



ESCUELA SUPERIOR POLITÉCNICA DEL LITORAL
FACULTAD DE INGENIERÍA EN ELECTRICIDAD Y COMPUTACIÓN

“IMPLEMENTACIÓN DE SISTEMAS DE COMUNICACIÓN PARA EL
LABORATORIO DE TELECOMUNICACIONES USANDO LA TARJETA DE
DESARROLLO XTREME DSP DEVELOPMENT KIT JUNTO A
MATLAB/SIMULINK Y XILINX DESIGN TOOL, COMO HERRAMIENTAS DE
SIMULACIÓN Y MODELADO”

INFORME DE PROYECTO DE GRADUACIÓN

Previa a la obtención del Título de:

INGENIERO EN ELECTRÓNICA Y TELECOMUNICACIONES

Presentado por:

Bryan David Ávila Zambrano

Paola Susana Mera Troya

GUAYAQUIL – ECUADOR

AÑO 2014

AGRADECIMIENTO

En primer lugar quisiera agradecer a Dios Todopoderoso por haberme dado la vida y la dicha de poder culminar un paso más en mi carrera profesional, sin él nada fuera posible.

Gracias a mis padres, que siempre han sido y son ese pilar fundamental en mi vida, sin sus sabios consejos, apoyo y orientación no sería la persona que soy ahora.

A mi enamorada por ser ese apoyo incondicional, por estar a mi lado en los buenos y malos momentos.

Por último quisiera agradecer a todos mis amigos y compañeros que nos ayudaron para la culminación de este proyecto.

Bryan David Ávila Zambrano

Quiero agradecer a Dios porque su amor guía e ilumina mi camino, al mismo tiempo es la fuerza que me empuja a seguir sin importar las adversidades.

A toda mi familia, en especial a mis padres ya que sin su esfuerzo, apoyo, motivación y amor no hubiera podido culminar esta etapa en mi vida.

También agradezco a todos mis compañeros y amigos que de una u otra manera dieron su granito de arena para ayudarme a llegar hasta aquí.

Paola Susana Mera Troya

DEDICATORIAS

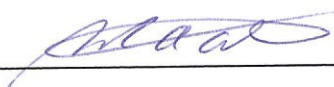
A toda mi familia, mi enamorada y amigos, todos ellos son los responsables de que yo pudiera lograr con éxito todos mis objetivos planteados.

Bryan David Ávila Zambrano

A mis padres y mis amigos por su ayuda brindada ya que sin ellos nada de lo conseguido podría haber sido posible.

Paola Susana Mera Troya

TRIBUNAL DE SUSTENTACIÓN



Ing. María Antonieta Álvarez

PROFESOR DEL PROYECTO DE GRADUACIÓN



Dr. Boris Ramos S.

MIEMBRO PRINCIPAL DEL TRIBUNAL



CIB - ESTOL

DECLARACIÓN EXPRESA

“La responsabilidad del contenido de este informe, nos corresponde exclusivamente; y el patrimonio intelectual del mismo a la ESCUELA SUPERIOR POLITÉCNICA DEL LITORAL”. (Reglamento de exámenes y títulos profesionales de la ESPOL).



Bryan David Ávila Zambrano



Paola Susana Mera Troya



CIS - ESPOL

RESUMEN

En el presente trabajo, “IMPLEMENTACIÓN DE SISTEMAS DE COMUNICACIÓN PARA EL LABORATORIO DE TELECOMUNICACIONES USANDO LA TARJETA DE DESARROLLO XTREME DSP DEVELOPMENT KIT JUNTO A MATLAB/SIMULINK Y XILINX DESIGN TOOL, COMO HERRAMIENTAS DE SIMULACIÓN Y MODELADO”, se muestra el diseño de prácticas de laboratorio realizando la simulación e implementación de diversos sistemas importantes para la rama de las telecomunicaciones, como la aplicación de filtros digitales, lazo de enganche de fase y un sistema modulador/demodulador, específicamente 16QAM, el software de apoyo para las simulaciones será Matlab/Simulink, ya que este programa es muy robusto y tiene la versatilidad de poder analizar y tratar señales complejas en el tiempo, para la implementación de los modelos se usará el software Fuse Probe, que permite cargar los archivos generados de los modelos en la tarjeta FPGA.

