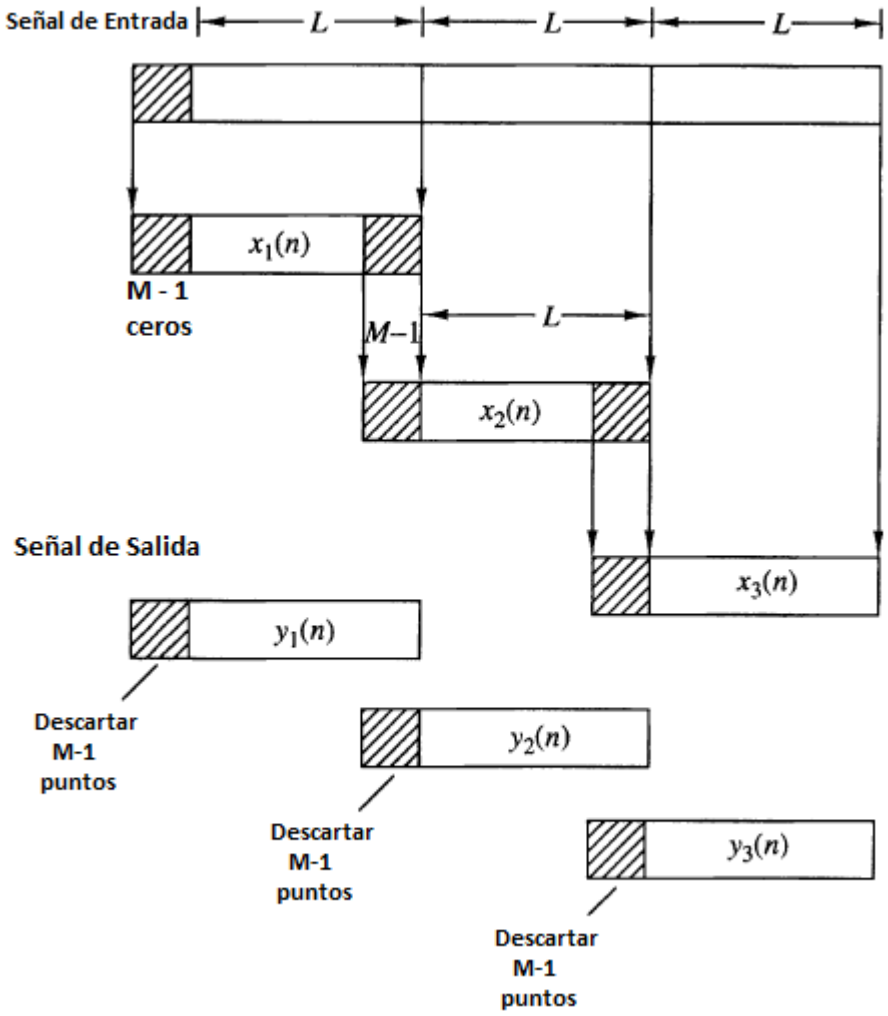


Figura 4.2: *Convolución lineal mediante método Overlap&Save.*

4.2. Ecualizador Adaptivo con Filtrado en Frecuencia

Lo visto en la Sección 4.1 puede aplicarse en el ecualizador utilizando el algoritmo BLMS analizado en el Capítulo 3. Trabajar con filtros adaptivos en el dominio de la frecuencia permite reducir considerablemente la complejidad computacional al trabajar con un elevado número de coeficientes, tal como se muestra en secciones posteriores.

Inicialmente sólo se realiza el filtrado en el dominio de la frecuencia, es decir la convolución entre la entrada y los coeficientes del ecualizador, mientras que la adaptación de los coeficientes se lleva a cabo en el tiempo. Por lo tanto, en cada iteración del algoritmo se debe computar la FFT del bloque de entrada correspondiente y la FFT de los coeficientes del ecualizador, y a partir de ellos calcular la salida correspondiente, para lo cual es necesario multiplicar ambas FFT y realizar la IFFT del resultado

obtenido.

Tanto los bloques de entrada como el vector de salida se van conformando respetando el método de Overlap&Save desarrollado en la Sección 4.1.6, es decir que cada bloque está formado por las últimas $M-1$ muestras del bloque anterior (siendo M la cantidad de coeficientes del ecualizador) y L nuevas muestras de entrada, teniendo un longitud total de $N = L+M-1$. Por esta razón todas las FFT e IFFT que se llevan a cabo son de N puntos. Para el caso de los coeficientes del ecualizador, se los rellena con ceros al final para que alcancen la longitud N deseada. A su vez, el vector de salida se confecciona a partir de las últimas L muestras generadas en cada iteración.

4.3. Ecualizador Adaptivo con Adaptación en Frecuencia

Así como es posible llevar a cabo el filtrado en el dominio de la frecuencia, para reducir aún más la complejidad computacional puede realizarse también la adaptación de los coeficientes en el dominio de la frecuencia, ya que para hacerlo se debe efectuar una correlación, operación bastante similar a la convolución y que por lo tanto puede llevarse a cabo aplicando los conceptos anteriormente vistos si se tienen algunas consideraciones adicionales.

A la hora de implementar un ecualizador adaptivo en el dominio de la frecuencia, se tienen dos alternativas, que pueden apreciarse en las Fig. 4.3 y 4.4.

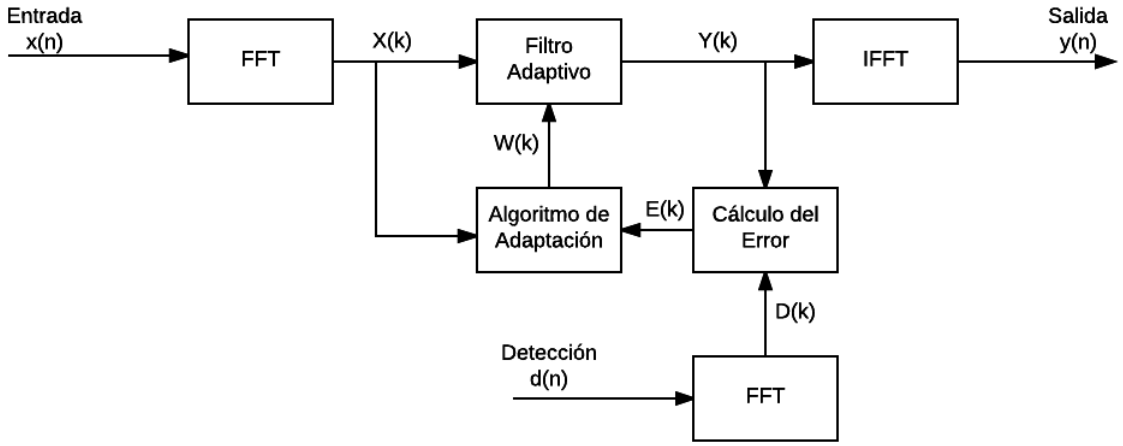


Figura 4.3: *Ecualizador Adaptivo en el Dominio de la Frecuencia con error calculado en frecuencia.*

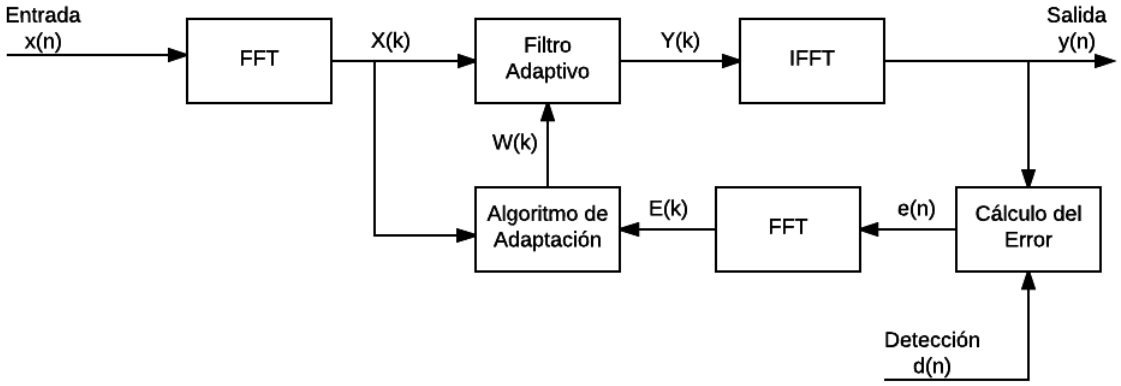


Figura 4.4: *Ecualizador Adaptivo en el Dominio de la Frecuencia con error calculado en tiempo.*

En la configuración de la Fig. 4.3, el error se calcula directamente en el dominio de la frecuencia y se lo utiliza en el algoritmo de adaptación, mientras que en la Fig. 4.4 el error se calcula en el dominio del tiempo y luego se computa su FFT para emplearlo en la adaptación de coeficientes.

En algoritmos de adaptación en los que el error es una función lineal de los datos de entrada, como el LMS que se utiliza en este caso, con ambas configuraciones suelen obtenerse resultados similares, por lo que se decide trabajar, sin un análisis más profundo, con la configuración de la Fig. 4.4, calculando el error en el dominio del tiempo.

4.3.1. Método Constrained

Considerando nuevamente un ecualizador con M coeficientes y que en cada iteración ingresan L nuevas muestras, siendo así $N = L + M - 1$, se define $W(k)$ como la Transformada de Fourier de N puntos del vector de coeficientes $w(n)$, rellenado con ceros hasta alcanzar una longitud igual a N (Ec. (4.28)), y $X(k)$ como la Transformada de Fourier de N puntos del bloque de entrada $x(n)$ que se procesará, confeccionado de la manera que ya se ha explicado en secciones anteriores para aplicar el método Overlap&Save.

$$W(k) = \mathcal{F}\{w^T(n),\, 0,\, \cdots,\, 0\}^T \tag{4.28}$$

A partir de $W(k)$ y $X(k)$, se determina $Y(k)$ y su Transformada Inversa $y_m(n)$, siendo las últimas L muestras de dicho vector las correspondientes a la salida del ecualizador, tal como ya se ha explicado.

$$Y(k) = X(k)\, W(k) \tag{4.29}$$

$$y(n) = \text{últimos L elementos de } \mathcal{F}^{-1}\{Y(k)\} \tag{4.30}$$

Para llevar a cabo la adaptación de coeficientes en el dominio de la frecuencia, debe implementarse una correlación lineal utilizando la FFT de manera similar a la forma en que se hizo con la convolución lineal.

Inicialmente se computa el error en el dominio del tiempo como la diferencia entre la salida obtenida y el símbolo detectado, de la misma forma en la que se lo hacía en el BLMS de secciones anteriores, pero teniendo en cuenta que solamente deben considerarse las últimas L muestras de la salida obtenida. De esta forma se obtiene el vector de error $e(n)$ de longitud L. Una vez que se tiene este vector, se computa su FFT de N puntos ($E(k)$), por lo que se lo debe rellenar con $M-1$ ceros. Como la correlación puede entenderse como una convolución al revés (ya que no se invierte ninguna de las secuencias), esta vez el relleno con ceros se hace al comienzo del vector y no al final.

$$E(k) = \mathcal{F}\{0,\ \cdots,\ 0,\ e^T(n)\}^T \tag{4.31}$$

A partir de este error, es posible calcular la estimación del gradiente $\hat{\nabla}(n)$ mediante el producto de $E(k)$ y el conjugado de $X(k)$, obteniendo así una correlación. Aplicando el mismo razonamiento de que la correlación es una convolución inversa, para obtener los mismos resultados que al hacerlo en el dominio temporal, se deben conservar las primeras M muestras en lugar de las últimas.

$$\hat{\nabla}(n) = \text{ primeros M elementos de } \mathcal{F}^{-1}\{X^*(k)\ E(k)\} \tag{4.32}$$

Finalmente, el gradiente en el tiempo de la Ec. 4.32 se transforma para obtener la estimación del gradiente en el dominio de la frecuencia y luego se lo utiliza para actualizar el valor de los coeficientes, también en el dominio de la frecuencia, llegando así a la Ec. (4.33) de adaptación de coeficientes.

$$W(k+1) = W(k) + 2\mu\, \mathcal{F}\{\hat{\nabla}^T(n),\ 0,\ \cdots,\ 0\}^T \tag{4.33}$$

Se debe notar que en la Ec. (4.33), los coeficientes se actualizan cada L muestras de entrada, ya que las transformadas se computan luego de que se han almacenado L nuevas muestras, por lo tanto los coeficientes permanecen fijos durante ese período de tiempo, tal como ocurre en el BLMS.

El procedimiento completo puede visualizarse en la Fig. 4.5.

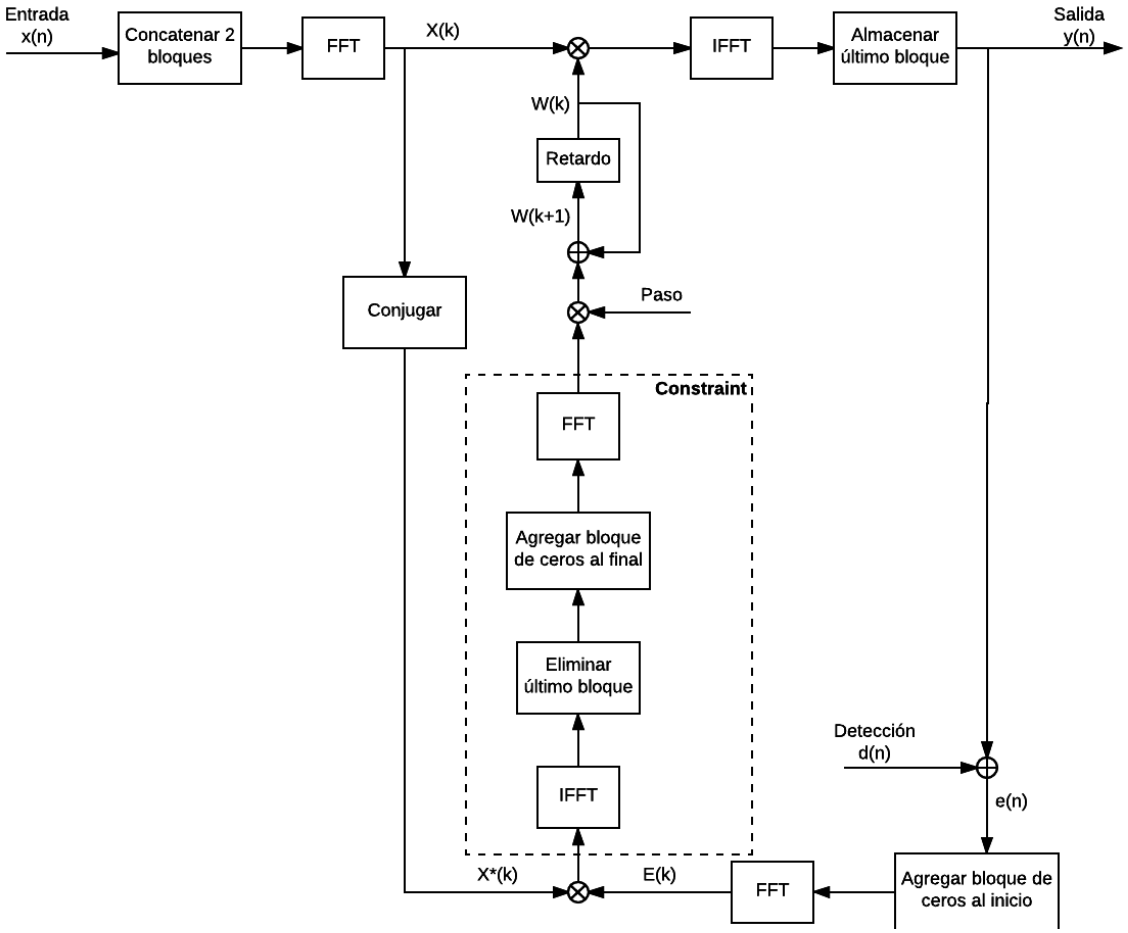


Figura 4.5: *Diagrama de bloques para un Constrained Frequency Domain Adaptive Equalizer.*

A partir de la Fig. 4.5, es posible observar que se necesitan realizar un total de cinco transformaciones (3 FFT y 2 IFFT) para llevar a cabo el proceso de ecualización. De esas cinco, dos (una IFFT y una FFT) corresponden a lo que se denomina *Gradient Constraint*, indicado también en la Fig. 4.5. Esto es necesario debido a que en el dominio del tiempo sólo se tienen M coeficientes cuya FFT de $N = L + M - 1$ puntos se calcula agregando ceros al final, pero los últimos términos de $\mathcal{F}^{-1}\{X^*(k)\,E(k)\}$ no suelen ser cero, por lo tanto es necesario antitransformar y luego transformar nuevamente garantizando que esos términos efectivamente sean cero, para así lograr una correcta adaptación de coeficientes mediante una correlación lineal.

4.3.2. Método Unconstrained

La adaptación en el dominio de la frecuencia puede llevarse a cabo sin utilizar el *Gradient Constraint*. En el diagrama de la Fig. 4.5, simplemente se eliminan los bloques enmarcados como *Constraint* y se trabaja directa-

mente con el producto obtenido, es decir que ya no se hace la IFFT ni la FFT, ni tampoco se reemplaza el final del bloque por ceros. De esta forma, la ecuación de adaptación puede escribirse de una manera más directa, como se puede ver en la Ec. 4.34.

$$W(k+1) = W(k) + 2\mu\, X^*(k)\, E(k) \tag{4.34}$$

De esta forma se logra reducir la complejidad computacional, ya que deben computarse 2 transformaciones menos que en el caso *constrained*, por lo que se necesitan un total de 3 transformaciones (2 FFT y 1 IFFT).

Sin embargo, esta reducción en la complejidad trae aparejada una degradación en el desempeño del algoritmo, debido a que ahora se está realizando una correlación circular en vez de una correlación lineal, por lo que esta adaptación en el dominio de la frecuencia deja de ser equivalente al algoritmo BLMS, lo cual se traduce en peores propiedades de convergencia y en un incremento en el error en estado estable.

4.4. Complejidad Computacional

Para analizar la complejidad de los diferentes algoritmos (LMS tradicional sin procesamiento por bloques, y las versiones constrained y unconstrained del ecualizador en el dominio de la frecuencia), se considera $L = M$ para simplificar el análisis, resultando así $N = 2M$.

El criterio a utilizar para determinar la complejidad de un algoritmo será el número de multiplicaciones reales que deben llevarse a cabo por cada M muestras procesadas. Se toma como criterio sólo a las multiplicaciones debido a que son las operaciones más costosas desde un punto de vista de hardware.

El algoritmo LMS tradicional, que lleva a cabo las convoluciones y correlaciones directamente mediante multiplicaciones y sumas, requiere M multiplicaciones para determinar el valor de la salida (correspondientes a la convolución) y M multiplicaciones adicionales para actualizar los coeficientes (correspondientes a la correlación), resultando así un total de $2M$ multiplicaciones por cada muestra de salida. Por lo tanto, para procesar las $L = M$ muestras que tendría un bloque de entrada en los otros algoritmos, se requieren $2M^2$ multiplicaciones.

Para determinar la complejidad de los ecualizadores en el dominio de la frecuencia, previamente se debe conocer la cantidad de multiplicaciones requeridas para computar las FFT. Existen algoritmos eficientes que permiten implementar cada FFT o IFFT de N puntos (ambas tienen la misma

complejidad), como *radix-2 FFT* que emplea $N \log_2 N$ multiplicaciones, tal como se demuestra en el Apéndice A.

Además de las FFT/IFFT, se deben realizar multiplicaciones adicionales debido al producto de las FFT para hacer la convolución o correlación. Debido a que la FFT da como resultado números complejos, cada producto entre FFT de N puntos requiere un total de $4N$ multiplicaciones reales, por lo que se necesitan un total de $8N$ multiplicaciones adicionales ($4N$ para la convolución y $4N$ para la correlación).

De esta forma, la cantidad total de multiplicaciones que se requieren para procesar $L = M$ muestras es la que se muestra en la Ec. 4.35, siendo a la cantidad de transformadas que se realizan.

$$\text{Multiplicaciones} = a\, N \log_2 N + 8N = a\, 2M \log_2(2M) + 16M \qquad (4.35)$$

Para el caso Constrained, se llevan a cabo 5 FFT por lo que la cantidad de multiplicaciones necesarias es $10M \log_2(2M) + 16M$, mientras que en la alternativa Unconstrained se emplean 3 FFT, resultando un total de $6M \log_2(2M) + 16M$ multiplicaciones.

En todos los casos, puede verse que la complejidad depende directamente de la cantidad de coeficientes del ecualizador (M), por lo que resulta interesante analizar el número de multiplicaciones que se requieren a medida que aumenta el valor de M, tal como puede observarse en la Fig. 4.6.

Es posible apreciar cómo para valores bajos en la cantidad de coeficientes del ecualizador, no resulta conveniente realizar una implementación en bloques en el dominio de la frecuencia ya que el algoritmo LMS tradicional tiene una complejidad menor. Pero a partir de los 64 coeficientes se puede notar una significativa reducción en la cantidad de multiplicaciones necesarias para procesar un bloque de M muestras. De hecho, la reducción es cada vez mayor a medida que se incrementa el número de coeficientes, debido a la eficiencia de la FFT y al procesamiento en bloques.

Por otra parte, puede observarse también que la versión Unconstrained tiene siempre una complejidad menor que la Constrained, independientemente de la cantidad de coeficientes del ecualizador, lo cual es lógico ya que ambos algoritmos llevan a cabo prácticamente las mismas operaciones a excepción de dos transformaciones.

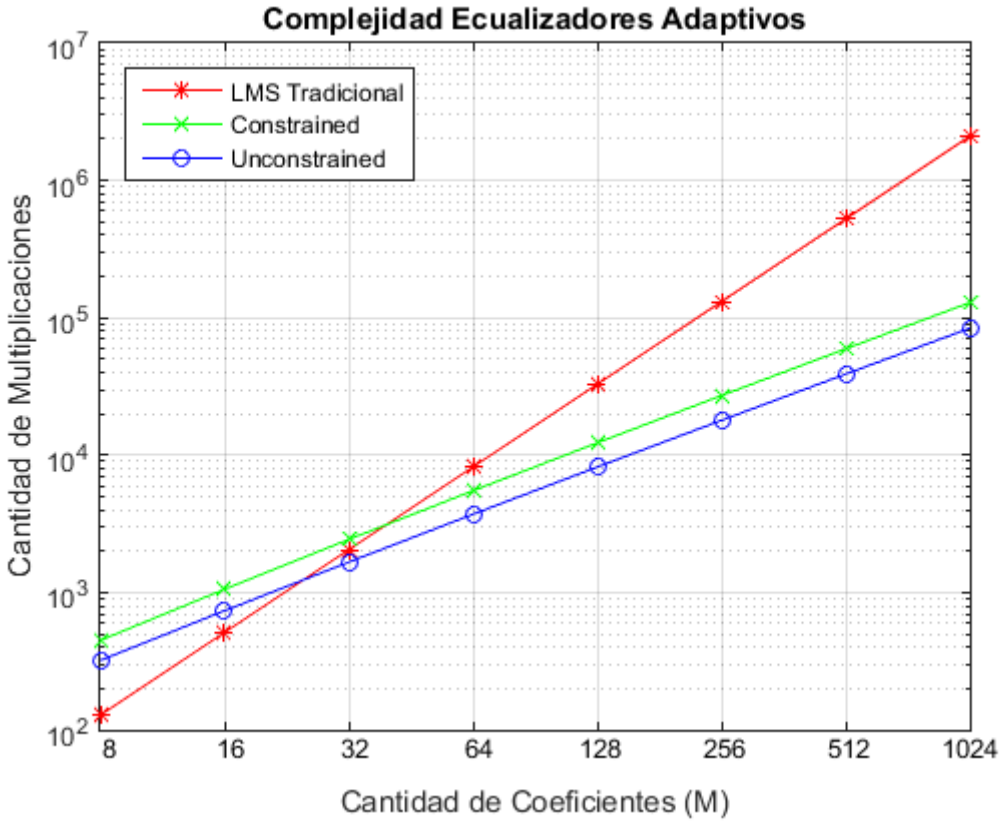


Figura 4.6: *Complejidad de los diferentes algoritmos para Ecualizadores Adaptivos.*

Una forma alternativa de analizar la complejidad es definir una relación de complejidad R entre los ecualizadores en el dominio de la frecuencia y el LMS tradicional en el dominio del tiempo.

$$R = \frac{a\, 2M \log_2(2M) + 16M}{2M^2} \tag{4.36}$$

De esta forma, si $R > 1$ significa que los algoritmos en el dominio de la frecuencia requieren un mayor número de multiplicaciones, mientras que si $R < 1$ el algoritmo en el dominio del tiempo será más complejo. En la tabla 4.1 se muestra el valor que toma R para diferentes valores de M en ambas variantes del ecualizador.

	R							
M	8	16	32	64	128	256	512	1024
Constrained	3.500	2.063	1.188	0.672	0.375	0.207	0.113	0.062
Unconstrained	2.500	1.438	0.813	0.453	0.250	0.137	0.074	0.040

Tabla 4.1: Relación de Complejidad R para diferentes valores de M.

Parte II

Marco Metodológico

Capítulo 5

Simulación de Algoritmos

Resumen

En este capítulo se llevan a cabo simulaciones de los algoritmos de adaptación de coeficientes analizados desde un punto de vista puramente teórico en los capítulos anteriores.

Se efectúan numerosas simulaciones en las que se varían los diferentes parámetros que definen el comportamiento de los algoritmos, trazando curvas de interés y analizando cómo se modifica el desempeño del ecualizador. A su vez, se verifica también que los resultados sean consistentes con la teoría.

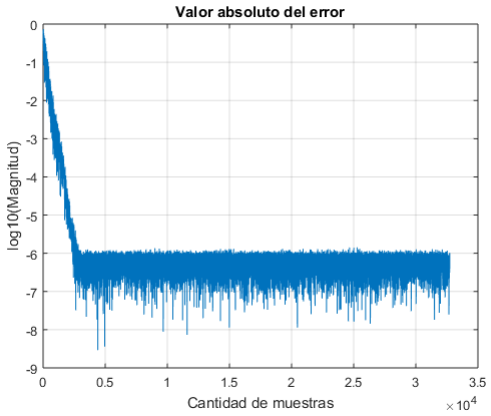
5.1. Algoritmo BLMS en el Dominio del Tiempo

En esta sección se analiza el comportamiento de un algoritmo BLMS, simulándolo en MATLAB. De esta forma se podrán observar los efectos que produce la paralelización del algoritmo LMS y cuáles son las diferencias con el mismo.

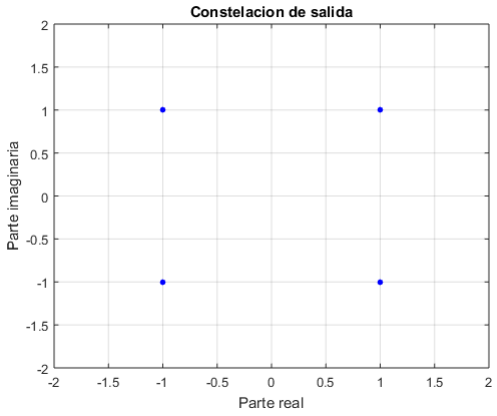
Al igual que el algoritmo LMS, el BLMS se encuentra influenciado por los parámetros originales que son el número de coeficientes del filtro M y el paso de adaptación μ, pero como ya se ha visto en las secciones anteriores, este también se encuentra caracterizado por la cantidad de elementos que posee cada bloque (L).

Para comenzar con este análisis, es necesario recordar el funcionamiento del algoritmo LMS nuevamente. Para esto es posible adaptar el programa utilizado para las simulaciones del BLMS, haciendo $L = 1$. De esta forma el sistema trabaja en serie ya que cada 'bloque' tiene un solo elemento. En este caso se trabaja con $M = 32$, $\mu = 0{,}005$ y un canal de transmisión de dos coeficientes [1, 0.4].

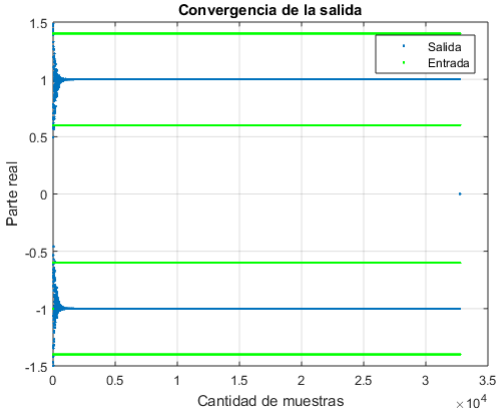
En la Fig. 5.1 se puede ver que el sistema funciona normalmente. Los datos convergen hacia los símbolos de la modulación QPSK, y el error también lo hace hasta un valor de 10^{-6}.



(a) *Valor absoluto del error.*



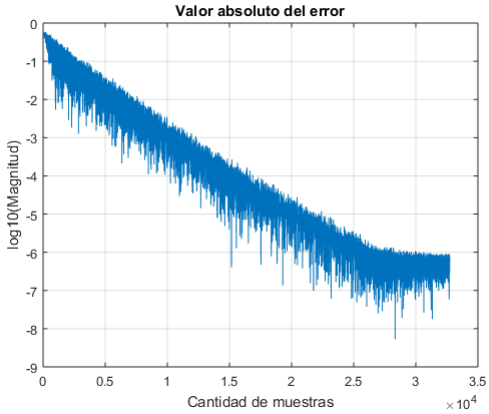
(b) *Constelación de salida.*



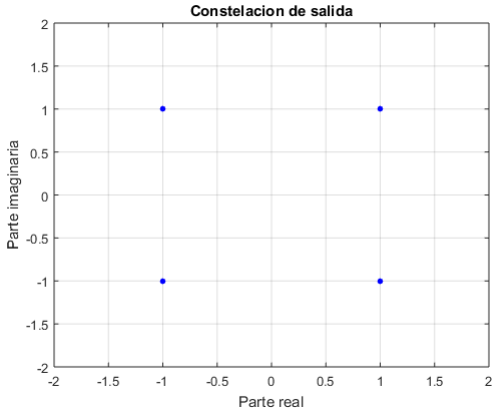
(c) *Convergencia de la salida.*

Figura 5.1: *Ecualizador adaptivo con algoritmo LMS ($L = 1$) para $M = 32$ y $\mu = 0.005$.*

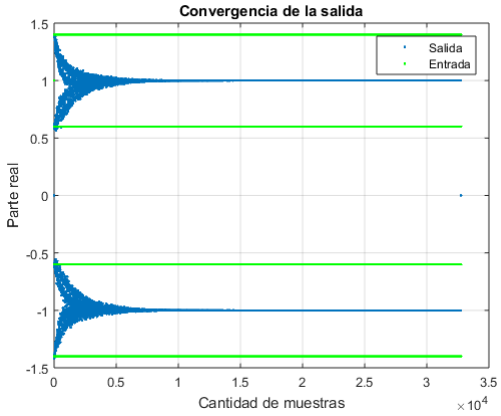
Si se utilizan los mismos parámetros que en el caso anterior pero con un paralelismo $L = 8$, se puede ver en la Fig. 5.2 que el comportamiento del algoritmo de adaptación es sutilmente diferente. Se sigue obteniendo la salida esperada, pero esta vez la convergencia es más lenta, debido precisamente al hecho de que para cada actualización de coeficientes se necesitan L muestras de entrada.



(a) *Valor absoluto del error.*



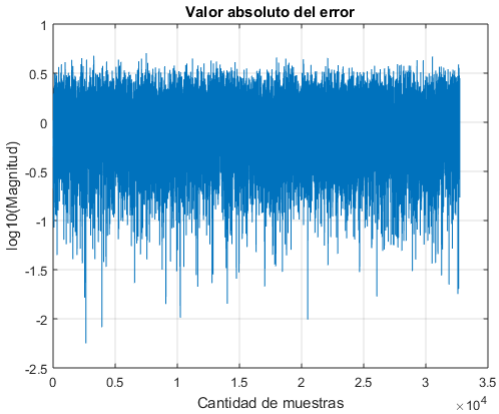
(b) *Constelación de salida.*



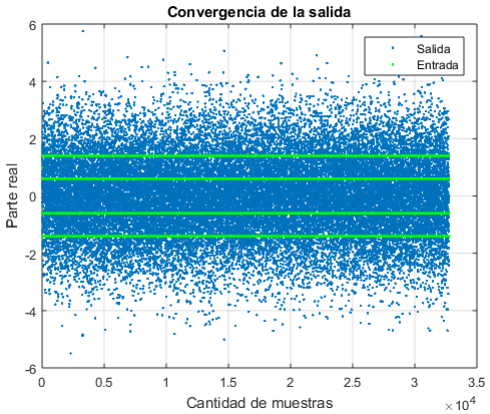
(c) Convergencia de la salida.

Figura 5.2: *Ecualizador adaptivo con algoritmo BLMS para $L = 8$, $M = 32$ y $\mu = 0.005$.*

Si se llevan al extremo los límites de convergencia del algoritmo LMS tradicional, haciendo $\mu = 0{,}03$ y manteniendo $M = 32$, se obtienen los resultados de la la Fig. 5.3, en la que se puede ver que el sistema deja de funcionar, debido a la magnitud del paso, y la salida diverge.



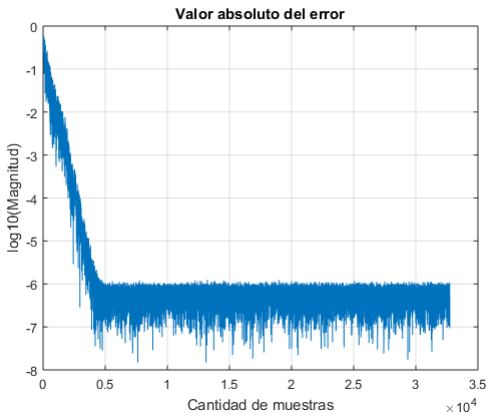
(a) *Valor absoluto del error.*



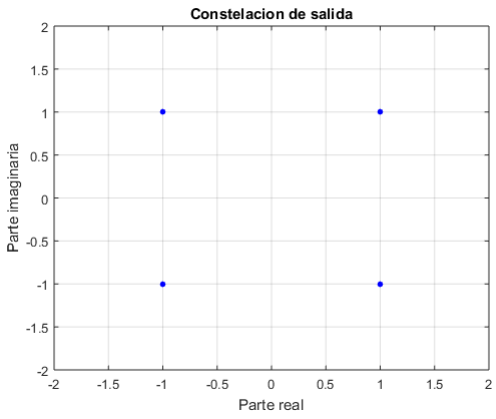
(b) *Convergencia de la salida.*

Figura 5.3: *Ecualizador adaptivo con algoritmo LMS ($L = 1$) para $M = 32$ y $\mu = 0.03$.*

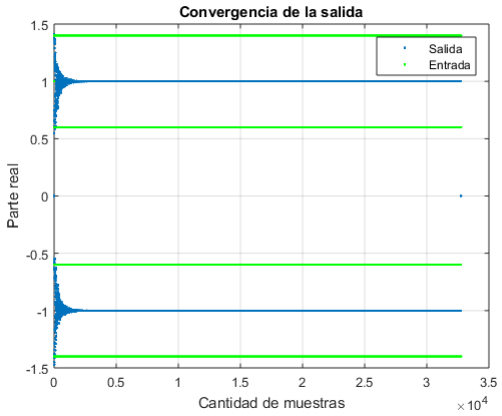
Si ahora se aumenta el valor de L, empleando $L = 8$ y manteniendo los parámetros del algoritmo LMS del ejemplo anterior ($\mu = 0{,}03$ y $M = 32$), en el que los coeficientes del filtro no llegaban a adaptarse en ningún momento y por lo tanto la salida del mismo no convergía, el resultado es el que se ve en la Fig. 5.4, el cual sí converge para el mismo valor de μ.



(a) *Valor absoluto del error.*



(b) *Constelación de salida.*



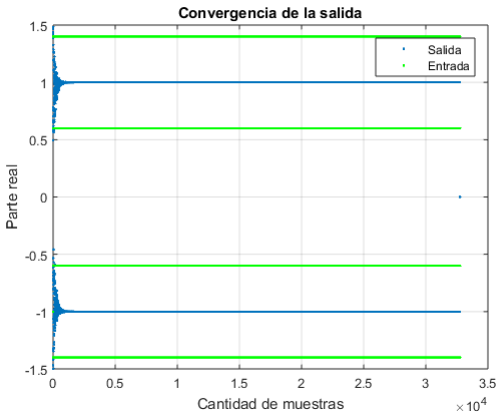
(c) *Convergencia de la salida.*

Figura 5.4: *Ecualizador adaptivo con algoritmo BLMS para $L = 8$, $M = 32$ y $\mu = 0.03$.*

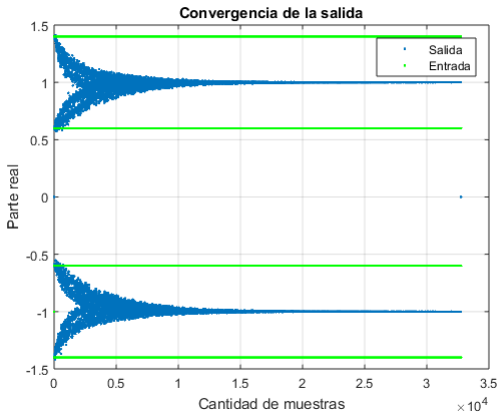
En el ejemplo de la Fig. 5.2 se explica que el sistema converge más lento debido a que se necesita una mayor cantidad de muestras para realizar la actualización de los coeficientes, lo cual conlleva a que el paso de adaptación tenga menos influencia en el algoritmo BLMS que en el LMS. Esta influencia va disminuyendo conforme se aumenta el valor de L ya que, como se vio en secciones anteriores, la ecuación de actualización de coeficientes ahora se encuentra afectada por el factor $\frac{1}{L}$ que divide al paso. Esta es la razón por la cual es posible hacer que el caso de la Fig. 5.3, que no convergía debido a la magnitud del paso, lo haga aumentando el valor de L como se muestra en la Fig. 5.4.

Otro aspecto a considerar en este tipo de ecualizadores es la influencia

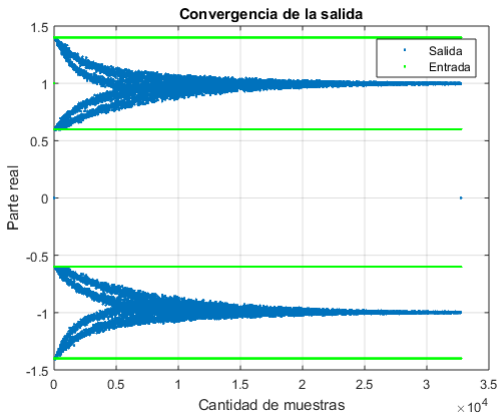
del tamaño del bloque L en la velocidad de convergencia de la salida. Este comportamiento se analiza en la Fig. 5.5, en la que se reduce el paso a $\mu = 0{,}005$ y se aumenta paulatinamente el valor de L, graficando en cada caso la salida del ecualizador a medida que se reciben nuevas muestras.



(a) L = 1.



(b) L = 16.



(c) L = 32.

Figura 5.5: Salida del ecualizador para M = 32 y $\mu = 0.005$, variando el valor L.

Como se puede ver, en la Fig. 5.5a, al utilizar el algoritmo LMS tradicional (es decir con $L = 1$), el sistema funciona correctamente y con una determinada velocidad de convergencia. A medida que se aumenta el valor de L, el tiempo que se requiere para alcanzar la convergencia es cada vez mayor, tal como puede observarse en las Fig. 5.5b y 5.5c. Con $L = 32$, se ve que la cantidad de símbolos graficados no es suficiente para que la salida del sistema se adapte exactamente a los valores de salida de los casos anteriores, aunque eventualmente converge con una mayor cantidad de muestras.

5.2. Algoritmo LMS en el Dominio de la Frecuencia

Ya se ha analizado el comportamiento del sistema utilizando el algoritmo BLMS para obtener un sistema paralelizado, capaz de procesar los datos a mayor velocidad debido a la utilización de bloques de entrada. Durante ese análisis, se comprobó cuáles eran los parámetros con los que se debía tener precaución en la utilización del algoritmo, teniendo en cuenta cómo influyen en su comportamiento. En virtud de esto, los efectos de la cantidad de taps que utiliza el ecualizador (M) y el valor del paso de adaptación (μ) serán los mismos, ya que aún trabajando en el dominio de la frecuencia se obtiene un comportamiento equivalente.

Utilizando la FFT, se trabaja con un procedimiento similar en lo que respecta al bloque de entrada, es decir que la información también se procesa en paralelo, ya que para aplicar la Transformada Discreta de Fourier se precisa una determinada cantidad de muestras de entrada igual a $L+M-1$, siendo L la cantidad de muestras nuevas que ingresan en cada iteración del algoritmo.

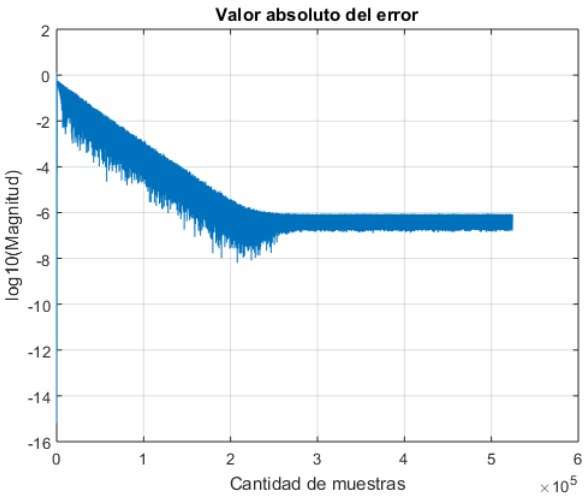
Además, en este caso también es necesario tener en cuenta otros parámetros relacionados al porcentaje de solapamiento de los datos que ingresan, que tienen que ver tanto con la memoria del sistema (asociada a la cantidad de coeficientes M) como con la cantidad de nuevas muestras L. Esto quiere decir que es de interés la relación entre M y L, debiendo cumplirse siempre $L \geqslant M$.

En esta sección se analiza el comportamiento del sistema realizando el filtrado en frecuencia, y luego combinándolo con el algoritmo de adaptación en frecuencia.

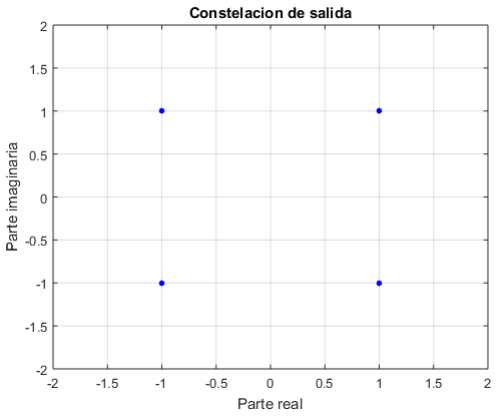
5.2.1. Filtrado en el Dominio de la Frecuencia

Como ya se ha explicado, el primer paso para realizar la ecualización en frecuencia es aplicar la Transformada de Fourier tanto a la señal de entrada como a los coeficientes, para obtener a partir de ellos la salida en el dominio de la frecuencia. Antitransformando la misma, se genera la salida del ecualizador en el dominio del tiempo.

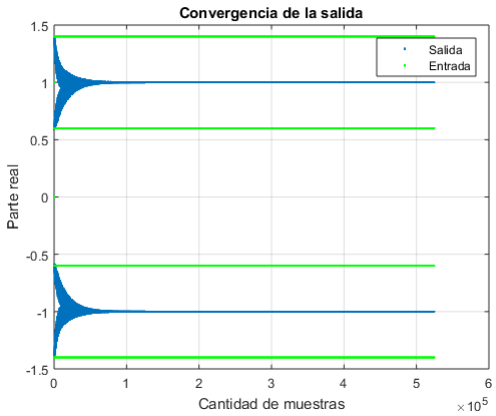
En primera instancia se analiza lo que sucede cuando $L = 64$ y $M = L/2 = 32$, con un paso $\mu = 0{,}005$ que se utiliza a lo largo de todas las simulaciones posteriores. Además, la primera simulación se realiza con un canal de tan solo dos coeficientes para mostrar que el funcionamiento es el mismo que para los casos analizados anteriormente. En la Fig. 5.6 se muestra el resultado con un canal descrito por los coeficientes [1 0,4].



(a) *Valor absoluto del error.*



(b) *Constelación de salida.*



(c) *Salida del ecualizador.*

Figura 5.6: *Ecualización con* $L = 64$ *y* $M = L/2$ *y un canal de dos coeficientes* [1 0,4].

Nuevamente se puede comprobar que el algoritmo converge a la solución correcta, resultado que también puede verificarse analizando la respuesta en frecuencia del sistema y comprobando que la misma sea la inversa del canal, tal como se ve en la Fig. 5.7.