El proyecto fue desarrollado utilizando la tarjeta de desarrollo Xtreme DSP Development Kit basado en un dispositivo FPGA Virtex IV de Xilinx, en el que se carga un archivo bitstream de los modelos creados en Simulink para su respectiva implementación.

El proyecto se lo ha estructurado en 4 capítulos que se detallan a continuación:

En el capítulo 1, damos a conocer los objetivos generales y específicos del proyecto, planteando los alcances y limitaciones.

En el capítulo 2, se describirá la parte teórica, dando a conocer los conceptos básicos y necesarios para poder entender y desarrollar las prácticas elaboradas.

En el capítulo 3, se muestra la descripción, diseño y la simulación del proyecto, el cual consiste en los sistemas de comunicación que se desarrollarán, describiendo detalladamente cada etapa del sistema.

En el capítulo 4, se presenta la implementación de cada uno de los sistemas de comunicación ensamblados en la tarjeta FPGA, usando equipos del laboratorio de telecomunicaciones para la respectiva visualización.

En el capítulo 5, se muestra un análisis detallado de cada sistema desarrollado, tanto en simulación como en implementación y se establece una comparación entre ellos analizando valores de SNR y BER.

ÍNDICE GENERAL

AGRADECIMIENTO.....	II
DEDICATORIA.....	IV
TRIBUNAL DE SUSTENTACIÓN.....	V
DECLARACIÓN EXPRESA.....	VI
RESUMEN.....	VII
ÍNDICE GENERAL.....	IX
ÍNDICE DE FIGURAS.....	XIV
ÍNDICE DE TABLAS.....	XX
ABREVIATURAS.....	XXIII
INTRODUCCIÓN.....	XXV
CAPÍTULO 1.....	1
PLANTEAMIENTO DEL PROBLEMA.....	1
1.1 Antecedentes.....	2
1.2 Objetivo General.....	2

1.3 Objetivos específicos.....	2
1.4 Justificación	3
1.5 Metodología	4
1.6 Descripción del proyecto.....	5
CAPÍTULO 2.....	7
MARCO TEÓRICO	7
2.1 Tarjeta “XTREME DSP DEVELOPMENT KIT”.....	7
2.1.1 Características del hardware.....	7
2.1.2 Especificaciones del software.....	12
2.2 Filtros.....	16
2.2.1 Clasificación de filtros.....	16
2.2.2 Filtro IIR tipo butterworth.....	19
2.2.3 Filtro FIR tipo LMS.....	21
2.3 Lazo de enganche de fase.....	26

2.3.1 Características y componentes.....	26
2.3.2 Aplicaciones.....	28
2.4 CDMA (Acceso Múltiple Por Division De Código).....	29
2.4.1 Breve historia.....	29
2.4.2 Características y desarrollo.....	30
2.4.3 Arquitectura de una red CDMA	33
CAPÍTULO 3.....	36
DISEÑO DE LOS MODELOS DE COMUNICACIÓN USANDO EL BLOCKSET DE XILINX EN SIMULINK.....	36
3.1 Diseño de filtros digitales	36
3.1.1 Diseño de filtro IIR tipo butterworth	39
3.1.1.1 Descripción y modelo.....	39
3.1.1.2 Simulación del modelo.....	46
3.1.2 Diseño de filtro FIR tipo LMS	48
3.1.2.1 Descripción y modelo.....	48