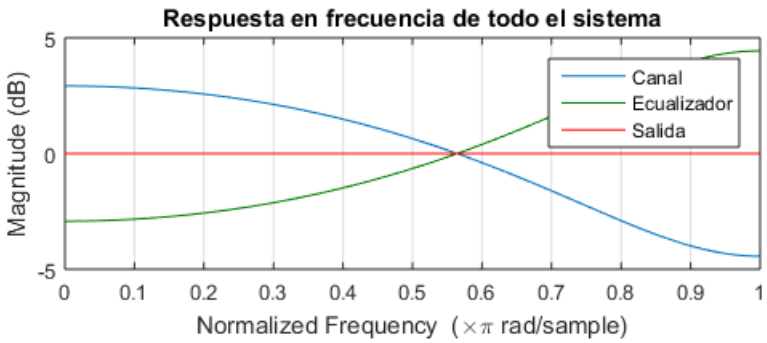


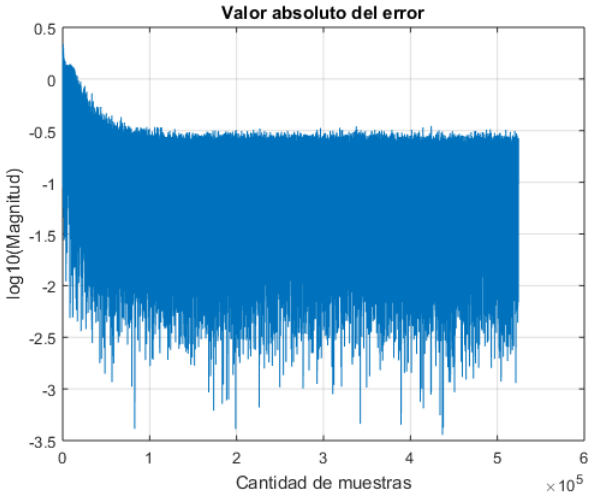
Figura 5.7: *Respuesta en frecuencia del sistema.*

Se debe mencionar que se obtiene un valor de error muy bajo debido

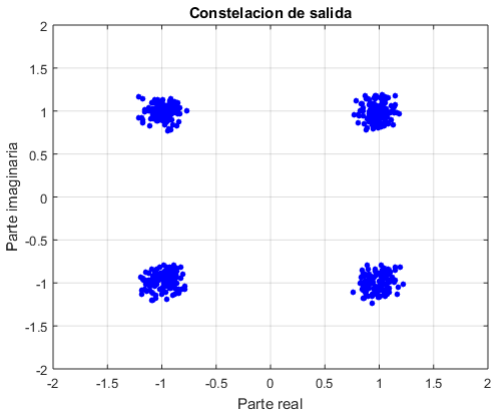
a que se tiene un gran número de coeficientes del ecualizador ($M = 32$), para compensar los efectos de un canal de tan sólo dos coeficientes.

En simulaciones anteriores, se utiliza este mismo canal y es por ello que se realiza esta prueba a modo de comparación, verificando que funciona de manera análoga al caso en el dominio del tiempo.

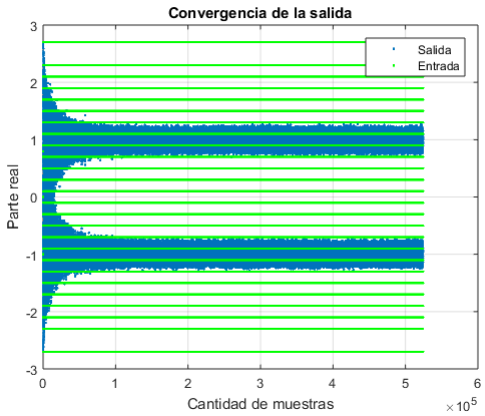
A continuación, se analiza lo que sucede con canales con una cantidad de taps mayor a la utilizada en casos anteriores, para aumentar el valor de ISI del mismo y comprobar que el ecualizador es capaz de compensar estos efectos. En la Fig. 5.8 se muestran los resultados de someter la señal de entrada a un canal de la forma $[-0{,}8\ 0{,}2\ \ -0{,}3\ 1\ 0{,}4]$ con 5 coeficientes, caracterizando al ecualizador con los mismos parámetros que en el caso anterior, es decir $L = 64$, $M = 32$ y $\mu = 0{,}005$.



(a) *Valor absoluto del error.*



(b) *Constelación de salida.*



(c) *Salida del ecualizador.*

Figura 5.8: *Ecualización con $L = 64$ y $M = L/2$ y un canal de cinco coeficientes* $[-0{,}8\ 0{,}2\ \ -0{,}3\ 1\ 0{,}4]$.

En esta situación se puede observar que el desempeño disminuye notablemente. A pesar de que la convergencia tiene lugar en un tiempo relati-

vamente corto, el error al que se arriba es evidentemente superior al de la Fig. 5.6a. Esto también puede verse reflejado en el análisis de frecuencia de la Fig. 5.9, donde se puede notar que los coeficientes del ecualizador no compensan perfectamente los efectos del canal, aunque sí lo describen de manera general.

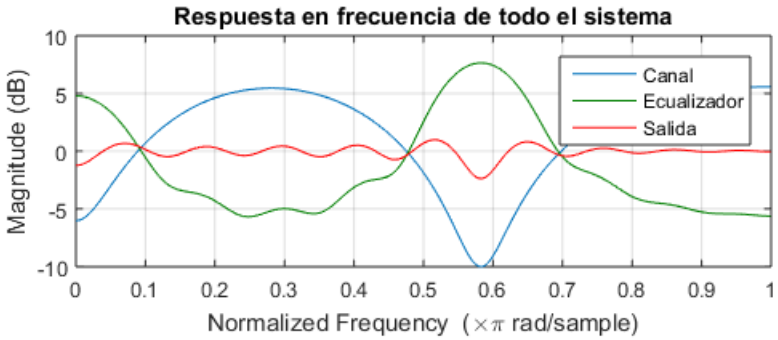


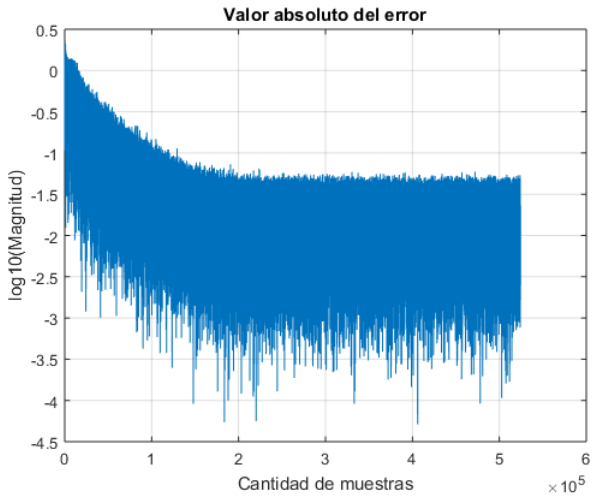
Figura 5.9: *Respuesta en frecuencia del sistema.*

Dado este canal, y conociendo las influencias de los parámetros del ecualizador, para realizar una mejor compensación de los efectos del canal cuando el mismo genera mucha interferencia, una opción es aumentar la cantidad de coeficientes del ecualizador a fin de lograr una mejor adaptación.

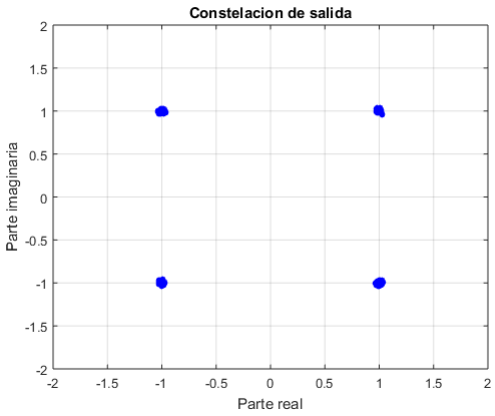
Recordando además que el límite para el valor de M es L, se analiza lo que sucede cuando $M = L = 64$. De esta forma se obtiene un solapamiento del 50 % ya que la cantidad de muestras de entrada será de $L+M-1 = 127$, donde sólo $L = 64$ serán muestras nuevas. Los resultados obtenidos para esta situación se muestran en la Fig. 5.10.

En este caso, el error que se obtiene es bastante menor y se puede notar cómo la salida converge de manera más precisa a los valores 1 y -1, tal como debe hacerlo para una correcta detección de símbolos a la salida.

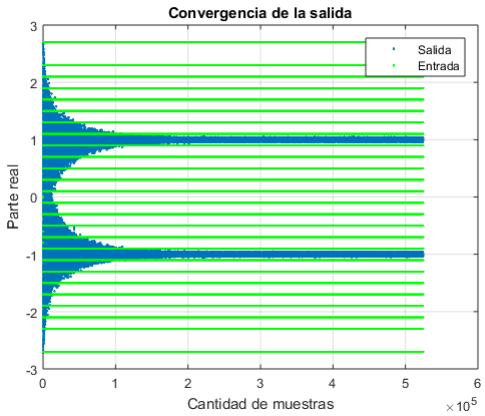
Si se observa una vez más qué es lo que sucede en el dominio de la frecuencia, también encontramos que la compensación del canal se logra de la manera esperada, obteniendo una constante casi perfecta como respuesta de todo el sistema (Fig. 5.11). Es evidente que el mayor problema se presenta en los picos de mayor magnitud generados en el canal, los cuales son mas difíciles de ecualizar.



(a) *Valor absoluto del error.*



(b) *Constelación de salida.*



(c) *Salida del ecualizador.*

Figura 5.10: *Ecualización con $L = 64$ y $M = L$ y un canal de cinco coeficientes* $[-0{,}8\ 0{,}2\ -0{,}3\ 1\ 0{,}4]$.

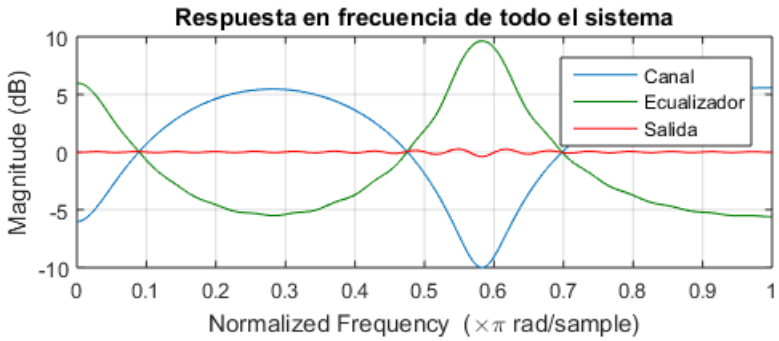
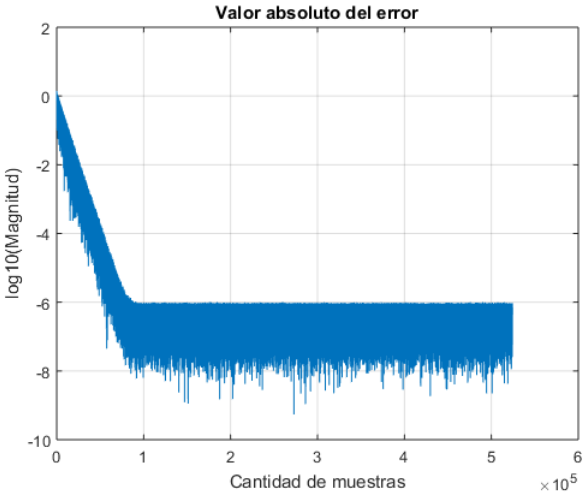


Figura 5.11: *Respuesta en frecuencia del sistema.*

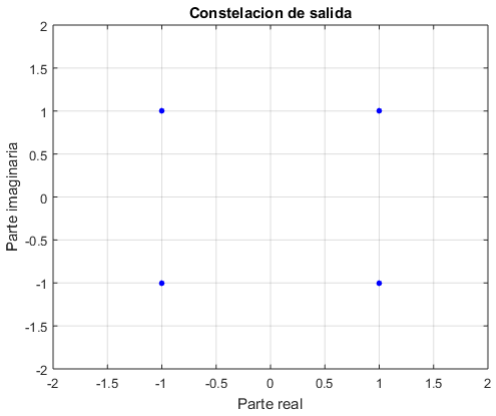
Habiendo realizado el incremento en la cantidad de taps, llevando el valor de M a su límite máximo, se tiene un error de una cierta magnitud, por lo que a continuación se analiza lo que sucede si se elimina uno de los coeficientes de mayor peso en el canal y que genera las complicaciones

mencionadas. Así, el nuevo canal con el que se trabaja es de la forma $[0{,}2\ -0{,}3\ 1\ 0{,}4]$.

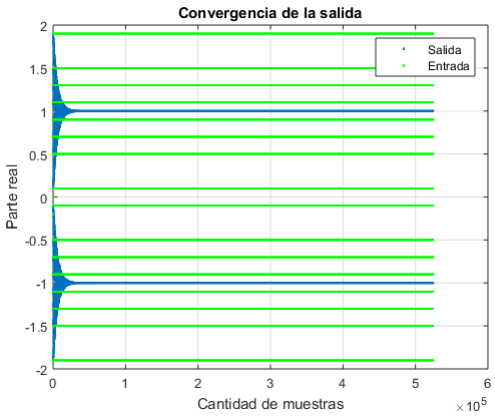
Eliminando este coeficiente, el desempeño del ecualizador mejora de manera notable. En la Fig. 5.12 se puede ver cómo se obtiene una excelente compensación del canal utilizando $M = L/2 = 32$ coeficientes del ecualizador. En la Fig. 5.12a se muestra cómo el error toma valores mucho menores llegando a converger en una magnitud de 10^{-6}. De esta forma, la salida converge rápidamente al valor deseado y se logra un muy buen funcionamiento del ecualizador.



(a) *Valor absoluto del error.*



(b) *Constelación de salida.*

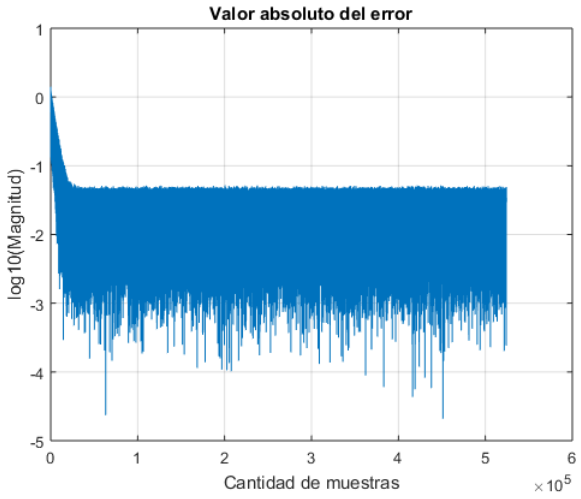


(c) *Salida del ecualizador.*

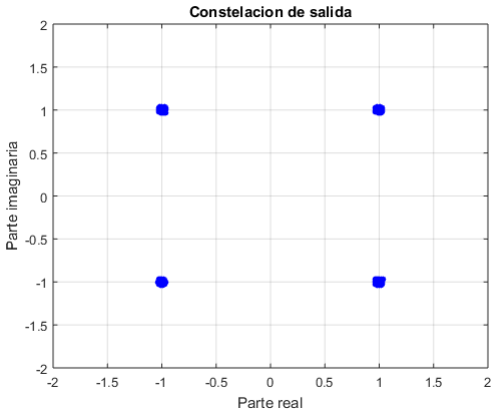
Figura 5.12: *Ecualización con* $L = 64$ *y* $M = L/2$ *y un canal de cuatro coeficientes* $[0{,}2\ -0{,}3\ 1\ 0{,}4]$.

En vista de estos resultados, es posible disminuir aún mas la cantidad de taps del ecualizador y seguir consiguiendo un desempeño aceptable del mismo. Teniendo en cuenta esto, se analiza qué sucede con valores de M menores, llegando a un caso límite en el que $M = L/8 = 8$. Con estos valores se logra un funcionamiento aceptable del ecualizador, con una buena

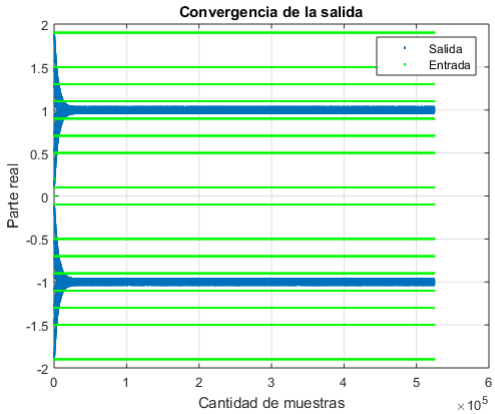
convergencia pero con un valor de error más alto, como se muestra en la Fig. 5.13.



(a) *Valor absoluto del error.*



(b) *Constelación de salida.*



(c) *Salida del ecualizador.*

Figura 5.13: *Ecualización con* $L = 64$ *y* $M = L/8$ *y un canal de cuatro coeficientes* $[0{,}2 \ -0{,}3 \ 1 \ 0{,}4]$.

Resulta sumamente evidente que la influencia de los coeficientes del canal sobre el ecualizador es muy importante y su funcionamiento depende fuertemente de ellos.

En la Fig. 5.14, se muestra la respuesta en frecuencia del sistema y se aprecia que la forma del canal tiene cambios mucho menos abruptos que en el canal analizado anteriormente (Fig. 5.11), lo que explica el hecho de que el desempeño del ecualizador sea ampliamente superior al momento de compensarlo.

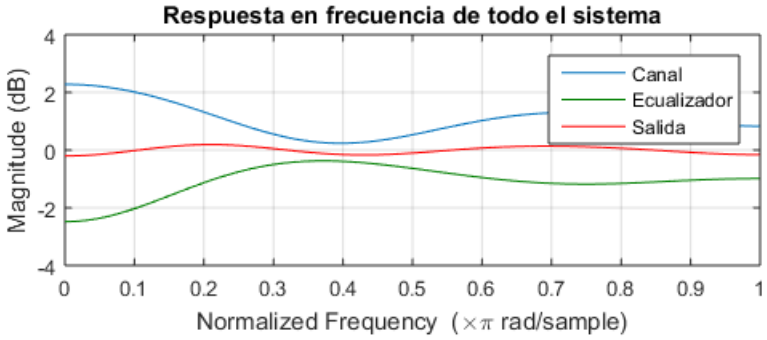
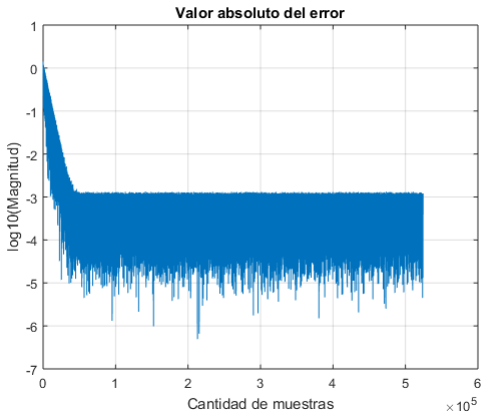
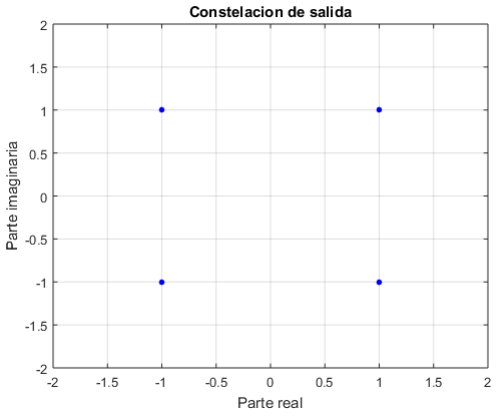


Figura 5.14: *Respuesta en frecuencia del sistema.*

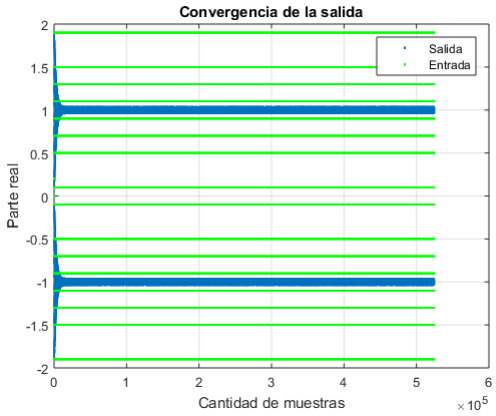
Por último, se estudia cómo influye el valor de L sobre este mismo canal, manteniendo la cantidad de coeficientes en $M = 8$. Se trabaja con $L = 32$ y se obtienen los resultados que se ven en la Fig. 5.15. El comportamiento es bastante similar al anterior, pudiéndose apreciar solamente una pequeña diferencia respecto al tiempo de convergencia, que está directamente relacionado con el grado de paralelismo que se utilice.



(a) *Valor absoluto del error.*



(b) *Constelación de salida.*



(c) *Salida del ecualizador.*

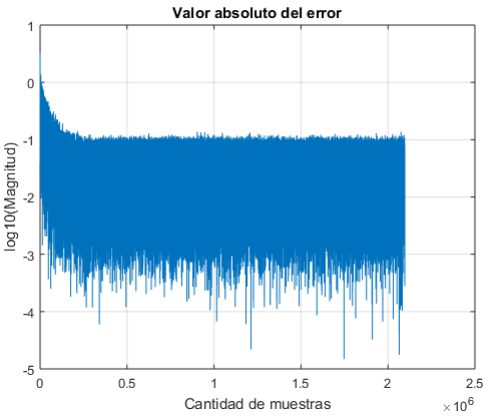
Figura 5.15: *Ecualización con $L = 32$ y $M = L/4$ y un canal de cuatro coeficientes* $[0{,}2\ \ -0{,}3\ 1\ 0{,}4]$.

5.2.2. Adaptación en el Dominio de la Frecuencia

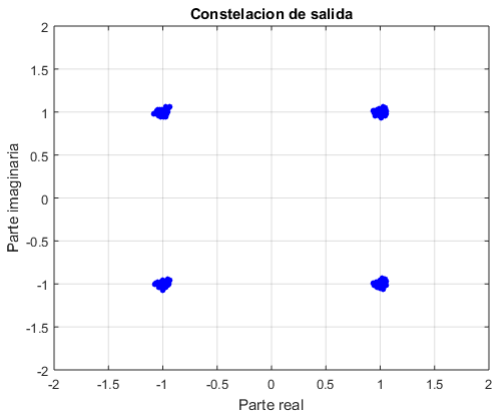
Como se explica en la Sección 4.3, existen dos métodos diferentes para llevar acabo la adaptación en frecuencia. En la implementación práctica de este proyecto se utiliza el método *Unconstrained*, el cual tiene un menor costo de implementación pero a su vez presenta un mayor nivel de error en estado estable que el *Constrained*.

En esta sección se hace énfasis en el método a implementar, es decir el método *Unconstrained*, pero de todos modos se efectúa una comparación con el otro caso para denotar la diferencia que existe entre ambos.

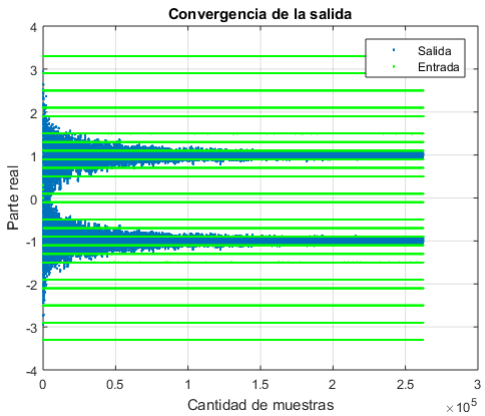
En lo que se refiere al método Unconstrained, es necesario tener especial cuidado en torno a la convergencia del sistema, ya que se puede perder la estabilidad del mismo en ciertas ocasiones en las que el método Constrained sí converge de manera correcta. Por ejemplo, en la Fig. 5.16 se ve cómo se comporta el ecualizador utilizando el método Constrained con los parámetros $L = M = 64$, $\mu = 0{,}0005$ y un canal de coeficientes $[0{,}2 \ -0{,}7 \ 1 \ 0{,}4 \ -1{,}0]$.



(a) *Valor absoluto del error.*



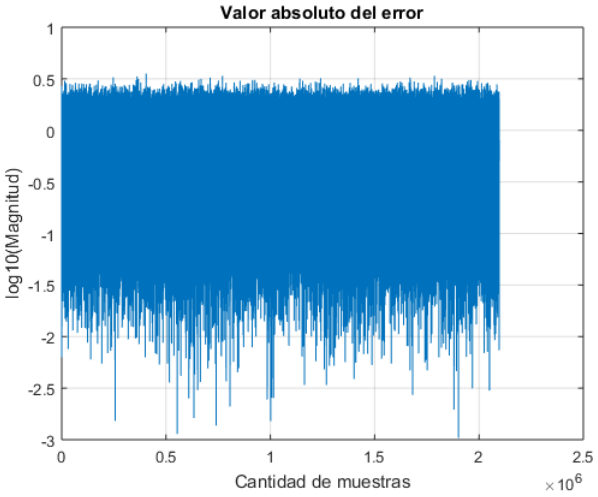
(b) *Constelación de salida.*



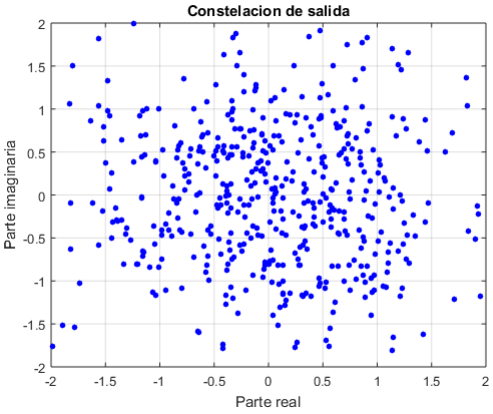
(c) *Salida del ecualizador.*

Figura 5.16: *Ecualización con* $L = 64$, $M = L$, $\mu = 0{,}0005$ *y un canal de coeficientes* $[0{,}2 \ -0{,}7 \ 1 \ 0{,}4 \ -1{,}0]$ *utilizando el método Constrained.*

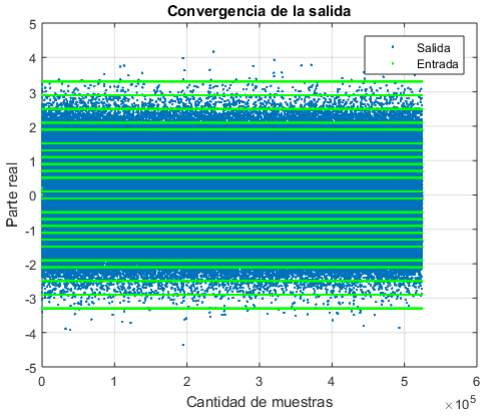
Se puede apreciar que el algoritmo converge y la salida del ecualizador se mantiene en los valores adecuados. Sin embargo, al implementar el sistema bajo las mismas condiciones pero esta vez utilizando el método Unconstrained, el algoritmo pierde la estabilidad y diverge, tal como se ve en la Fig. 5.17.



(a) *Valor absoluto del error.*



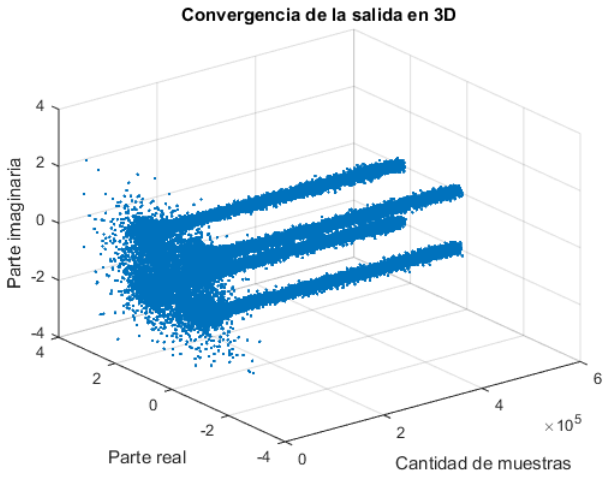
(b) *Constelación de salida.*



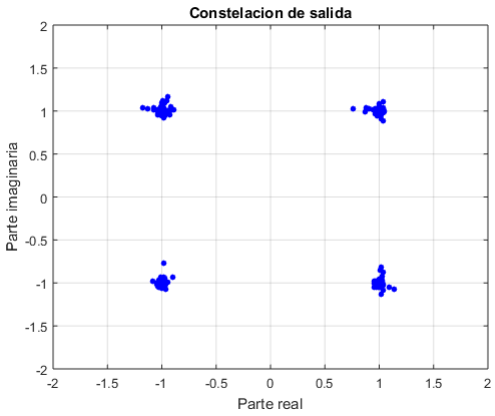
(c) *Salida del ecualizador.*

Figura 5.17: *Ecualización con* $L = M = 64$, $\mu = 0{,}0005$ *y un canal de coeficientes* $[0{,}2\ -0{,}7\ 1\ 0{,}4\ -1{,}0]$ *utilizando el método Unconstrained.*

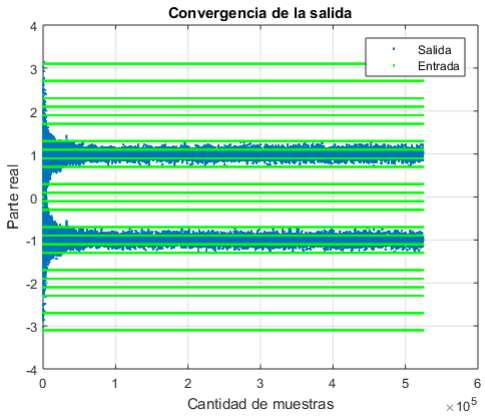
Si en cambio el canal fuese levemente diferente, intercambiando el valor de uno de los coeficientes por otro menor, obteniendo un canal de la forma $[0{,}2\ -0{,}5\ 1\ 0{,}4\ -1{,}0]$, se vuelve a conseguir la estabilidad de la salida del ecualizador utilizando los mismos parámetros que en el caso anterior. Este comportamiento puede verse en la Fig. 5.18.



(a) *Salida del ecualizador en 3 dimensiones.*



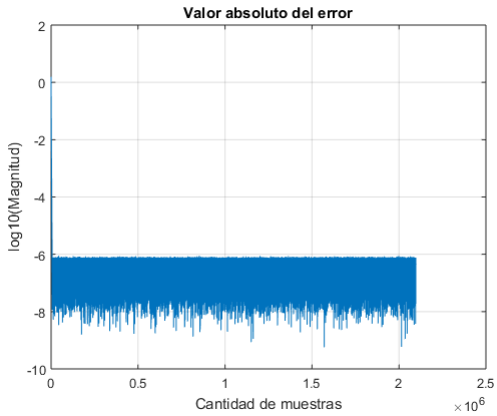
(b) *Constelación de salida.*



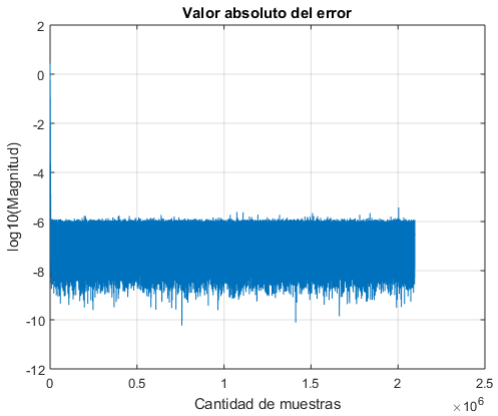
(c) *Salida del ecualizador.*

Figura 5.18: *Ecualización con* $L = M = 64$, $\mu = 0{,}0005$ *y un canal de coeficientes* $[0{,}2\ -0{,}5\ 1\ 0{,}4\ -1{,}0]$ *con el método Unconstrained.*

Se puede establecer nuevamente la misma comparación utilizando otros parámetros y modificando el canal para que quede en evidencia cómo se comporta el error en cada método. En la Fig. 5.19 se muestra la comparación de los errores para los parámetros $L = M = 32$, $\mu = 0{,}005$ y un canal de coeficientes $[0{,}2\ -0{,}5\ 1\ 0{,}4]$. Es posible notar que para el caso Unconstrained, el error en estado estable toma un valor un poco mayor al caso Constrained tal como ya se había explicado



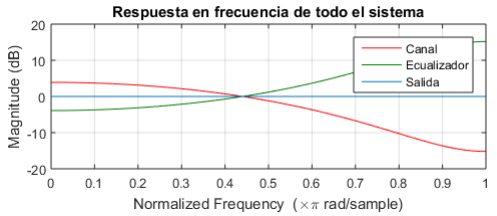
(a) *Error para el Método Constrained.*



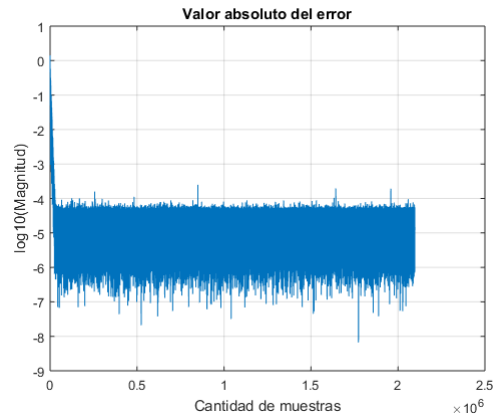
(b) *Error para el Método Unconstrained.*

Figura 5.19: *Comparación del error para los dos métodos planteados.*

Por otro lado, es posible comprobar que utilizando un mismo canal pero variando los parámetros del ecualizador se consiguen resultados diferentes con ambos métodos. Por ejemplo, trabajando con un canal de coeficientes [0,3482 0,8704 0,3482] y evaluando el comportamiento del sistema con $M = L = 32$ y $\mu = 0{,}005$ se puede ver cómo se obtienen buenos resultados con el método Constrained (Fig. 5.20), mientras que con el método Unconstrained el algoritmo no converge (Fig. 5.21).

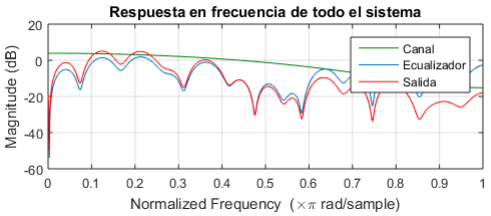


(a) *Respuesta en frecuencia.*

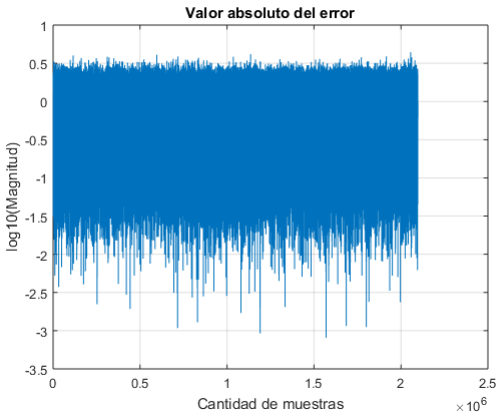


(b) *Valor absoluto del error.*

Figura 5.20: *Aplicación del método Constrained con* $L = M = 32$, $\mu = 0{,}005$ *y un canal con coeficientes* [0,3482 0,8704 0,3482].



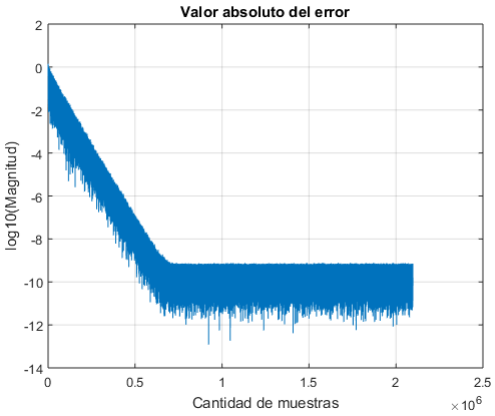
(a) *Respuesta en frecuencia.*



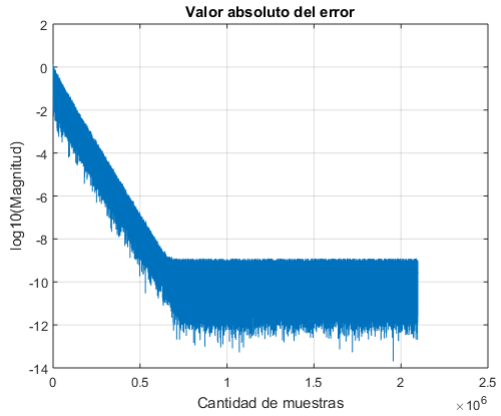
(b) *Valor absoluto del error.*

Figura 5.21: *Aplicación del método Unconstraint con* $L = M = 32$, $\mu = 0{,}005$ *y un canal con coeficientes* [0,3482 0,8704 0,3482].

Este comportamiento se debe al mayor nivel de error que proporciona el método Unconstrained. Esto puede apreciarse una vez más al trabajar con un paso $\mu = 0{,}0005$ (siendo necesario además aumentar la cantidad de taps a $L = M = 64$ para que el sistema se mantenga estable), ambos métodos convergen a los errores de la Fig. 5.22. De esta forma, se alcanza la estabilidad en decremento de la velocidad de convergencia.



(a) *Error para el Método Constrained.*



(b) *Error para el Método Unconstrained.*

Figura 5.22: *Comparación del error para los dos métodos planteados con* $L = M = 128$, $\mu = 0{,}0005$ *y un canal de coeficientes* [0,3482 0,8704 0,3482].

Capítulo 6

Diseño de la Arquitectura

Resumen

En este capítulo se comienza con la etapa de diseño, en la cual se plantea la arquitectura del proyecto y nuevamente se realizan simulaciones, pero esta vez de todo el sistema propuesto. Se llevan a cabo los estudios correspondientes tanto en punto flotante como en punto fijo, y se determinan los valores óptimos de todos los parámetros y resoluciones que finalmente se utilizan.

Además, se trazan curvas de BER para analizar el desempeño del sistema y la degradación del mismo al trabajar en punto fijo (con respecto a la curva en punto flotante).

6.1. Arquitectura del Proyecto

Se propone como diseño del sistema el de la Fig. 6.1 en la que se puede ver que los símbolos generados se dirigen hacia un transmisor, para luego pasar por el canal y finalmente llegar al ecualizador. A partir de allí solo se necesita que la señal ya ecualizada atraviese el detector para obtener la señal que ha sido transmitida.

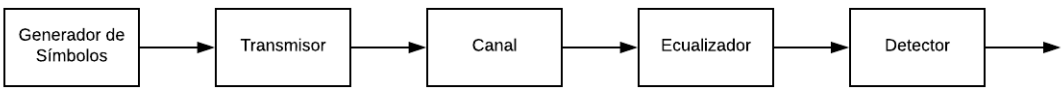


Figura 6.1: *Diagrama en bloque de la arquitectura del proyecto.*

6.2. Arquitectura Paralela

A la hora de llevar a cabo la implementación del sistema, uno de los problemas más importantes que se pueden presentar es el incumplimiento de las restricciones de tiempos especificadas, más conocidas como *Timing Constraints*, lo cual limita la máxima velocidad de trabajo del sistema desarrollado.

Los problemas de timing aparecen debido a que el tiempo de propagación de la señal a través de los bloques lógicos es mayor que el período de reloj. Este camino (*path*) se denomina camino crítico (*critical path*), y se define entre registros, entre un puerto de entrada y un registro o entre un registro y un puerto de salida. Por ejemplo, en el diagrama de la Fig. 6.2, se tiene una señal que debe pasar de un registro a otro a través de cierta lógica combinacional.

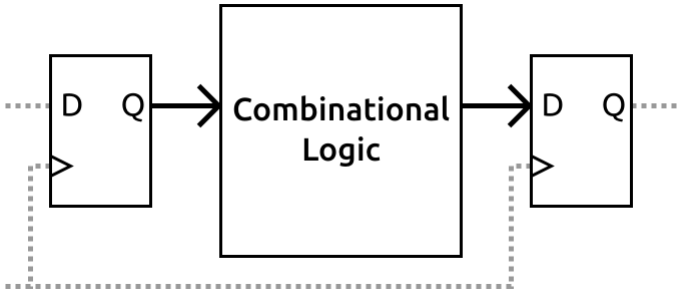


Figura 6.2: *Diagrama de bloques para una señal propagándose entre registros.*

El tiempo que demora la señal en propagarse desde el primer registro hasta el segundo, pasando por la lógica combinacional, debe ser menor al tiempo que tarda en llegar el próximo flanco de clock a los registros, es decir que debe ocurrir dentro de un mismo ciclo de clock. De lo contrario se estaría perdiendo información o teniendo información inválida. Esto es lo que principalmente limita la frecuencia máxima de trabajo del sistema, ya que al aumentar la frecuencia se reduce el ciclo de reloj y por lo tanto se reduce el margen en los tiempos de propagación de las señales. En la práctica, también se deben tener en cuenta numerosas consideraciones adicionales (ya contempladas por las herramientas utilizadas), como el camino recorrido por el clock de la fuente (*source clock path*), el camino recorrido por el clock del destino (*destination clock path*), incertidumbres del clock (*clock uncertainty*), etc.

En este caso, dado que la señal de entrada debe atravesar dos filtros antes de llegar al ecualizador (filtro transmisor y canal), lo cual conlleva un gran número de multiplicaciones en el dominio temporal, es probable que

las señales no cuenten con el tiempo suficiente para llegar a sus diferentes destinos. Esta situación se vuelve aún mas problemática si se considera una alta frecuencia de trabajo y un gran número de coeficientes en el filtro.

Una posible solución a este problema es la incorporación de registros intermedios entre los caminos críticos que generan los problemas de timing. Pero si el tiempo necesario para la propagación de una determinada señal es mucho mayor al disponible, esta solución se torna poco eficiente e incluso inviable. Otra alternativa sería disminuir la frecuencia de trabajo del sistema y, consecuentemente, también la tasa de transmisión de símbolos, lo cual en este caso no es una opción ya que no se pretenden cambiar las especificaciones de diseño.

En la práctica, lo que se suele hacer en estos casos es abandonar la estructura serial y pasar a una implementación en paralelo. Esto consiste en agregar múltiples instancias de los módulos que se desean paralelizar, a fin de que los mismos trabajen a una velocidad menor mientras mayor sea el nivel de paralelismo, pero sin afectar el baudrate a la salida. Así, la frecuencia del sistema en paralelo queda definida por la Ec. (6.1).

$$F_P = \frac{F_S}{P} \tag{6.1}$$

Siendo F_P la frecuencia en paralelo (a la que trabaja cada uno de los módulos individuales que se instancian), F_S la frecuencia en serie y P el nivel de paralelismo.

A pesar de que cada módulo otorga muestras válidas a su salida a una menor velocidad (F_P), al tener una cantidad P de módulos trabajando simultáneamente, se obtiene la misma cantidad de salidas en ese tiempo, que si el procesamiento se realizara en serie a una frecuencia mayor F_S. Es decir, que en cada ciclo de reloj se computan múltiples salidas en paralelo.

Para que este método funcione, es importante que se cumpla un procedimiento análogo al que se llevaría a cabo en forma serial, con lo cual la dependencia entre muestras de un mismo bloque paralelo debe ser igual a que si las mismas fueran procesadas una detrás de la otra. Esto conlleva a un nuevo diseño de los módulos que se desean paralelizar, lo cual se explica con mayor detalle en la Sección 7.3.

En este caso, interesa implementar la estructura paralela solamente en los módulos anteriores al ecualizador, ya que en principio, en el ecualizador en sí no sería necesario. Por un lado, porque el IP Core utilizado para el cálculo de las FFT/IFFT trabaja en forma serial, y por otro lado, porque el mismo trabaja en el dominio de la frecuencia, y como ya se ha explicado,

esto implica menor cantidad de multiplicaciones de las que serían necesarias en el dominio del tiempo y por lo tanto los problemas de timing disminuyen.

Se debe mencionar que los mayores beneficios de paralelizar el sistema se obtendrían si se contara con un IP Core que permitiese realizar una FFT/IFFT en la que un bloque de muestras ingresara en simultáneo a él, permitiendo así tasas mucho mayores. En el caso de este proyecto, podría alcanzarse un paralelismo de hasta 64, ingresando en cada ciclo de clock 64 muestras nuevas al ecualizador (128 en total debido al método *Overlap&Save*), y entregando a su salida 64 muestras válidas por ciclo de clock, logrando así una tasa de símbolos mucho mayor a la frecuencia de trabajo de cada módulo individual.

Para llevar a cabo la mencionada paralelización, se decide utilizar un paralelismo de 8 y se confecciona una nueva arquitectura que se presenta en la Fig. 6.3.

Debido a que el ecualizador es fraccionalemente espaciado y en el filtro transmisor se le aplica un *upsampling* a las muestras de entrada, se tiene el doble de paralelismo a la salida del filtro de coseno realzado y para el canal de transmisión. El resto del sistema sigue trabajando de forma serial, debido a las restricciones impuestas por el IP Core empleado para computar las FFT/IFFT.