3.1.2 Simulación del modelo.....	58
3.2 Diseño de Lazo de enganche de fase.....	61
3.2.1 Descripción y modelo.....	61
3.2.2 Simulación del modelo.....	66
3.3 Diseño de un modulador/demodulador 16QAM.....	66
3.3.1 Descripción y modelo.....	68
3.3.2 Simulación del modelo.....	94
CAPÍTULO 4.....	101
IMPLEMENTACIÓN DE LOS MODELOS DE COMUNICACIÓN USANDO LA TARJETA "XTREME DSP DEVELOPMENT KIT".....	101
4.1 Implementación de filtros digitales.....	101
4.1.1 Implementación de filtro IIR tipo butterworth.....	101
4.1.2 Implementación de filtro FIR tipo LMS.....	103
4.2 Implementación de Lazo de enganche de fase.....	103
4.3 Implementación de un modulador/demodulador 16QAM.....	105

CAPÍTULO 5.....	107
PRUEBAS Y ANÁLISIS DE LOS RESULTADOS.....	107
5.1 Filtros digitales.....	107
5.1.1 Filtro IIR tipo butterworth.....	108
5.1.2 Filtro FIR tipo LMS.....	109
5.2 Lazo de enganche de fase.....	109
5.3 Modulador/Demodulador 16QAM.....	112
CONCLUSIONES	
RECOMENDACIONES	
BIBLIOGRAFÍA	
ANEXOS	

ÍNDICE DE FIGURAS

Figura 2.1 Tarjeta XTREME DSP DEVELOPMENT KIT	8
Figura 2.2 Diagrama de la tarjeta Xtreme DSP Development Kit, Main y reloj de la FPGA	10
Figura 2.3(a) Bloque sintetizable	12
Figura 2.3(b) Bloque no sintetizable	12
Figura 2.4 Ventana de configuración del System Generator	13
Figura 2.5 Blockset de Xilinx y convertidores ADC, DAC	14
Figura 2.6 Ventana del Fuse Probe	15
Figura 2.7 Proceso de filtrado digital de una señal analógica	17
Figura 2.8 Componentes de un sistema de filtrado adaptativo	18
Figura 2.9 Esquema de implementación de un filtro IIR	19
Figura 2.10 Gráfica de respuesta en frecuencia para tres tipos de filtros pasabajos Butterworth	20
Figura 2.11 Esquema de implementación de un filtro FIR	21
Figura 2.12 Gráfica de la superficie de error	23

Figura 2.13 Modelo filtro FIR adaptativo LMS.....	25
Figura 2.14 Diagrama de bloques del PLL.....	26
Figura 2.15 Sistema CDMA con códigos ortogonales.....	32
Figura 2.16 Arquitectura de una red CDMA	34
Figura 3.1 Generación de audio con ruido blanco gaussiano	37
Figura 3.2 Configuración de parámetros del FDATool.....	38
Figura 3.3 Exportación del filtro a Simulink con bloques básicos.....	41
Figura 3.4 Primera sección de la estructura del filtro pasa bajos creado con bloques de Simulink	42
Figura 3.5 Primera sección de la estructura del filtro pasa bajos creado con el blockset de Xilinx.	43
Figura 3.6 Diseño para la simulación del filtro IIR tipo Butterworth.....	44
Figura 3.7 Simulación de la señal con ruido y señal filtrada.	47
Figura 3.8 Espectro de la señal de salida.	47
Figura 3.9 Algoritmo LMS	55
Figura 3.10 Diseño de la simulación del filtro FIR tipo LMS.....	57

Figura 3.11 Simulaciones de las gráficas de entrada, memoria llena, salida, memoria vacía y habilitadora del bloque FIFO	58
Figura 3.12 Simulaciones de las gráficas de entrada, salida y del puerto rdf del filtro “variable”	59
Figura 3.13 Espectro de la señal $Y(n)$	60
Figura 3.14 Diseño del integrador con bloques de Xilinx	64
Figura 3.15 Diseño del PLL con bloques de Xilinx.....	65
Figura 3.16 Simulación de la señal de entrada y salida del PLL	66
Figura 3.17 Diagrama de bloques de un modulador QAM.....	67
Figura 3.18 Diagrama de bloques de un demodulador QAM.....	67
Figura 3.19 (a) Tabla de verdad del canal I	69
Figura 3.19 (b) Tabla de verdad del canal Q	70
Figura 3.20 Conexión de los multiplexores para agrupar los canales I-Q	70
Figura 3.21 Conexión de los bloques que conforman el convertidor 2 a L niveles para el canal I	75
Figura 3.22 Conexión de los bloques que conforman el convertidor 2 a L niveles para el canal Q	76