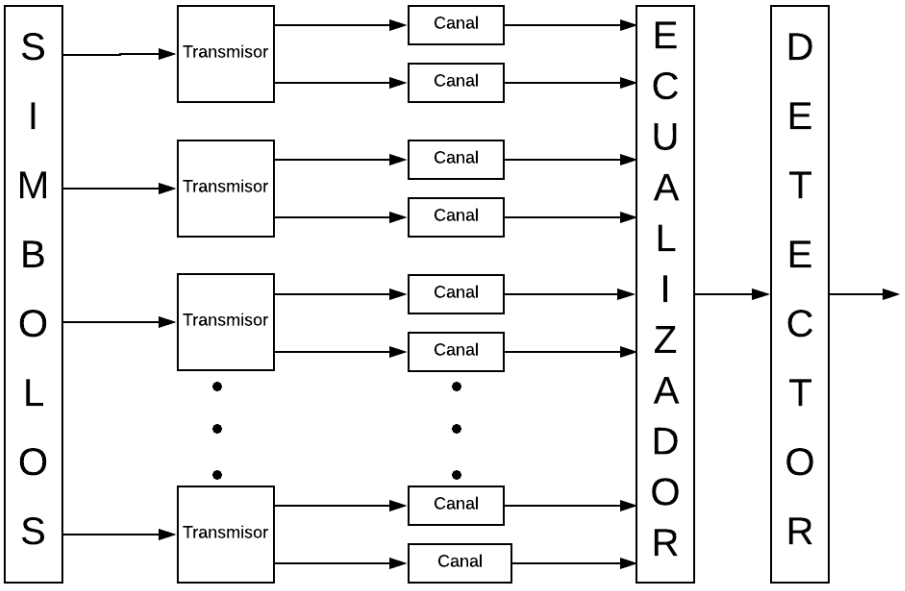


Figura 6.3: *Diagrama en bloques de la Arquitectura en Paralelo.*

A partir de este punto, el resto de las pruebas y análisis que deban realizarse, se hacen utilizando esta arquitectura, la cual se comporta de manera totalmente equivalente a su versión serial por lo que todos los

estudios realizados previamente continúan siendo válidos.

6.3. Punto Flotante

En la Sección 5.2, se simula el funcionamiento de un ecualizador adaptivo en el dominio de la frecuencia como un módulo aislado, para así comprender su comportamiento frente a la variación de sus diferentes parámetros. En cambio, en esta sección se le incorpora a dicho simulador el resto del sistema propuesto en la Fig. 6.3 utilizando la arquitectura en paralelo. Además, se trabaja con un Ecualizador Fraccionalmente Espaciado, lo que implica agregar, en este caso, un sobremuestreo de 2 (*oversampling = 2*) a la muestras de entrada y consecuentemente luego tomar intercaladamente las muestras de salida.

Este nuevo simulador se lleva a cabo en *Python* en punto flotante, es decir, trabajando con la máxima resolución posible de los datos con los que se opera, por lo que no existe pérdida de información debida a la forma en que se representan los diferentes valores. Posteriormente se ve que al momento de la implementación en FPGA no resulta práctico realizarlo de esta manera, siendo necesario tener otros parámetros en cuenta.

El objetivo de estas simulaciones es encontrar los parámetros óptimos para un funcionamiento correcto y eficiente del sistema. Esto permite tener mayores garantías a la hora de implementar el proyecto en FPGA, conociendo de antemano cómo es el comportamiento esperado del mismo.

Los parámetros de los bloques complementarios al ecualizador se definen con anterioridad y se encuentran detallados al final de esta sección, concentrando de esta forma el análisis en el diseño del ecualizador específicamente.

Se parte del requerimiento de que la cantidad de coeficientes del filtro ecualizador debe ser $M = 65$, para que el mismo sea capaz de compensar diversos canales de transmisión y debido a que es a partir de esa cantidad de coeficientes que las implementaciones en el dominio de la frecuencia comienzan a resultar más prácticas que aquellas en el dominio del tiempo, como se ve en la Fig. 4.6; y un solapamiento (*overlap*) del 50 %, resultando así $L = 64$. A partir de esto, se genera un algoritmo para encontrar el paso óptimo necesario para que los coeficientes del ecualizador converjan a un valor que proporcione la salida con el menor Error Cuadrático Medio.

Para ello, se evalúa el desempeño de una serie de pasos diferentes y se calcula para cada uno de ellos el error cuadrático medio de la salida respecto a los símbolos generados (que en este caso son conocidos), obteniendo

así la gráfica de la Fig. 6.4. Cabe mencionar que esta metodología para calcular el paso óptimo depende fuertemente de la cantidad de símbolos evaluados, por lo que el valor encontrado variará de acuerda al número de símbolos generados. A pesar de esto, proporciona una estimación lo suficientemente precisa y general, por lo que se utilizan directamente los resultados obtenidos.

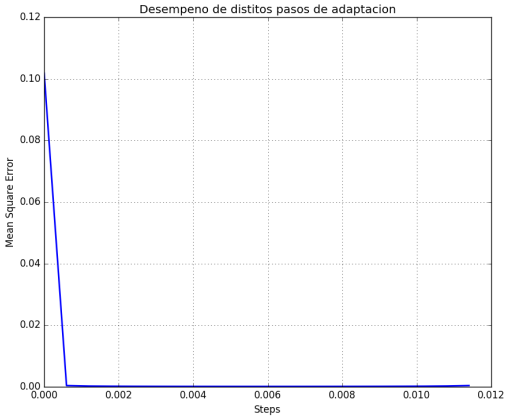


Figura 6.4: *Error Cuadrático Medio para diferentes pasos de adaptación.*

Analizando detenidamente la Fig. 6.4, se puede ver cómo inicialmente el error disminuye rápidamente al aumentar el paso, posteriormente continúa reduciéndose muy lentamente hasta alcanzar un valor mínimo y luego comienza a incrementarse. De esta evaluación surge que el paso óptimo es $\mu = 0{,}006$, el cual proporciona el menor error a la salida con una rápida velocidad de convergencia, pero tiene una magnitud que puede provocar que la salida del ecualizador se estabilice en un valor menos preciso y que incluso ante variaciones lo suficientemente bruscas del canal el sistema se desestabilice.

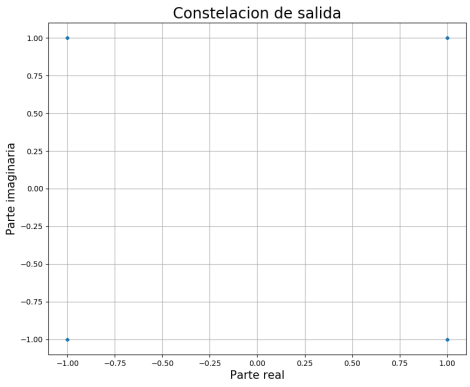
Por esta razón, en la implementación se decide recurrir al *Gear-Shift* (ya mencionado en la Sección 2.2.9), utilizando el paso óptimo para lograr la convergencia de los coeficientes rápidamente y disminuyendo luego su valor en un orden de magnitud para obtener una mayor precisión. Este cambio se efectúa cuando la salida ya se encuentra en estado estable, por lo tanto se emplea un valor de $\mu = 0{,}0006$ una vez que los coeficientes ya se han adaptado. De esta forma, se consigue una convergencia veloz y bajos niveles de error en estado estable.

Comportamiento del Sistema en Punto Flotante

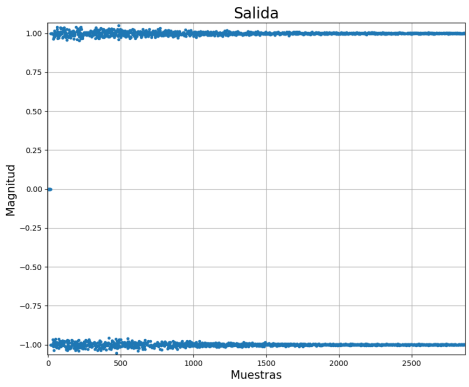
A continuación se muestran los resultados de las simulaciones del FSE con el paso óptimo elegido a fin de conocer su funcionamiento con una representación en punto flotante, que luego servirá como punto de referencia para encontrar la resolución óptima en punto fijo que proporcione el menor error posible en el sistema.

En primera instancia se verifica el sistema utilizando como canal de transmisión un impulso, y sin agregar ruido gaussiano. Esto constituye una primera prueba de funcionamiento, y en caso de obtener los resultados esperados se procede a realizar pruebas más exhaustivas con canales de transmisión y diferentes niveles de ruido. Los resultados obtenidos se pueden apreciar en la Fig. 6.5, en la que el desempeño es prácticamente perfecto, con un error despreciable y una constelación de salida puntual.

(a) *Valor absoluto del error.*



(b) *Constelación de salida.*



(c) *Salida del ecualizador.*

Figura 6.5: *Ecualización con* $L = 64$, $M = 65$ *y un impulso como canal de transmisión.*

Este es precisamente el comportamiento que se espera obtener considerando que un impulso no modifica la señal que ingresa al ecualizador, por

lo tanto los coeficientes no tienen la necesidad de compensar sus efectos y no se adaptan, ya que se inicializan como un impulso.

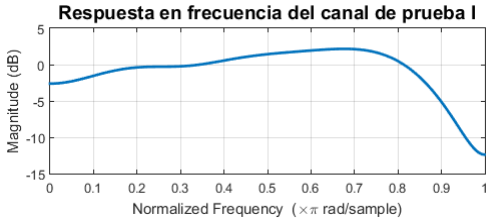
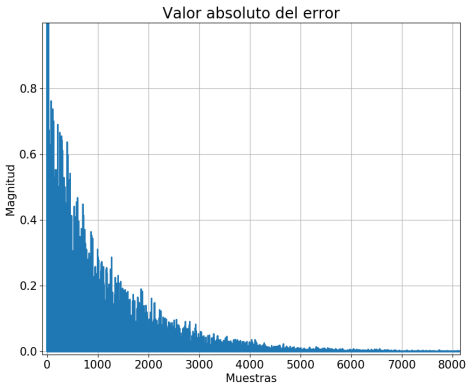
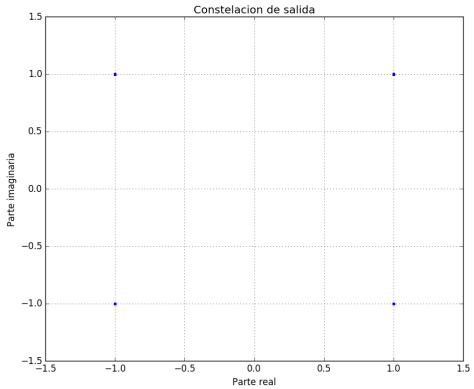


Figura 6.6: *Magnitud de la respuesta en frecuencia del canal de prueba I.*

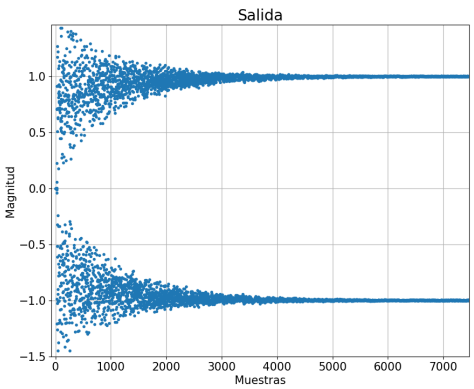
Una vez comprobado el correcto funcionamiento con un impulso unitario en el canal, se procede a realizar la misma evaluación, nuevamente sin ruido gaussiano, pero utilizando un canal de prueba de 11 coeficientes $[0{,}04, -0{,}05, 0{,}07, -0{,}21, -0{,}5, 0{,}72, 0{,}36, 0, 0{,}21, 0{,}03, 0{,}07]$ cuya respuesta en frecuencia se puede ver en la Fig. 6.6.



(a) *Valor absoluto del error.*



(b) *Constelación de salida.*

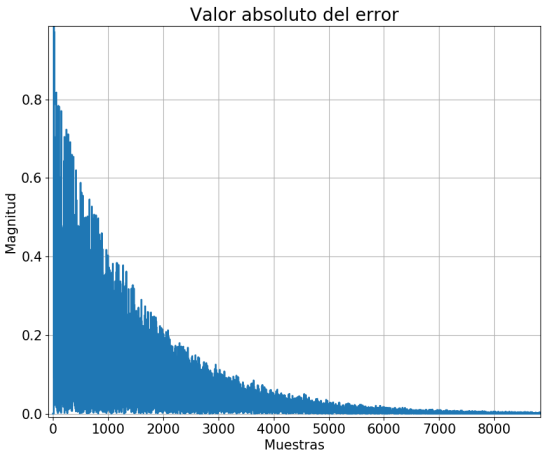


(c) *Salida del ecualizador.*

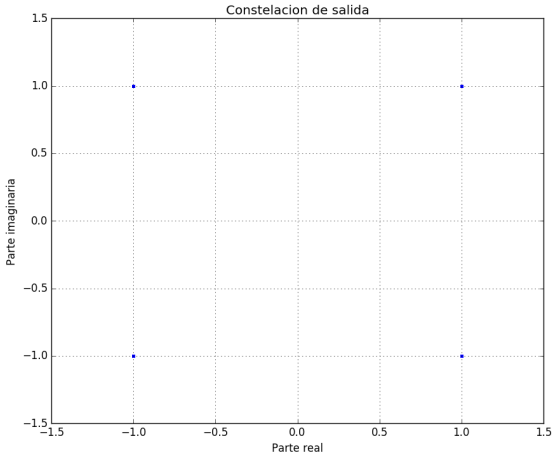
Figura 6.7: *Ecualización con $L = 64$ y $M = 65$ utilizando el canal de prueba.*

Como se puede observar en la Fig. 6.7, el funcionamiento sigue siendo correcto y la salida converge a los valores deseados.

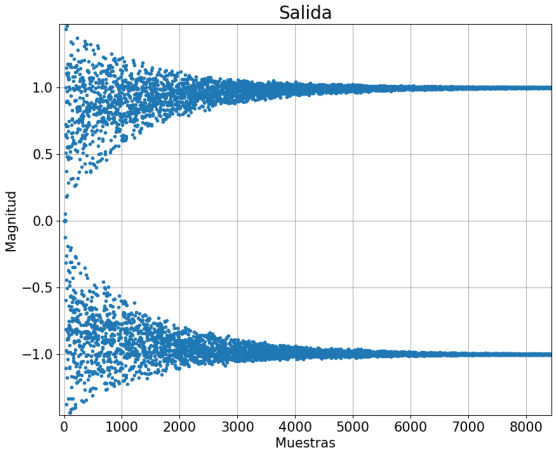
Para verificar que el funcionamiento es realmente el correcto, se contrastan estos resultados con los obtenidos en iguales condiciones con un simulador del ecualizador en el dominio del tiempo y con una arquitectura serial. Los mismos pueden verse en la Fig. 6.8.



(a) *Valor absoluto del error.*



(b) *Constelación de salida.*



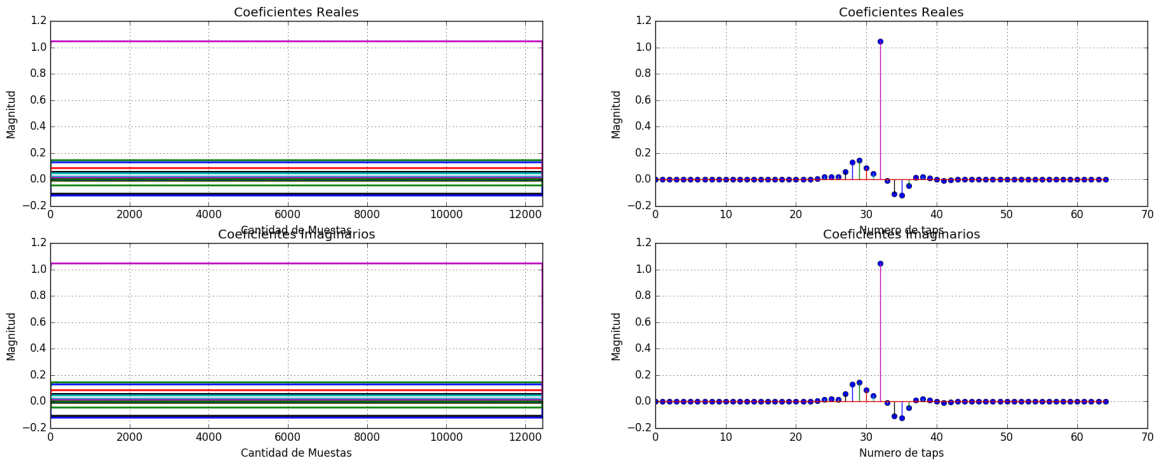
(c) *Salida del ecualizador.*

Figura 6.8: *Ecualización en el dominio del tiempo con* $L = 64$ *y* $M = 65$ *utilizando el canal de prueba.*

Es posible notar que el desempeño es prácticamente idéntico, habiendo sutiles diferencias en los tiempos de convergencia. En concordancia con los análisis teóricos anteriores, esto se debe por un lado a que en el dominio de la frecuencia el procesamiento se realiza con bloques de muestras y por lo tanto la adaptación es más lenta, y por otro lado a que en el simulador en frecuencia se utiliza el método *Unconstrained* que, como se explica en la Sección 4.3, degrada ligeramente el desempeño del ecualizador, lo cual no resulta significativo debido a que a los fines prácticos se puede ver que

el algoritmo converge y se estabiliza correctamente.

Retomando el análisis en el dominio de la frecuencia, se analiza también qué es lo que sucede con los coeficientes del ecualizador, mostrando su evolución en el tiempo en la Fig. 6.9a y el estado final de los mismos cuando ya se encuentran adaptados en la Fig. 6.9b. Es de suma importancia que los coeficientes se mantengan estables en el tiempo una vez alcanzada la convergencia ya que en este caso el canal no varía en el tiempo. Esta estabilidad se ve favorecida por el cambio de paso realizado, el cual minimiza las variaciones una vez que los coeficientes ya están adaptados.

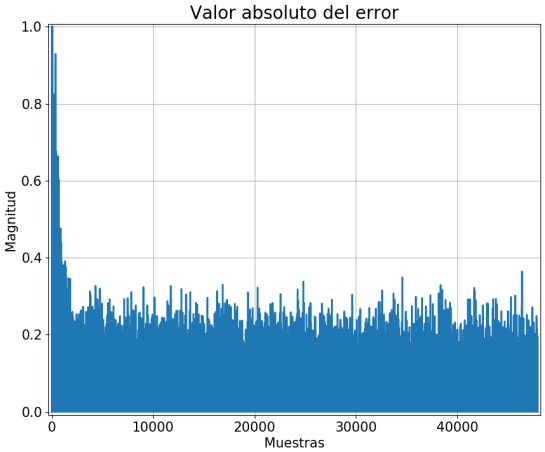


(a) *Evolución de los coeficientes en el tiempo.* (b) *Coeficientes adaptados.*

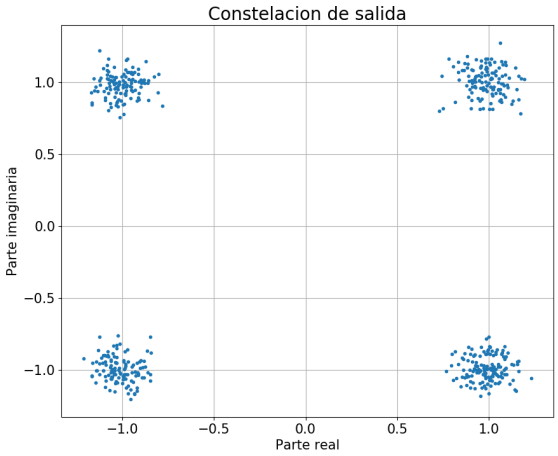
Figura 6.9: Comportamiento de los coeficientes del Ecualizador sin ruido.

Como última prueba se realiza la simulación del ecualizador agregando ruido gaussiano a la salida del canal de prueba. Se emplea una $SNR = 18dB$ para simular el ruido que se suma a la señal antes de ingresar al ecualizador.

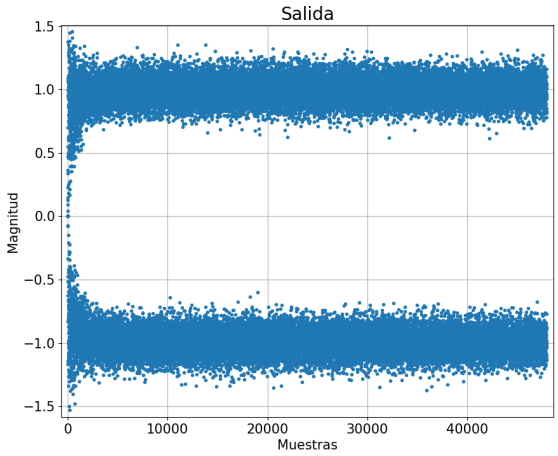
En la Fig. 6.10 se ve cómo en esta situación los datos de salida se encuentran mucho más dispersos en torno a los valores ± 1 y el error aumenta notablemente. Aún así la constelación sigue estando bien definida por lo que la detección de símbolos se hará correctamente en la gran mayoría de los casos. Esto puede verificarse agregando un contador de errores y graficando las curvas de BER para cada valor de SNR. Este análisis se realiza posteriormente en la Sección 6.5.



(a) *Valor absoluto del error.*



(b) *Constelación de salida.*

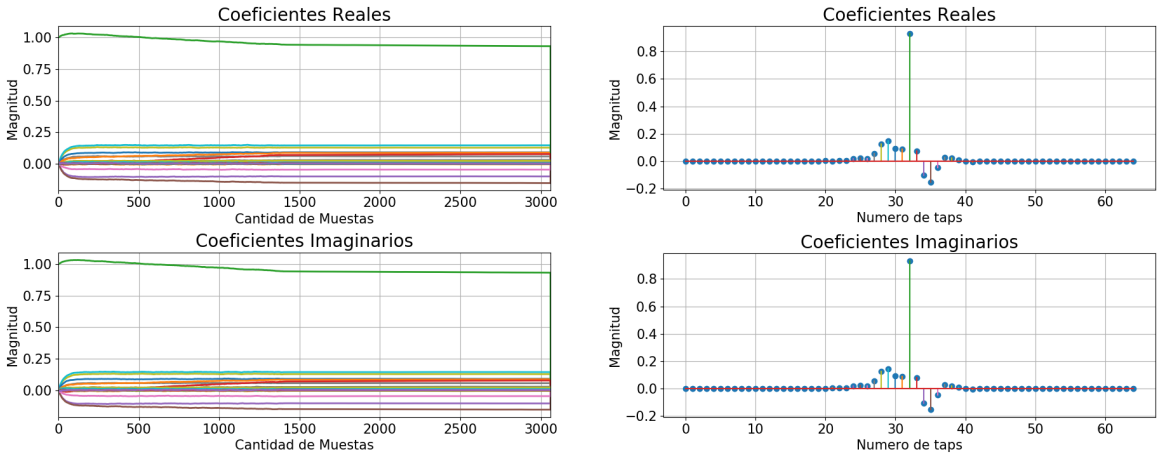


(c) *Salida del ecualizador.*

Figura 6.10: *Ecualización con* $L = 64$ *y* $M = 65$ *utilizando el canal de prueba con SNR = 18dB.*

En las Fig. 6.11a y 6.11b puede verse que en este caso los coeficientes también se mantienen estables a pesar de la presencia de ruido en el canal, aunque convergen a valores ligeramente diferentes debido justamente a esto.

También cabe destacar cómo al principio del gráfico en la Fig. 6.11a, los coeficientes tienen mayores variaciones. Esto se debe a que durante ese tiempo las muestras han sido procesadas con el paso $\mu = 0{,}006$, es decir, antes de realizar el Gear-Shift, por lo que los resultados son menos precisos. Luego de disminuir el valor del paso los coeficientes quedan más estables.



(a) *Evolución de los coeficientes en el tiempo.* (b) *Coeficientes adaptados.*

Figura 6.11: Comportamiento de los coeficientes del Ecualizador con ruido.

6.4. Punto Fijo

Como ya se ha mencionado, a nivel de implementación no es práctico trabajar con los datos representados en punto flotante y resulta más conveniente operar con representaciones en punto fijo.

Utilizar aritmética de punto fijo implica una significativa pérdida de precisión, pero a cambio se obtiene una considerable disminución en la complejidad del hardware requerido al momento de la implementación.

Es necesario hacer énfasis en que esto es simplemente una forma de interpretar los valores representados por una determinada secuencia de bits, por lo cual la cantidad de bits y la forma de interpretarlos durante su procesamiento dependerá del uso que se le dé y del dato con el que se trabaje. El hardware que lleva a cabo las operaciones interpreta todas sus entradas como si fueran valores puramente enteros, siendo el diseñador el que se encarga de interpretar estos supuestos enteros de una manera escalada, para que así puedan representar números con parte entera y parte fraccional.

Para dejar en claro la notación a utilizar y justificar las diferentes resoluciones utilizadas al llevar a cabo operaciones en punto fijo, en el Apéndice B se realiza una breve introducción a la representación de variables en punto fijo.

Una vez aclarado esto, es posible continuar con el desarrollo del sistema, pero ya trabajando en punto fijo. Evidentemente, el comportamiento no será el mismo debido a que sólo es posible representar una cierta can-

tidad de números y con una precisión particular, por lo tanto parte de la información se perderá durante este proceso de cuantización.

El objetivo es determinar la resolución óptima necesaria para que este error sea lo más pequeño posible y no impacte de manera significativa en el desempeño del sistema.

6.4.1. Resoluciones Óptimas

Lo primero que se debe hacer es encontrar la resolución óptima de todos los datos con los que se trabaja durante la implementación, no sólo del ecualizador en sí, sino también del resto de los módulos anteriores al mismo, ya que también es importante que los datos de entrada se encuentren representados de forma adecuada para ser trabajados luego en el ecualizador. Las variables que se deben cuantificar son las que se muestran a continuación.

- Coeficientes del Filtro Transmisor.
- Salida del Filtro Transmisor.
- Coeficientes del Canal de Transmisión.
- Salida del Canal de Transmisión.
- Pasos de Adaptación.
- Coeficientes del Ecualizador en el Dominio de la Frecuencia.
- Salida del Ecualizador en el Dominio del Tiempo y de la Frecuencia.
- Error en el Dominio del Tiempo y de la Frecuencia.

Las resoluciones de los módulos anteriores al ecualizador ya se encuentran definidas con anterioridad y no hacen al diseño del ecualizador en sí, por lo tanto no se muestran los resultados del análisis para ellas.

Para encontrar la manera más eficiente de representar los valores en cuestión, se decide utilizar el método de *Maximización de la SNR*. El mismo consiste en tomar como referencia el dato que se desea analizar en punto flotante, para compararlo con el mismo dato pero con una representación en punto fijo. Para ello se calcula la energía media de los mismos y se toma como 'señal' los valores en punto fijo, y como 'ruido' la diferencia entre lo obtenido en punto flotante y el resultado realizando la cuantización de los datos. Se calcula la relación entre ambos y se repite el procedimiento

para diferentes valores de *NB* y *NBF*, a fin de encontrar una combinación que maximice la SNR. En la Fig. 6.12 se muestra un ejemplo sencillo que explica esquemáticamente la forma en que se realiza este procedimiento. Cabe mencionar que en el caso del ecualizador el método cambia y se evalúa siempre la salida del ecualizador ya que cada dato depende de los anteriores y lo que interesa es el resultado obtenido a la salida.

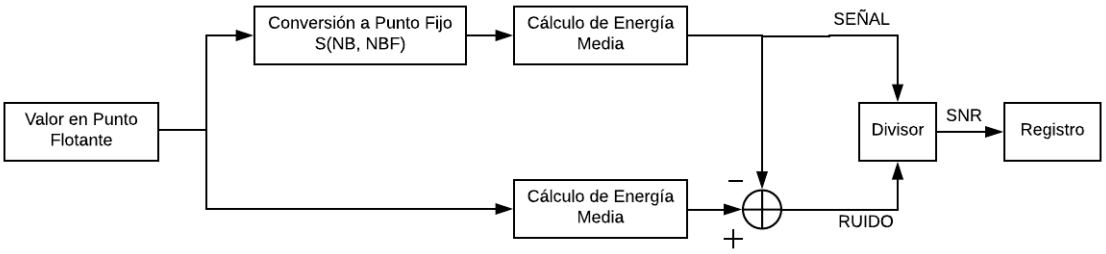


Figura 6.12: *Ejemplo de implementación del método de maximización de SNR.*

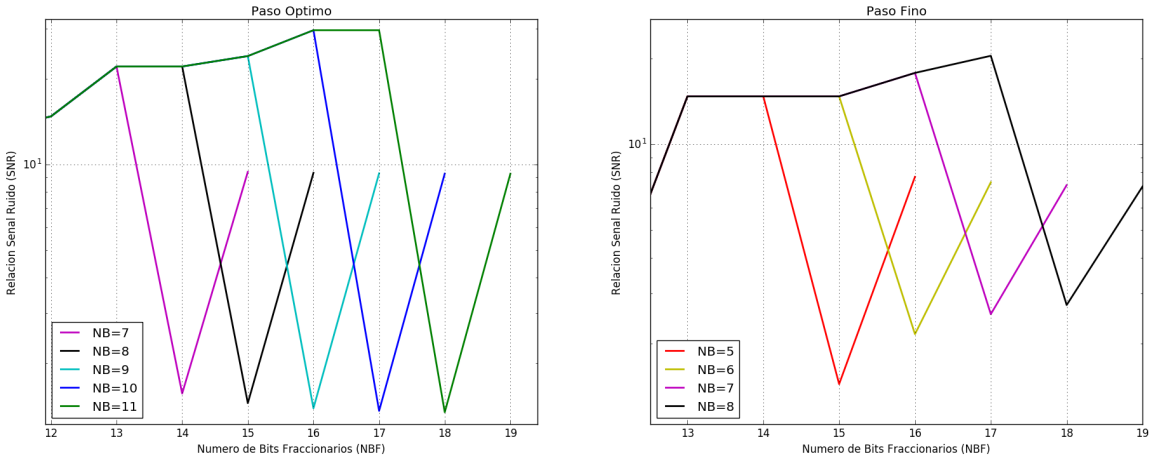
A continuación se muestran los gráficos obtenidos a partir de la aplicación de este método para cada variable que se busca cuantizar. En algunas oportunidades se decide utilizar una resolución diferente a la que el gráfico indica como la más eficiente. Esto se debe al funcionamiento del IP Core utilizado para el cálculo de las FFT/IFFT, para el cual las entradas y salidas (en tiempo y frecuencia, o viceversa) deben cumplir ciertas condiciones en relación a su parte entera y fraccionaria. Todo esto se explica con mayor detalle en la Sección 7.5.1, pero por el momento sólo es importante demostrar que las resoluciones elegidas otorgan un valor de SNR razonablemente alto y que los resultados obtenidos con las mismas son similares a sus análogos en punto flotante.

Paso de Adaptación

En este caso es necesario definir la resolución de los dos pasos de adaptación elegidos para realizar la adaptación de coeficientes. Por un lado el paso óptimo (Fig. 6.13a), y por el otro el paso fino que se utiliza luego del Gear-Shift (Fig. 6.13b).

Ya que los dos pasos se emplean indistintamente en el algoritmo de adaptación, para realizar las operaciones implicadas sería conveniente que ambos tengan la misma resolución. De esta forma la alineación de la parte fraccionaria y entera se hace siempre de la misma forma, evitando la necesidad de hacer cambios de resoluciones en los resultados. Por esta razón se decide representar tanto al paso óptimo como al paso fino con una

resolución $\mathbf{S(8, 13)}$.

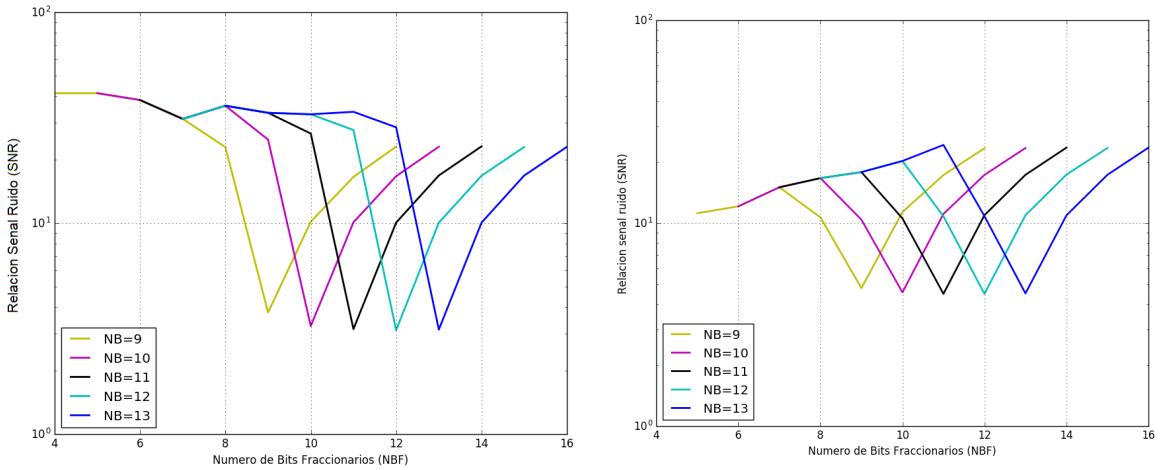


(a) *Paso óptimo.* (b) *Paso fino.*

Figura 6.13: *Resoluciones para los pasos de adaptación.*

Coeficientes del Filtro

Para encontrar la resolución óptima de los coeficientes del filtro en el dominio de la frecuencia, se realizan dos análisis distintos, uno con el impulso (Fig. 6.14a) y otro utilizando el canal de prueba (Fig. 6.14b), ya que interesa que los mismos tengan un buen desempeño en ambos casos.



(a) *Con un impulso.* (b) *Con el canal de prueba.*

Figura 6.14: *Resoluciones para los coeficientes del ecualizador.*

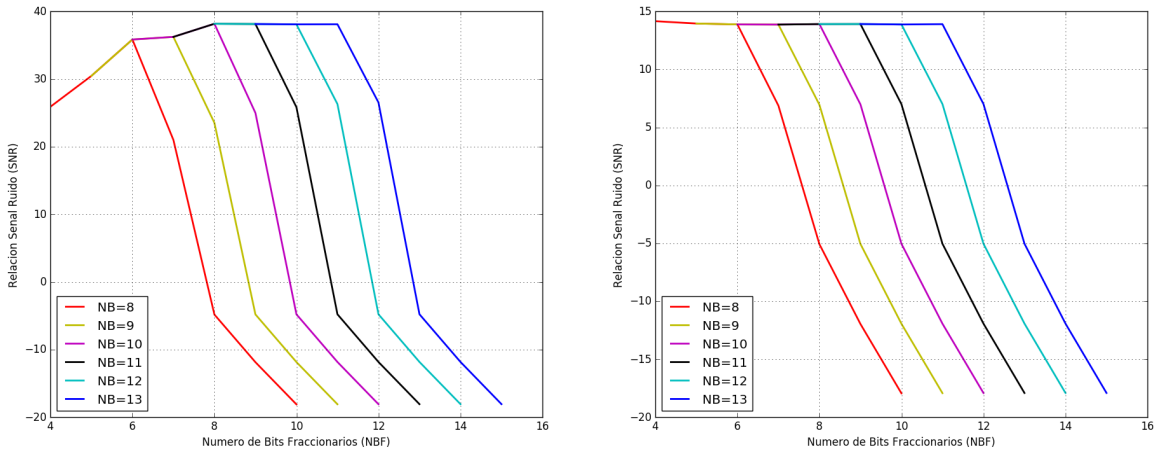
Observando las imágenes se define la resolución $\mathbf{S(12, 10)}$ como la óptima considerando estos dos casos. Cabe destacar que el valor de los coeficientes juega un papel muy importante en la adaptación del sistema y

por ello se decide utilizar redondeo en lugar de truncado para los mismos, logrando así una mejora en el desempeño del ecualizador.

Salida del Ecualizador

Para la salida del ecualizador se realiza un procedimiento similar, analizando los dos casos anteriores. Esta vez es necesario obtener las resoluciones para la salida tanto en el dominio del tiempo como en el dominio de la frecuencia. Para ambos casos se utiliza truncado en la representación de punto fijo.

En el dominio del tiempo el examen para diferentes resoluciones da como resultado lo que se observa en la Fig. 6.15.

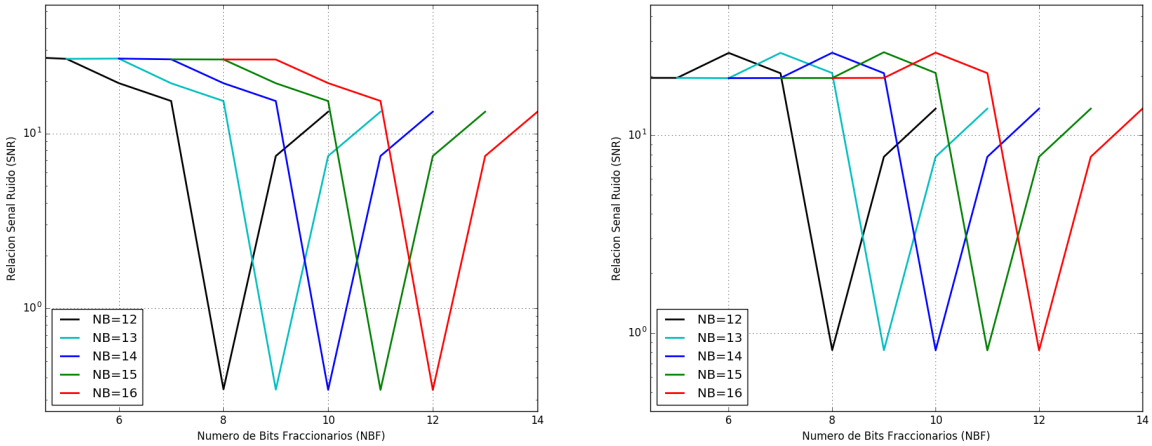


(a) *Con un impulso.* (b) *Con el canal de prueba.*

Figura 6.15: *Resoluciones para la salida del ecualizador en el dominio del tiempo.*

Dado que en este caso se debe utilizar una IFFT para obtener la salida en el tiempo a partir de la misma en frecuencia, es necesario tener en cuenta no sólo lo que se aprecia en el gráfico sino también lo ya mencionado acerca de las condiciones de funcionamiento del IP Core. Por ello, luego de realizar algunas pruebas de funcionamiento se decide utilizar la resolución $\mathbf{S(12,8)}$.

Por otro lado, en la Fig. 6.16 se muestra el estudio realizado para el caso de la salida en el dominio de la frecuencia. Analizando el gráfico y por la misma razón que en el caso anterior, se define la resolución $\mathbf{S(15,8)}$ como la óptima para representar esta variable.

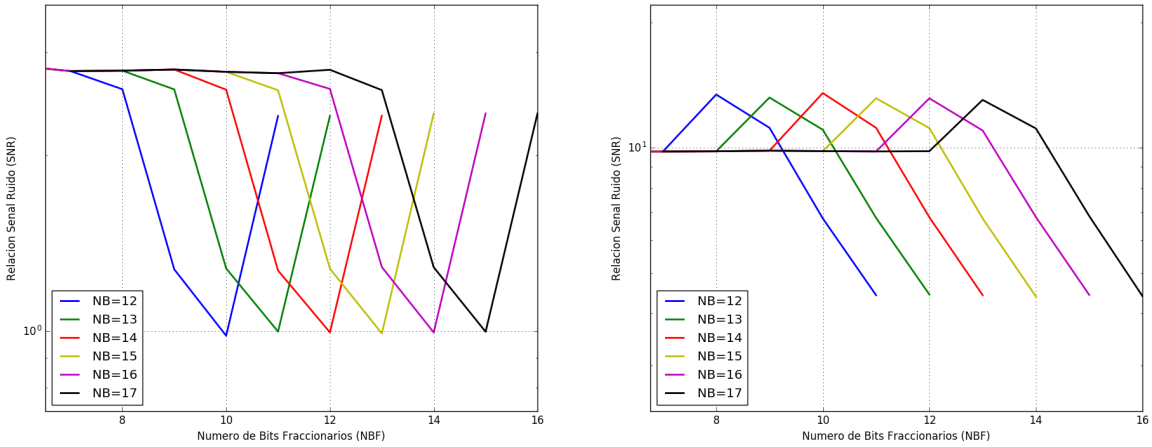


(a) *Con un impulso.* (b) *Con el canal de prueba.*

Figura 6.16: *Resoluciones para la salida del ecualizador en el dominio de la frecuencia.*

Error a la Salida del Ecualizador

Para el error también es necesario realizar el análisis en el dominio del tiempo y de la frecuencia. Sin embargo, para el primer caso, dado que el error se calcula como una resta entre la salida del ecualizador y el símbolo detectado a partir de la misma, se puede trabajar con la máxima resolución, ya que esta consiste en agregar un bit entero a la ya definida en el caso anterior. Por lo tanto la resolución para representar al error en el dominio del tiempo será $\mathbf{S(13, 8)}$.



(a) *Con un impulso.* (b) *Con el canal de prueba I.*

Figura 6.17: *Resoluciones para el error en el dominio de la frecuencia.*

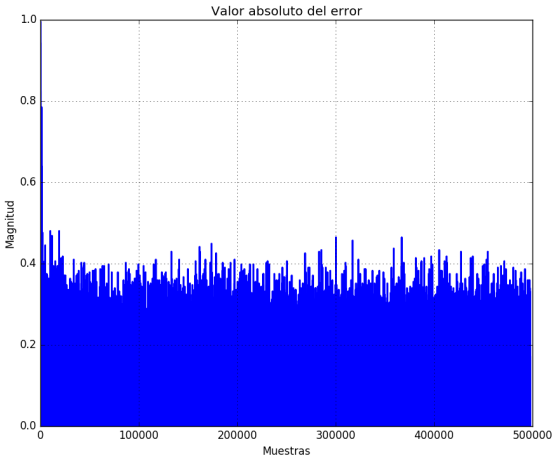
En el segundo caso, al igual que en los anteriores, se realiza el análisis de resoluciones utilizando el método de maximización de SNR, arrojando

los resultados que se ven en la Fig. 6.17.

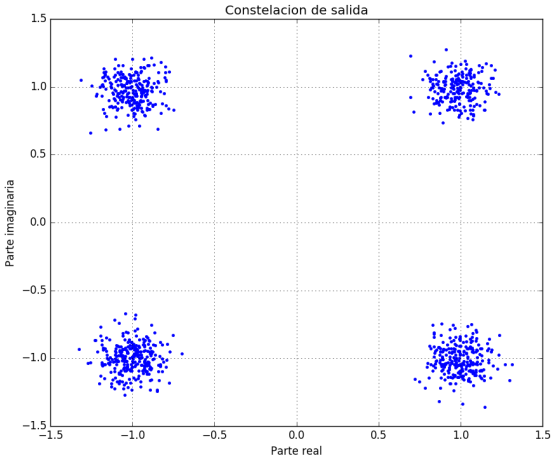
Dadas las resoluciones ya establecidas para la salida del ecualizador y por lo tanto las del error en el tiempo, por las restricciones del IP Core, se decide tomar en este caso la resolución $\mathbf{S(17,8)}$, a pesar de que no represente la más optima para representar este dato.

Comportamiento del Sistema en Punto Fijo

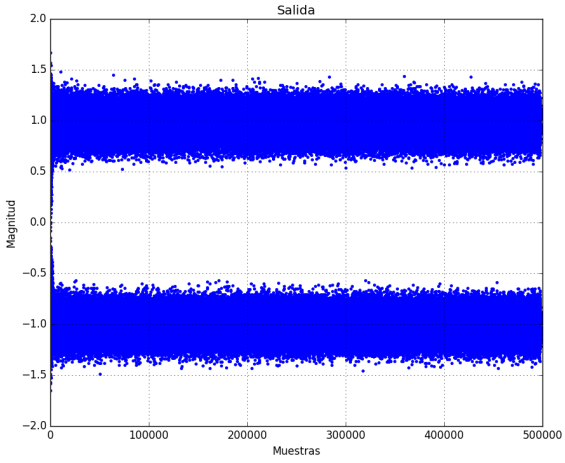
Llegado este punto, sólo queda comprobar que las resoluciones obtenidas en los pasos anteriores son adecuadas para el correcto funcionamiento del sistema.



(a) *Valor absoluto del error.*



(b) *Constelación de salida.*



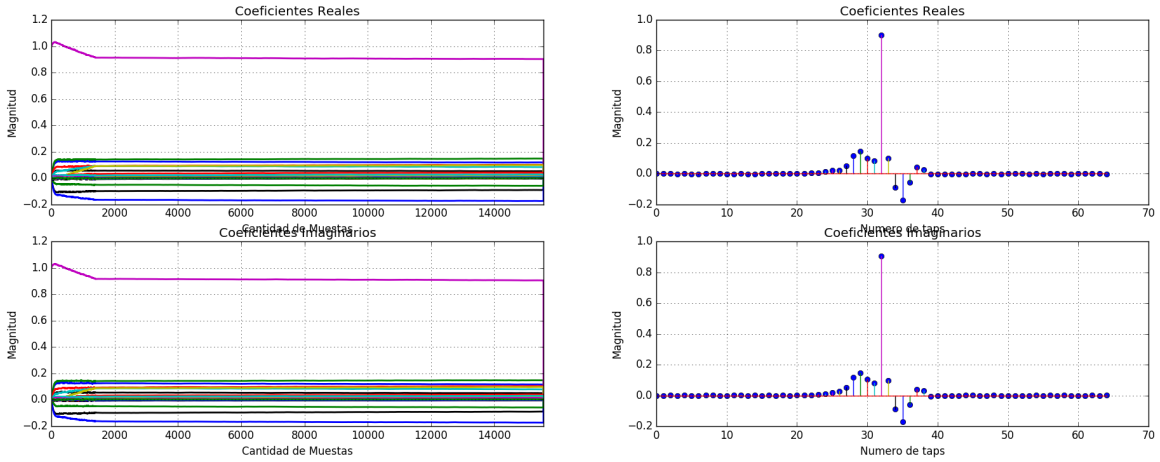
(c) *Salida del ecualizador.*

Figura 6.18: *Ecualización en punto fijo con* $L = 64$ *y* $M = 65$ *utilizando el canal de prueba con SNR = 18dB.*

En la Fig. 6.18 se puede observar el comportamiento del sistema en punto fijo, con 18 dB de ruido gaussiano y utilizando el canal de prueba como

medio de transmisión. Comparándolo con la Fig. 6.10 en la que se muestra el funcionamiento bajo las mismas condiciones pero con una representación en punto flotante, se puede decir que los mismos son lo suficientemente similares y que el error no es significativo a los fines prácticos.