Figura 3.23 Conexión de los bloques del oscilador local y desfasador	79
Figura 3.24 Conexión del bloque multiplicador del modulador	81
Figura 3.25 Conexión del bloque sumador del modulador.....	82
Figura 3.26 Conexión del bloque multiplicador del demodulador.....	84
Figura 3.27 Conexión del filtro del demodulador.....	87
Figura 3.28 Conexión de los bloques del Convertidor L a 2 niveles para el canal I y Q respectivamente	90
Figura 3.29 Conexión del sumador del demodulador	93
Figura 3.30 Gráficas del canal I, canal Q, señal de entrada y reloj.....	94
Figura 3.31 Gráfica del canal I, canal I multinivel, canal Q y canal Q multinivel respectivamente.....	95
Figura 3.32 Gráfica del canal I multinivel, canal I con portadora, canal Q multinivel y canal Q con portadora.....	95
Figura 3.33 Gráfica del canal I-Q con portadora y señal modulada	96
Figura 3.34 Constelación 16QAM en simulación	96
Figura 3.35 Gráfica de la señal modulada, portadora y canal I en alta frecuencia	97

Figura 3.36 Gráfica de la señal modulada, portadora y canal Q en alta frecuencia	98
Figura 3.37 Gráfica del canal I-Q en alta frecuencia y del filtro	98
Figura 3.38 Gráfica del canal I con nivel DC, ajuste y recuperación de los bits I.....	99
Figura 3.39 Gráfica del canal Q con nivel DC, ajuste y recuperación de los bits Q.	99
Figura 3.40 Gráfica del canal I recuperado, canal Q recuperado, la señal recuperada y la señal binaria inicial.....	100
Figura 4.1 Señal de audio con ruido gaussiano.....	102
Figura 4.2 Señal con la acción del filtro IIR.....	102
Figura 4.3 Señal con la acción del filtro FIR.	103
Figura 4.4 Señal referencia y VCO (8Khz).....	104
Figura 4.5 Señal referencia y VCO (10Khz).....	104
Figura 4.6 Señal modulada 16QAM.....	105
Figura 4.7 Constelación 16QAM en implementación.....	106
Figura 4.8 Señal demodulada	106

Figura 5.1 Enganche del PLL de altas a bajas frecuencias.	110
Figura 5.2 Enganche del PLL de bajas hacia altas frecuencias.....	110
Figura 5.3 Determinación del rango de enganche para el límite de baja frecuencia en simulación.	111
Figura 5.4 Determinación del rango de enganche para el límite en alta frecuencia en simulación.	112
Figura 5.5 Enganche del PLL desde el límite inferior.....	113
Figura 5.6 Enganche del PLL desde el límite superior.....	113
Figura 5.7 Determinación del rango de enganche para el límite de baja frecuencia en implementación.	114
Figura 5.8 Determinación del rango de enganche para el límite de alta frecuencia en implementación.	115
Figura 5.9 Sistema con canal AWGN.	115
Figura 5.10 Esquema del canal AWGN.	116

ÍNDICE DE TABLAS

Tabla I – Relación de la numeración mostrada con los pines de la FPGA ...	11
Tabla II – Configuración de parámetros del FDATool para el diseño del filtro IIR	39
Tabla III – Configuración de parámetros del diseño del filtro IIR tipo Butterworth.....	45
Tabla IV – Parámetros del FDATool para el diseño del filtro de renovación de coeficientes	48
Tabla V – Parámetros del FDATool para el filtro que genera la señal deseada $d(n)$	50
Tabla VI – Configuración de parámetros del FIR Compiler 5.0 del filtro que genera la señal deseada.....	52
Tabla VII – Configuración de parámetros del FIR Compiler 5.0 del filtro variable.	53
Tabla VIII – Configuración de parámetros del bloque FIFO.....	56
Tabla IX – Configuración del FPATool para el filtro FIR pasabajo	62
Tabla X – Configuración del FIR Compiler para el diseño del filtro.....	63

Tabla XI – Configuración del bloque Slice	65
Tabla XII – Agrupación de los bits en los canales I-Q.....	69
Tabla XIII – Configuración de parámetros de los bloques que forman el bloque Divisor de datos.	72
Tabla XIV – Agrupación de bits para obtener la señal unipolar multinivel	73
Tabla XV – Tabla de verdad para los valores de la señal multinivel unipolar para 16QAM.....	74
Tabla XVI – Tabla de verdad para los valores de la señal multinivel bipolar para 16QAM.....	75
Tabla XVII – Configuración de parámetros de los bloques que conforman el convertidor 2 a L niveles.	77
Tabla XVIII – Configuración de parámetros de los bloques que conforman el oscilador y desfasador.	80
Tabla XIX – Configuración de parámetros del bloque multiplicador del modulador.....	82
Tabla XX – Configuración de parámetros del bloque sumador del modulador	83

Tabla XXI – Configuración de parámetros del bloque multiplicador del demodulador.....	85
Tabla XXII – Configuración de parámetros del bloque FDATool.....	86
Tabla XXIII – Configuración de parámetros del filtro del demodulador	88
Tabla XXIV – Separación de los bits de la señal multinivel unipolar.....	89
Tabla XXV – Configuración de los bloques del convertidor L a 2 niveles	92
Tabla XXVI – Configuración del bloque sumador del demodulador.....	93
Tabla XXVII – Valores de SNR simulados para diferentes potencias de ruido	114
Tabla XXVIII – Valores de SNR implementados para diferentes potencias de ruido.....	116

ABREVIATURAS

ADC	Convertidor Analógico Digital
AWGN	Ruido Aditivo Blanco Gaussiano
BER	Tasa de Error por Bit
BSC	Controlador de Estación Base
BSM	Administrador de Estación Base
CDMA	División de Código por Acceso Múltiple
DAC	Convertidor Digital Analógico
DS	Secuencia Directa
FH	Salto de Frecuencia
FIFO	Primera Entrada Primera Salida
FIR	Respuesta Finita al Impulso
FPGA	Field Programmable Gate Array
HLR	Registro de Ubicación Base
IIR	Respuesta Infinita al Impulso

J	Superficie de Error
LMS	Algoritmo de Mínimos Cuadrados
MC	Multiportadora
MS	Estación Móvil
MTX	Central de Telefonía Móvil
PLL	Lazo de Enganche de Fase
PSK	Modulación por Desplazamiento de Fase
PSTN	Red de Teléfono Conmutada Pública
QPSK	Modulación por Desplazamiento Cuadrafásica
SNR	Relación Señal Ruido
TH	Salto en el Tiempo
W	Vector de Coeficientes

INTRODUCCIÓN

Es imprescindible que un estudiante de la carrera de Telecomunicaciones entienda el funcionamiento de cada bloque que conforma un sistema de telecomunicación, sea del tema o alcance que sea, los conceptos teóricos deben ser sólidos para incluso desarrollar nuevos modelos, y la mejor forma de afianzar el aprendizaje es mediante prácticas.

En el laboratorio de telecomunicaciones se desarrollan prácticas de modulaciones en las que se emplean módulos, donde solo se realiza la captura de gráficas en cada sección de los bloques descritos en el mismo, lo ideal sería que cada estudiante pueda programar cada uno de estos bloques para obtener la modulación o señal deseada.

Actualmente muchas tecnologías están basadas en los dispositivos FPGA (Field Programmable Gate Array), por su versatilidad y por permitir a los usuarios programar de acuerdo a sus requerimientos y necesidades.