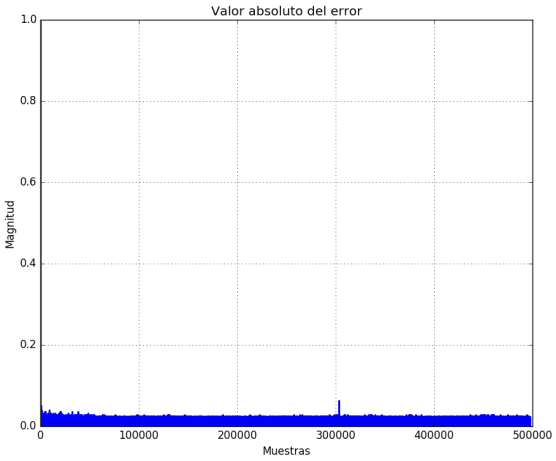
Por otro lado también se puede ver en la Fig. 6.19 el comportamiento de los coeficientes del ecualizador utilizando las resoluciones definidas a lo largo de esta sección. En esta imagen se puede ver que los mismos se mantienen estables en el tiempo sin grandes variaciones y por lo tanto se puede concluir que los resultados obtenidos del análisis de resoluciones son los correctos y se procede a la implementación de los mismos en RTL.



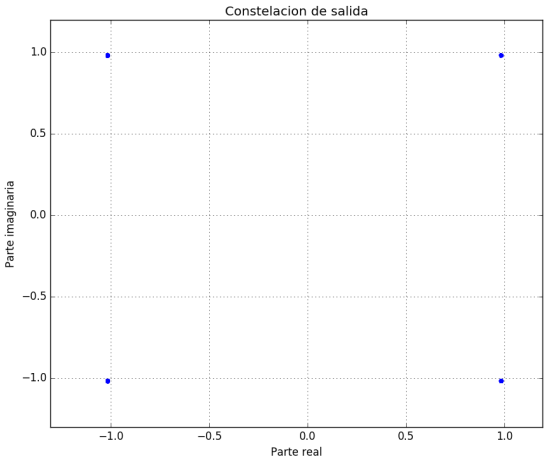
(a) *Evolución de los coeficientes en el tiempo.* (b) *Coeficientes adaptados.*

Figura 6.19: *Coeficientes en punto fijo con* $L = 64$ *y* $M = 65$ *utilizando el canal de prueba con SNR = 18dB.*

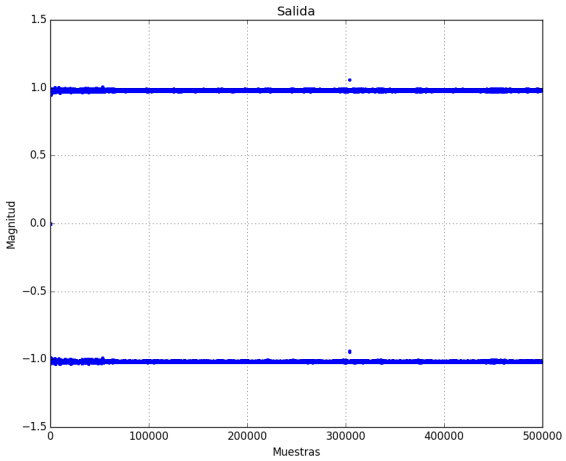
A modo de completar la información sobre el funcionamiento del simulador en punto fijo y poder contrastarlo con los resultados obtenidos en la implementación en FPGA, en la Fig. 6.20 se puede ver la salida y el error del ecualizador cuando el canal de transmisión es un impulso y no hay presencia de ruido gaussiano.



(a) *Valor absoluto del error.*



(b) *Constelación de salida.*



(c) *Salida del ecualizador.*

Figura 6.20: *Ecualización en punto fijo con* $L = 64$ *y* $M = 65$ *con un impulso como canal de transmisión.*

Nuevamente, esta figura puede compararse con su análoga en punto flotante (Fig. 6.5), notando una vez más que existe una gran similitud entre las mismas y dando por finalizado el análisis en punto fijo.

6.5. Curvas de BER

La forma más efectiva de comprobar que el funcionamiento del sistema tanto en punto fijo como en punto flotante es correcto, es realizar la comparación de sus curvas de BER. Como primera instancia se deben calibrar las curvas para verificar que los niveles de ruido incorporados sean los correctos. Para ello se contrastan los resultados obtenidos en las simulaciones con la curva teórica que fue presentada en la Sección 2.1.5, y la cual para

el caso de una modulación DQPSK se encuentra descrita por la Ec. 6.2, presentada en [17]. Con esta ecuación es posible calcular cada uno de los puntos de la curva para diferentes valores de SNR.

$$BER_{DQPSK} = erfc\left(\sqrt{\frac{1}{2} * 10^{\frac{SNR}{10}}}\right) - \frac{1}{4} * erfc\left(\sqrt{\frac{1}{2} * 10^{\frac{SNR}{10}}}\right)^2 \quad (6.2)$$

A partir de las simulaciones y de realizar el cociente entre la cantidad de errores y la cantidad de bits contados se obtienen las curvas simuladas, tanto para punto fijo como para punto flotante. Esta simulación, para ser comparable con la teórica debe realizarse sin canal de transmisión y sin ecualizador. La comparación de los tres casos se puede ver en la Fig. 6.21 donde se aprecia que las 3 curvas coinciden casi perfectamente, con una ligera desviación para el caso en punto fijo, debido precisamente al error introducido por la cuantización de los datos, que comienza a tener mayor relevancia a medida que aumenta la SNR.

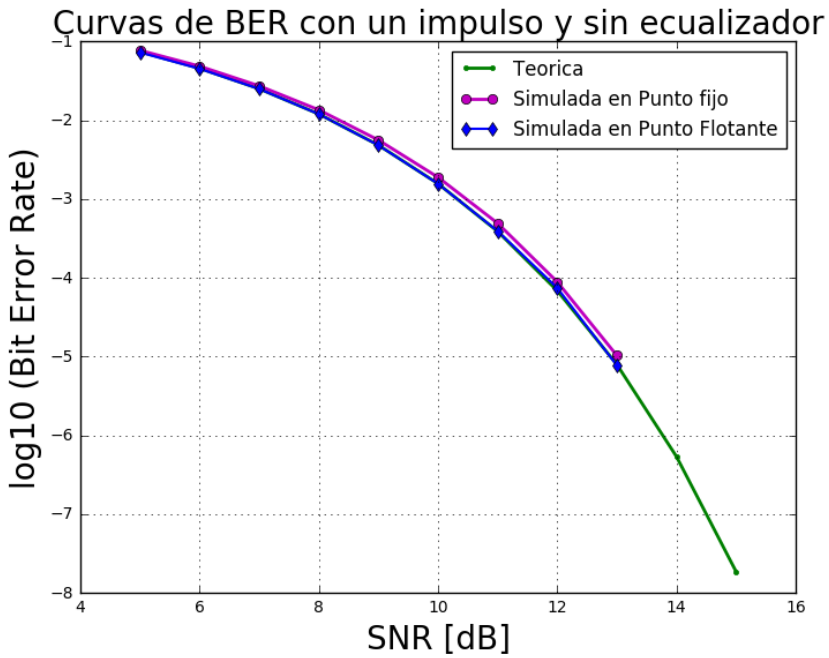
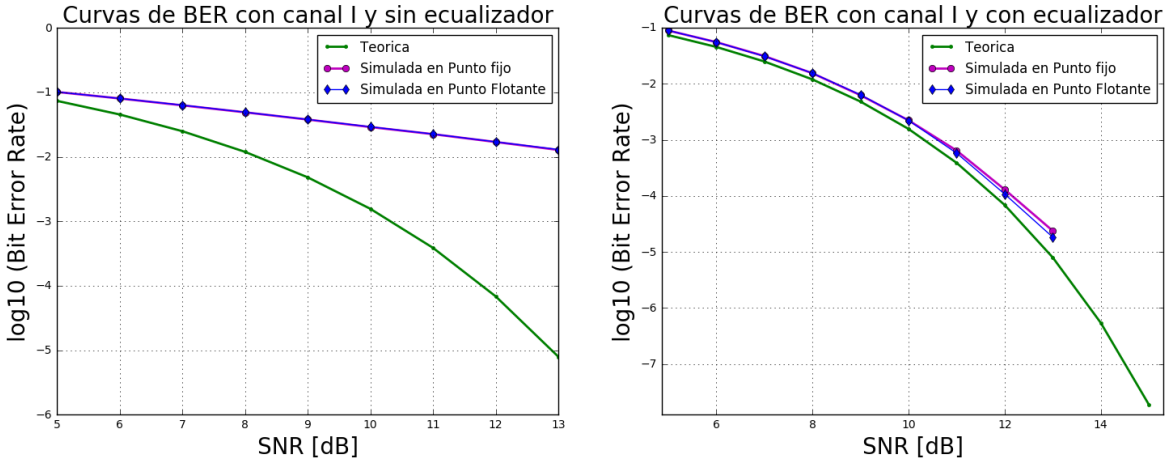


Figura 6.21: *Curvas de BER para el caso sin ecualizador y sin canal de transmisión*

De esta forma se comprueba no sólo que los niveles de SNR son los correctos sino también que los módulos complementarios al ecualizador se encuentran bien diseñados y con las resoluciones adecuadas.

A partir de este momento, todos los datos de Bit Error Rate que se grafiquen se comparan con la curva teórica que representa el menor error posible.

Si se agrega al sistema el canal de prueba I, se puede ver en la Fig. 6.22a que la cantidad de errores a la salida del mismo aumenta considerablemente, y por lo tanto la curva simulada se ubica muy por encima de la teórica, desviándose más mientras mayor sea la SNR.



(a) *Sin ecualizador.* (b) *Con ecualizador.*

Figura 6.22: *Curvas de BER simuladas con canal de prueba I.*

Para mejorar el desempeño del sistema, se agrega al simulador el ecualizador y se toman nuevamente los datos para diferentes niveles de ruido. En la Fig. 6.22b se muestran los resultados para las simulaciones de punto fijo y punto flotante utilizando el ecualizador y con el canal de prueba I. Ahora el desempeño mejora sustancialmente, haciendo que la curva tienda a alinearse con la teórica, pero siempre manteniéndose por encima de ella, como es de esperar ya que la misma representa el error cuando no existe canal de transmisión. Con este estudio es posible concluir que el ecualizador funciona correctamente y que las resoluciones elegidas representan adecuadamente a cada uno de los datos, ya que la diferencia entre la curva en punto flotante y en punto fijo es prácticamente despreciable.

6.6. Especificaciones

En base a los análisis hechos tanto en punto flotante como en punto fijo, quedan determinadas las especificaciones del sistema a implementar.

El ecualizador que se desarrolla se integra como parte de un sistema de comunicaciones que cuenta con otros módulos complementarios, detallados en la Sección 7.3, por lo que es necesario plantear no sólo las especificaciones del ecualizador propiamente dicho, sino también las del resto del sistema.

6.6.1. Especificaciones Globales

En la Tabla 6.1, se muestran las especificaciones que afectan tanto a los módulos complementarios al ecualizador, como a todo el sistema en general.

Cabe mencionar, respecto a la frecuencia de trabajo utilizada, que finalmente la tasa de datos a la salida del sistema será de $48Mbps$ equivalentes a $24Mbaudios$. Por otro lado, debido al upsampling, a la salida del transmisor y del canal se tendrán 16 datos nuevos por cada ciclo de clock, a pesar de que al comenzar la transmisión en la PRBS el paralelismo sea igual a 8.

Modulación	DQPSK	
Paralelismo	8	
PRBS	9	
Transmisor	Coseno Realzado	roll-off = 0.5
	Entrada	U(1,0)
	Oversampling	2
	Número de Baudios	8
	16 Coeficientes	S(8,7)
	Salida	S(8,6)
Canal	Entrada	S(8,6)
	11 Coeficientes (configurables)	S(8,7)
	Ruido (SNR configurable)	Gaussiano
	Salida	S(9,6)
Frecuencia de Trabajo	MicroBlaze - Register File	96MHz
	PRBS - Transmisor - Canal	3MHz
	Ecualizador	96MHz
	Slicer - BER	24MHz

Tabla 6.1: *Especificaciones Globales.*

6.6.2. Especificaciones Ecualizador

Respecto al ecualizador en sí, en la Tabla 6.2 se pueden ver todos los datos relacionados al mismo y que se utilizan en la implementación del proyecto.

Fraccionalmente Espaciado	OS = 2	
Paso de Adaptación	0.006 - 0.0006	S(8,13)
64 Muestras de Entrada	S(9,6)	
65 Coeficientes	S(12,10)	
32 Muestras de Salida	Tiempo	S(12,8)
	Frecuencia	S(15,8)
Error	Tiempo	S(13,8)
	Frecuencia	S(17,8)
FFT/IFFT	128 puntos	

Tabla 6.2: *Especificaciones del Ecualizador.*

Capítulo 7

Implementación en FPGA

Resumen

En este capítulo se detalla la implementación en FPGA de la arquitectura propuesta. Se precisan las herramientas utilizadas, se muestra cómo se desarrollan los diferentes módulos adicionales que requiere el sistema y cómo se confecciona la plataforma de verificación que permite comunicarse con la FPGA para configurar el sistema y extraer datos del mismo. Además, se explica detalladamente cómo se implementa el módulo ecualizador propiamente dicho.

7.1. Implementación de la Arquitectura

En este trabajo se propone implementar en FPGA los bloques presentados en la Fig. 7.1, en la que se puede apreciar la totalidad del sistema, desde la generación de símbolos mediante el módulo *PRBS* hasta el ecualizador adaptivo propiamente dicho, pasando por el filtro transmisor (*TX rcosine*) y el canal (*FIR* y *GNG*). Se cuenta además con un *contador de BER*, mediante el cual es posible determinar el desempeño del sistema.

En la parte inferior de la Fig. 7.1 también es posible apreciar la plataforma de verificación utilizada tanto para configurar el sistema como para extraer datos en diferentes puntos del mismo.

En secciones anteriores se explica desde un punto de vista teórico el funcionamiento de los bloques de la Fig. 7.1, tanto de los módulos complementarios como del ecualizador adaptivo en el dominio de la frecuencia. En este capítulo se explica detalladamente la implementación de cada uno de estos módulos, haciendo énfasis en la implementación del ecualizador.

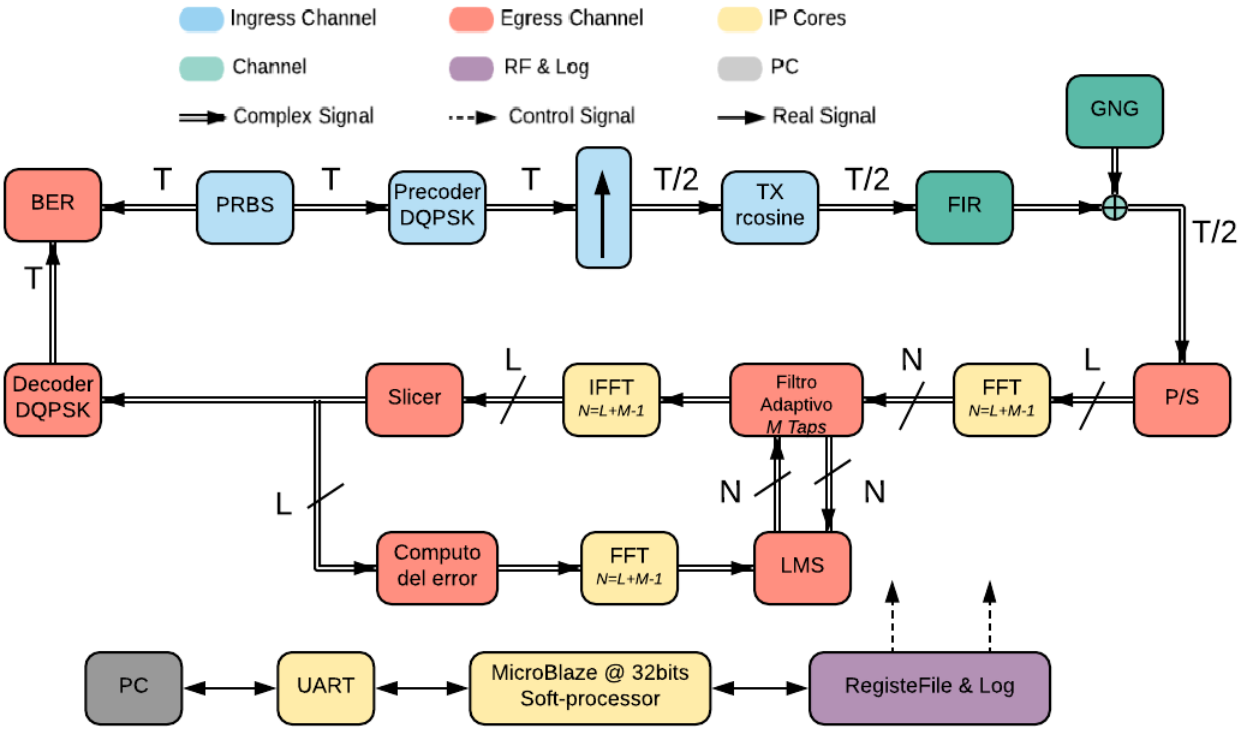


Figura 7.1: *Diagrama en bloques de la arquitectura general del proyecto.*

7.2. Herramientas y Hardware utilizado

Para la descripción en RTL de los módulos a implementar y las simulaciones de su comportamiento funcional, como así también para llevar a cabo la síntesis, implementación y generación de bitstream, se utiliza como entorno de trabajo las herramientas brindadas por *Xilinx*, ya que el dispositivo sobre el cual se trabaja es una FPGA comercializada por dicha empresa.

Concretamente, se emplea *Vivado 2013.4* junto con su Kit de Desarrollo de Software *Xilinx SDK 2016.1* (para la programación del microprocesador embebido). Se utiliza una versión antigua de Vivado debido a que la FPGA que se emplea requiere de la compra de una licencia, la cual ya había sido adquirida en el pasado pero solamente hasta la versión de Vivado utilizada. La versión del SDK sí puede ser una versión más moderna, ya que no se requieren licencias adicionales al utilizar la plataforma y el binario generados por Vivado.

Para las simulaciones del comportamiento en punto flotante y en punto fijo del sistema, se emplea Python 2.7 (no se utiliza Python 3 debido a incompatibilidades con la clase de Fixed Point utilizada) y MATLAB en su versión R2015A. Por otra parte, para la programación del microprocesador

se utiliza el lenguaje C.

Con respecto al hardware utilizado, se utiliza el kit de evaluación de Xilinx *KC705*, el cual cuenta con una *Kintex-7*, además de otros componentes que facilitan su utilización, entre los cuales se encuentran numerosos conectores, memorias, un display LCD, pulsadores, switches y LEDs. Este kit puede observarse en la Fig. 7.2.

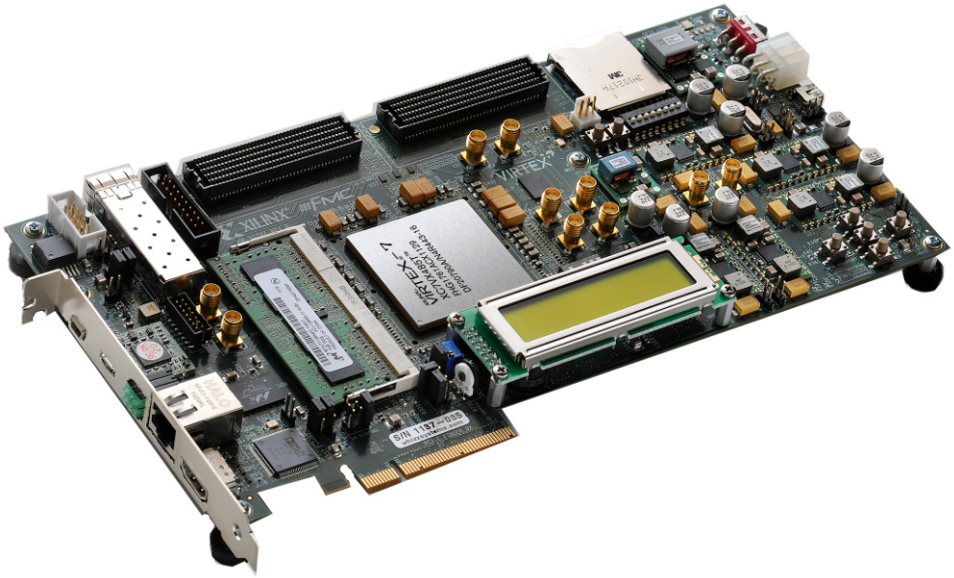


Figura 7.2: *Kit de evaluación Xilinx Kintex-7 FPGA KC705.*

7.2.1. MyHDL

MyHDL es un paquete libre y de código abierto que permite utilizar Python como un lenguaje de descripción y verificación de hardware, aprovechando todas la potencia y las facilidades de un lenguaje de alto nivel como Python para simplificar el flujo de desarrollo.

La idea básica de MyHDL es utilizar los *generators* de Python para modelar la concurrencia del hardware, comportándose los *generators* de manera similar a un bloque *always* en Verilog. Con esta idea, los módulos de hardware son modelados como funciones que retornan *generators*, permitiendo así tener numerosas características análogas a los módulos de Verilog (o VHDL). MyHDL provee además diferentes clases para modelar señales que se comunican entre *generators* o para soportar operaciones orientadas a nivel de bit.

Una de las principales características de MyHDL, y la que mayormente se utiliza en este proyecto, es la capacidad de *Cosimulación* con Verilog, en la que se verifica el diseño RTL a través de Python, pudiendo aprovechar

toda su capacidad de *Unit Testing* para hacer un análisis más exhaustivo con mayor facilidad.

Esto permite confeccionar todo el testbench en Python, solamente indicando en Verilog qué señales o puertos provienen de MyHDL y cuáles son enviadas hacia MyHDL. De esta forma, es posible generar estímulos y mandarlos directamente a un puerto sin archivos de por medio, leer puertos de salida y graficarlos de manera directa, e iterar diferentes simulaciones variando parámetros de manera automática.

MyHDL también permite visualizar formas de onda mediante programas externos como *GTKWave* a los que se le cargan archivos específicos generados por el testbench.

Por último, al usarlo como lenguaje de descripción de hardware permite la conversión del código en Python a Verilog o VHDL, el cual incluso puede ser sintetizable si se siguen ciertas directivas al programar el módulo. Esta característica no es utilizada en el proyecto.

7.3. Módulos Complementarios

Si bien el proyecto se enfoca en la implementación en FPGA de un ecualizador en el dominio de la frecuencia, para poder analizar y verificar su correcto funcionamiento resulta necesario también desarrollar una serie de módulos complementarios que permitan insertar el ecualizador diseñado en un sistema de comunicaciones básico. En la Fig. 7.1 puede apreciarse un diagrama de bloques mostrando la totalidad del sistema a desarrollar.

7.3.1. Generación de Símbolos y Modulación

Los símbolos a transmitir a través del sistema de comunicaciones se generan mediante el bloque denominado PRBS (*PseudoRandom Binary Sequence*), es decir una Secuencia de Bits Pseudoaleatoria, llamada así debido a que se genera a partir de un algoritmo determinístico pero exhibe un comportamiento estadístico similar al de una secuencia verdaderamente aleatoria.

Dependiendo del tipo de PRBS utilizada, la secuencia de bits se repite cada un cierto número de bits generados. Así, por ejemplo, en el caso de PRBS7 la secuencia se repite cada $127 = 2^7 - 1$ bits, y de manera general, para una PRBSn la secuencia se repite cada $2^n - 1$ si se cumplen ciertas condiciones.

La implementación de este bloque se lleva a cabo mediante un Registro

de Desplazamiento con Realimentación Lineal (*Linear Feedback Shift Register - LFSR*), es decir que el bit de entrada del registro de desplazamiento es una función lineal del estado anterior del mismo, concretamente es una operación lógica XOR (OR exclusiva) entre 2 bits determinados del shift register, tal como se ilustra en la Fig. 7.3 para el caso PRBS7, aunque en el proyecto la que se utilizó es una PRBS9.

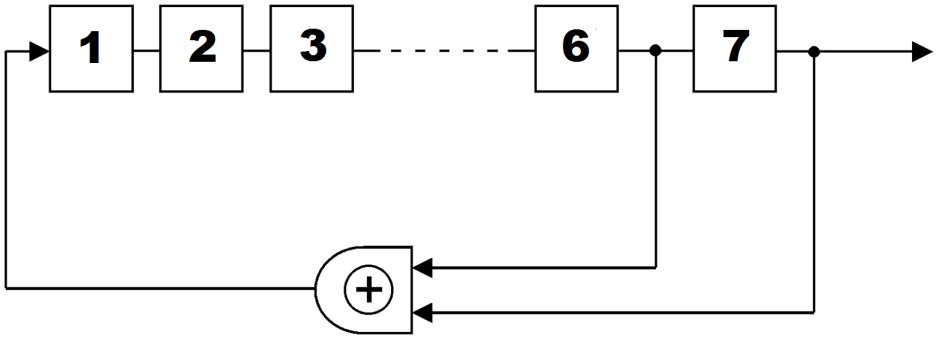


Figura 7.3: *Registro de Desplazamiento con Realimentación Lineal para PRBS7.*

La elección de los bits entre los cuales se lleva a cabo la XOR no es trivial, ya que para garantizar la secuencia de máxima longitud (es decir aquella que se repita cada $2^n - 1$), deben ser 2 bits específicos de acuerdo al tipo de PRBS. Por ejemplo, para una PRBS7 la XOR debe realizarse entre los bits 6 y 7 (comenzando la numeración en 1), para una PRBS9 entre los bits 5 y 9, para una PRBS23 entre los bits 18 y 23, para una PRBS31 entre los bits 28 y 31, etc. Los bits entre los cuales se hace la XOR se pueden indicar con la notación de polinomios de la Ec. (7.1).

$$\begin{aligned} \text{PRBS7} &= x^7 + x^6 + 1 \\ \text{PRBS9} &= x^9 + x^5 + 1 \\ \text{PRBS23} &= x^{23} + x^{18} + 1 \\ \text{PRBS31} &= x^{31} + x^{28} + 1 \end{aligned} \tag{7.1}$$

Para generar una modulación QPSK, basta con utilizar diferentes bloques de PRBS para generar 2 secuencias de bits independientes, pasando cada una por su respectivo filtro transmisor y obteniendo sus respectivas salidas, representando cada secuencia las componentes I y Q. En el diagrama de bloques de la Fig. 7.1, estas 2 componentes están representadas en la señal compleja que va de un bloque a otro, ya que cada componente puede asociarse a la parte real e imaginaria de un número complejo.

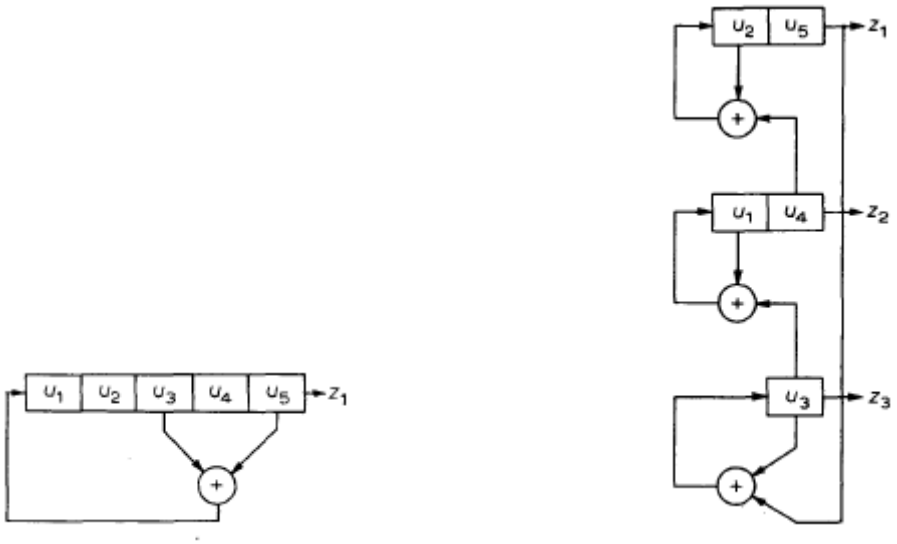
Ya que el sistema a implementar trabaja en forma paralela, es necesario generar una cantidad mayor de símbolos a transmitir en cada ciclo de

clock, de manera que ingresen en simultáneo a las múltiples instancias de los módulos posteriores.

Debido a que las especificaciones planteadas requieren un paralelismo igual a 8, lo que se hace es implementar una PRBS que genere palabras de 8 bits en cada ciclo de clock, de manera tal que la serialización de estas palabras conforme una secuencia idéntica a la generada por la PRBS serial explicada anteriormente.

En [10] se detalla cómo llevar a cabo una implementación genérica de una PRBS en paralelo, dependiendo del tipo de PRBS y del grado de paralelismo requerido. A partir de dicha explicación, se utiliza un *script* en Python que de manera automática genera los polinomios adecuados para cada uno de los bits que salen en paralelo, escritos de manera tal que pueden ser empleados directamente en Verilog.

En la Fig. 7.4 se ejemplifica la implementación de una PRBS5 sin paralelismo (Fig. 7.4a) y con un paralelismo de 3 (Fig. 7.4b).



(a) *Sin paralelismo.* (b) *Paralelismo 3.*

Figura 7.4: *Implementación de una PRBS5 con diferentes paralelismos.*

Resta determinar cómo se implementa el codificador y decodificador necesarios para lograr una modulación diferencial.

Para la codificación (*Precoder DQPSK*) se instancian módulos a los cuales ingresan tanto la parte real como la parte imaginaria de los símbolos generados, se realizan a partir de ellas determinadas operaciones con lógica combinacional y se entrega a su salida los símbolos con modulación diferencial.

El decodificador (*Decoder DQPSK*) funciona de manera análoga, reci-

biendo a su entrada los símbolos con modulación diferencial, detectando la diferencia de fase entre símbolos consecutivos mediante lógica combinacional, y entregando a su salida los símbolos ya decodificados.

En la tabla 7.1 se indican los valores de salida que debe tener el codificador (y la diferencia de fase que representan) de acuerdo a los símbolos de entrada y a los valores anteriores del codificador.

No Codificado	Anterior	Codificado
00	00	00 (0°)
01	00	10 (270°)
10	00	01 (90°)
11	00	11 (180°)
00	01	01 (90°)
01	01	00 (0°)
10	01	11 (180°)
11	01	10 (270°)
00	10	10 (270°)
01	10	11 (180°)
10	10	00 (0°)
11	10	01 (90°)
00	11	11 (180°)
01	11	01 (90°)
10	11	10 (270°)
11	11	00 (0°)

Tabla 7.1: Mapeo de bits en modulación DQPSK.

El comportamiento de la tabla puede lograrse utilizando lógica combinacional, implementando las operaciones de las Ec. 7.2 y 7.3, en las que $b_1 b_0$ son los bits a codificar, I_{k-1} y Q_{k-1} son las salidas del codificador reales e imaginarias anteriores respectivamente, I_k y Q_k son los valores codificados.

$$I_k = \overline{b_0 \oplus b_1} \times (b_0 \oplus I_{k-1}) \; + \; (b_0 \oplus b_1) \times (b_1 \oplus Q_{k-1}) \tag{7.2}$$

$$Q_k = \overline{b_0 \oplus b_1} \times (b_1 \oplus Q_{k-1}) \; + \; (b_0 \oplus b_1) \times (b_0 \oplus I_{k-1}) \tag{7.3}$$

La decodificación se lleva a cabo a partir del símbolo actual y del anterior utilizando los valores de las fórmulas de las Ec. 7.4 y 7.5.

$$A = I_k \times I_{k-1} \; + \; Q_k \times Q_{k-1} \tag{7.4}$$

$$B = I_{k-1} \times Q_k \; - \; I_k \times Q_{k-1} \tag{7.5}$$

Siempre ocurre que uno de los valores de las Ec. 7.4 y 7.5 tiene un mayor valor absoluto que el otro, ya sea A o B (uno da 0 y el otro ± 2). De acuerdo a cuál de los dos sea el mayor, los bits $b_1 b_0$ que se desean determinar pertenecerán al conjunto $\{00, 11\}$ o al conjunto $\{01, 10\}$. Para determinar cuál de los dos elementos del conjunto es el correcto, se utiliza el signo del mayor valor.

7.3.2. Filtro Transmisor

Al contar sólo con un filtro transmisor (y no también con uno receptor), el mismo será de tipo Coseno Realzado, cuyos coeficientes serán generados en Python con la resolución y parámetros adecuados y luego cargados en el módulo mediante el uso de la plataforma de verificación.

La forma más simple de implementarlo es mediante un filtro FIR con la estructura de la Fig. 7.5, en el que van ingresando muestras al bloque, se van colocando en un shift register y se hace una convolución con los coeficientes del filtro para obtener la salida del transmisor.

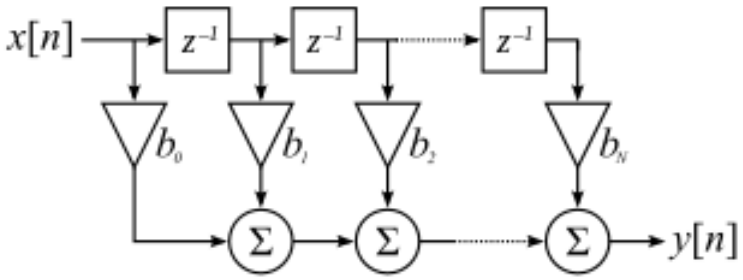


Figura 7.5: *Estructura de un filtro FIR con N coeficientes.*

Sin embargo, dado que se requiere realizar un sobremuestreo a los símbolos generados, se decide utilizar una estructura polifásica debido a su mayor eficiencia.

Un sobremuestreo tradicional se llevaría a cabo colocando una determinada cantidad de ceros entre cada símbolo generado (en este caso se trabaja con un factor de *oversampling* de 2, por lo que se colocaría 1 cero entre cada símbolo generado, duplicando así la tasa), siendo esta nueva secuencia la que ingresa al filtro FIR. Pero esta implementación resulta ineficiente, ya que una gran cantidad de operaciones darán como resultado un valor nulo, debido a la multiplicación de los coeficientes por los ceros agregados en la secuencia. Por esta razón, se estarían llevando a cabo operaciones que de antemano ya es posible saber que darán cero como resultado.

Para evitar estas operaciones innecesarias, considerando una cantidad de coeficientes N y un factor de sobremuestreo OS, en una implementación polifásica no se agregan ceros a la secuencia original, sino que esta ingresa de manera simultánea a una cantidad igual a OS de subfiltros, cada uno de los cuales cuenta con N/OS coeficientes trabajando a la tasa de muestreo original (de menor valor). Cada subfiltro cuenta con los coeficientes C_i, C_{i+os}, C_{i+2os} y así sucesivamente hasta completar el tamaño adecuado. Las salidas de los subfiltros se conectan a un multiplexor que trabaja a la tasa de muestreo superior y se encarga de concatenar las salidas de los subfiltros, obteniendo así exactamente la misma salida que con la estructura de la Fig. 7.5.

En la Fig. 7.6 puede apreciarse la estructura de un filtro polifásico con $N = 4$ coeficientes y sobremuestreo $OS = 2$, observándose que hay 2 subfiltros, cada uno con los correspondientes coeficientes C_i y C_{i+2}.

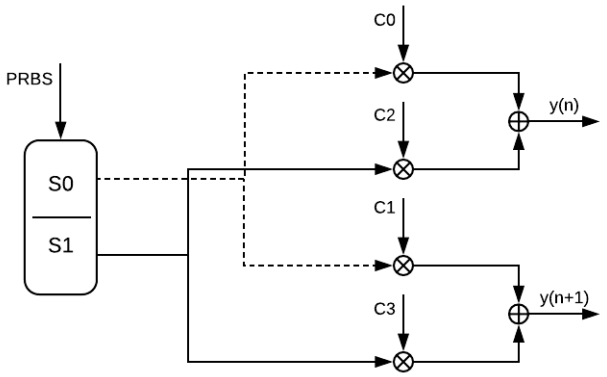


Figura 7.6: *Estructura de un filtro polifásico con $N = 4$ coeficientes y factor de sobremuestreo $OS = 2$.*

Para este sistema, se utiliza un filtro Coseno Realzado con Sobremuestreo $OS = 2$ y Número de Baudios $Nbaud = 8$, resultando un total de 16 coeficientes con resolución $S(8, 7)$, habiéndose determinado mediante simulaciones en punto fijo que ésta era la resolución óptima.

Aprovechando el upsampling, en este punto se incrementa el paralelismo del sistema a 16, eliminando el multiplexor de salida y entregando ambas salidas en simultáneo mediante sus respectivos puertos de salida.

Por otro lado, en la Fig. 7.5 se puede ver que las muestras de entrada ingresan de forma serial, y como ya se ha explicado, en este proyecto se trabaja con una arquitectura en paralelo, con lo cual, en un mismo instante de tiempo se contará con todas las muestras necesarias y no se requerirá la presencia del shift register mencionado. Sin embargo, para que el fun-

cionamiento sea análogo al caso serial y las muestras ingresen en el orden correcto a cada transmisor, será necesario agregar un buffer a la entrada de los mismos que almacenará las nuevas muestras provenientes de la PRBS, más una cierta cantidad de muestras del ciclo anterior definidas por $Nbaud * OS - 1$.

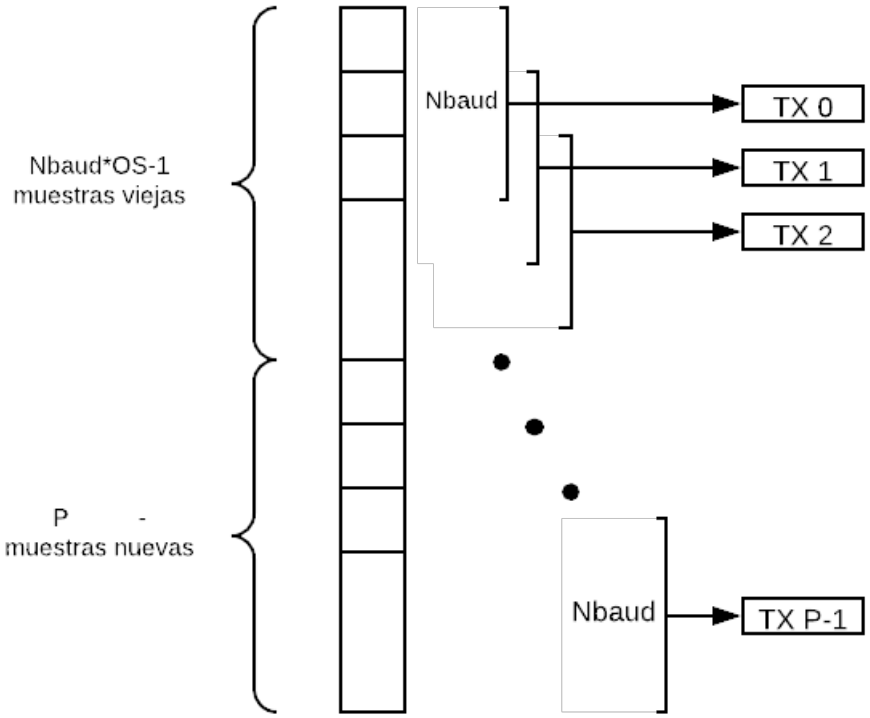


Figura 7.7: *Estructura del buffer a la entrada de los filtros transmisores.*

De este buffer se toman las muestras de a $Nbaud = N/OS$ para ingresar al transmisor como se muestra en la Fig. 7.7. Esto se debe precisamente a la utilización del filtro polifásico que se explica anteriormente.

7.3.3. Canal

El canal de transmisión que los bits atravesarían se simula mediante dos bloques: un filtro FIR que representa la respuesta en frecuencia del canal, y un Generador de Ruido Gaussiano (*Gaussian Noise Generator - GNG*) que se encarga de generar el ruido que el canal incorpora a la señal.

El filtro FIR se implementa utilizando una estructura similar a la de la Fig. 7.5, teniendo en cuenta nuevamente que la entrada se encuentra paralelizada, con lo cual una vez más será necesario incorporar un buffer a la entrada del canal, con la única diferencia de que esta vez se tiene el doble del paralelismo debido al upsampling y ya no es posible utilizar un

filtro polifásico. Por lo tanto, se tendrá como resultado una entrada de la forma que se muestra en la Fig. 7.8.

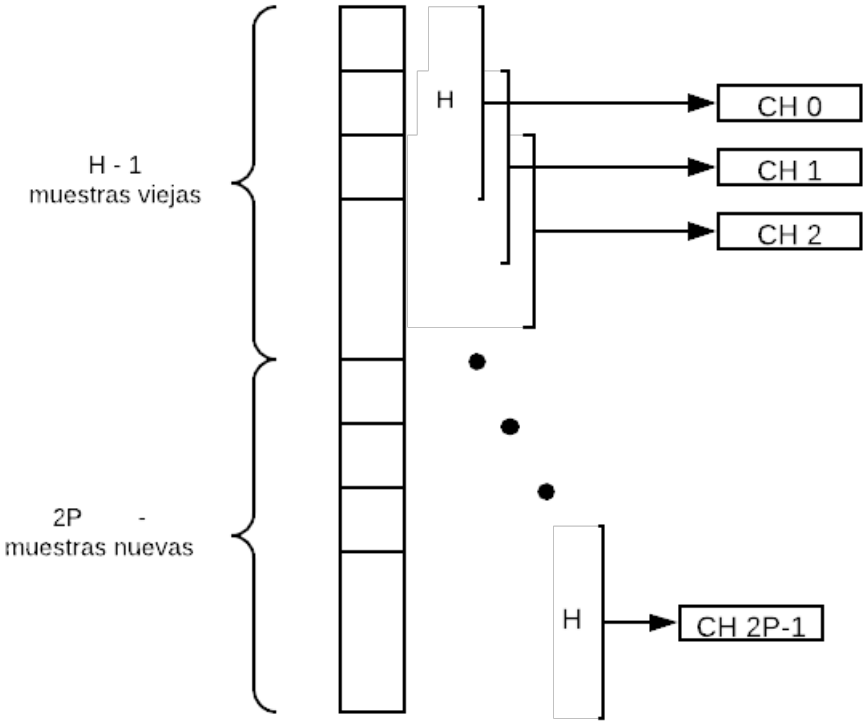


Figura 7.8: *Estructura del buffer a la entrada de los filtros del canal.*

En este caso cada filtro contará con un número de coeficiente $H = 11$ que se pueden configurar mediante uno de los puertos de entrada del módulo. La resolución utilizada para los coeficientes es $S(8, 6)$.

Para el generador de ruido gaussiano, se utiliza un IP Core obtenido de Open Cores. El IP Core utilizado genera una salida de 16 bits empleando una resolución en punto fijo de $S(16, 11)$, con una distribución normal en un rango de $\pm 9{,}1\sigma$ y repitiéndose la secuencia con un período de aproximadamente 2^{176}. Internamente cuenta con un generador de números aleatorios de 64 bits con semillas iniciales parametrizables, por lo que se pueden obtener diferentes secuencias de ruido gaussiano con las mismas propiedades.

Para ajustar el nivel de SNR, se multiplica la salida del IP Core por una constante en resolución $S(11, 10)$, cuyo valor depende del valor de SNR deseado y se configura mediante un puerto de entrada del módulo.

7.3.4. Contador de BER

Para determinar el Bit Error Rate del sistema, es necesario contabilizar tanto la cantidad de bits procesados como la cantidad de errores que tienen lugar al procesar dicha cantidad de bits.

El módulo cuenta con una PRBS interna inicializada con la misma semilla que la PRBS que genera los símbolos transmitidos, y simplemente compara bit a bit los símbolos detectados con los símbolos generados por su propia PRBS interna, incrementando en 1 la cantidad de errores en caso de que ambos símbolos difieran, llevando siempre la cuenta de la cantidad de bits que se van procesando.

El principal desafío radica en sincronizar la PRBS interna con los símbolos detectados que recibe, ya que habrá un determinado retardo debido al sistema de comunicaciones que atravesó. Por ello es que el contador de BER tiene un período inicial de adaptación, en el que compara los símbolos detectados con sucesivas secuencias de la PRBS desfasadas en diferentes valores. Aquella secuencia que genere la menor cantidad de errores es la que se encuentra sincronizada con los símbolos recibidos y la que se utiliza para la comparación luego del período de adaptación.

El conteo de errores y de bits procesados recién comienza una vez finalizado el período de adaptación, lo cual se indica mediante un puerto de salida del módulo.

7.4. Plataforma de Verificación

Para determinar si el sistema diseñado se comporta de la manera esperada, y más concretamente el ecualizador adaptivo, es necesario comunicarse de alguna forma con la FPGA tanto para configurar los diferentes módulos como para extraer datos en diferentes puntos del sistema, es decir para llevar a cabo el logueo de datos.

7.4.1. Register File

Para lograr esta comunicación, por un lado se crea un módulo especial que forma parte del sistema llamado Registro de Archivo (*Register File - RF*), el cual recibe diferentes comandos en su puerto de entrada y los traduce en órdenes para los distintos módulos que forman parte del sistema de comunicaciones propiamente dicho.

La entrada principal del RF es de 32 bits, utilizándose 8 para el comando en sí, 1 para habilitación (enable E) y 23 para datos que se requieran para ciertas configuraciones o que se deseen pasar a algún módulo determinado.

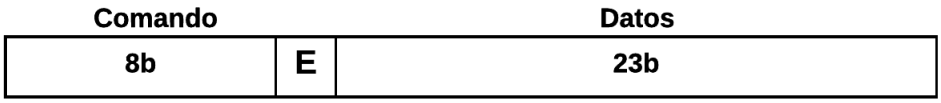


Figura 7.9: *Formato de entrada para comandos en Register File.*

El Register File cuenta también con otras entradas para recibir datos provenientes de los distintos módulos. Además cuenta con diferentes salidas conectadas a los módulos que se quieran comandar y a las cuales se les asigna distintos valores mediante las órdenes que recibe.

A grandes rasgos, las tareas que el RF permite llevar a cabo son las siguientes:

- Reiniciar el sistema, enviando una señal de reset a todos los módulos.
- Habilitar en forma individual cada uno de los módulos, enviando la correspondiente señal de *enable* a ellos (PRBS, filtro transmisor, codificador/decodificador, ecualizador, contador de BER, ruido del canal y filtro FIR del canal).
- Cargar coeficientes del filtro transmisor y del filtro FIR que representa la respuesta del canal.
- Configurar nivel de SNR.
- Extraer datos en diferentes puntos del sistema (salida del filtro transmisor, salida del canal, salida del ecualizador, señal de error del ecualizador, coeficientes del ecualizador, cantidad de bits procesados por el contador de BER y cantidad de errores contados).

En la Fig. 7.10 se puede observar cómo el Register File se interconecta con el resto de los módulos del sistema para permitir configurarlos y extraer datos en sus diferentes puntos.

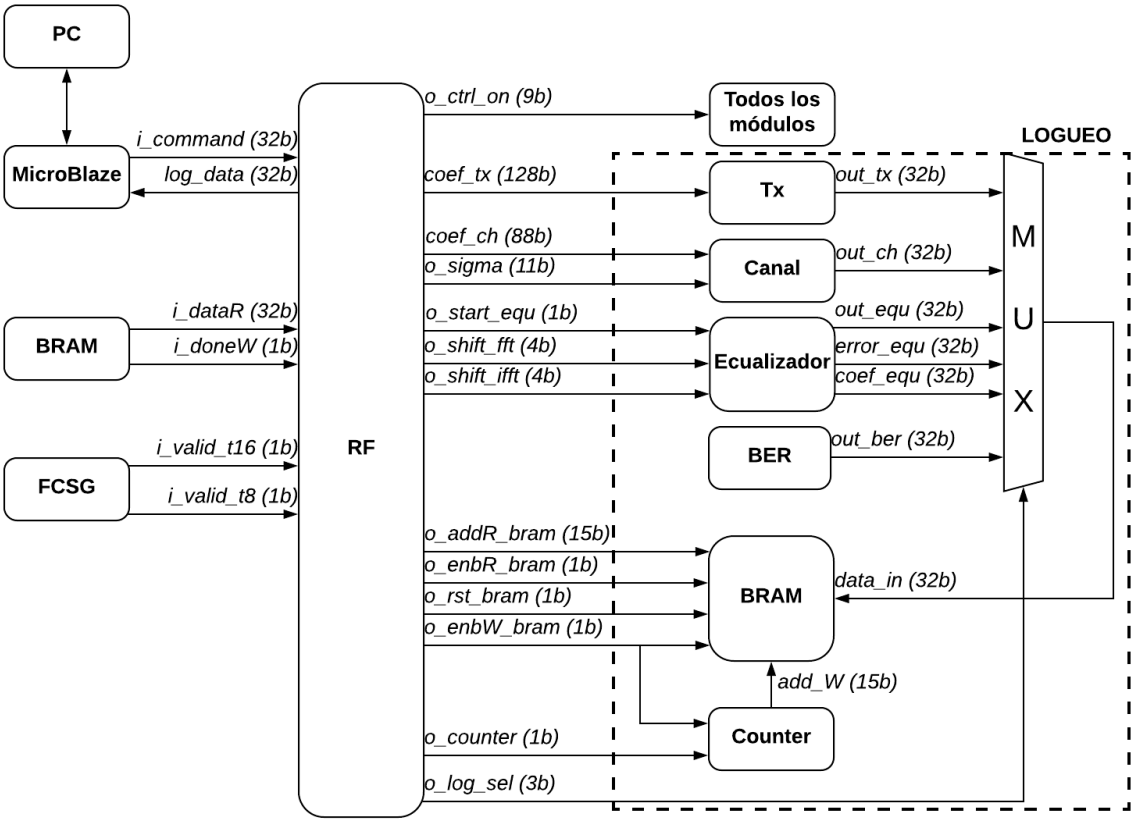


Figura 7.10: *Interconexión del Register File con el resto de los módulos del sistema.*

El funcionamiento de la mayor parte de los comandos es relativamente simple, limitándose los mismos a colocar en un determinado puerto de salida (indicado por el comando que recibe) un determinado valor (indicado también por el comando). Por su mayor complejidad, resulta necesario profundizar en la forma en la que se extraen los datos de la FPGA.

Para el logueo de datos primero se envía un comando indicando que se desea loguear una determinada salida del sistema (especificando previamente qué salida mediante otro comando) y en ese momento comienza a escribirse una memoria RAM con los datos solicitados (el RF se encarga de habilitar la escritura de la memoria), hasta que la misma se llena. La memoria RAM es de 2^{15} palabras de 32 bits cada una, la cual es implementada internamente en la FPGA utilizando módulos de memoria con los que cuenta la misma llamados Block RAM (*BRAM*).

Una vez que se ha llenado la BRAM, se envía un comando indicando que se desea leer el contenido de la memoria, especificando la dirección de la misma a la cual se quiere acceder, por lo que se debe enviar un comando para cada posición de memoria leída.

A diferencia de la lectura, en la que en el propio comando se especifica la dirección a leer, para el caso de la escritura de la BRAM se utiliza

un contador (*Counter*) cuya salida se encuentra conectada a la dirección de escritura de la BRAM, y que comienza a contar cuando se recibe el comando de logueo (el RF habilita el contador), y detiene su cuenta cuando alcanza la cantidad total de palabras de la BRAM. Cuando alcanza dicho valor, envía una señal al RF y este deshabilita la escritura de la BRAM, evitando que se sobrescriban datos. Este contador trabaja en sincronía con el resto del sistema de comunicaciones, por lo que garantiza que se tomen muestras consecutivas sin que se pierdan datos entre medio. El mismo no siempre incrementa su cuenta a la misma velocidad, sino que la misma varía de acuerdo al punto del sistema que se desee loguear, ya que se trabaja a diferentes velocidades a lo largo de todo el sistema. Para lograr esto, el RF le indica al *Counter* a qué velocidad debe contar, dependiendo de la variable que se haya seleccionado para loguear.

Las consideraciones anteriores no son importantes al momento de la lectura, debido a que las muestras consecutivas ya han sido tomadas, por eso es posible especificar en el propio comando la dirección a leer.

Por otra parte, al paralelizar el sistema, en un mismo instante de tiempo se cuentan con múltiples muestras a la salida del filtro transmisor y a la salida del canal, por lo que es necesario serializar dichas muestras antes de que las mismas puedan ser logueadas. Una vez efectuada la serialización, el logueo de las mismas se efectúa de la misma forma que para el resto de los puntos de interés del sistema.

Cabe mencionar también que la señal de reset del sistema, como así también las señales de enable de diferentes módulos, se encuentran conectadas a diferentes LEDs de la placa, a los fines de tener una indicación visual del estado en el que se encuentra el sistema.

7.4.2. MicroBlaze y Comunicación Serial

Si bien el Register File controla el sistema de acuerdo a los comandos que recibe, resta determinar cómo el usuario envía dichos comandos al RF. Para poder enviar comandos desde una PC, se instanció en la FPGA un Soft-Processor cuyo puerto de salida se conecta al puerto de entrada del Register File, y desde la PC se envían comandos al procesador utilizando comunicación serial.

Un Soft-Processor o Microprocesador Embebido es un microprocesador descripto en un lenguaje HDL que se implementa utilizando recursos de la FPGA. En este caso se utiliza *Micro-Blaze*, un soft-processor ofrecido por *Xilinx* como un *IP Core* que simplemente se debe instanciar como un

módulo adicional del proyecto.

Este microprocesador cuenta con un módulo UART utilizado para la comunicación por puerto serie con una PC, y otros adicionales como el *Clock Wizard*, al cual ingresa una señal de clock generada por la propia FPGA y el mismo se encarga de generar la señal de clock que utiliza internamente el mismo microprocesador y que además gobierna a todo el sistema de comunicaciones.

Además, se cuenta con un puerto de reset (entrada conectada a un interruptor físico de la FPGA que se utiliza para resetear el procesador), un puerto de entrada y otro de salida, ambos de propósitos generales de 32 bits (gpio_i_data y gpio_o_data respectivamente) y un último puerto utilizado para la comunicación serial. Los GPIO se encuentran conectados al Register File, utilizándose el de salida para enviar los comandos y el de entrada para recibir los datos que se desean loguear.

Una vez instanciado el microprocesador, es necesario programarlo utilizando el lenguaje C, para lo cual se utilizan también las herramientas brindadas por Xilinx, concretamente el *Software Development Kit* (*Xilinx SDK*).

El programa en C es sencillo. Para el caso del envío de comandos, se encarga de recibir por puerto serie secuencias de 32 bits (que representan el comando) y colocar dicho valor en su GPIO de salida. Dado que sólo es posible transmitir por puerto serie 8 bits a la vez, el programa espera hasta que se reciban 4 conjuntos de 8 bits, los acomoda adecuadamente y luego se escriben los 32 bits en el GPIO.

Para el caso de la recepción de datos provenientes del RF, inicialmente desde la PC se envía un comando indicando que se desea leer una determinada posición de memoria en la BRAM (previamente llenada con los datos deseados). Al recibir el comando, el Register File se encarga de colocar en el GPIO de entrada del procesador el contenido de dicha dirección. Luego desde el procesador se lee el contenido de dicho puerto y se lo envía por puerto serie a la PC. Al igual que en el caso anterior, se lo envía de a grupos de 8 bits y en el programa de Python en la PC lo reconstruye. A su vez, ya que el procesador simplemente recibe secuencias de bits y las coloca en su correspondiente puerto, el programa en PC debe encargarse también de armar la trama adecuada de acuerdo a la acción que se desee llevar a cabo.

Para la comunicación desde la PC, se confeccionó en Python un programa que recibe comandos escritos por parte del usuario a través de un terminal, y de acuerdo al comando ingresado arma la secuencia de bits adecuada y la envía por puerto serie.

Al momento de leer la BRAM, este programa es quien se encarga de ir solicitando y guardando una a una las diferentes posiciones de memoria de la BRAM (no se solicita la siguiente hasta que no se haya recibido la que se pidió previamente).

También permite realizar distintos gráficos a partir de los datos extraídos para visualizarlos mejor, como por ejemplo diagramas de ojo a la salida del transmisor o del canal, la evolución en el tiempo de la salida del ecualizador, su error o los distintos coeficientes.

Se puede ver que mediante el Register File, el soft-processor y el programa en Python es posible configurar y controlar completamente el sistema que se implemente en FPGA, siendo posible obtener información acerca de los valores que toman diferentes variables dentro del sistema para determinar su correcto funcionamiento.

7.5. Ecualizador Adaptivo

7.5.1. FFT/IFFT IP Core

La implementación en el dominio de la frecuencia requiere de la utilización de la Transformada de Fourier (tanto FFT como IFFT), la cual por lo tanto debe ser implementada también en un lenguaje de descripción de hardware.

Debido a la complejidad del módulo y a los fines de agilizar el diseño, se decide emplear un IP Core ya existente creado por terceros, siendo conscientes de que esto conllevaría el problema de tener que generar en el simulador un comportamiento lo más parecido posible al del IP Core.

Se analizan diferentes alternativas de código abierto y libre uso disponibles en *Open Cores*. Para determinar cuál se utilizaría, se tienen en cuenta variables como recursos utilizados, capacidad de paralelización y parametrización, latencia, y el hecho de si permite o no un flujo continuo de datos (es decir que permita que ingrese una muestra nueva por clock sin que sea necesario introducir demoras adicionales).

De esta forma, el IP Core que se decide utilizar es el llamado *Pipelined FFT/IFFT 128 points processor*, el cual tiene las características que se presentan a continuación.

- FFT/IFFT de 128 puntos, radix-8.
- 310 ciclos de clock de latencia, con una salida por ciclo de clock.
- Entrada, salida y twiddle factors parametrizables de 8 a 16 bits.

- Etapas de normalización y detectores de overflow.

Para tener una estimación de los recursos utilizados por el IP Core, se lo sintetiza para una FPGA Kintex-7, que es la que se utiliza para implementar el proyecto, con 12 bits de entrada, 16 bits de salida y 8 bits para los twiddle factors, obteniendo así los valores de la Tabla 7.2.

Recurso	Utilización	% de utilización
LUT	2667	1.31
Memory LUT	124	0.19
FF	3958	0.97
DSP	4	0.48
BRAM	1	0.22

Tabla 7.2: Recursos utilizados en una Kintex-7 por el IP Core utilizado para el cómputo de FFT/IFFT.

En la Fig. 7.11 se puede observar el esquemático del módulo con sus respectivas señales de entrada y salida, tal como se indica en su hoja de datos ([15]).

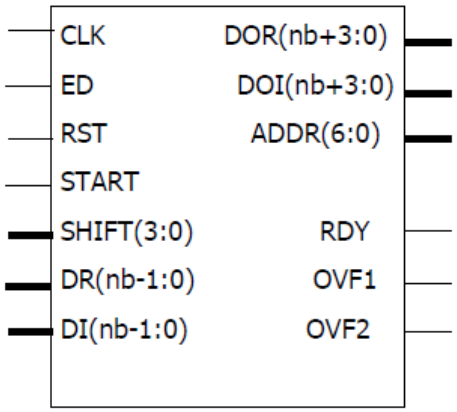


Figura 7.11: *Esquemático del IP Core para cómputo de FFT/IFFT.*

Los puertos *CLK*, *RST* y *ED* corresponden a las señales de clock, reset y habilitación respectivamente.

Las muestras de entrada y salida son números enteros complejos en complemento a 2 de *nb* (entrada) y *nb+4* (salida) bits, que pueden interpretarse adecuadamente como números en representación en punto fijo, mientras se mantenga la misma parte fraccional para la entrada y la salida. Los puertos *DR* y *DI* se corresponden con las partes reales e imaginarias respectivamente de las muestras de entrada, mientras que los puertos *DOR* y *DOI* tienen un comportamiento análogo para las muestras de salida.

El puerto de entrada *SHIFT* permite hacer ajustes de amplitud para evitar que ocurra un overflow en las etapas internas del módulo, lo cual se indica mediante los puertos de salida *OVF1* y *OVF2*.

El puerto *START* se utiliza para indicar el inicio de la entrada de muestras, las cuales se empiezan a tomar con el flanco de bajada de la señal de *START*, y a partir de dicho momento se toma una muestra nueva por cada ciclo de clock, tal como puede apreciarse en la Fig. 7.12.

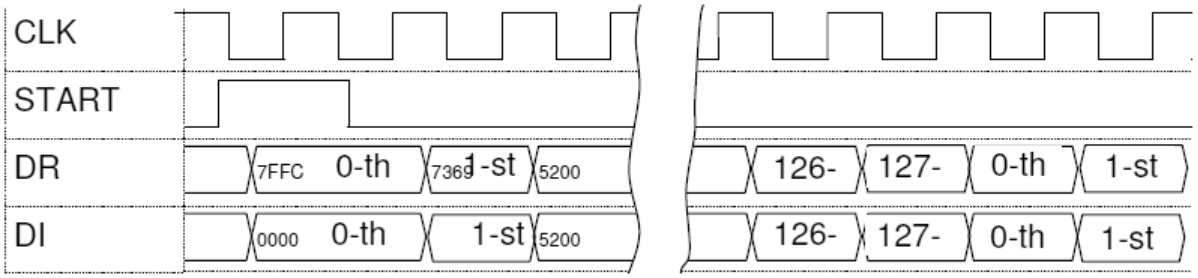


Figura 7.12: *Formas de onda de los datos de entrada del IP Core.*

Las muestras de salida se obtienen en los puertos correspondientes a partir del momento en que la señal *RDY* se pone en *1*, tal como se observa en la Fig. 7.13, en la que también es posible apreciar que la señal *ADDR* indica el número de muestra de salida al cual corresponde cada valor (por lo tanto es un número entre 0 y 127). Una vez que se llega a la muestra de salida 127 de un determinado bloque de entrada, la siguiente salida corresponde a la muestra de salida 0 del siguiente bloque que haya ingresado.

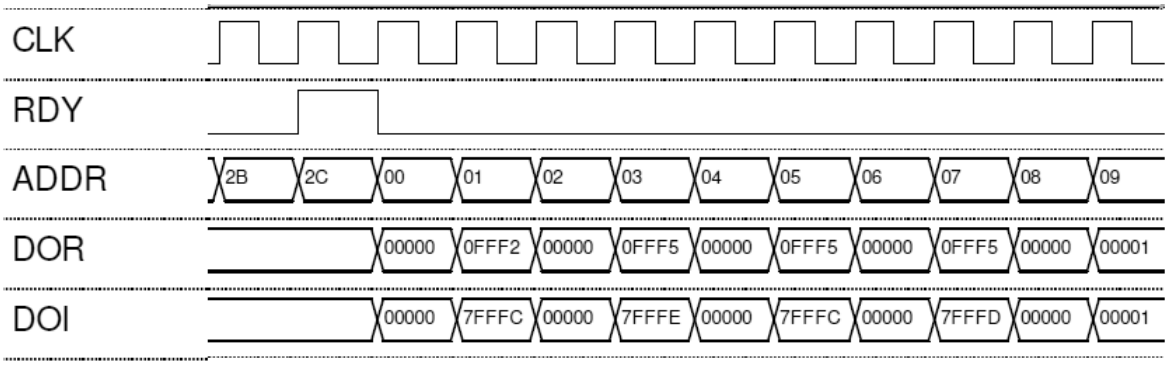
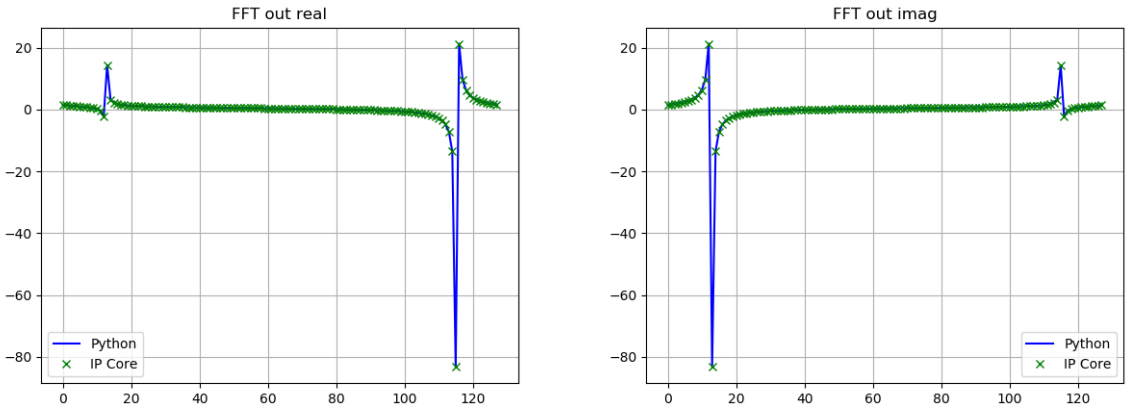


Figura 7.13: *Formas de onda de los datos de salida del IP Core.*

Una vez comprendido el funcionamiento del IP Core mediante la lectura de la documentación provista con el mismo, se realizan algunas pruebas utilizando solamente el módulo de manera individual para verificar que se comporta de la manera esperada.

Inicialmente se prueba su funcionamiento como FFT, colocando en su entrada una señal senoidal en punto fijo, la cual se genera en Python, y se compara su salida con la salida generada por la función FFT del paquete *NumPy* de Python. Colocando a la entrada un seno con una única frecuencia (con la misma parte real e imaginaria) en resolución $S(8,4)$, se obtienen las salidas en el dominio de la frecuencia de la Fig. 7.14, en la que se realiza una comparación entre la salida del IP Core y los valores obtenidos en Python, pudiéndose ver que ambos coinciden aproximadamente aunque no son exactamente iguales. También se probaron señales de entrada más complejas, obteniendo en todos los casos resultados satisfactorios.



(a) *Salida real de la FFT.* (b) *Salida imaginaria de la FFT.*

Figura 7.14: *Comparación de resultados entre FFT de Python y el IP Core utilizado, colocando un seno complejo a su entrada.*

Una vez comprobado su correcto funcionamiento como FFT, se decide colocar en cadena una FFT y una IFFT, debiendo obtener a la salida de la concatenación lo mismo que a la entrada. La interconexión se lleva a cabo conectando las salidas reales e imaginarias de la FFT a las entradas reales e imaginarias de la IFFT, y utilizando la señal de *RDY* de la FFT como señal de *START* de la IFFT. Al probar esta interconexión, no se obtuvieron los resultados esperados, no siendo posible recuperar la forma de onda de entrada. Ante esto, se decide probar la IFFT de manera aislada, colocando a su entrada la salida de la FFT generada en Python, no pudiendo obtener tampoco los resultados esperados.

Luego de numerosos intentos infructíferos, se concluye que la IFFT no estaba implementada de manera correcta. Para evitar modificar el IP Core a los fines de corregir esta problemática, se decide aprovechar el hecho de que es posible calcular una IFFT a partir de una FFT sin incrementar demasiado la complejidad del sistema.

En la Fig. 7.15 se puede ver cómo es posible realizar una IFFT a partir de una FFT de una manera relativamente sencilla, simplemente cruzando las conexiones de las partes reales e imaginarias, y dividiendo el resultado por N, siendo N la cantidad de puntos de la FFT/IFFT. Debido a que N es una potencia de 2, esto simplemente se reduce en un desplazamiento de bits hacia la derecha.

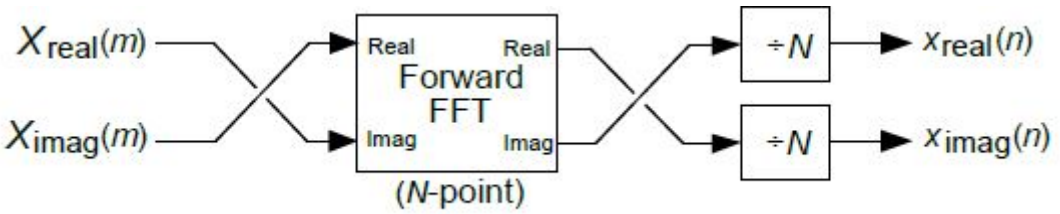


Figura 7.15: *Cómputo de una IFFT a partir de una FFT.*

La interconexión de la Fig. 7.15 se replica utilizando el IP Core tal como se muestra en la Fig. 7.16, en la que se puede apreciar cómo la salida real de la primera FFT (DOR) se conecta a la entrada imaginaria de la segunda FFT (DI), y la salida imaginaria (DOI) se conecta a la entrada real (DR). Luego las salidas de la segunda FFT se dividen por N, por lo que se conectan a módulos que realizan dicha división mediante un desplazamiento binario que conserva el signo. Finalmente se puede ver que la salida imaginaria de la segunda FFT (una vez dividida por N) se conecta al puerto de salida real (DOR_shift), y la salida real de la FFT al puerto de salida imaginario (DOI_shift).

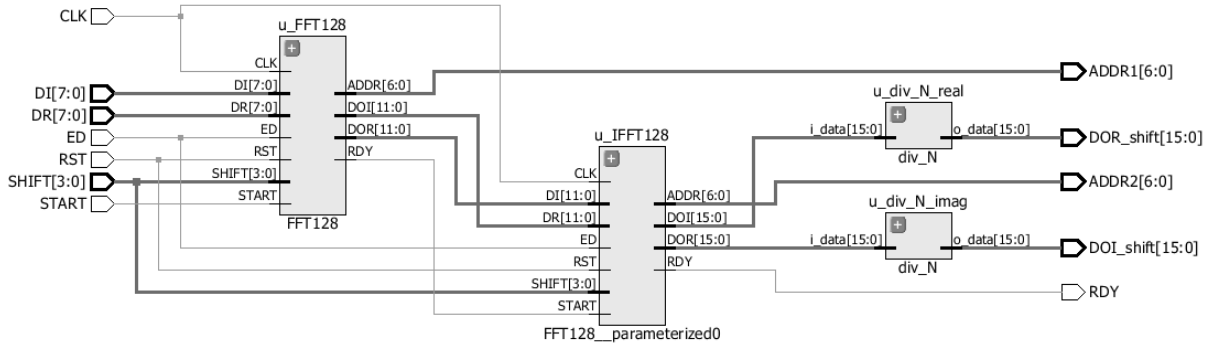


Figura 7.16: *Cómputo de una IFFT a partir de una FFT con IP Cores.*

Un aspecto importante que se debe tener en cuenta y que incorpora ciertas limitaciones al momento de diseñar el sistema, es el hecho de que el IP Core no permite ajustar la resolución de sus datos, por lo que impone una relación prefijada entre la resolución de entrada y de salida de sus muestras. Concretamente, si la entrada tiene la forma $S(NB, NBF)$, la

salida es de la forma $S(NB+4, NBF)$, es decir se agregan 4 bits a la parte entera y se mantiene la misma cantidad de bits para la parte fraccional. Esto es tenido en cuenta al momento de realizar las gráficas de resolución óptima en la etapa de diseño de la Sección 6.4.1.

7.5.2. Overlap & Save

Como se vio en la Sección 4.1.6, si se desea realizar un filtrado en frecuencia en el que la secuencia de entrada tiene una gran longitud, mucho mayor a la del filtro, se debe recurrir al método de Overlap&Save.

Tal como se explica en la mencionada sección, a la FFT de N puntos ingresan bloques de entrada de tamaño $N = L+M-1$ constituidos por las últimas $M-1$ muestras del bloque anterior seguidas por L nuevas muestras de la secuencia de entrada. Luego se realiza el procesamiento en el dominio de la frecuencia y se computa la IFFT de N puntos, descartándose las primeras $M-1$ muestras y conservando las últimas L.

Si bien los módulos se diseñan para que sean parametrizables, se considera el caso concreto a implementar en el que el tamaño de los bloques de entrada es $L = 64$ y la cantidad de coeficientes del ecualizador es $M = 65$, resultando así un tamaño total de $N = L + M - 1 = 128$, que es también la cantidad de puntos de la FFT.

Se diseña por un lado un módulo encargado de acomodar adecuadamente las muestras de entrada, llamado *overlap_in*, y por otro lado uno que se ocupa de entregar las muestras correctas a la salida, ya procesadas en el dominio de la frecuencia, llamado *save_out*.

En la Fig. 7.17 se pueden observar los puertos de entrada y salida del módulo *overlap_in*. Se debe notar que la salida del módulo es serial, debido a que la entrada a la FFT debe ser serial también, lo cual supone un problema debido a que en el mismo tiempo en el que llegan 64 nuevas muestras, debe haber ingresado serialmente un bloque de 128 muestras a la FFT, por lo que es necesario que la FFT trabaje al doble de velocidad que el resto del sistema.

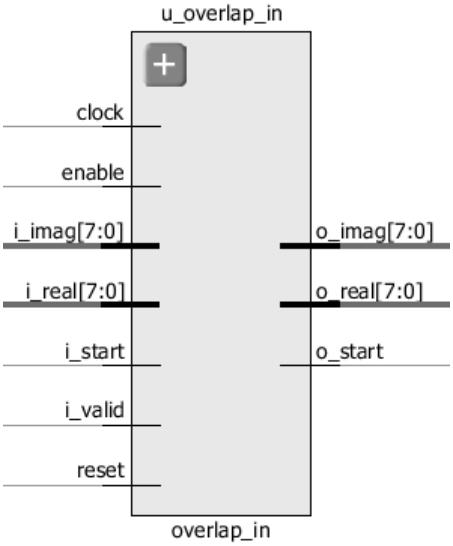


Figura 7.17: *Esquemático del módulo para concatenar los bloques de entrada a la FFT.*

Internamente, el módulo cuenta con 2 buffers (4 en realidad, 2 para la parte real y 2 para la parte imaginaria, pero tienen un comportamiento totalmente análogo). Uno de los buffers es de tamaño 64 y se encarga de almacenar las nuevas muestras que llegan al módulo, almacenando una nueva y desplazando todo el buffer cada vez que lo indica la señal del puerto *i_valid*, la cual es más lenta que la señal de *clock* (la mitad concretamente). El otro buffer, de tamaño 128, es donde se realiza la concatenación de $M-1$ muestras anteriores con L muestras nuevas, y son colocadas serialmente en el puerto de salida, una por ciclo de clock mediante un multiplexor.

Como el clock es el doble de rápido que la señal de valid, en el mismo tiempo en el que se barren las 128 muestras del buffer de salida, se almacenan 64 muestras nuevas en el buffer de entrada. Internamente se utiliza un contador que actualiza el buffer de salida cada 128 ciclos, desplazando las muestras viejas (que se encuentran en la parte inferior del buffer de salida) a la parte superior del buffer de salida (para que sean las primeras en ingresar a la FFT), y colocando las muestras nuevas (que se encuentran en el buffer de entrada) en la parte inferior del buffer de salida. Este mismo contador es el que se utiliza como señal de control en el multiplexor para determinar la muestra de salida actual.

El módulo cuenta también con un puerto de entrada *i_start*, cuyo flanco de bajada debe ocurrir en el momento en que comienzan a ingresar muestras nuevas por primera vez. Debido a que inicialmente debe llenarse el buffer de salida antes de comenzar a extraer serialmente las muestras que ingresarán a la FFT, se recurre al puerto *o_start*, el cual se conecta a la señal de *START* de la FFT y su flanco de bajada tiene lugar recién cuando el

buffer de salida se encuentra listo, logrando así una correcta sincronización para que la FFT comience a funcionar en el momento adecuado.

Una vez realizado el procesamiento en el dominio de la frecuencia, es necesario regresar al dominio del tiempo mediante una IFFT de N puntos, obteniéndose así N muestras de salida, de las cuales se descartan las primeras $M-1$ y se conservan las últimas L, que se corresponden con las respectivas L nuevas muestras que ingresaron al módulo *overlap_in* descripto previamente.

Las muestras de entrada ingresan a la mitad de la velocidad a la que trabajan estos módulos, por lo que las muestras de salida también deberían salir a la mitad de la frecuencia de trabajo de la IFFT, que entrega sus muestras de salida de manera serial. Al descartar la mitad de las muestras de salida, se consigue una tasa neta igual a la de entrada, pero con períodos en lo que no se transmite muestra alguna (las $M-1$ que se descartan) y períodos en los que se transmiten las muestras válidas a gran velocidad (las L que se conservan). Sin embargo, este no es el comportamiento que se busca, puesto que se requiere un flujo continuo de nuevas muestras de salida a la velocidad correspondiente.

Para lograr este comportamiento deseado, internamente se utilizan 2 buffers de tamaño $L = 64$. Cuando una señal externa (conectada al puerto de entrada *i_ready*) indica que comienza la llegada de muestras desde la IFFT (es decir que este puerto está directamente conectado a la señal de *RDY* de la IFFT), se inicia un contador. Cuando este contador llega a 64, se comienzan a almacenar las muestras que ingresan en el buffer de entrada, ignorando de esta forma las primeras $M - 1 = 64$ muestras que entrega la IFFT. Cuando el contador llega a 128, se reinicia la cuenta y se colocan los valores del buffer de entrada en el buffer de salida. Este contador, al igual que el llenado del buffer de entrada, trabaja a la misma velocidad que la IFFT (es decir al doble de lo requerido para las muestras finales de salida). Mientras ingresan muestras y se actualiza el buffer de entrada, se colocan serialmente y a la velocidad adecuada en los puertos de salida las muestras que se encuentran en el buffer de salida. Como se trabaja a la mitad de velocidad, en el tiempo en el que ingresan $N = 128$ muestras desde la IFFT (de las cuales solamente $L = 64$ son válidas), se colocan en los puertos de salida las $L = 64$ muestras válidas que se encuentran almacenadas en el buffer de salida.

Por otra parte, debido a que el ecualizador es Fraccionalmente Espaciado, las 64 muestras de salida deben tomarse intercaladamente, conservando de esta forma sólo 32 muestras. Para lograr esto, simplemente se coloca el

valor correspondiente del buffer de salida sólo cuando el contador es múltiplo de 2, para lo cual basta verificar si el último bit del mismo es 0 ó 1.

En la Fig. 7.18 se pueden observar los puertos de entrada y salida del módulo *save_out*.

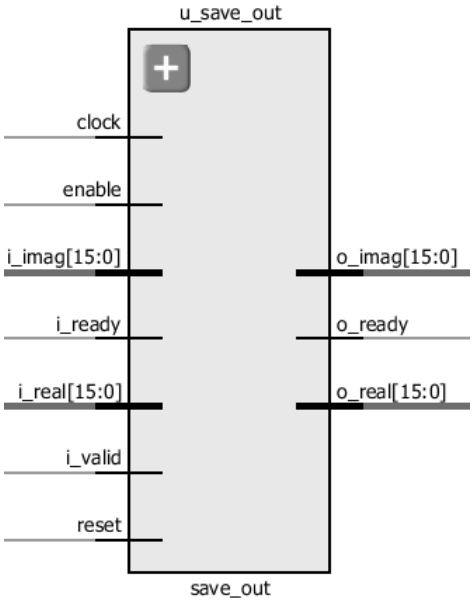


Figura 7.18: *Esquemático del módulo para descartar muestras de salida de la IFFT.*

7.5.3. Producto Complejo

Como el algoritmo trabaja en el dominio de la frecuencia, se deben llevar a cabo multiplicaciones complejas debido al hecho de que las salidas de la FFT son señales complejas. Por lo tanto, es necesario diseñar un módulo que realice dichas multiplicaciones, el cual se utiliza para realizar tanto el filtrado en frecuencia como la adaptación de coeficientes.

Para confeccionar el módulo, es necesario expresar el producto complejo como sumas y multiplicaciones reales, tal como se muestra en la Ec. (7.6), en la que se expresan los números complejos a partir de sus respectivas partes reales (A_r y B_r) e imaginarias (A_i y B_i).

$$(A_r + jA_i) \times (B_r + jB_i) = (A_rB_r - A_iB_i) + j(A_rB_i + A_iB_r) \qquad (7.6)$$

Se observa que para realizar un producto complejo, es necesario llevar a cabo 4 multiplicaciones y 2 sumas reales.

El módulo diseñado es puramente combinacional y lleva a cabo las operaciones de la Ec. (7.6) a partir de las respectivas señales complejas que le

ingresan por sus puertos de entrada, y entrega a su salida las correspondientes partes reales e imaginarias del resultado.

El multiplicador diseñado trabaja en punto fijo, por lo que es necesario pasarle como parámetro las resoluciones de ambos factores (pueden tener resoluciones diferentes) y del resultado. Internamente trabaja en máxima resolución y luego realiza una operación de truncado/saturación de acuerdo a la resolución de salida especificada.

7.5.4. Filtrado en el Dominio de la Frecuencia

Interconectando los módulos de las Secciones 7.5.1, 7.5.2 y 7.5.3, ya es posible efectuar el filtrado en el dominio de la frecuencia.

La interconexión básica consiste en colocar las muestras que llegan desde el resto del sistema (concretamente la salida del canal) en el puerto de entrada del módulo *overlap_in*, el cual se encarga de concatenar adecuadamente los bloques de muestras nuevos con los anteriores, colocándolos adecuadamente a su salida, la cual se encuentra conectada al puerto de entrada de la FFT. La salida de la FFT se conecta a una de las entradas del multiplicador complejo, poniendo en la otra entrada los coeficientes del filtro (en frecuencia), obteniendo así a la salida del multiplicador las muestras en el dominio de la frecuencia ya filtradas. La salida del multiplicador complejo se conecta a la entrada de la IFFT, para pasar nuevamente al dominio del tiempo. Finalmente, la salida de la IFFT se conecta al módulo *save_out*, que se encarga de descartar las muestras correspondientes para entregar a su salida los resultados del filtrado efectuado en el dominio de la frecuencia. Esta interconexión se puede observar en la Fig. 7.19.

Se debe notar que todo el proceso tiene lugar de manera serial debido a que las muestras de entrada y salida del IP Core utilizado para computar las FFT/IFFT son seriales, por lo tanto es necesario serializar las entradas paralelas que le llegan desde el resto del sistema. Como consecuencia de esto, para realizar el filtrado en frecuencia propiamente dicho (es decir el producto por los coeficientes), basta con instanciar un único multiplicador complejo, al cual le ingresan serialmente a una de sus entradas las salidas de la FFT, y en la otra entrada se coloca el coeficiente correspondiente a la muestra actual. Para colocar el coeficiente correcto, se utiliza un multiplexor al cual ingresan todos los coeficientes y se emplea como variable de control del mismo la salida *addr* de la FFT, la cual indica el número de muestra (valor entre 0 y $N - 1 = 127$) que se encuentra a la salida de la FFT en ese momento, garantizando así que la salida número i sea

multiplicada por el coeficiente w_i.

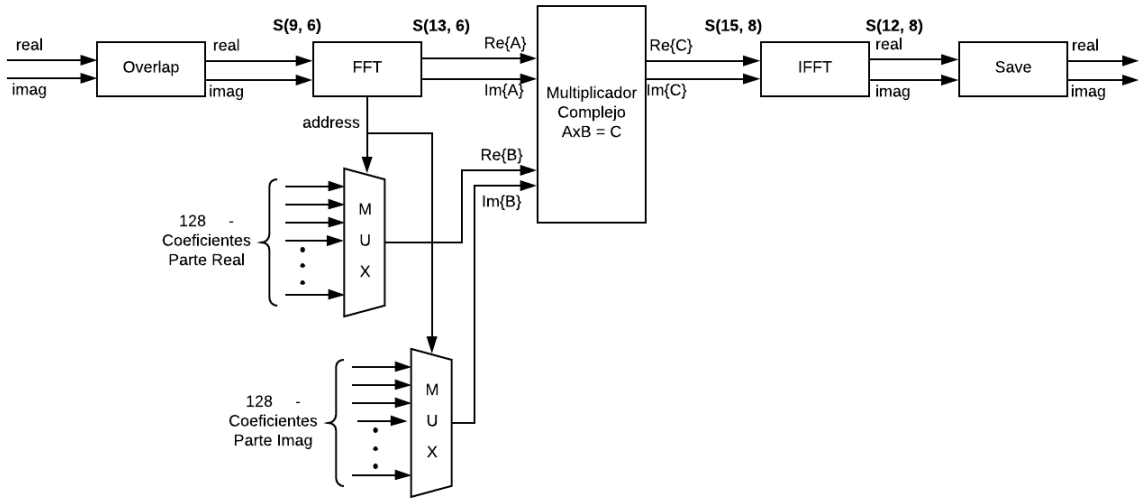


Figura 7.19: *Interconexión de módulos para filtrado en frecuencia.*

Llegado este punto, al realizar el filtrado en el dominio de la frecuencia básicamente se consigue obtener un ecualizador en el dominio de la frecuencia cuyos coeficientes son estáticos (no se adaptan).

Para comprobar el correcto funcionamiento, inicialmente se ponen todos los coeficientes del ecualizador en $1+j0$ (en el dominio de la frecuencia, por ello tienen parte real e imaginaria), de manera tal que no produzcan efecto alguno al multiplicar las muestras de entrada en frecuencia. Al realizar esto, se obtiene a la salida del ecualizador exactamente lo mismo que a su entrada, por lo que el funcionamiento es el adecuado.

La siguiente prueba que se realiza consiste en incorporar los datos de Python en la simulación funcional del diseño a nivel RTL. Lo que se hace es colocar a la entrada del ecualizador las muestras de salida del canal (es decir lo que llegaría desde el resto del sistema) obtenidas del simulador en punto fijo de Python, y se inicializan los coeficientes del ecualizador en los valores ya adaptados, obtenidos también desde Python. Debido a que los coeficientes adaptados corresponden a las mismas muestras que se colocan en el ecualizador a nivel RTL, la salida del mismo debe tomar valores en torno a ± 1, como si ya estuviera adaptada. Este efectivamente es el comportamiento obtenido, tal como se observa en la Fig. 7.20, en la que se muestra la entrada al ecualizador y la salida del mismo, aunque no se corresponden debido a la latencia del sistema (es decir, en cada punto la salida que se ve no es la obtenida a partir de la muestra de entrada en ese instante de tiempo).

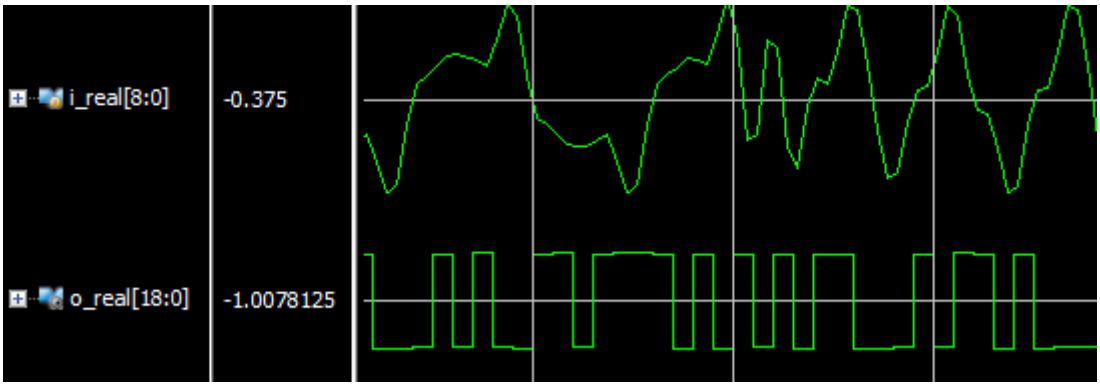


Figura 7.20: *Forma de onda a la salida del ecualizador con coeficientes estáticos adaptados.*

7.5.5. Slicer y Cómputo de Error

Para llevar a cabo la detección final del símbolo recibido, se ingresa la salida del ecualizador a un *slicer* o detector, que simplemente extrae el valor del bit más significativo y lo entrega a su salida. De esta forma, si la salida del ecualizador es 1 (o simplemente positiva en realidad) el símbolo detectado es un 0, mientras que si la salida es -1 (o negativa) el símbolo detectado es un 1. Este comportamiento puede observarse en la Fig. 7.21.

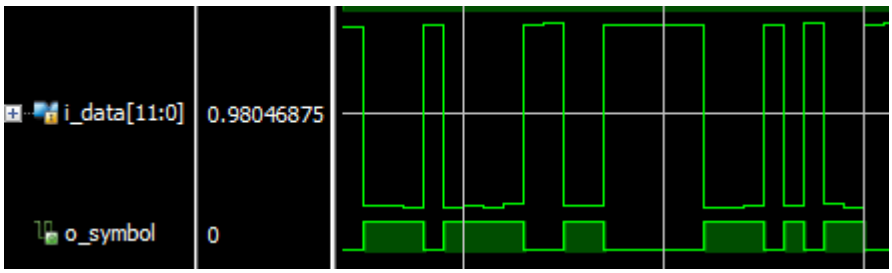


Figura 7.21: *Forma de onda a la salida del slicer.*

Este módulo se encarga también de computar el error entre el símbolo detectado (la salida de este módulo) y la salida del ecualizador (la entrada de este módulo), colocando el resultado en un puerto de salida. Para calcular el error simplemente se efectúa la resta, teniendo en cuenta que deben alinearse las comas para obtener el resultado correcto.

7.5.6. Buffer de Errores

Para realizar la adaptación en el dominio de la frecuencia, primero es necesario almacenar una determinada cantidad de muestras del error computado en el dominio del tiempo, y armar el buffer adecuado para luego hacer una FFT de $N = 128$ puntos.

Debido a que el ecualizador es fraccionalmente espaciado, se cuenta con bloques de error cuyo tamaño es solamente 32 (ya que se obtienen 32 salidas válidas), debiendo llenar con ellos un buffer de tamaño $N = 128$ que luego ingresa a la FFT. Para ello, el buffer de error debe tener la forma de la Ec. (7.7), en la que se puede ver que 64 + 32 posiciones valen siempre 0 y los errores que van ingresando al módulo se colocan en las posiciones 64, 66, 68, ..., 124, 126.

$$\text{buffer\_error} = [\ \underbrace{0, \ldots, 0}_{\text{M-1 = 64 ceros}},\ \underbrace{e_0,\ 0,\ e_1,\ 0,\ \ldots,\ e_{30},\ 0,\ e_{31},\ 0}_{\text{32 muestras de error y 32 ceros}}\ ] \tag{7.7}$$

El módulo se encarga precisamente de confeccionar este buffer de tamaño $N = 128$, inicializando todas las posiciones en 0 y luego colocando los errores en los posiciones adecuadas. Para esto se utiliza un contador cuya cuenta comienza cuando llega una señal de *ready* desde el módulo *save_out*, indicando que comienzan a ingresar muestras de error válidas, ya que el cómputo del error es puramente combinacional, por lo que se tienen muestras de error válidas en el mismo ciclo de clock en el que se tienen muestras de salida válidas en el ecualizador. Luego, una señal de validación proveniente también del módulo *save_out* indica cuando llega cada nueva muestra de error, por lo que se detectan sus flancos de subida para incrementar el contador y colocar en el buffer de entrada el error que se encuentra en el puerto de entrada en ese momento.

Una vez que el contador llega a 32, el mismo se reinicia, se copia el contenido del buffer de entrada en un buffer de salida también de tamaño $N = 128$ y se activa una señal de *ready* indicando que el módulo comienza a entregar muestras válidas a su salida, las cuales son extraídas del buffer de salida al mismo tiempo que el buffer de entrada se llena con los nuevos errores que ingresan.

Este módulo funciona correctamente debido a que en el tiempo en que ingresan 32 nuevas muestras de error, deben haberse colocado a la salida del módulo las 128 posiciones del buffer de salida (entre las cuales se encuentran los 32 valores anteriores del error).

La señal de *ready* de este módulo se utiliza como señal de *START* para la FFT que se encarga de pasar el error al dominio de la frecuencia, para de esta forma realizar luego la adaptación en dicho dominio. Junto con la señal de *ready*, la salida del módulo (extraída del buffer de salida) ingresa a los puertos de entrada de la FFT. La señal de *RDY* de esta FFT indica el momento en el que se comienzan a tener muestras de error válidas en el

dominio de la frecuencia y por lo tanto puede comenzar la adaptación de coeficientes.

7.5.7. Adaptación de Coeficientes

Una vez que se tiene el error en el dominio de la frecuencia, ya es posible comenzar con la adaptación de los $N = M + L - 1 = 128$ coeficientes en el dominio de la frecuencia.

Para realizar la adaptación, se recurre a la Ec. (4.34), que se reescribe a continuación por conveniencia, en la que $W(k+1)$ es el nuevo valor que toman los coeficientes en el dominio de la frecuencia, $W(k)$ el valor actual de los mismos, μ el paso de adaptación, $X^*(k)$ el conjugado de las muestras de entrada en el dominio de la frecuencia, y $E(k)$ el error en el dominio de la frecuencia.

$$W(k+1) = W(k) + 2\mu\, X^*(k)\, E(k) \tag{7.8}$$

La complejidad al momento de implementar la Ec. (7.8) radica en la sincronización entre $X^*(k)$ y $E(k)$, ya que deben emplearse las muestras de entrada correspondientes al error que se está utilizando. Por lo tanto, se debe considerar la latencia del sistema desde el momento en el que se encuentran listas las primeras muestras de entrada en el dominio de la frecuencia, hasta que se encuentran listas las primeras muestras del error en el dominio de la frecuencia.

Para solventar este problema de sincronización, es posible aprovechar las señales de *ready* con las que cuentan todos los módulos utilizados, analizando la latencia entre la señal de RDY de la FFT a la que ingresan las muestras de entrada del ecualizador, y la señal de RDY de la FFT a la que ingresa el error en el dominio del tiempo.

La idea básica consiste en almacenar los sucesivos bloques de 128 muestras de entrada en el dominio de la frecuencia hasta que se detecte la señal de RDY que indica que ya está disponible la primera muestra de error en el dominio de la frecuencia. En ese momento, es posible aplicar la Ec. (7.8) utilizando las primeras muestras de entrada que se han almacenado, y las muestras de error que comienzan a llegar. De esta forma la adaptación ya queda correctamente sincronizada, utilizando a partir de esa sincronización inicial los diferentes bloques de 128 muestras almacenadas y los 128 errores que entregue la FFT correspondiente.

Para almacenar los valores de entrada en el dominio de la frecuencia, se utiliza un buffer de un tamaño que debe determinarse de antemano. A

los fines de conocer la longitud que este buffer debe tener, se recurre a un contador auxiliar cuya cuenta comienza en el momento en que la FFT de entrada activa su señal de *RDY* y se detiene cuando la FFT del error activa su señal de *RDY*, determinando así la cantidad de ciclos de clock transcurridos. Para que este procedimiento funcione adecuadamente, las diferentes señales de *ready* deben propagarse correctamente a lo largo de todo el sistema, que comienza con la FFT de entrada, pasa por el multiplicador complejo, ingresa a la IFFT de salida, sale de la misma, pasa por el buffer de error y finalmente termina ingresando a la FFT del error. Al realizar esta cuenta, se obtiene que el buffer debe tener un tamaño igual a 1140, es decir deben almacenarse 1140 muestras de entrada en el dominio de la frecuencia.

El módulo que realiza la adaptación posee entradas que indican cuándo está lista la primera muestra de entrada en el dominio de la frecuencia (*i_ready_data*) y cuándo lo está la primera señal de error en frecuencia (*i_ready_error*). Cuando llega la señal de *i_ready_data*, comienza a llenarse el buffer de entrada (de tamaño 1140), guardando las nuevas muestras que ingresan y desplazando las ya almacenadas. Por su parte, cuando llega la señal de *i_ready_error* comienza la adaptación de coeficientes, calculando el gradiente utilizando la muestra más antigua del buffer (la que se ubica en la posición 0) y el error que ingresa en ese momento.

Como se puede observar en la Ec. (7.8), las muestras de entrada en el dominio de la frecuencia $X(k)$ deben conjugarse para obtener $X^*(k)$, para lo cual es necesario multiplicar por -1 la parte imaginaria. A los fines de reducir las operaciones necesarias, se aprovecha el hecho de que también es necesario multiplicar dichas muestras por el paso, el cual tiene un valor prefijado. Esto permite multiplicar la parte real de $X(k)$ por el paso normal (μ), y la parte imaginaria de $X(k)$ por el valor negativo del paso ($-\mu$), logrando así la conjugación y la multiplicación por el paso en una misma operación. Una vez obtenido el valor de la operación anterior, es posible ingresarlo al módulo multiplicador complejo de la Sección 7.5.3 junto con el valor del error que ingresa en ese momento al módulo, obteniendo así el valor final que se debe emplear para actualizar los coeficientes.

El valor que se extrae del módulo multiplicador complejo debe utilizarse para actualizar el valor del coeficiente correspondiente al número de muestra utilizada dentro del bloque de $N = 128$. Para determinar cuál es este número y por lo tanto qué coeficiente debe actualizarse, se emplea la señal *i_addr_error*, que está conectada a la señal de *addr* de la FFT encargada de computar la transformada del error. Esta señal indica de manera direc-

ta qué número de coeficiente debe actualizarse con el valor calculado en el momento actual. De esta forma es posible almacenar de manera consecutiva todas las muestras de entrada en el dominio de la frecuencia (sin importar dónde comienza y termina cada bloque de 128), ya que la señal *i_addr_error* se encarga de marcar el inicio y el fin de cada bloque.

Todas las operaciones anteriormente mencionadas se realizan en máxima resolución, aumentando según corresponda la cantidad de bits tanto de la parte entera como de la parte fraccionaria. Los coeficientes actualizados se almacenan temporalmente en máxima resolución, y una vez procesado un bloque de 128 muestras (es decir que ya se han actualizado todos los coeficientes) se los almacena en el vector de coeficientes definitivo, realizándoles previamente un proceso de redondeo y saturación.

Con respecto al paso de adaptación utilizado, el módulo también efectúa el ya explicado cambio de paso (*gear-shift*) luego de que se ha procesado una cierta cantidad de muestras. Para ello, se inicia un contador en el momento en que comienza la adaptación de coeficientes (es decir cuando llega la señal de *i_ready_error*) y se incrementa su valor cada vez que ingresa una nueva muestra. Mientras este contador sea menor a un determinado número, se emplea el paso grueso para la adaptación de coeficientes. Una vez superado ese valor, habiendo pasado ya el tiempo suficiente para que los coeficientes estén razonablemente adaptados, se comienza a utilizar el paso fino.

Además de la adaptación de coeficientes, el módulo se encarga de determinar el coeficiente que debe utilizarse en cada momento para efectuar el filtrado en frecuencia. Para ello, se utiliza la señal *i_addr_data*, conectada a la señal de *addr* de la FFT de las muestras de entrada, como señal de control de un multiplexor al que le ingresan los 128 coeficientes ya adaptados (con parte real e imaginaria), y entrega a su salida, conectada a puertos de salida del módulo, el coeficiente adecuado listo para ser utilizado por el multiplicador que efectúa el filtrado en el dominio de la frecuencia.

En la Fig. 7.22 se puede observar el módulo que se encarga de llevar a cabo la adaptación de coeficientes, con todos los puertos de entrada y salida mencionados anteriormente. Las señales de error (*i_error_real* e *i_error_imag*) ya se encuentran en el dominio de la frecuencia.

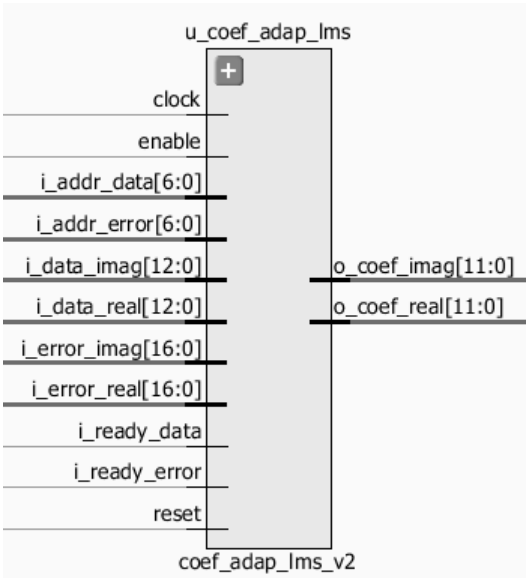


Figura 7.22: *Esquemático del módulo de Adaptación de Coeficientes con algoritmo LMS.*

7.5.8. Interconexión Final de Módulos

A partir de la interconexión de la totalidad de los módulos desarrollados en las subsecciones anteriores es posible confeccionar la implementación final del ecualizador adaptivo en el dominio de la frecuencia. En la Fig. 7.23 es posible apreciar cómo deben conectarse para lograr el comportamiento deseado.

Se observa que las entradas ingresan primero al módulo *overlap_in*, que se encarga de concatenar bloques de muestras nuevas con bloques de muestras antiguas, para luego ingresar de manera serial a la FFT de las muestras de entrada. La salida de esta FFT ingresa al multiplicador complejo junto con los coeficientes, provenientes del módulo que se encarga de realizar la adaptación de los mismos. El resultado del multiplicador ingresa a la IFFT de salida, que a su vez tiene conectadas sus salidas al módulo *save_out* que se encarga de descartar las muestras necesarias y entregar las muestras de salida finales a la tasa adecuada, tomadas de manera intercalada. La salida de este módulo ingresa al *slicer* que se encarga de detectar el símbolo recibido, además de computar el error (en el dominio del tiempo) entre la salida propiamente dicha del ecualizador y el símbolo detectado. Este error ingresa al buffer que se encarga de armar el bloque adecuado para su ingreso a la FFT del error, cuya salida ingresa junto con las muestras de entrada en el dominio de la frecuencia al módulo que se encarga de efectuar la adaptación de coeficientes.

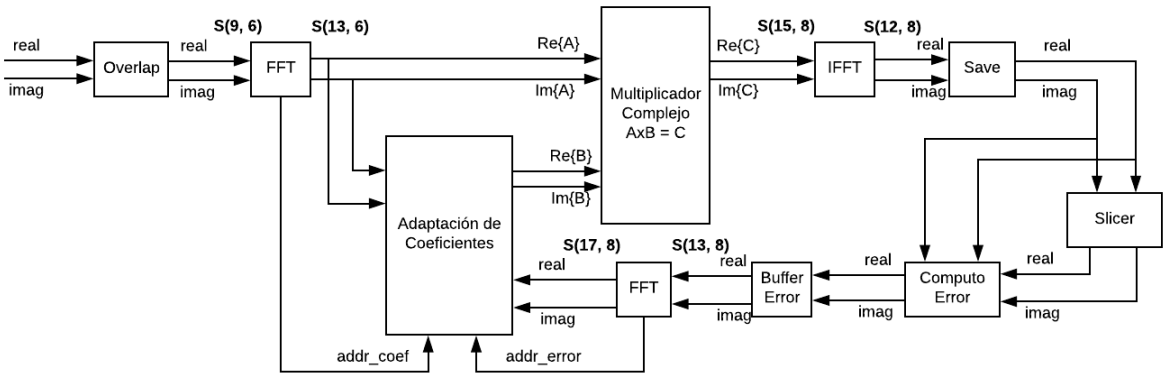


Figura 7.23: *Interconexión del Ecualizador Adaptivo en el Dominio de la Frecuencia.*

Por simplicidad no se han colocado en el esquema de la Fig. 7.23 las diferentes señales de *ready* y de *validación*, las cuales entran y salen de los sucesivos módulos, permitiendo así una adecuada sincronización de todo el sistema para que funcione de manera correcta. El esquemático del módulo con sus puertos de entrada y salida puede observarse en la Fig. 7.24.

La entrada *i_channel* es la salida del canal serializada sincronizadamente y se utiliza cuando el módulo no se encuentra habilitado. Las entradas *i_real* e *i_imag* son las muestras de entrada del ecualizador y se encuentran conectadas a la salida del canal (internamente son las entradas del módulo *overlap_in*). El puerto *i_error_real* corresponde al error en el dominio del tiempo entre la salida del ecualizador y el símbolo detectado, calculado por conveniencia fuera del módulo. La señal *i_start* indica cuándo comienzan a ingresar muestras válidas y se encuentra conectada internamente a la señal de *start* del módulo *overlap_in*. Por último, las entradas *i_shift_fft*, *i_shift_fft_error* e *i_shift_ifft* se utilizan para realizar escalamientos en las FFT/IFFT para garantizar que no haya overflow durante su cómputo. Su valor óptimo se determina previamente mediante simulaciones y se configura a nivel de síntesis desde el Register File. Con respecto a las salidas del módulo, *o_real* es la salida propiamente dicha del ecualizador, mientras que las otras tres salidas que se observan se utilizan para el logueo de los coeficientes.

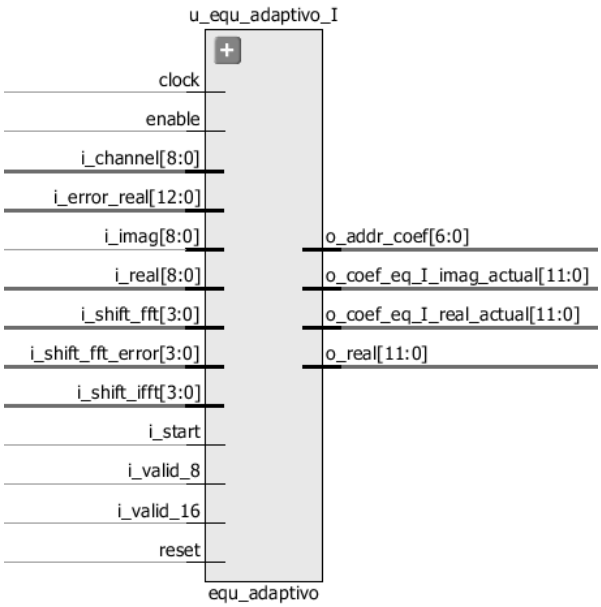
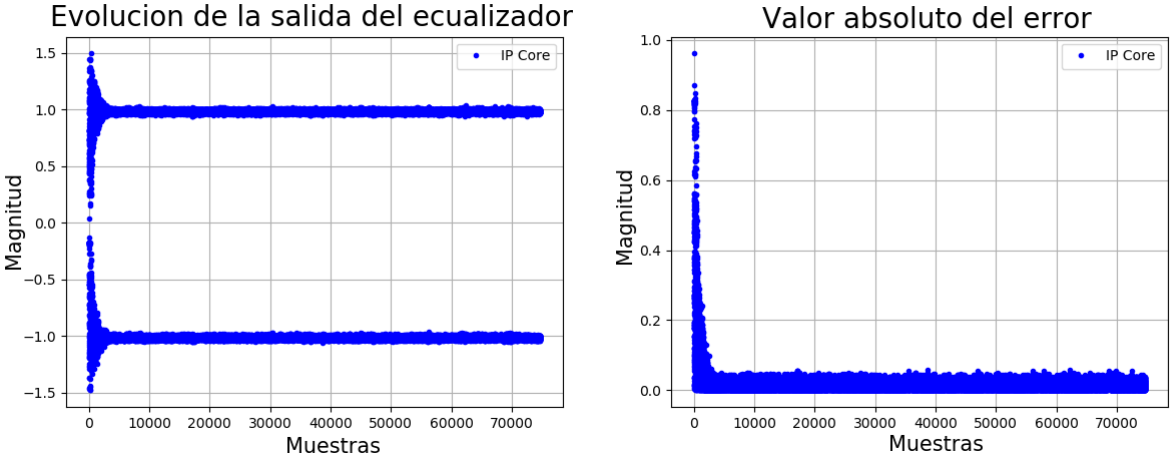


Figura 7.24: *Esquemático del módulo Ecualizador Adaptivo.*

Llegado este punto, ya se cuenta con el ecualizador implementado en su totalidad, por lo que es posible proceder a comprobar su correcto funcionamiento a nivel de simulación funcional. Para ello, se generan desde el simulador en punto fijo de Python los estímulos que ingresan a los puertos de entrada del ecualizador. De esta forma, las muestras de entrada ya han pasado por el codificador, el filtro transmisor y el canal, permitiendo analizar el funcionamiento del ecualizador como un módulo aislado, sin necesidad de conectarlo todavía al resto del sistema. Al realizar estas pruebas, se obtienen los resultados de la Fig. 7.25, en la que se aprecia cómo la salida converge a ± 1 mientras que el error tiende a 0, con una rápida velocidad de convergencia debido al valor del paso.



(a) *Salida del ecualizador.* (b) *Error a la salida del ecualizador.*

Figura 7.25: *Resultados del simulador a nivel funcional con paso* $\mu = 0{,}006$.

Capítulo 8

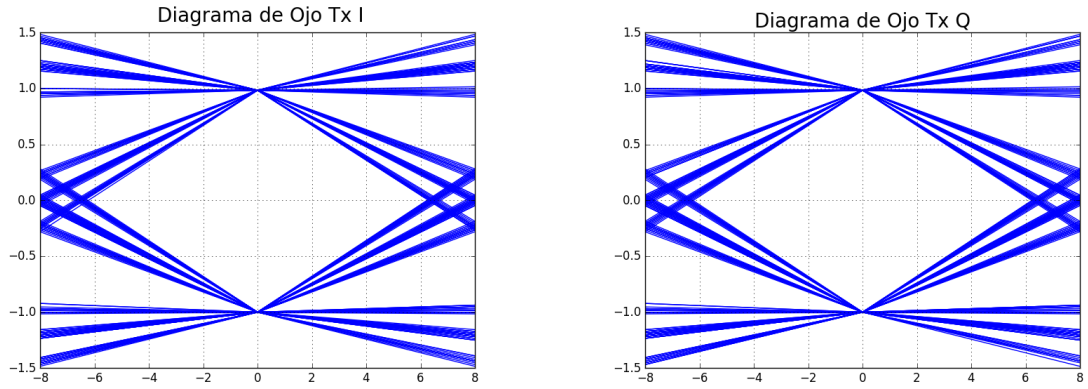
Resultados

Resumen

En este capítulo se presentan los resultados obtenidos a partir de los datos extraídos del sistema implementado en FPGA, realizando diferentes gráficas de interés y trazando las respectivas curvas de BER, comparándolas con las de simulación. Además, se analizan los recursos utilizados por los diferentes módulos del sistema.

8.1. Extracción de datos de la FPGA

Una vez finalizada la implementación y ya habiendo bajado el proyecto a la FPGA, se deben realizar todas las pruebas necesarias para comprobar su correcto funcionamiento.



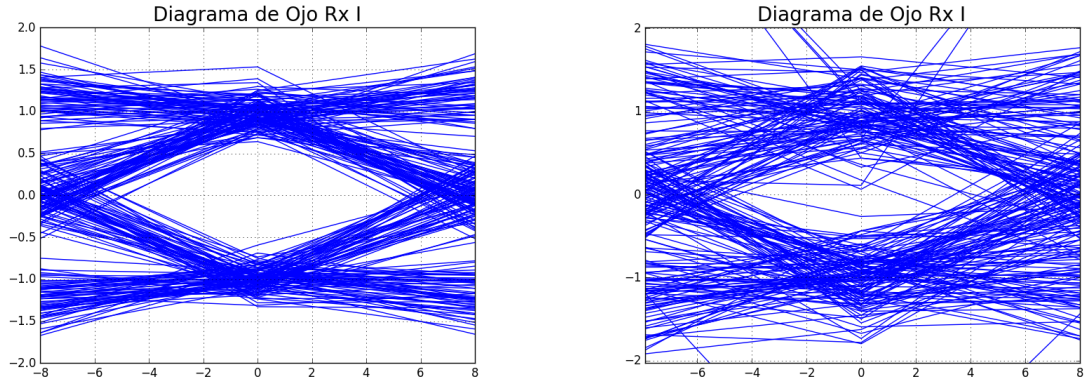
(a) *Filtro transmisor I.* (b) *Filtro transmisor Q.*

Figura 8.1: *Diagrama de Ojo a la salida del filtro transmisor en la FPGA.*

Inicialmente, se verifica el transmisor implementado (PRBS, codificador DQPSK y filtro transmisor), como así también el Register File y el sistema de logueo de datos. Para ello, se envían los comandos necesarios para cargar

los coeficientes del filtro transmisor y habilitarlo luego junto con la PRBS y el encoder. Posteriormente se extraen de la placa los datos a la salida del filtro transmisor (tanto la parte real como la parte imaginaria de la modulación DQPSK) y a partir de ellos se obtienen los diagramas de ojo de la Fig. 8.1.

Luego se incorpora ruido al sistema y se grafica nuevamente el diagrama de ojo a la salida del canal. Concretamente, se asigna una $SNR = 13dB$ y se obtienen los resultados de la Fig. 8.2a, en la que se puede observar una forma similar a la de la Fig. 8.1 pero con mayores niveles de ruido. A medida que se disminuye la SNR, la apertura es cada vez menor (como se ve en la Fig. 8.2b) y por lo tanto la detección es menos precisa, incrementándose la cantidad de errores contabilizados.



(a) *SNR = 13dB.* (b) *SNR = 7dB.*

Figura 8.2: *Diagrama de Ojo a la salida del canal con diferentes valores de SNR.*

Habiendo comprobado el adecuado funcionamiento de estas etapas, el próximo paso consiste en garantizar que los niveles de ruido del sistema se encuentren correctamente calibrados, de manera similar a lo realizado en la etapa de diseño. Para ello, la única forma de tener una referencia indefectiblemente correcta es realizando comparaciones con respecto a la curva teórica de la modulación utilizada. Si los niveles de ruido se encuentran correctamente calibrados, tanto la curva teórica, como la simulada y la extraída de la FPGA deben coincidir.

Como se observa en la Fig. 8.3, todas las curvas coinciden de manera satisfactoria. Se ve cómo a medida que se incrementa la SNR, la curva de la FPGA comienza a separarse ligeramente de la curva teórica, lo cual es razonable debido a que los errores de cuantización toman mayor relevancia. Otro aspecto a destacar es la gran ventaja que brinda la implementación del sistema en FPGA, permitiendo llegar hasta valores de Bit Error Rate

mucho menores, prácticamente imposibles de alcanzar en simulación (o al menos no en un tiempo razonable).

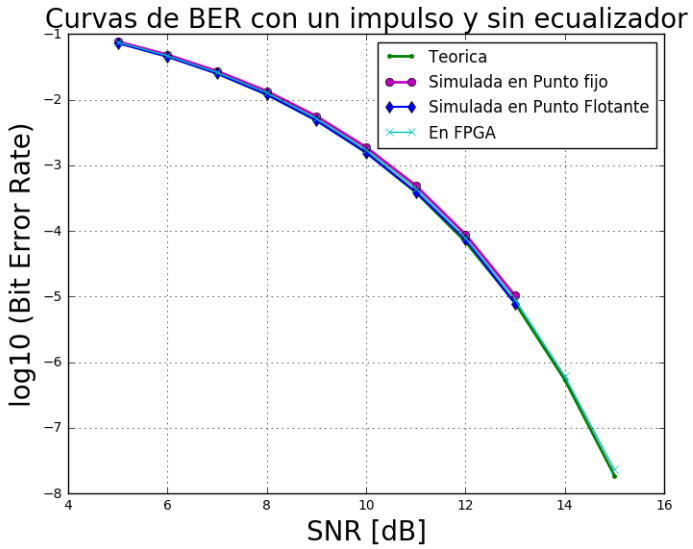


Figura 8.3: *Comparación de Curvas de BER sin canal y sin ecualizador.*

Con el sistema correctamente calibrado, es posible comenzar con las pruebas de funcionamiento del ecualizador utilizando diferentes canales.

8.1.1. Impulso Unitario como Canal

La primera prueba se lleva a cabo utilizando un impulso unitario como canal de transmisión. Bajo estas condiciones, los coeficientes del ecualizador no deberían sufrir modificaciones, debido a que se los inicializa como un impulso y por lo tanto no deberían adaptarse.

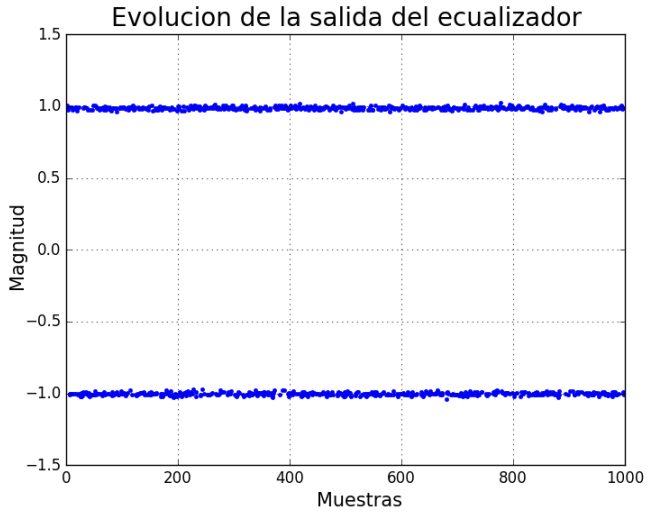


Figura 8.4: *Salida del ecualizador con un impulso unitario como canal.*

Primero se grafica en el tiempo la salida real del ecualizador, pudiéndose ver en la Fig. 8.4 que converge a ± 1 como es de esperarse.

En concordancia con la salida obtenida, al graficar el perfil de los coeficientes del ecualizador se obtiene un impulso unitario. Esto puede verse en la Fig. 8.5, en la que los coeficientes se adaptan ligeramente a valores cercanos a cero, debido a los diferentes errores de cuantización y en el cómputo de las FFT/IFFT.

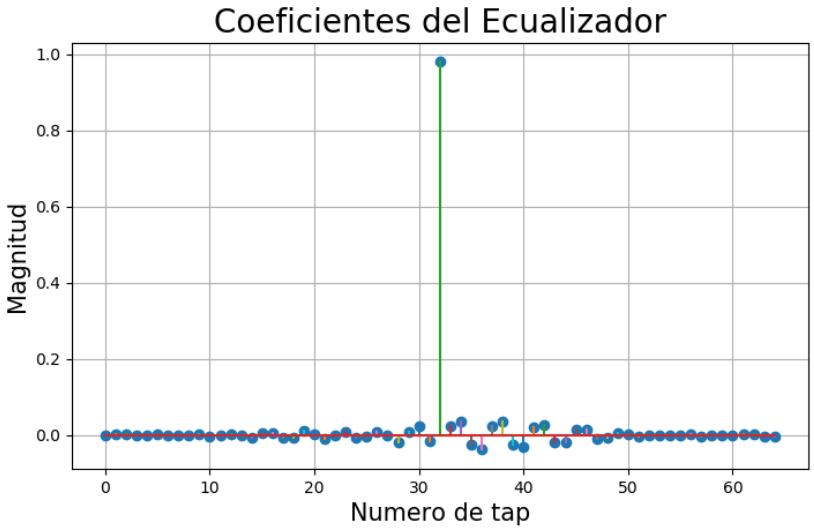
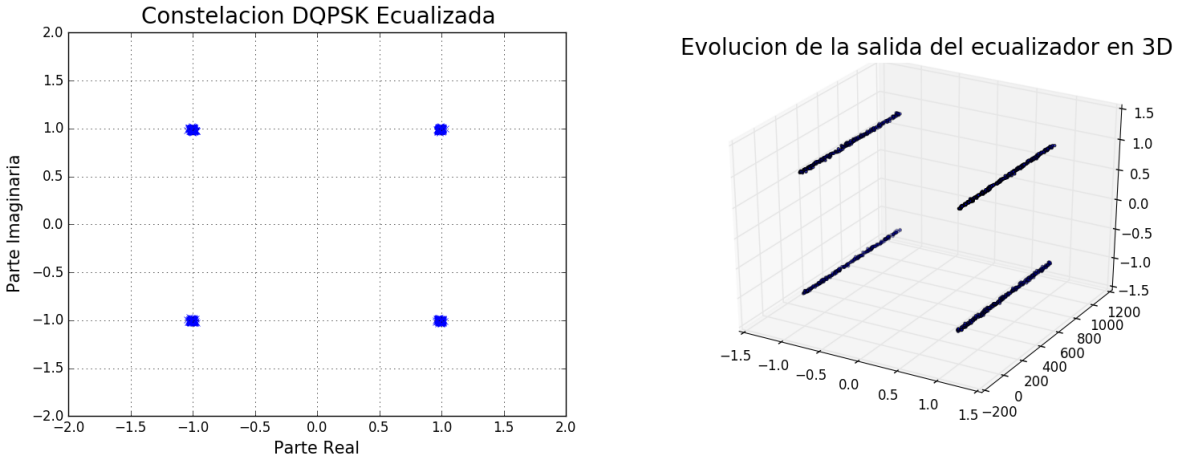


Figura 8.5: *Perfil de los coeficientes del ecualizador con un impulso unitario como canal.*

En la Fig. 8.6 se analiza también la constelación a la salida del ecualizador y su evolución en el tiempo.



(a) *Constelación ecualizada.* (b) *Evolución en el tiempo de la constelación.*

Figura 8.6: *Constelación ecualizada con un impulso como canal.*

Finalmente, se extraen de la FPGA los valores del error entre la salida

del ecualizador y el símbolo detectado, graficando su logaritmo en base 10 en la Fig. 8.7.

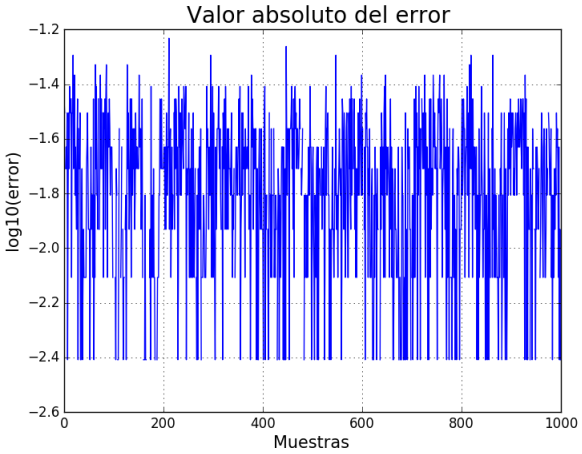


Figura 8.7: *Error a la salida del ecualizador con un impulso unitario como canal.*

8.1.2. Canal de Prueba I

Comprobado el correcto funcionamiento utilizando un impulso unitario, se procede a realizar pruebas con un nuevo canal cuyos coeficientes son *[0.04, -0.05, 0.07, -0.21, -0.5, 0.72, 0.36, 0, 0.21, 0.03, 0.07]* y cuya respuesta en frecuencia se muestra en la Fig. 8.8. Este es el canal que se ha utilizado durante el desarrollo de toda la etapa de diseño.

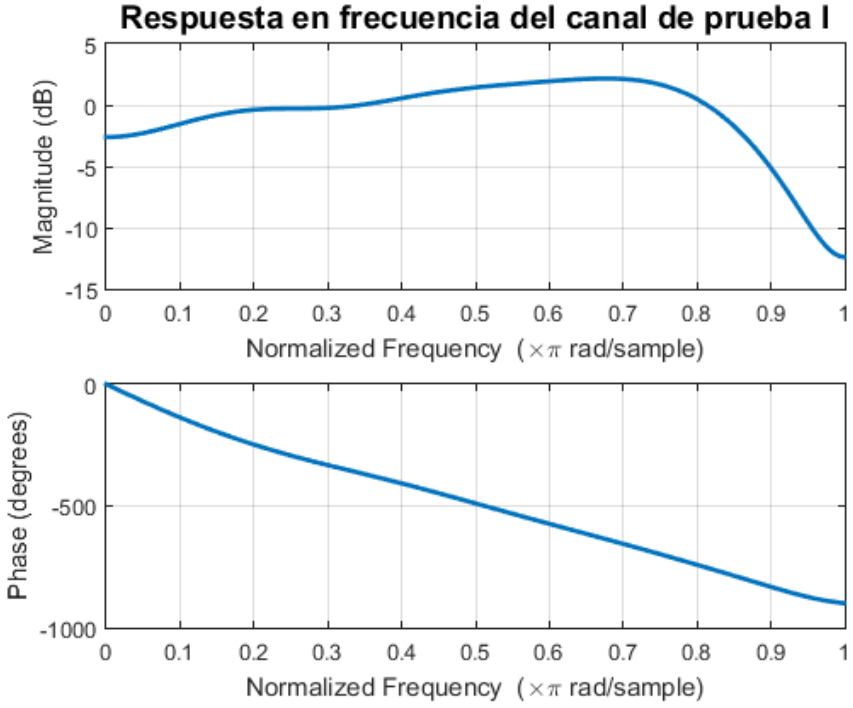


Figura 8.8: *Respuesta en frecuencia del canal de prueba I.*

Nuevamente se grafica la salida del ecualizador, verificando que converge satisfactoriamente a ± 1.

Figura 8.9: *Salida del ecualizador con el canal de prueba I.*

Para comprobar que el ecualizador se comporta de la manera esperada, se extraen los valores de los coeficientes y se grafica su perfil en la Fig. 8.10. Los mismos se extraen en diferentes momentos y se verifica que no se modifican, debido a que el ecualizador ya se encuentra adaptado.

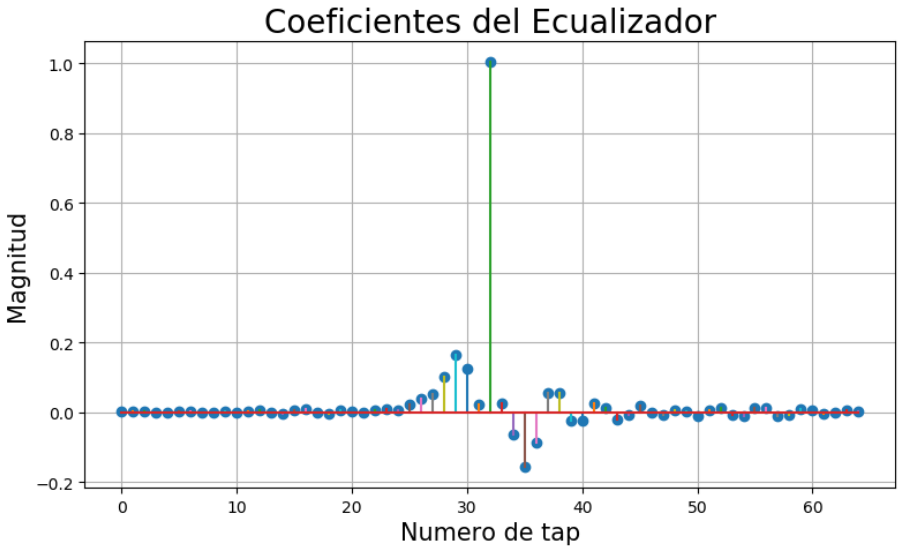


Figura 8.10: *Perfil de los coeficientes del ecualizador adaptado con canal de prueba I.*

A partir del valor de los coeficientes, es posible graficar su respuesta en frecuencia, junto con la respuesta del canal y la respuesta conjunta de ambos. En la Fig. 8.11 puede verse cómo para frecuencias bajas, el

ecualizador compensa correctamente los efectos del canal y por ello la salida del ecualizador converge adecuadamente a ± 1.

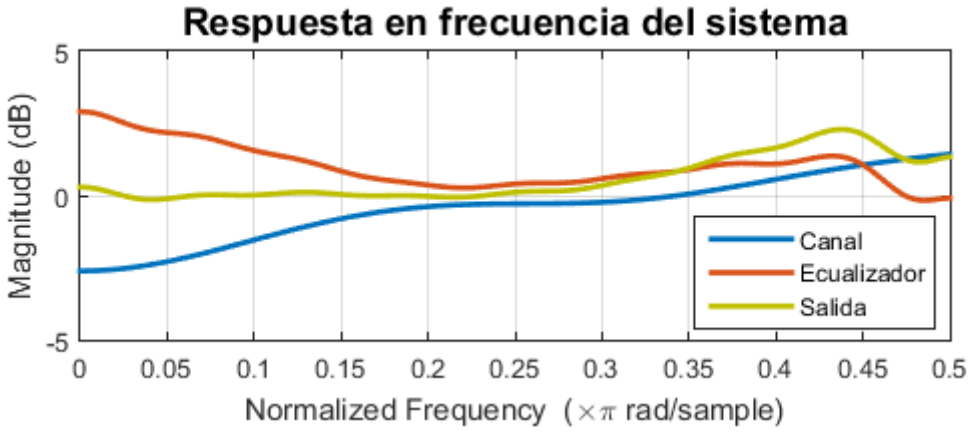
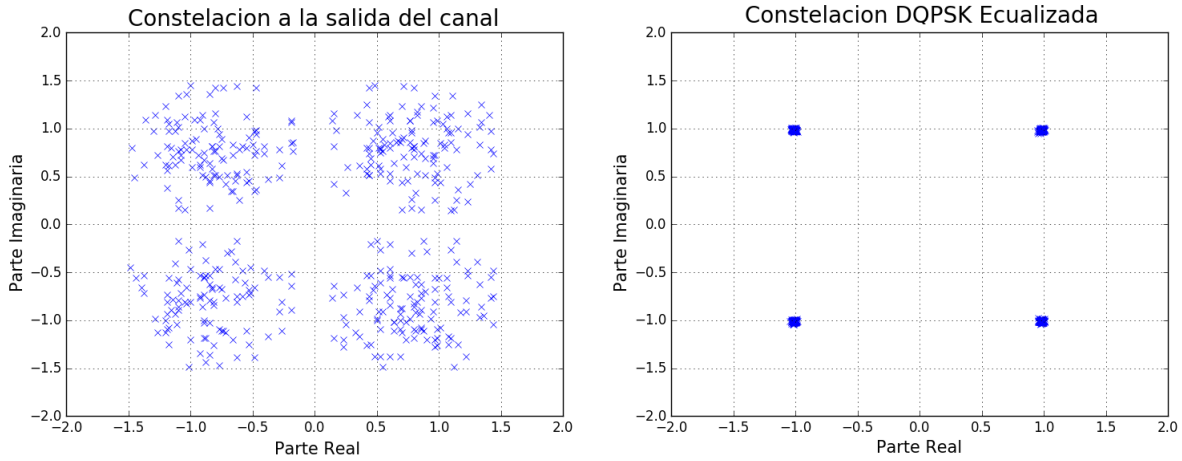


Figura 8.11: *Respuesta en frecuencia del canal, del ecualizador y de todo el sistema.*

El ecualizador solamente se adapta para frecuencias bajas debido a que en su entrada no le ingresan componentes de alta frecuencia y por lo tanto no puede determinar cómo compensar la respuesta del canal más allá de una cierta frecuencia.

De manera análoga al caso del impulso unitario, en la Fig. 8.12 se muestra la constelación que queda conformada a la salida del ecualizador, contrastándola con la constelación no ecualizada que ingresa a él.



(a) *No ecualizada.* (b) *Ecualizada.*

Figura 8.12: *Constelación antes y después del ecualizador, con el canal de prueba I.*

Al graficar el error en la Fig. 8.13, se observa que es ligeramente superior al caso con el impulso unitario como canal.

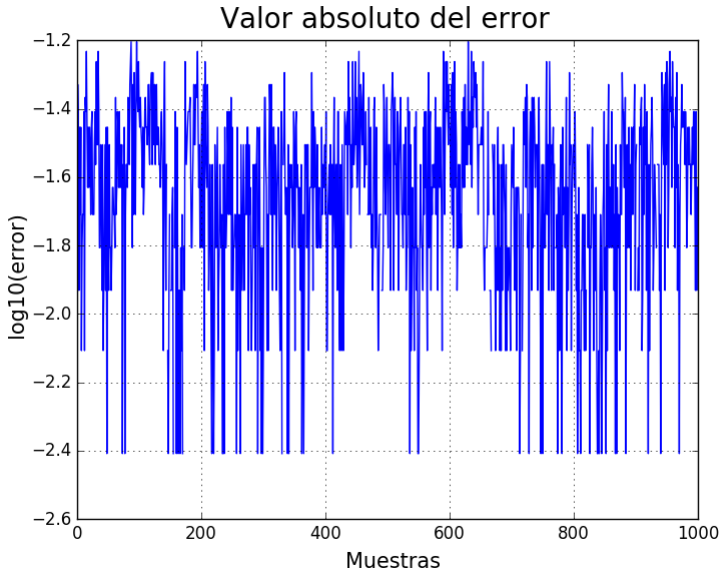
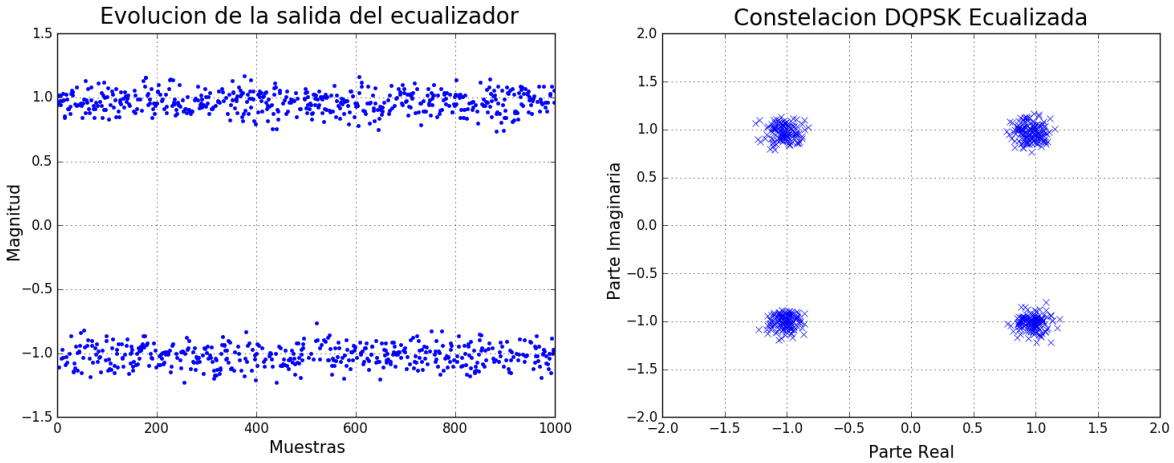


Figura 8.13: *Error a la salida del ecualizador con el canal de prueba I.*

Los resultados obtenidos hasta el momento corresponden a un funcionamiento del sistema sin ruido, es decir habilitando solamente el filtro FIR que representa la respuesta del canal. Por ello luego se agrega ruido al sistema, comenzando con $SNR = 18dB$. La salida y la constelación obtenidas con este nivel de SNR se observan en la Fig. 8.14, en la que se aprecia cómo la salida continúa convergiendo a ± 1 y la constelación sigue centrada en los valores correctos, pero los puntos se encuentran más dispersos.



(a) *Salida del ecualizador.* (b) *Constelación ecualizada.*

Figura 8.14: *Funcionamiento del ecualizador con SNR = 18dB.*

En concordancia con los resultados de la Fig. 8.14, el error también se estabiliza en un valor mayor debido a la presencia del ruido, tal como puede observarse en la Fig. 8.15.

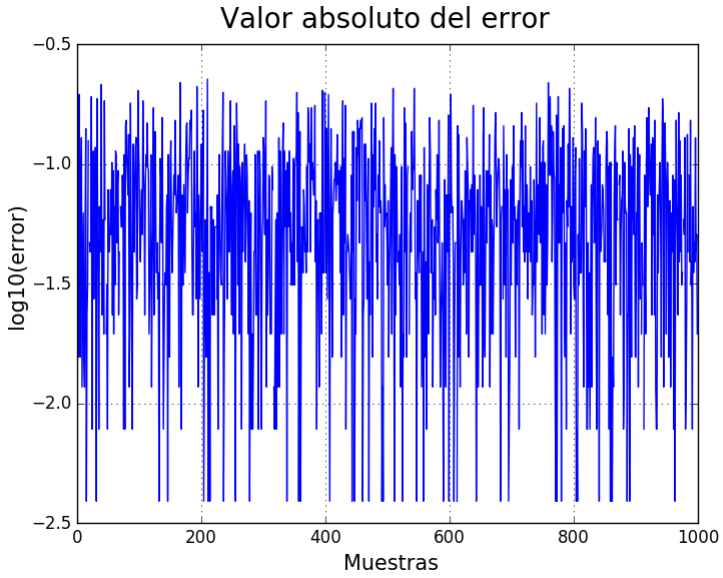
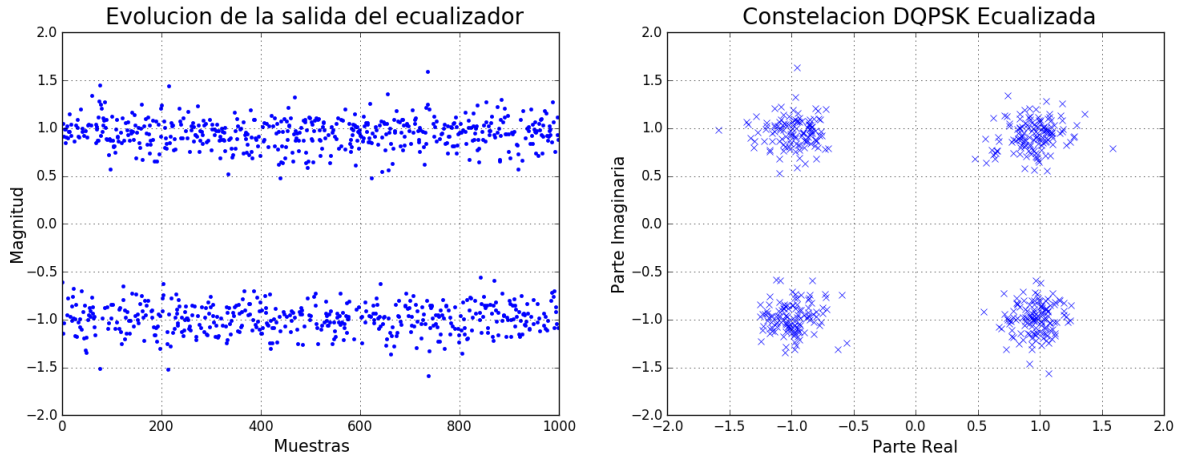


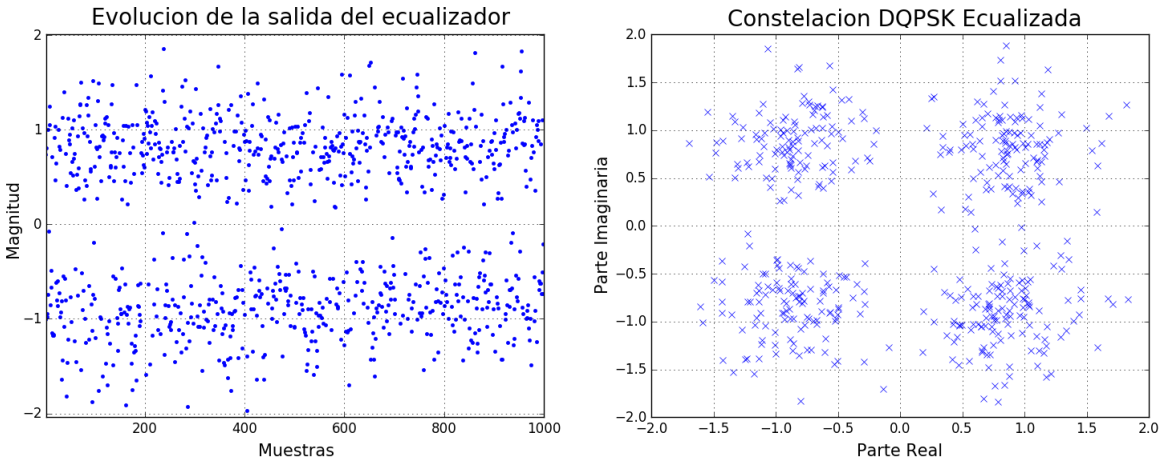
Figura 8.15: *Error a la salida del ecualizador con el canal de prueba I y SNR = 18dB.*

Si se continúa disminuyendo el nivel de SNR, el ecualizador sigue funcionando de la manera esperada pero con un error cada vez mayor y por lo tanto con una constelación cada vez más dispersa. Este comportamiento se observa en las Fig. 8.16 y 8.17, en las que se trabaja con $SNR = 12dB$ y $SNR = 5dB$ respectivamente.



(a) *Salida del ecualizador.* (b) *Constelación ecualizada.*

Figura 8.16: *Funcionamiento del ecualizador con SNR = 12dB.*



(a) *Salida del ecualizador.* (b) *Constelación ecualizada.*

Figura 8.17: *Funcionamiento del ecualizador con SNR = 5dB.*

8.1.3. Canal de Prueba II

A los fines de continuar explotando las capacidades adaptivas del ecualizador, se realizan pruebas utilizando otro canal para verificar una vez más su correcta adaptación.

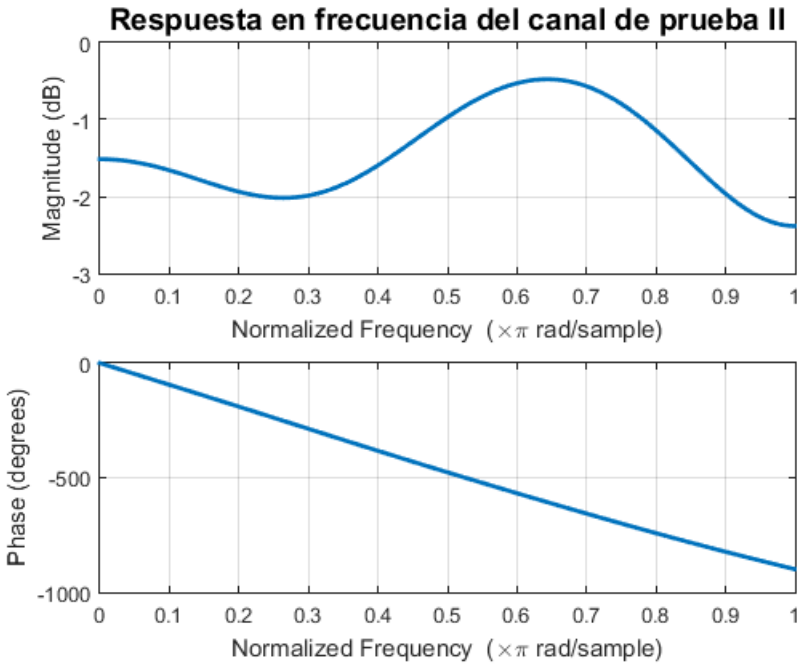


Figura 8.18: *Respuesta en frecuencia del canal de prueba II.*

Los coeficientes del nuevo canal a utilizar son *[0.0, 0.0, 0.02, 0.05, -0.2, 0.8, 0.2, -0.05, 0.02, 0.0, 0.0]* y su respuesta en frecuencia se muestra en la Fig. 8.18, en la que se puede ver que es un canal que atenúa unos cuantos dB para todas las frecuencias.

Al loguear la salida del ecualizador, se comprueba que converge satisfactoriamente a ± 1, tal como se ve en la Fig. 8.19.

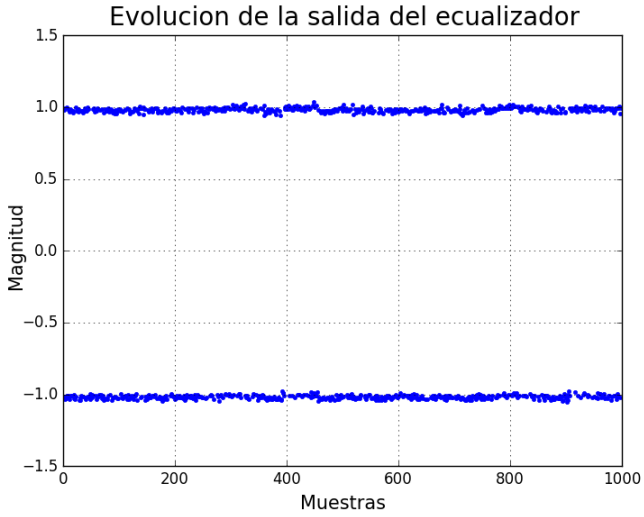
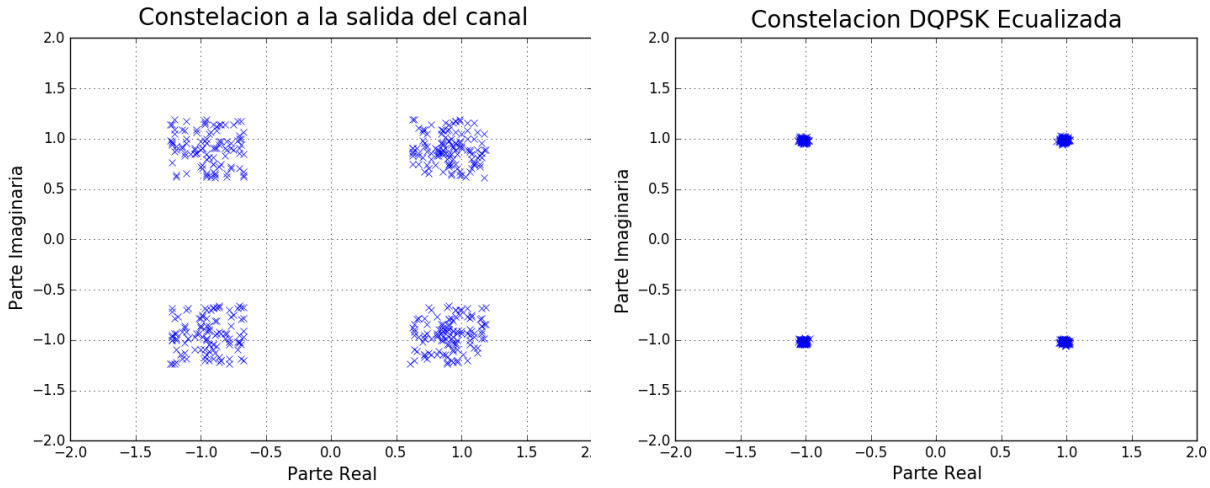


Figura 8.19: *Salida del ecualizador con el canal de prueba II.*

Al igual que en el caso anterior, en la Fig. 8.20 se analiza la constelación no ecualizada (a la salida del canal) y se la contrasta con la constelación ya ecualizada.



(a) *No ecualizada.* (b) *Ecualizada.*

Figura 8.20: *Constelación antes y después del ecualizador, con el canal de prueba II.*

A los fines de comprobar la adaptación de los coeficientes del ecualizador, se realiza un logueo de los mismos y se obtiene el perfil de la Fig. 8.21.

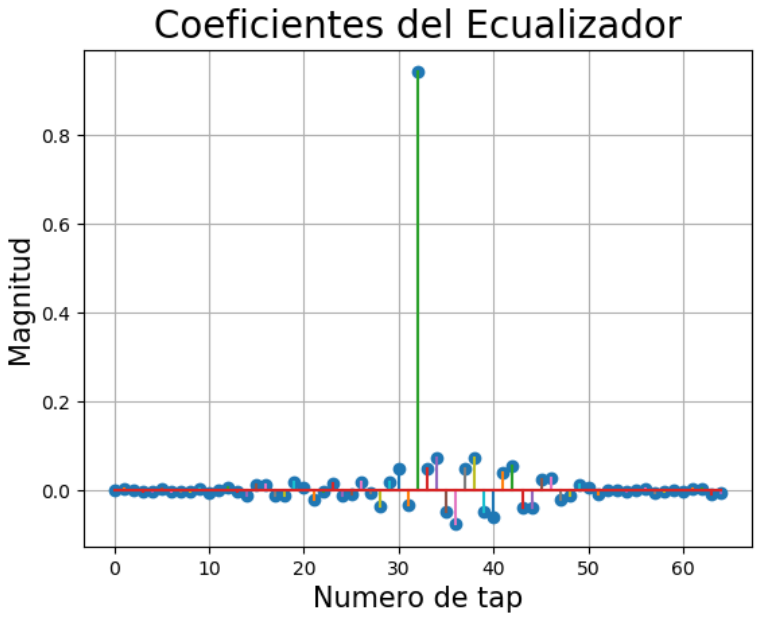
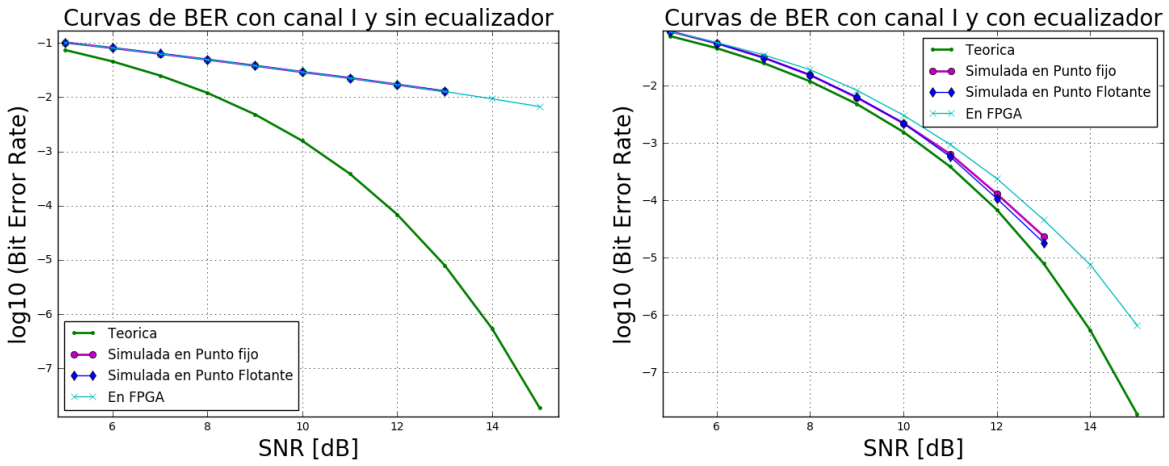


Figura 8.21: *Perfil de los coeficientes del ecualizador adaptado con canal de prueba II.*

8.2. Curvas de BER

Para determinar el desempeño del sistema utilizando el ecualizador implementado, se extraen de la FPGA los datos necesarios para trazar las curvas de BER para diferentes valores de SNR, contrastándolos con los valores obtenidos a partir de simulaciones tanto en punto fijo como en punto flotante. Para ello, se configuran valores sucesivos de SNR entre 5dB y 15dB, y para cada caso se extrae del módulo contador de BER la cantidad de bits procesados y la cantidad de bits contabilizados, calculando a partir de estos valores el Bit Error Rate. Este proceso se lleva a cabo para ambos canales de prueba.

En la Fig. 8.22 se muestran los resultados obtenidos empleando el canal de prueba I. Se puede apreciar que la tasa de errores disminuye notablemente gracias a la presencia del ecualizador, logrando acercarse a la curva teórica (sin canal).



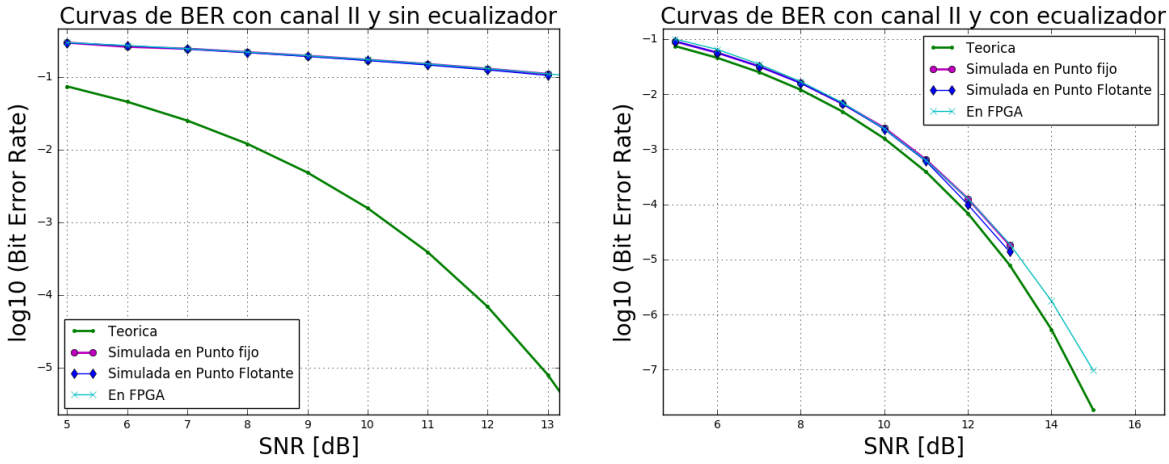
(a) *Sin ecualizador.* (b) *Con ecualizador.*

Figura 8.22: *Curvas de BER utilizando el canal de prueba I.*

Se observa también en la Fig. 8.22b que la curva extraída de la FPGA comienza a separarse de las curvas simuladas (y de la teórica) a medida que se incrementa la SNR del sistema. Este comportamiento se debe a las imprecisiones, normalizaciones y posibles saturaciones en el cómputo de las FFT/IFFT efectuados por el IP Core utilizado, ya que el mismo no fue modelado en el simulador y su funcionamiento no es idéntico al de la función *fft* de NumPy utilizada en el simulador de Python. Debido a que el IP Core es utilizado como una *caja negra*, no es posible controlar las resoluciones de las operaciones internas que realiza, lo cual provoca que ante ciertos canales el comportamiento no sea totalmente análogo a las simulaciones efectuadas.

De manera similar al caso anterior, en la Fig. 8.23 se traza la curva de BER obtenida a partir de los datos de la FPGA utilizando el canal de prueba II. Se puede observar una vez más que el desempeño mejora significativamente al incorporar el ecualizador al sistema, consiguiendo aproximarse a la curva teórica aún más que en el caso anterior.

Se puede apreciar además que en este caso no se produce una separación de la curva obtenida de la FPGA con respecto a la teórica o a las simuladas. Esto se debe a que en este caso los niveles de señal y la respuesta del canal no generan problemas significativos en el cómputo de las FFT/IFFT..



(a) *Sin ecualizador.* (b) *Con ecualizador.*

Figura 8.23: *Curvas de BER utilizando el canal de prueba II.*

8.3. Recursos Utilizados

Finalmente se analizan los recursos utilizados por la totalidad del sistema implementado, los cuales pueden observarse en la Tabla 8.1. Se aprecia que el recurso más utilizado son los bloques DSP, los cuales se encargan de realizar ciertas operaciones (multiplicaciones principalmente) de manera eficiente. Debido a la gran cantidad de multiplicaciones que se llevan a cabo tanto en el ecualizador como en el resto del sistema, es lógico el elevado consumo de este recurso.

Recurso	**Utilización**	**% de utilización**
Flip Flop	64571	16
LUT	53166	26
Memory LUT	3217	5
DSP	492	59
BRAM	60	13

Tabla 8.1: Recursos utilizados por el sistema implementado.

Los recursos utilizados son considerables debido no sólo a la implementación del ecualizador, sino también al paralelismo de todas las etapas anteriores, en las que se multiplica el hardware utilizado con respecto a una implementación serial. Por ello, resulta interesante determinar también cuánto ocupa cada uno de los módulos con respecto a la totalidad de los recursos utilizados, para lo cual se recurre al número de celdas usados por cada módulo de manera individual, teniendo en cuenta que una celda

puede ser desde una simple LUT hasta un bloque de BRAM o DSP. A partir del reporte de síntesis brindado por la herramienta, se extrae el número de celdas de cada módulo y se confecciona el gráfico de la Fig. 8.24.

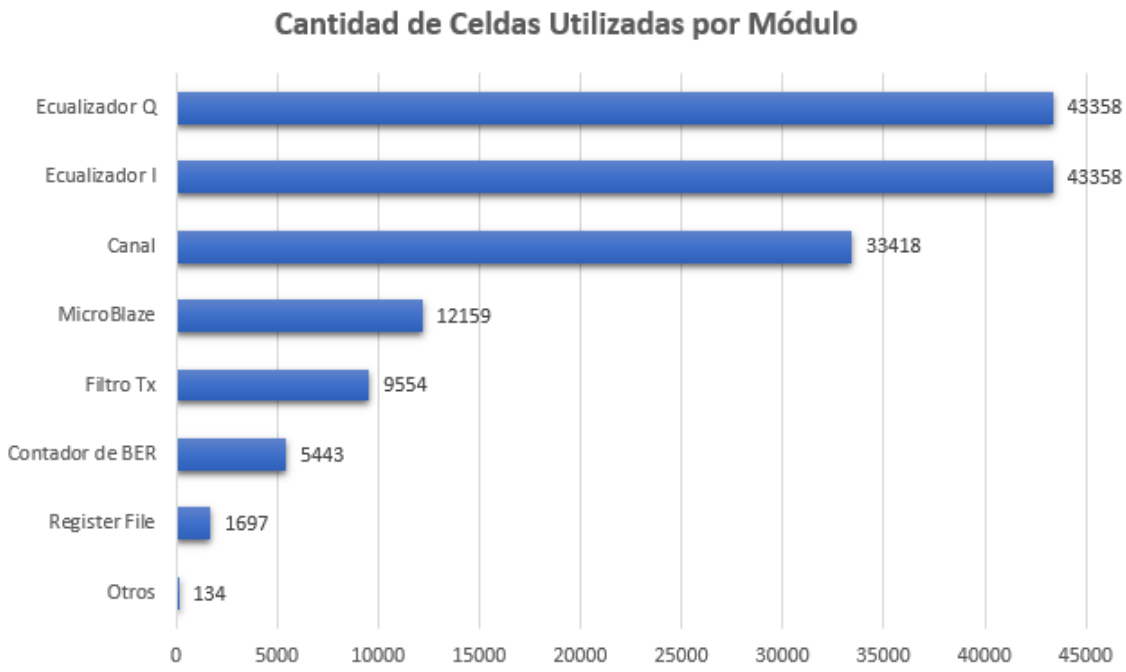


Figura 8.24: *Cantidad de celdas utilizadas por los diferentes módulos del sistema.*

Como es de esperarse se puede observar que el ecualizador utiliza la mayor cantidad de recursos, debido principalmente a las FFT/IFFT y a la gran cantidad de muestras de entrada que deben almacenarse para la correcta sincronización al momento de adaptar los coeficientes. Luego del ecualizador, el módulo que más recursos necesita es el canal. Esto se debe a su implementación en paralelo, en la que se requieren 16 módulos en paralelo, cada uno con sus respectivos Generadores de Ruido Gaussiano, los cuales son los que más consumen en el canal.

También resulta de interés analizar la distribución de recursos dentro de los propios módulos que forman parte del ecualizador, en virtud de lo cual se realiza el gráfico de la Fig. 8.25.

Se puede observar que la mayor utilización se encuentra en la adaptación de coeficientes, debido por un lado a la gran cantidad de muestras de entrada en el dominio de la frecuencia que se deben almacenar, lo cual implica utilizar un gran número de registros. Además, para actualizar el valor de los coeficientes se llevan a cabo una serie de multiplicaciones y sumas manteniendo la máxima resolución, lo cual lleva a emplear una elevada cantidad de bits a los fines de minimizar el arrastre de errores de cuantización al efectuar las diferentes operaciones.

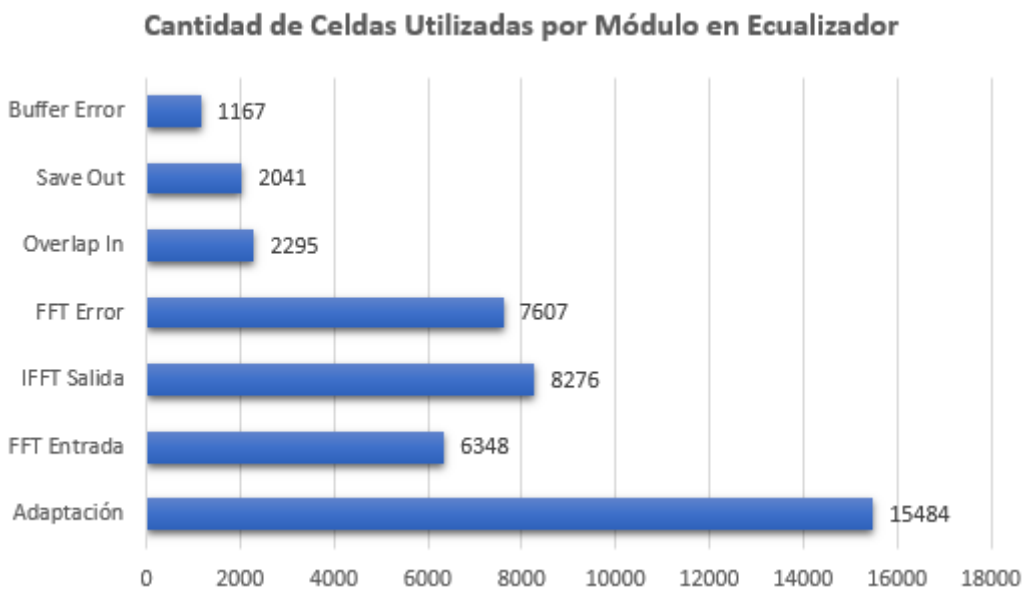


Figura 8.25: *Cantidad de celdas utilizadas por los diferentes módulos del ecualizador.*

Por otra parte, en la Fig. 8.25 se aprecia también el elevado número de celdas utilizado por el IP Core que computa las FFT/IFFT. Cada instancia necesita una cantidad de celdas diferentes debido a que trabajan con distinto número de bits de entrada y salida. Así, la FFT de entrada trabaja con 9 bits de entrada y 13 de salida, por lo que es la que menos celdas requiere. La IFFT de salida utiliza 15 bits de entrada y 19 de salida, siendo así la de mayor utilización. Por último, la FFT que computa la transformada del error trabaja con 13 bits de entrada y 17 de salida, por ello tiene un consumo de recursos intermedio con respecto a las otras dos instancias del IP Core.

Por último, resulta interesante mostrar cuánto ocupa en el diseño implementado cada uno de los diferentes módulos instanciados en la FPGA. Esto puede observarse en la Fig. 8.26, en la que se aprecia cómo la utilización de cada uno de los diferentes módulos es consistente con la cantidad de celdas de la Fig. 8.24.

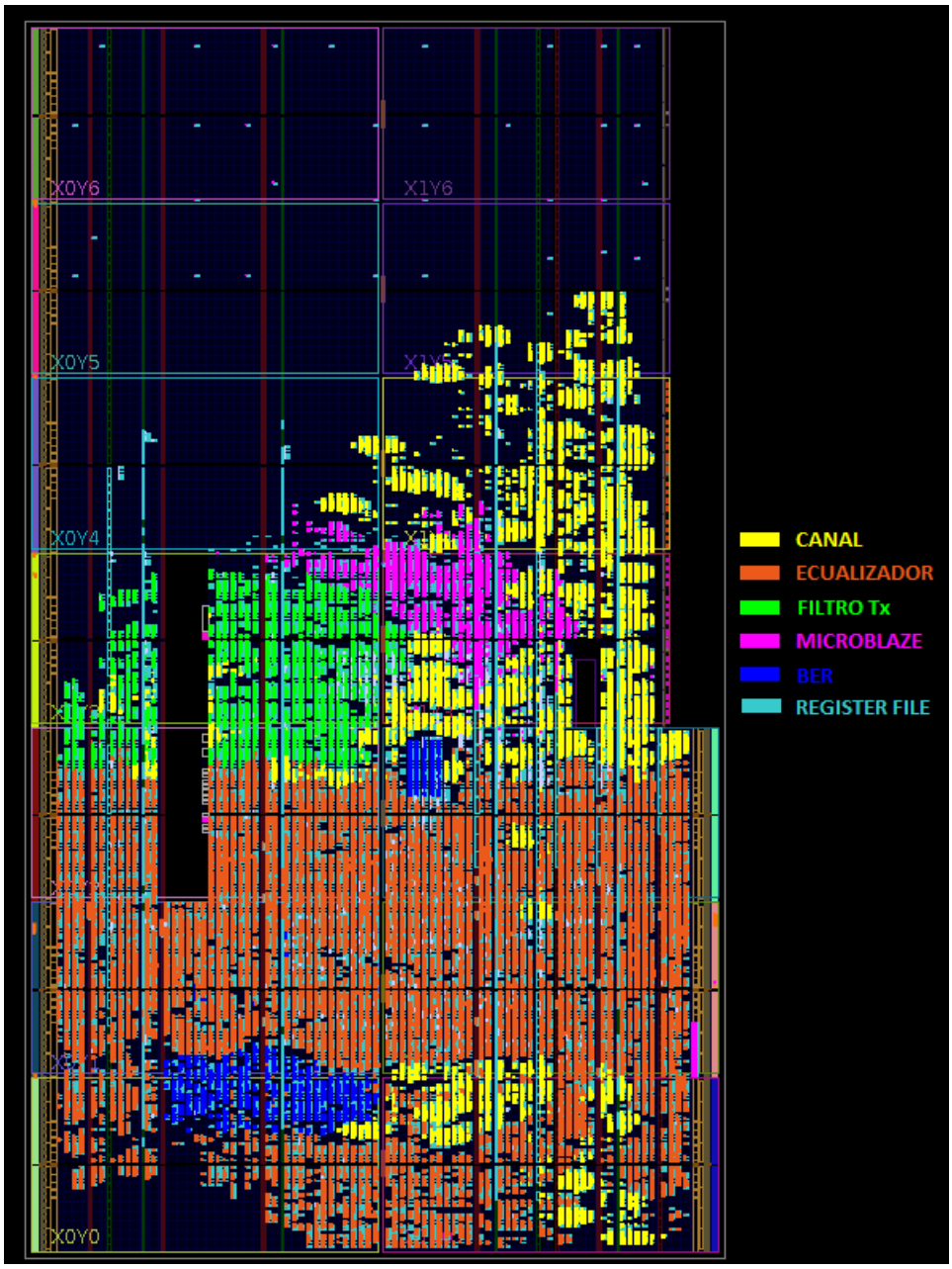


Figura 8.26: *Utilización de la FPGA en el diseño implementado.*

Capítulo 9

Conclusiones

Mediante la realización del presente Proyecto Integrador hemos sido capaces de aplicar una gran cantidad de conocimientos tanto teóricos como prácticos aprendidos a lo largo de toda la carrera, principalmente los referidos a las comunicaciones digitales. De esta forma, fue posible llevar a cabo un trabajo integral que pone en evidencia las habilidades adquiridas para nuestra futura vida profesional.

Entendiendo la importancia que las comunicaciones digitales tienen en el contexto actual, el desarrollo de este proyecto resulta de especial relevancia. El creciente aumento en el tráfico de datos a nivel mundial, fruto, entre otras cosas, de la globalización y de la proliferación de servicios en la nube, impone la necesidad de utilizar nuevas y mejores técnicas que permitan satisfacer esa demanda.

En vista de esto, se logró diseñar e implementar un ecualizador que posibilitaría un aumento en las capacidades de transmisión de los medios disponibles, concretamente en fibra óptica. Este ecualizador permite también una implementación eficiente en el dominio de la frecuencia, logrando mejorar notablemente el desempeño del sistema a las velocidades exigidas actualmente, con un costo de hardware menor que en el caso de una implementación en el dominio del tiempo.

A partir de los datos obtenidos del sistema implementado en FPGA, fue posible comprobar que los resultados concuerdan razonablemente con las simulaciones efectuadas previamente, consiguiendo así las mejoras esperadas en el desempeño. De esta forma se pudo determinar que la implementación se llevó a cabo de manera adecuada.

Debido a la magnitud de este proyecto, se requirió seguir una serie de pasos específicos para lograr un desarrollo prolijo y ordenado. En virtud de esto, se recurrió a un flujo de trabajo muy similar al que efectivamente es utilizado en la industria, partiendo de análisis teóricos y especificaciones

requeridas, pasando por simulaciones de funcionamiento, hasta llegar a la implementación final del proyecto y la verificación de resultados.

Siguiendo esta metodología de trabajo, fue posible prevenir potenciales errores con la mayor anticipación posible, evitando así retrocesos innecesarios en etapas posteriores del desarrollo, y garantizando también un correcto diseño.

Cabe mencionar la importancia de los estudios teóricos realizados antes de iniciar la ejecución práctica, ya que estos nos proporcionaron un profundo entendimiento del sistema, brindándonos así las herramientas necesarias para afrontar con mayor facilidad los problemas que fueron surgiendo.

Otro aspecto a destacar en el flujo de trabajo es la utilización del paquete MyHDL, el cual permitió realizar verificaciones funcionales de forma práctica y eficiente, ahorrando tiempo y facilitando notablemente el análisis de los resultados.

Si bien en principio el proyecto planteado consistió únicamente en la realización de un Ecualizador Adaptivo en el Dominio de la Frecuencia, también fue de gran importancia el desarrollo de todo un sistema de comunicaciones que complemente al ecualizador y permita analizar su funcionamiento en un entorno más cercano a la realidad. Teniendo en cuenta que el objetivo de los módulos adicionales fue probar el comportamiento del ecualizador, solamente se desarrollaron aquellos que fueron estrictamente necesarios para modelar los efectos que repercuten directamente en su funcionamiento, dejando de lado otros bloques que sí estarían presentes en un sistema de comunicaciones real. Tanto el ecualizador como los módulos complementarios se diseñaron de manera altamente parametrizable, con el objetivo de permitir modificar las condiciones de funcionamiento planteando diferentes escenarios posibles.

Como en todo proyecto, a lo largo del mismo surgieron diferentes problemas que fue necesario solucionar. Entre ellos se destacan por un lado los inconvenientes en torno al IP Core de la FFT/IFFT, debido a la necesidad de ajustar su funcionamiento al algoritmo utilizado en simulación, y a que el mismo trabajaba de manera serial. Por otro lado, también resultó problemática la migración de una implementación en serie hacia una en paralelo (tanto a nivel de simulación como en la descripción en RTL), debiendo lograr un comportamiento equivalente entre ambas versiones. Si bien la implementación en paralelo representó una complejidad adicional, la misma permitió reducir considerablemente los requerimientos de timing de la parte del sistema paralelizado.

Con respecto a las posibles mejoras que podrían introducirse en el siste-

ma, la principal que se puede mencionar es la utilización de un ecualizador que trabaje de manera paralela, para lo cual sólo sería necesario que las FFT/IFFT tengan un funcionamiento paralelo. De esta forma, podrían aprovecharse mejor los beneficios de paralelizar el sistema, pudiéndose incluso incrementar el paralelismo hasta 64, ingresando en cada ciclo de clock 64 muestras nuevas y entregando a su salida 64 muestras válidas, permitiendo así tasas mucho mayores.

Otro aspecto a mejorar radica en la utilización de los IP Core para computar las FFT/IFFT. Al trabajar con una modulación DQPSK, las 2 secuencias generadas se tratan de manera independiente una vez hecha la codificación diferencial, es decir que se tienen 2 secuencias reales. Sin embargo, el IP Core trabaja con números complejos a su entrada, por lo que sería posible aprovechar las propiedades de la Transformada de Fourier para computar las FFT de 2 secuencias reales a partir de una única FFT compleja, reduciendo así a la mitad el uso del IP Core y por lo tanto disminuyendo la cantidad de recursos utilizados.

A modo de cierre de este Proyecto Integrador, se exponen algunas valoraciones personales respecto al trabajo realizado. Este proyecto tiene como objetivo aplicar y profundizar los conocimientos obtenidos a lo largo de toda la carrera, para obtener finalmente el título de grado de Ingeniero/a Electrónico/a. Durante la ejecución del mismo, fue necesario demostrar constancia, responsabilidad, capacidad de resolución de problemas y criterio profesional, para hacer frente a los desafíos que se fueron presentando y que se presentarán a lo largo de nuestra vida profesional.

Bibliografía

[1] John R. Barry, Edward A. Lee y David G. Messerschmitt. *Digital Communication*. Springer Science+Business Media, 2004.

[2] Mark Borgerding. "Turning Overlap-Save into a Multiband Mixing, Downsampling Filter Bank". En: *IEEE Signal Processing Magazine* (2006).

[3] Eleanor Chu y Alan George. *Inside the FFT Black Box: Serial and Parallel Fast Fourier Transform Algorithms*. CRC Press, 2000.

[4] Gregory A. Clark, Sanjit K. Mitra y Sydney R. Parker. "Block Implementation of Adaptive Digital Filters". En: *IEEE Transactions on Circuits and Systems* Cas-28.6 (1981), págs. 584-592.

[5] James Donovan. *Dispersion in Multimode Optical Fiber and Intersymbol Interference*. Commscope Infrastructure Academy. 2017.

[6] Shoab A. Khan. *Digital Design of Signal Processing Systems: A Practical Approach*. Wiley, 2011.

[7] Guangxi Liu. *Gaussian Noise Generator Core Specification*. Open Cores.

[8] Richard Lyons. *DSP Tricks: Computing inverse FFTs using the forward FFT*. 2010. URL: `https://www.embedded.com/design/configurable-systems/4210789/DSP-Tricks--Computing-inverse-FFTs-using-the-forward-FFT`.

[9] Vijay Madisetti. *The Digital Signal Processing Handbook*. CRC Press, 1997.

[10] John J. O Reilly. "Series-parallel generation of m-sequences". En: *The Radio and Electronic Engineer* 45.4 (1975), págs. 171-176.

[11] Alan V. Oppenheim y Ronald W. Schafer. *Tratamiento de Señales en Tiempo Discreto*. Pearson, 2011.

[12] Wayne T. Padgett y David V. Anderson. *Fixed-Point Signal Processing*. Morgan Claypool Publishers, 2009.

[13] Keshab K. Parhi. *VLSI Digital Signal Processing Systems: Design and Implementation*. Wiley, 1999.

[14] John G. Proakis y Dimitris G. Manolakis. *Digital Signal Processing: Principles, Algorithms and Applications*. Pearson.

[15] Anatolij Sergiyenko. *Pipelined FFT/IFFT 128 points (Fast Fourier Transform) IP Core User Manual*. Unicore Systems.

[16] John J. Shynk. "Frequency-Domain and Multirate Adaptive Filtering". En: *IEEE SP Magazine* (1992).

[17] Marvin K. Simon. "On the Bit-Error Probability of Differentially Encoded QPSK and Offset QPSK in the Presence of Carrier Synchronization". En: *IEEE Transactions on Communications* 54.5 (2006), págs. 806-812.

[18] Sergey Ten y Edwards Merrion. *An Introduction to the Fundamentals of PMD in Fibers*. Corning Incorporated. 2006.

[19] Xilinx. *Design Analysis and Closure Techniques*. Vivado Design Suite User Guide. Ver. 2017.4.

[20] Xilinx. *Embedded Processor Hardware Design*. Vivado Design Suite User Guide. Ver. 2017.3.

[21] Xilinx. *KC705 Evaluation Board for the Kintex-7 FPGA User Guide*. Ver. 1.7.

[22] Xilinx. *Xilinx 7 Series FPGA and Zynq-7000 All Programmable SoC Libraries Guide for HDL Designs*. Ver. 14.7.

[23] Omid Zia-Chahabi, Raphael Le Bidan y Michel Morvan. "Efficient Frequency-Domain Implementation of Block-LMS/CMA Fractionally-Spaced Equalization for Coherent Optical Communications". En: *IEEE Photonics Technology Letters* (2011).

Apéndice A

Cómputo FFT con Diezmado en el Tiempo

La reducción de complejidad de los ecualizadores en el dominio de la frecuencia radica en el hecho de que existen técnicas sumamente eficientes para implementar la Transformada Discreta de Fourier, conocidas en su conjunto como Transformada Rápida de Fourier o FFT (Fast Fourier Transform), en las que no se computa de manera directa la DFT/IDFT a partir de sus definiciones, que se muestran en las Ec. A.1 y A.2, en las que $W_N = e^{-j(2\pi/N)}$ es el factor de twiddle (o simplemente twiddle).

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

$$x(n) = \frac{1}{N} \sum_{k=0}^{N-1} X(k)\, W_N^{-kn} \tag{A.2}$$

Considerando que tanto $x(n)$ y $X(k)$ pueden ser complejos, es posible apreciar que ambas expresiones son muy similares, sólo difiriendo en el factor de escala $1/N$ y en el exponente del factor de twiddle W_N.

Una evaluación directa de una DFT de N puntos requeriría N multiplicaciones complejas (o $4N$ multiplicaciones reales) y $N-1$ sumas complejas para cada valor de k, por lo que para computar la totalidad de los N valores de la DFT se requerirían N^2 multiplicaciones complejas y $N(N-1)$ sumas complejas.

Las mejoras en la eficiencia a la hora de calcular la FFT se logran aprovechando las propiedades de periodicidad y simetría de W_N^{kn} que se muestran en las Ec. (A.3) y (A.4), ya que hacen que se produzcan operaciones redundantes y que por lo tanto pueden evitarse de antemano.

$$\text{Simetría Conjugada Compleja:} \quad W_N^{k(N-n)} = W_N^{-kn} = (W_N^{kn})^* \tag{A.3}$$

$$\text{Periodicidad en } n \text{ y } k: \quad W_N^{kn} = W_N^{k(n+N)} = W_N^{n(k+N)} \tag{A.4}$$

Concretamente, el algoritmo que se utilizará es el que se conoce como **Diezmado en el Tiempo** (*Decimation in Time - DIT*), en los que se descompone sucesivamente la secuencia de entrada $x(n)$ en secuencias más pequeñas, debiéndose calcular así una DFT de menor tamaño.

Considerando $N = 2^v$, es decir que N es una potencia de 2 y por lo tanto es un número par, se puede calcular $X(k)$ a partir de dos secuencias de $N/2$ puntos que combinadas conforman la secuencia original $x(n)$. Específicamente, las secuencias que se utilizan son $g(n) = x(2n)$ y $h(n) = x(2n+1)$, formadas por las muestras $x(n)$ de índice par e impar respectivamente. Este procedimiento se ilustra en la Fig. A.1.

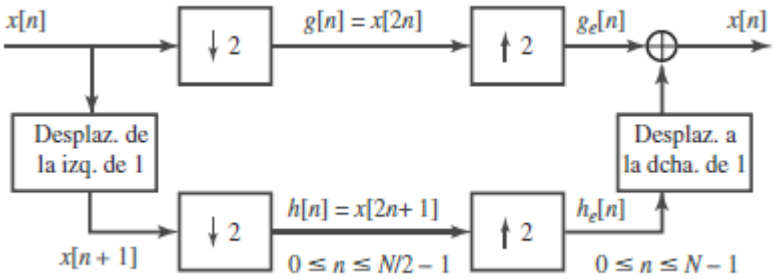


Figura A.1: *Diagrama de bloques para diezmado en el tiempo.*

Aplicando propiedades de la Transformada de Fourier, es posible expresar las transformadas de $g(n)$ y $h(n)$ en función de la de $x(n)$.

$$G(e^{jw}) = \frac{1}{2}\left(X(e^{jw/2}) + X(e^{j(w-2\pi)/2})\right) \tag{A.5}$$

$$H(e^{jw}) = \frac{1}{2}\left(X(e^{jw/2})e^{jw/2} + X(e^{j(w-2\pi)/2})e^{j(w-2\pi)/2}\right) \tag{A.6}$$

Siguiendo el diagrama de la Fig. A.1, es posible determinar las transformadas de $g_e(n)$ y $h_e(n)$, las cuales han sido expandidas en el tiempo, por lo que por propiedades de la transformada de Fourier es posible definir de manera directa $G_e(e^{jw}) = G(e^{j2w})$ y $H_e(e^{jw}) = H(e^{j2w})$, obteniendo finalmente el valor de $X(e^{jw})$ a partir de ellas.

$$X(e^{jw}) = G_e(e^{jw}) + e^{-jw}H_e(e^{jw}) = G(e^{j2w}) + e^{-jw}H(e^{j2w}) \tag{A.7}$$

Si en la Ec. (A.7) se reemplazan los valores de las Ec. (A.5) y (A.6), se puede observar que efectivamente se obtiene $X(e^{jw})$, demostrándose que

es posible calcular la transformada de la secuencia original a partir de las secuencias en las que se la divide.

La transformada discreta $X(k)$ se obtiene valuando $X(e^{jw})$ en las frecuencias discretas $w = 2k\pi/N$ con $k = 0,\ 1,\ \cdots,\ N-1$, obteniendo así la Ec. (A.8).

$$X(k) = X(e^{j2k\pi/N}) = G(e^{j(2k\pi/N)2}) + e^{-j2k\pi/N} H(e^{j(2k\pi/N)2}) \tag{A.8}$$

La expresión de la Ec. (A.8) puede plantearse en función de la secuencia original si se tienen en cuenta las definiciones de las transformadas involucradas.

$$G(e^{j(2k\pi/N)2}) = \sum_{n=0}^{N/2-1} x(2n)e^{-j(2k\pi/N)2n} = \sum_{n=0}^{N/2-1} x(2n)e^{-j(2k\pi/(N/2))n}$$

$$G(e^{j(2k\pi/N)2}) = \sum_{n=0}^{N/2-1} x(2n)W_{N/2}^{kn} \tag{A.9}$$

$$H(e^{j(2k\pi/N)2}) = \sum_{n=0}^{N/2-1} x(2n+1)W_{N/2}^{kn} \tag{A.10}$$

Reemplazando los valores de las Ec. (A.9) y (A.10) en la Ec. (A.8), se llega a la expresión deseada de $X(k)$.

$$X(k) = \sum_{n=0}^{N/2-1} x(2n)W_{N/2}^{kn} + W_N^k \sum_{n=0}^{N/2-1} x(2n+1)W_{N/2}^{kn} \quad \text{con k = 0, 1, ..., N-1} \tag{A.11}$$

$$X(k) = G(k) + W_N^k H(k) \quad \text{con k = 0, 1, ..., N-1}$$

La Ec. (A.11) deja en clara evidencia cómo es posible calcular una DFT de N puntos ($X(k)$) a partir de dos DFT de $N/2$ puntos ($G(k)$ y $H(k)$).

Se debe notar que las DFT de $N/2$ puntos se están calculando para un valor de k que llega hasta $N-1$, en vez de $N/2-1$ como se haría normalmente. Sin embargo, una DFT de $N/2$ puntos es periódica con período $N/2$, por lo tanto $G(k)$ y $H(k)$ se calculan para k hasta $N/2-1$ y luego se extiende periódicamente sin necesidad de realizar operaciones adicionales.

Este procedimiento puede apreciarse en la Fig. A.2, en la que se calcula una DFT de $N = 8$ puntos a partir de dos DFT de $N/2 = 4$ puntos.

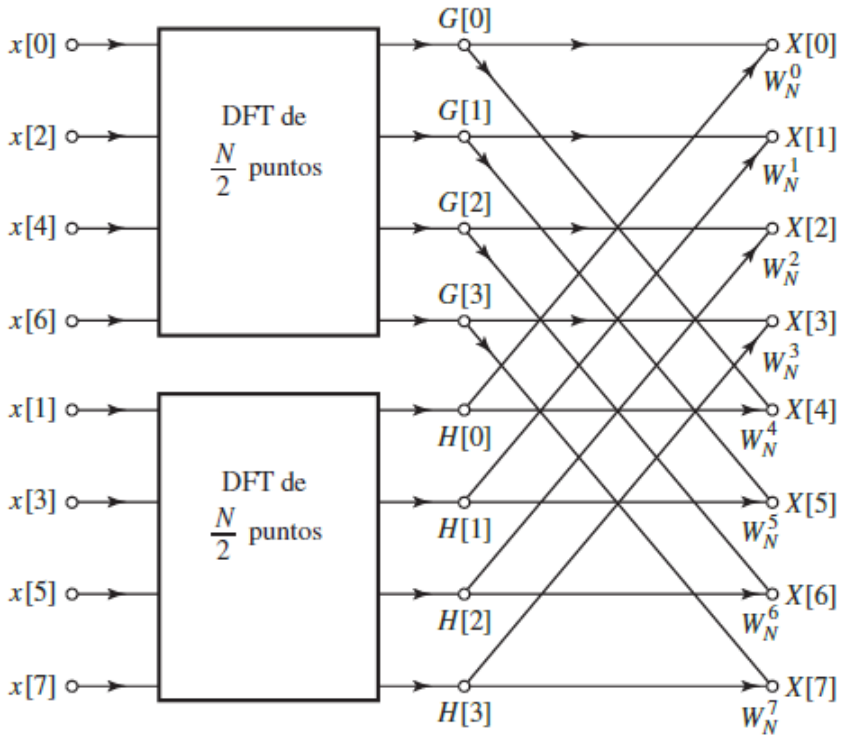


Figura A.2: *Grafo de flujo para el cálculo de una DFT de $N = 8$ puntos a partir de dos DFT de $N/2 = 4$ puntos.*

Se debe notar que se calculan solamente $N/2 = 4$ valores de $G(k)$ y $H(k)$, aprovechando así la periodicidad de la DFT (por ejemplo, como $G(0) = G(4)$, cuando se debe utilizar $G(4)$ se emplea el valor de $G(0)$ ya calculado previamente).

Con este método, se calculan de manera directa dos DFT de $N/2$ puntos, requiriendo ambas en conjunto un total de $2(N/2)^2$ multiplicaciones complejas y $2(N/2)^2 = N^2/2$ sumas complejas (considerando $N \simeq N - 1$). Una vez calculadas se las debe combinar para formar $X(k)$, lo cual requiere N multiplicaciones adicionales (para multiplicar por la secuencia impar por el factor de twiddle) y N sumas adicionales para sumar ambos resultados parciales. De esta forma, se requieren en total $N + N^2/2$ multiplicaciones complejas y $N + N^2/2$ sumas complejas. Así, para todo $N > 2$, este método requiere menos operaciones que el cálculo directo.

Así como la DFT original de N puntos se dividió en dos DFT de $N/2$ puntos logrando una reducción en la complejidad, cada una de las DFT de $N/2$ puntos puede computarse a partir de dos DFT de $N/4$ puntos (considerando que N es una potencia de 2 y que por lo tanto puede continuar subdividiéndose), que se combinan luego para obtener las de $N/2$ puntos, siguiendo un proceso totalmente análogo. De esta forma, los 4 puntos de $G(k)$ se computarían como se muestra en la Fig. A.3.

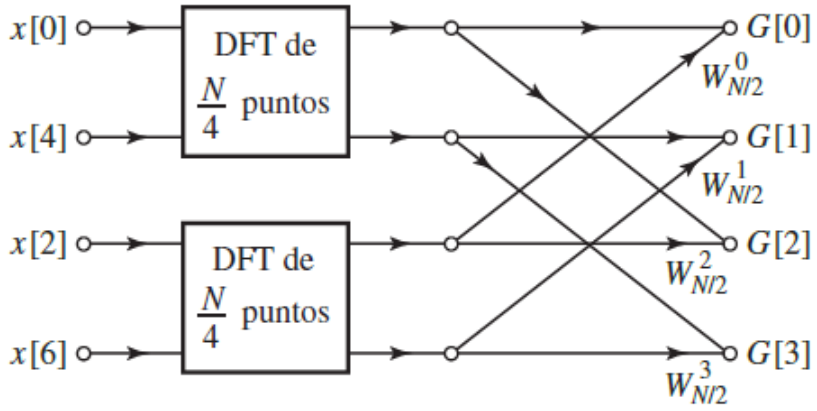


Figura A.3: *Grafo de flujo para el cálculo de una DFT de $N/2 = 4$ puntos a partir de dos DFT de $N/4 = 2$ puntos.*

La Fig. A.3 representa los bloques de DFT de $N/2$ puntos de la Fig. A.2, por lo que se pueden combinar ambos grafos, teniendo en cuenta que $W_{N/2} = W_N^2$. De esta forma se obtiene el grafo de la Fig. A.4.

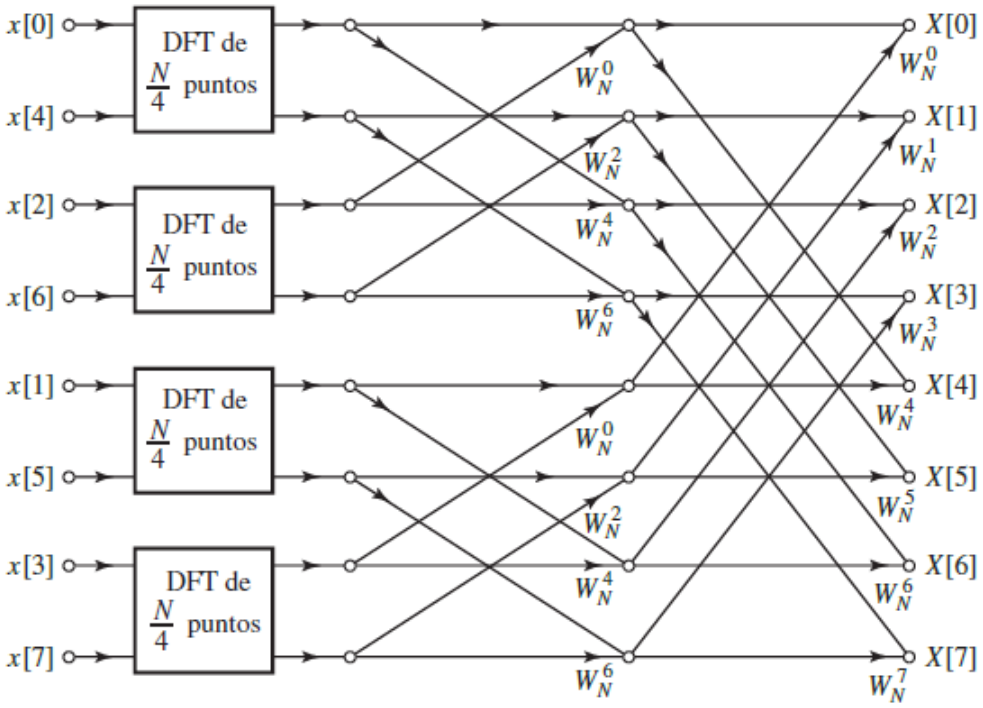


Figura A.4: *Grafo de flujo para el cálculo de una DFT de $N = 8$ puntos a partir de múltiples DFT de $N/2 = 4$ y $N/4 = 2$ puntos.*

A su vez, las DFT de $N/4$ puntos pueden continuar dividiéndose, utilizando para cada una dos DFT de $N/8$ puntos, y así sucesivamente hasta que finalmente se llega a una DFT de 2 puntos, que se calcula de manera directa como se muestra en la Fig. A.5, requiriéndose un total de $\log_2 N$ etapas de cómputo.

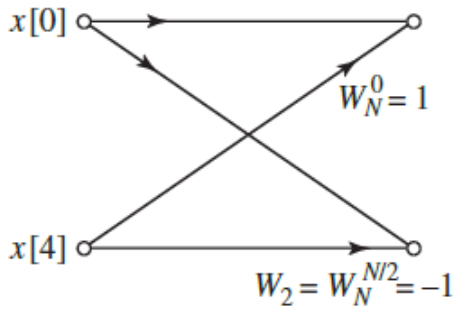


Figura A.5: *Grafo de flujo para el cálculo de una DFT de $N = 2$ puntos.*

En el caso de una DFT de $N = 8$ puntos, con $N/4$ ya se llega a una DFT de 2 puntos que se computa de manera directa, obteniendo finalmente el diagrama de la Fig. A.6.

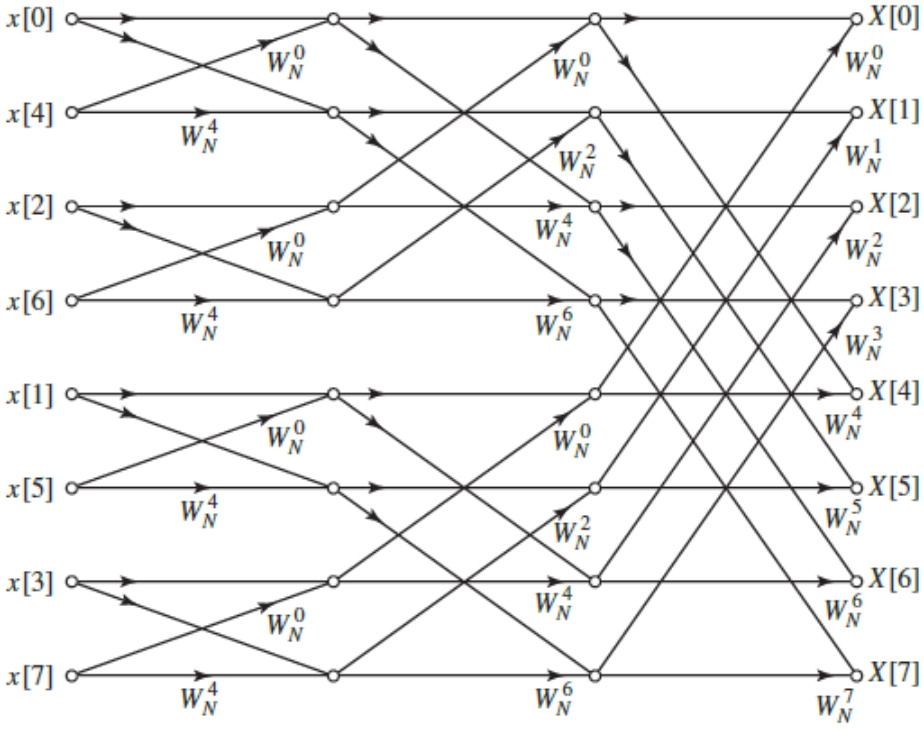


Figura A.6: *Grafo de flujo para el cálculo de una DFT de $N = 8$ puntos mediante diezmado en el tiempo.*

Este método en el que se va dividiendo por la mitad el tamaño de la DFT hasta llegar a una de tamaño 2 se denomina **Radix-2 FFT**.

Su complejidad se puede analizar a partir del cálculo hecho anteriormente en el que se subdivide una única vez, calculando de manera directa las DFT de $N/2$ puntos, resultando una cantidad de multiplicaciones y sumas complejas igual a $N + 2(N/2)^2$. Si las DFT de $N/2$ puntos se calculan a partir de transformadas de $N/4$ puntos, el factor $(N/2)^2$ debe reemplazarse por $N/2 + 2(N/4)^2$, resultando un total de $N + N + 4(N/4)^2$. Con el mismo

razonamiento, si las DFT de $N/4$ puntos se calculan a partir de dos DFT de $N/8$ puntos, el factor $(N/4)^2$ debe reemplazarse por $N/4 + 2(N/8)^2$, obteniendo un total de $N + N + N + 8(N/8)^2$.

Repitiendo este reemplazo $\log_2 N$ veces (siendo N una potencia de 2), se llega finalmente a que para el cálculo de una DFT de N puntos se requieren $N \log_2 N$ multiplicaciones y sumas complejas. Este resultado es consistente con la Fig. A.6, en la que se puede apreciar que se tienen $\log_2 N$ etapas con N multiplicaciones en cada uno de ellas.

Apéndice B

Representación en Punto Fijo

Cuando se trabaja en punto fijo (*fixed point*), se tiene una cierta cantidad de bits totales para la representación del valor deseado, que incluye además un bit de signo opcional, es decir que el dato puede ser signado (*signed* - *S*) o no signado (*unsigned* - *U*). El numero total de bits (*NB*) se encuentra dividido en parte fraccional (*NBF*) y parte entera (*NBI* = *NB*-*NBF*), por lo que, a diferencia de la representación de números en punto flotante, en la que el punto decimal puede estar en diferentes lugares de acuerdo a la magnitud del número a representar, al trabajar en punto fijo el punto decimal se encuentra siempre en el mismo lugar, de allí el nombre.

La *resolución* del dato a representar se forma a partir de la cantidad de bits totales y de la cantidad de bits fraccionales, siendo la forma de simbolizar esta resolución la que se muestra en la Ec. (B.1).

$$\underbrace{S(NB, NBF)}_{Signado} \qquad \underbrace{U(NB, NBF)}_{No\ signado} \tag{B.1}$$

Los bits se ordenan colocando el bit de signo en la posición más significativa (si el número es signado), seguido por los bits enteros (In) y luego los bits fraccionarios (Fn), como se muestra en la Fig. B.1. Si la representación es no singada, el conjunto comienza con los bits que representan la parte entera.

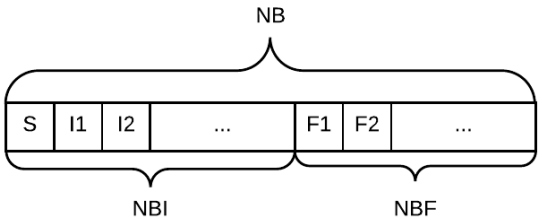


Figura B.1: *Representación de un número signado en punto fijo.*

De esta manera se puede representar un cierto rango de números, con

una precisión determinada por NB, NBF y NBI.

En la Tabla B.1 se sintetiza la manera de calcular el valor de un número x en punto fijo de acuerdo a si éste es o no signado, mostrándose también el rango de valores posibles a representar y la precisión con la cual es posible trabajar.

	Signado	**No Signado**
Símbolo	$S(NB, NBF)$	$U(NB, NBF)$
Valor	$x = \frac{1}{2^{NBF}}\left(-2^{NB-1} + \sum_{n=0}^{NB-2} 2^n b_n \right)$	$x = \frac{1}{2^{NBF}} \sum_{n=0}^{NB-1} 2^n b_n$
Rango	$-2^{NB-1-NBF} \leqslant x \leqslant 2^{NB-1-NBF} - 2^{-NBF}$	$0 \leqslant x \leqslant 2^{NBI} - 2^{-NBF}$
Precisión	2^{-NBF}	2^{-NBF}

Tabla B.1: Representación en punto fijo signada y no signada.

Al intentar expresar un número con una determinada resolución, es posible que el mismo no se encuentre dentro de la serie de valores que se pueden representar con dicha resolución, por lo tanto es necesario definir un criterio para determinar cuál de los 2 valores más cercanos, posibles de representar, tomará nuestro dato.

Para resolver esta situación, existen 2 técnicas: *Truncado* y *Redondeo*. La primera consiste en 'cortar' cierta cantidad de bits de la parte fraccionaria para que se cumpla con el NBF requerido. La segunda, realiza este mismo corte pero a su vez analiza el primer bit de la sección cortada. Si este es un '1', el valor se incrementa en una unidad, si el bit es '0' el valor queda como está.

A nivel de implementación, realizar el truncado de un número es un poco más eficiente y menos costoso, pero implica una pérdida de información ligeramente mayor. El redondeo ofrece una aproximación mas precisa al valor que se desea representar pero su costo de implementación es superior. En virtud de esto, se utiliza el truncado para aplicaciones que no sean tan críticas y el redondeo cuando se requiera mayor precisión.

Otra situación que puede presentarse al trabajar en punto fijo, es que el valor a representar supere el límite inferior o superior del rango definido por la resolución con la cual se trabaja.

Para estos casos existen 2 criterios posibles: *Wrapping* y *Saturación*. En el primer caso, si el valor sobrepasa el límite superior, comienza a tomar valores desde el límite inferior, y sucede lo contrario si el valor está por debajo del límite inferior (Fig.B.2a). En el segundo caso, al superarse el rango, el valor se estanca en el límite inferior o superior dependiendo de cuál

de ellos se haya sobrepasado (Fig. B.2b). Durante toda la implementación del sistema se utiliza el criterio de saturación para estos casos.

(a) *Wrapping.* (b) *Saturación.*

Figura B.2: *Criterios para manejar la superación del rango máximo determinado por la resolución de trabajo.*

Además, a lo largo del trabajo no sólo es necesario representar los valores en una resolución adecuada, si no que también se deben realizar operaciones con dichos valores, los cuales no necesariamente tendrán la misma cantidad de bits enteros o fraccionarios. En estas situaciones, se debe trabajar el resultado de las operaciones en máxima resolución, para luego obtener su representación óptima en punto fijo, evitando así una mayor pérdida de información.

Para una operación de *Suma o Resta*, el resultado tendrá como máximo un bit más que la cantidad de bits del operando con mayor resolución.

Para una operación de *Multiplicación*, la cantidad de bits total del resultado en máxima resolución será la sumatoria del número de bits de los dos factores, es decir la cantidad de bits enteros será la suma de los bits enteros de cada operando, y la cantidad de bits fraccionales será la suma de los bits fraccionales.

Para cada operación que se desee realizar, se debe tener en cuenta que el punto decimal deberá estar alineado entre los datos a operar. Si los mismos no poseen la misma resolución se deberá agregar ceros a la derecha y completar con el bit de signo a la izquierda.

Apéndice C

Anexos del Proyecto Integrador

C.1. Solicitud de Aprobación de Tema

Facultad de Ciencias Exactas, Físicas y Naturales
AREA INGENIERÍA
ESCUELA DE ELECTRONICA
C.C. 755 - Correo Central - 5000 - CÓRDOBA
Tel. Directo (0351) 33-4147 int 110
Conmutador: 433-4141 y 33-4152 - Interno 10

Sr. Director de la Escuela de Ingeniería Electrónica
Ing.: RODRIGO BRUNI

Me dirijo a Ud. a fin de solicitar la **aprobación del tema del *Proyecto Integrador*** (***PI***) que propongo a continuación:

TEMA

NOMBRE DEL PROYECTO: SIMULACIÓN E IMPLEMENTACIÓN EN FPGA DE ECUALIZADOR FRACCIONALMENTE ESPACIADO EN EL DOMINIO DE LA FRECUENCIA CON ALGORITMO LMS PARA ADAPTACIÓN DE COEFICIENTES

DESCRIPCIÓN: VER ANEXO.

DESARROLLO DE PROTOTIPO: se implementará el diseño en FPGA.

AREA TEMATICA DEL PI: Comunicaciones, Digitales.

ASIGNATURAS: Teoría de Señales y Sistemas Lineales, Electrónica Digital I, Teoría de las Comunicaciones, Electrónica Digital II.

Director de PI

Nombre: Hueda, Mario Rafael
Cargo: Titular de Cátedra Teoría de Señales y Sistemas Lineales
Dirección Personal o Laboral: Huiliches 781 Casa 40, B° Portales del Sur
TE: 0351-156812027
E-Mail: mario.hueda@unc.edu.ar
Firma del Director / Co-Director: .....

Co-Director de PI

Nombre: Pola, Ariel Luis
Cargo: Ingeniero Principal
Dirección Personal o Laboral: Ernesto Romagosa 518
TE: 0358-154184934
E-Mail: arielpola@gmail.com
Firma del Co-Director: .....

Datos del Estudiante

Nombre y Apellido: Matías Nicolás Brignone
Matrícula: 38503382
Materias que faltan aprobar: ninguna
Dirección: José Barros Pazos 3290, B° Bajo Palermo
Localidad: Córdoba **Provincia**: Córdoba
E-mail: mnbrignone@gmail.com
Teléfono: 351 2676789

Firma: .....

Datos del Estudiante

Nombre y Apellido: Lucía Fernanda Rodríguez
Matrícula: 38203622
Materias que faltan aprobar: ninguna
Dirección: Independencia 834, B° Nueva Córdoba
Localidad: Córdoba **Provincia**: Córdoba
E-mail: luferrogom@gmail.com
Teléfono: 383 4694050

Firma: .....

Objetivo General:
El objetivo es diseñar, simular e implementar en FPGA un Ecualizador Fraccionalmente Espaciado en el dominio de la frecuencia utilizando el algoritmo LMS para la adaptación de coeficientes, y realizar las pruebas de funcionamiento en un sistema de comunicaciones completo, agregando no sólo los módulos principales para realizar la transmisión (transmisor, modulador, contador de errores) sino también aquellos bloques que permiten simular el comportamiento del canal de comunicaciones para evaluar el funcionamiento del ecualizador frente a los problemas que afectan a la transmisión de datos.

Objetivos Específicos:

- Estudiar el algoritmo Block Least Mean Square (BLMS) para la adaptación de coeficientes.
- Simular el funcionamiento de un ecualizador adaptivo en el dominio del tiempo utilizando el algoritmo Block Least Mean Square (BLMS) para la adaptación de coeficientes.
- Estudiar el método Overlap&Save para realizar el filtrado en el dominio de la frecuencia.
- Simular un ecualizador adaptivo realizando el filtrado en el dominio de la frecuencia, utilizando el método de Overlap&Save.
- Simular en un lenguaje de alto nivel un ecualizador adaptivo realizando tanto el filtrado como la adaptación de coeficientes en el dominio de la frecuencia.
- Utilizar el paquete MyHDL de Python para durante la verificación funcional de la implementación de cada bloque.
- Diseñar en RTL el módulo ecualizador en el dominio de la frecuencia verificando su correcto funcionamiento a nivel de síntesis y timing.
- Implementar en FPGA un ecualizador adaptivo en el dominio de la frecuencia, utilizando el algoritmo LMS para la adaptación de coeficientes.
- Desarrollar una plataforma de verificación que permita configurar y controlar el sistema.

Antecedentes de Proyectos similares: -

Duración y Fases de las tareas previstas:

Inicio 1/12/2017	SEMANAS											
	1|2	3|4	5|6	7|8	9|10	11|12	13|14	15|16	17|18	19|20	21|22	23|24
Análisis Teórico y Simulaciones												
Desarrollo de la Plataforma de Verificación												
Implementación del sistema en FPGA												
Verificación de Resultados y Elaboración de Informe												

Metodología

Lugar previsto de realización: Fundación Fulgor (Ernesto Romagosa 518, Bº Colinas V. Sarsfield).

Requerimiento de Instrumental y Equipos: computadoras con software apropiado instalado (MATLAB, Python con los paquetes necesarios, Vivado 2016.1 o posterior junto con su Kit de Desarrollo de Software Xilinx SDK), FPGA (Kit KC705 – Kintex-7).

Inversión estimativa prevista por el alumno: -

Apoyo Económico externo a la Facultad: estipendio mensual otorgado por la Fundación Fulgor.

Referencias Bibliográficas o de Software:

1. OPPENHEIM, Alan V., SCHAFER, Ronald W., (2011), *Tratamiento de Señales en Tiempo Discreto*, Madrid, España: Pearson Educación.
2. BARRY, John R., LEE, Edward A., MESSERSCHMITT, David G., (2004), *Digital Communication*, Nueva York, Estados Unidos: Springer Science+Business Media.
3. CLARK, Gregory A., MITRA, Sanjit K. y PARKER, Sydney R. (1981), *Block Implementation of Adaptive Digital Filters*. IEEE Transactions on Circuits and Systems, Cas-28 (6), 584-92.
4. SHYNK, John J. (1992), Frequency-Domain and Multirate Adaptive Filtering. IEEE SP Magazine.

Recibido Cátedra PI

.....
Firma

Córdoba, / / .

ANEXO

1. Descripción Detallada del Proyecto

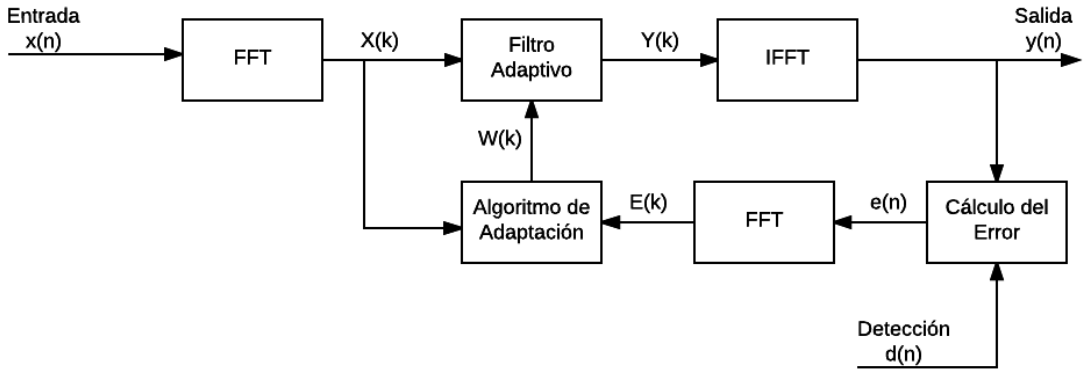
Desde hace varias décadas, las comunicaciones digitales han tomado una enorme importancia debido a las considerables ventajas que presentan frente a una comunicación analógica, aun cuando lo que se desea transmitir es una señal inherentemente analógica (en cuyo caso debe ser representada mediante una secuencia de bits con una determinada precisión).
El principal beneficio de las comunicaciones digitales radica en el hecho de poder aplicar algoritmos de procesamiento digital de señales (DSP) y técnicas de codificación, permitiendo así incrementar drásticamente las velocidades de transmisión que un determinado medio puede soportar. Además, los avances en el diseño y fabricación de circuitos integrados permiten que el procesamiento digital sea conveniente también desde un punto de vista económico.
Para permitir transmisiones a tasas cada vez mayores, es necesario corregir los efectos que el canal produce en las señales transmitidas, los cuales dependerán del tipo de medio por el que se realice la transmisión (cable coaxial, fibra óptica, aire, etc.).
Uno de los principales problemas que surge es la *Interferencia Intersímbolo* (ISI), en la que, debido a las distorsiones generadas por el canal, símbolos adyacentes al símbolo que se desea muestrear en un momento dado contribuyen al valor que se mide, pudiendo provocar una detección errónea.

En este proyecto el enfoque está orientado a transmisiones de alta velocidad por fibra óptica, en las que los bits de datos, representados mediante pulsos de luz, se dispersan en el tiempo a medida que viajan a lo largo de la fibra óptica, causando ISI en caso de que lleguen a superponerse unos con otros. Esto no suponía un problema cuando las transmisiones eran a velocidades más bajas, ya que había una mayor separación entre bits, pero a medida que las velocidades se fueron incrementando, la Interferencia Intersímbolo se convirtió en un factor crítico al momento de determinar el máximo ancho de banda admisible.
La dispersión de los pulsos de luz en la fibra óptica, y por lo tanto la ISI, se debe principalmente a dos fenómenos: dispersión cromática y PMD (*Polarization Mode Dispersion* o Dispersión por Modo de Polarización).
La dispersión cromática se debe al hecho de que la luz viaja por la fibra a diferentes velocidades de acuerdo a su longitud de onda, lo cual resulta problemático debido a que la fuente generadora de pulsos de luz no suele ser una única longitud de onda, sino que está compuesta de varias longitudes de onda en torno a una central.
La PMD es la que mayor dispersión causa en la fibra óptica y está asociada a la polarización de la luz, ya que la misma es una onda electromagnética cuyo campo eléctrico puede estar alineado con el eje horizontal (polarización horizontal), con el eje vertical (polarización vertical) o puede ser una composición de ambos en caso de que no esté alineado con ninguno de los ejes. A su vez, la velocidad con la que los pulsos se propagan a lo largo de la fibra óptica depende del índice de refracción de la misma.
Idealmente, en una fibra perfectamente simétrica, la velocidad de propagación no dependería de la polarización de la luz ya que el índice de refracción sería el mismo independientemente del modo de polarización. Sin embargo, esto no ocurre en la realidad y

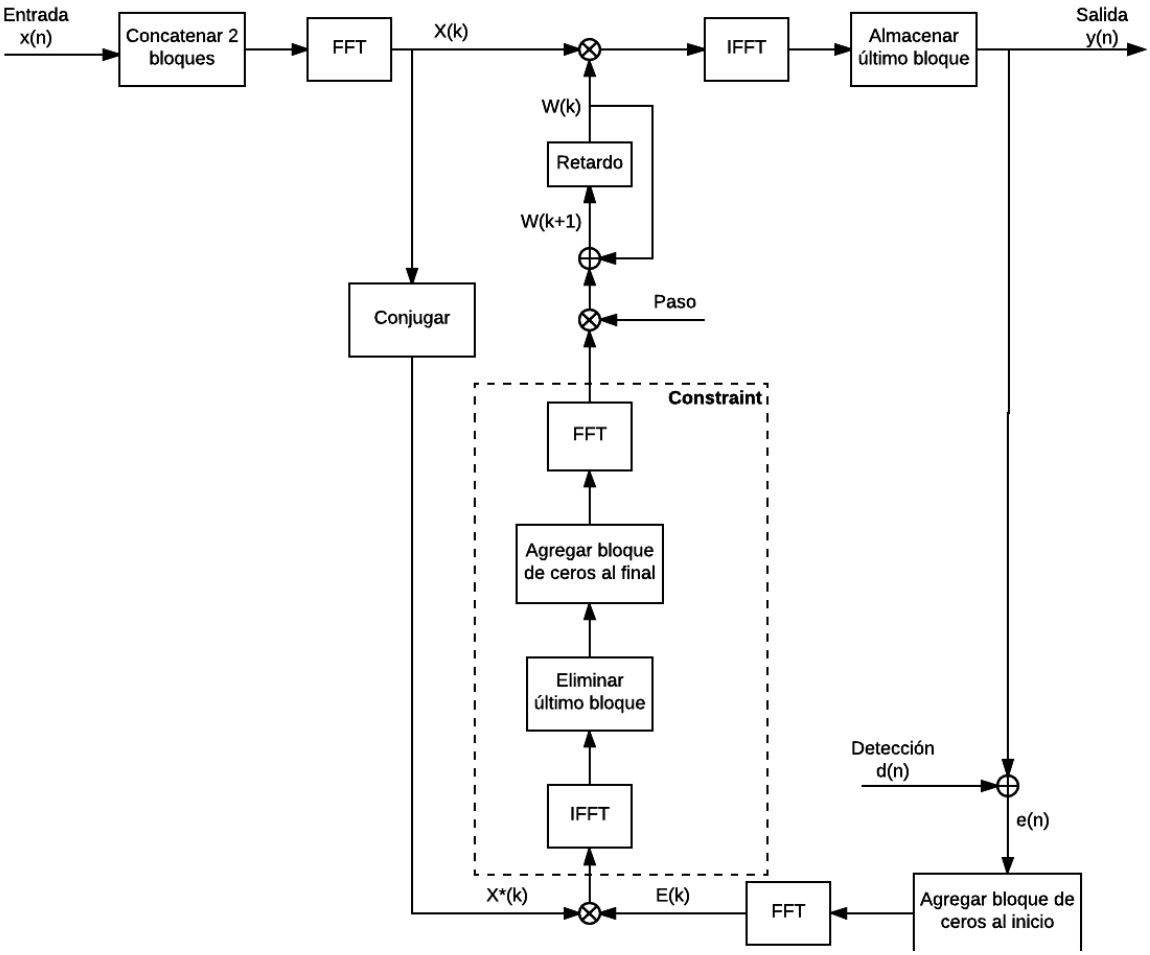
no se tienen los mismos índices de refracción en las direcciones horizontales y verticales, lo cual da como resultado que la luz con polarización horizontal se propague a una velocidad diferente que la que tiene una polarización vertical, introduciéndose así un retardo entre ambas polarizaciones denominado Differential Group Delay (DGD), produciendo un estiramiento del pulso.
La PMD empeora a medida que las distancias recorridas son mayores, varía a lo largo de la fibra y ocurre por múltiples causas, como asimetrías en el núcleo de la fibra, fuerzas externas aplicadas sobre ella, curvas y torceduras en su disposición al instalarla, cambios de temperatura, o vibraciones mecánicas (debidas por ejemplo a un tren que pasa cerca). Por lo tanto, puede verse que la ISI causada por PMD tiene una naturaleza aleatoria y dependiente del tiempo.

Además de mejoras constructivas en la fibra óptica, a los fines de corregir la ISI se utilizan técnicas de ecualización que buscan contrarrestar en el receptor los efectos del canal para reducir lo máximo posible la probabilidad de error.
Para compensar la Interferencia Intersímbolo producida por PMD, no es posible emplear un ecualizador con coeficientes conocidos y estáticos debido a la mencionada naturaleza aleatoria (por lo que no se podría conocer de antemano los valores necesarios para contrarrestar sus efectos) y dependiente del tiempo. Para solventar este problema, se emplea un Ecualizador Adaptivo, el cual posee coeficientes cuyos valores se modifican de acuerdo al error detectado a la salida del módulo ecualizador, utilizando diferentes algoritmos a los fines de minimizar el error debido tanto a la ISI como al ruido introducido por el canal.
El filtrado y la adaptación de coeficientes llevados a cabo por el ecualizador puede realizarse en el dominio del tiempo o en el dominio de la frecuencia. Cuando la correcta compensación de la ISI requiere que el ecualizador tenga un elevado número de coeficientes, la complejidad de hardware en el dominio del tiempo aumenta rápidamente mientras mayor sea la cantidad de coeficientes, resultando poco práctica su implementación. Es por ello que se recurre a realizar una ecualización en el dominio de la frecuencia, lo cual permite reducir considerablemente la cantidad de operaciones necesarias para el procesamiento de los datos.

Teniendo esto en consideración surge el proyecto propuesto, en el cual se diseña, simula e implementa en FPGA un Ecualizador Fraccionalmente Espaciado en el Dominio de la Frecuencia, utilizando el algoritmo LMS para la adaptación de coeficientes, teniendo una potencial aplicación en la compensación de ISI debida a la PMD en fibras ópticas.
Se utiliza un ecualizador lineal con un algoritmo de Minimización del Error Cuadrático Medio (MMSE o Minimum Mean-Squared Error), calculado a partir de la salida del ecualizador y el símbolo posteriormente detectado, permitiendo que quede una cierta ISI residual, pero logrando minimizar a cambio el error debido tanto a la ISI como al ruido.
Tanto el filtrado (convolución) como la adaptación de coeficientes (correlación) se llevan a cabo en el dominio de la frecuencia, empleando el método denominado Overlap & Save.
A continuación, se presenta un diagrama en bloques básico del ecualizador, en el cual se puede observar cómo tanto el filtrado como la adaptación de coeficientes se realizan en el dominio de la frecuencia (luego de los bloques de FFT), debiéndose tener en cuenta que el error se calcula en el dominio del tiempo y luego se lo transforma.



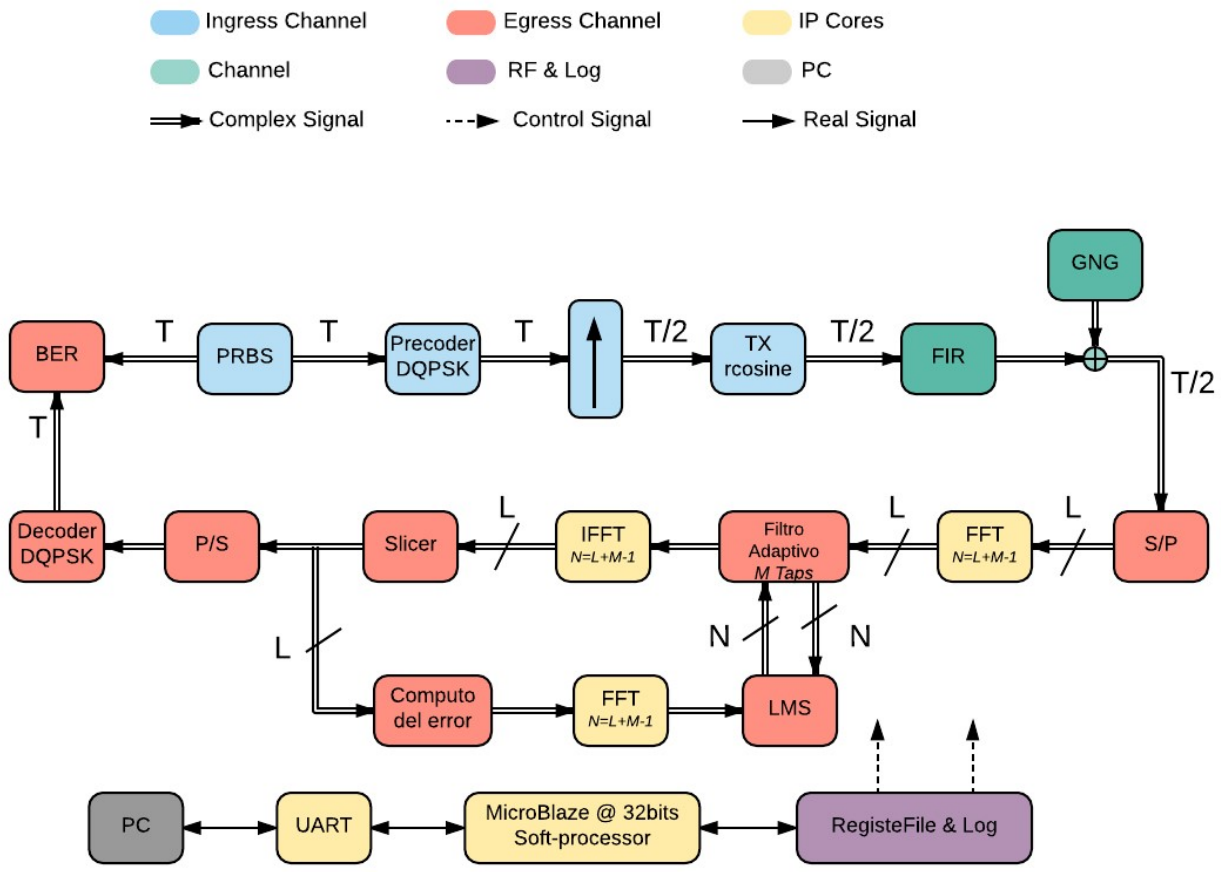
El diagrama anterior se presenta con mayor nivel de detalle a continuación, mostrando las operaciones que se deben llevar a cabo para obtener los mismos resultados que trabajando en el dominio del tiempo.



En el diagrama hay un bloque enmarcado como *Constraint*, necesario para que el gradiente obtenido sea exactamente igual al que se obtendría en el dominio del tiempo. Sin embargo, es posible no utilizar ese bloque, disminuyendo ligeramente el desempeño del ecualizador pero debiendo realizar dos transformadas menos, con lo cual se logra una reducción

significativa en el consumo de recursos. Esto se conoce como *Uncontrained* y es lo que se implementará en el presente proyecto.

Para comprobar el correcto funcionamiento del ecualizador, es necesario probarlo en el marco de un sistema de comunicaciones más completo, por lo cual también es necesario llevar a cabo la implementación del mismo en FPGA, contando con un generador de símbolos, un codificador y decodificador DQPSK (ya que es la modulación a utilizar en el sistema), un filtro transmisor (de tipo coseno realzado), un modelo del canal (filtro FIR representando su respuesta y un generador de ruido gaussiano), un contador de BER (Bit Error Rate) y una plataforma de verificación que permite la interacción con el usuario desde una PC por puerto serie, comunicándose con un microprocesador instanciado en la propia FPGA, pudiendo así configurarse el sistema y extraer datos en diferentes puntos de interés.
A continuación, se presenta un diagrama de bloques del sistema completo a implementar en el proyecto propuesto.



Cabe mencionar que, durante las Prácticas Supervisadas, se llevó a cabo un proyecto que también implicaba la implementación en FPGA de partes de un sistema de comunicaciones digitales, por lo que es posible tomar como referencia algunos módulos previamente desarrollados (contador de BER, filtro transmisor y register file concretamente), debiéndoselos adaptar y modificar significativamente para un uso adecuado en este nuevo proyecto.

C.2. Informes Mensuales

INFORME DE AVANCE DE PROYECTO INTEGRADOR

El presente informe corresponde al Proyecto Integrador titulado *"Simulación e implementación en FPGA de Ecualizador Fraccionalmente Espaciado en el dominio de la frecuencia con algoritmo LMS para adaptación de coeficientes"*, realizado por Brignone Matías Nicolás y Rodríguez Lucía Fernanda.
En el momento de presentación de este informe de avance, ya se han finalizado con éxito las siguientes actividades planteadas como objetivos:

- Estudio del algoritmo Block Least Mean Square (BLMS) para la adaptación de coeficientes.
- Simulación del funcionamiento de un ecualizador adaptivo en el dominio del tiempo utilizando el algoritmo Block Least Mean Square (BLMS) para la adaptación de coeficientes.
- Estudio del método Overlap&Save para realizar el filtrado en el dominio de la frecuencia.
- Simulación de un ecualizador adaptivo realizando el filtrado en el dominio de la frecuencia, utilizando el método de Overlap&Save.
- Simulación en Python del ecualizador adaptivo realizando tanto el filtrado como la adaptación de coeficientes en el dominio de la frecuencia.
- Desarrollo de la plataforma de verificación para configurar y controlar el sistema.

Por otra parte, actualmente las actividades en las que se está trabajando de manera simultánea son las siguientes:

- Diseño a nivel RTL del módulo ecualizador en el dominio de la frecuencia.
- Utilización del paquete MyHDL de Python para durante la verificación funcional de la implementación de cada bloque.

21/12/2017

Universidad Nacional de Córdoba

Facultad de Ciencias Exactas, Físicas y Naturales

Escuela de Ingeniería Electrónica

Quien suscribe el Profesor Hueda, Mario R. en su carácter de Director del Proyecto Integrador de los Estudiantes Brignone, Matías N. y Rodríguez, Lucía F., denominado: SIMULACIÓN E IMPLEMENTACIÓN EN FPGA DE ECUALIZADOR FRACCIONALMENTE ESPACIADO EN EL DOMINIO DE LA FRECUENCIA CON ALGORITMO LMS PARA ADAPTACIÓN DE COEFICIENTES, considera que el desarrollo del trabajo se ha completado según lo especificado en la Solicitud de Aprobación de Tema y se encuentra en condiciones de tramitar su defensa.

A los efectos de quien corresponda, en fecha ...../...../2018.

Firma y aclaración del Director