La formación integral de las bases conceptuales de un estudiante se complementa con la práctica, para ello se han diseñado prácticas nuevas de laboratorio para complementar y afianzar la teoría vista en las sesiones de clases a lo largo de la carrera de telecomunicaciones.

CAPÍTULO 1

PLANTEAMIENTO DEL PROBLEMA

1.1 Antecedentes

Desde sus inicios, la problemática que siempre se ha tratado de mitigar en el campo de las telecomunicaciones es recuperar de forma total y eficiente la información que ha sido enviada desde algún punto transmisor, ya sean bits u ondas electromagnéticas a un receptor, el objetivo es de poder captar la señal enviada en su plenitud con la mínima cantidad de errores posible, debido a esta necesidad este sector ha progresado y evolucionado muy rápido durante estos últimos años, se han desarrollado diferentes elementos importantes para poder solucionar este

problema, tales como moduladores, demoduladores, filtros y diferentes sistemas de comunicación y a medida que pasa el tiempo se siguen desarrollando nuevas tendencias que cada vez mejoran este campo tan complejo de las telecomunicaciones.

1.2 Objetivo General

Simular sistemas de comunicación a través del blockset de Xilinx en Simulink/Matlab e implementar dichos sistemas usando la tarjeta XTREME DSP DEVELOPMENT KIT, con el fin de desarrollar prácticas para el Laboratorio de Telecomunicaciones de la Facultad de Ingeniería de Electricidad y Computación de la ESPOL.

1.3 Objetivos Específicos

- Mostrar la aplicación del PLL en los sistemas de comunicación utilizados en cada práctica y la importancia para los sistemas en comunicaciones en general.
- Determinar la eficiencia de la modulación y demodulación 16QAM calculando el SNR.
- Determinar la probabilidad de error (BER) y la relación señal ruido (SNR) en la modulación y demodulación CDMA.

- Desarrollar dos tipos de filtros (IIR tipo butterworth, FIR tipo lsm) hacer una comparación de la relación señal ruido (SNR) y la probabilidad de error (BER).

1.4 Justificación

A través de los años se han ido desarrollando una variedad de sistemas de telecomunicaciones, con el desarrollo de este proyecto se quiere implementar algunas de ellas, tanto como multiplexación, modulación y demodulación con el fin de garantizar un mejor aprendizaje respecto a las telecomunicaciones y transmisión de datos. El elemento principal a utilizar para desarrollar las modulaciones es la tarjeta XTREME DSP DEVELOPMENT KIT, lo cual es una ventaja ya que actualmente este tipo de tarjetas son muy utilizadas para generar prototipos, pruebas, experimentos e incluso nuevas tecnologías, además de que Xilinx es una de las marcas líderes en venta de FPGAs junto a Altera, Atmel, QuickLogic, entre otros.

Esta guía es un aporte muy importante para la carrera, ya que se pone en práctica los conocimientos y conceptos adquiridos en cursos anteriores como modulación, demodulación, multiplexación, técnicas de transmisión y uso de filtros digitales.

Con el uso de Simulink podemos diseñar y simular algoritmos que nos permita un entendimiento del tema a tratarse, con señales creadas dentro del programa, pero, usando el toolbox de "Xilinx Design Tool " se crea un código bitstream para FPGA's, este permitirá crear un prototipo que trabaje con señales reales y verificar los resultados obtenidos en ambos casos.

Teniendo en cuenta el alto nivel de procesamiento de la tarjeta electrónica se puede tener una mayor precisión de datos a tiempo real, elevando así su fidelidad, además el blockset de Xilinx Design Tool puede trabajar con bloques de Simulink teniendo en cuenta ciertas especificaciones.

1.5 Metodología

Cada práctica constará de un formato, y se tomará en cuenta el mismo esquema para desarrollar los temas en cada una de ellas.

ESPECIFICAR LOS OBJETIVOS: para conocer e investigar (de ser el caso) los conceptos a tomar en cuenta para el desarrollo de la práctica.

FUNDAMENTO TEÓRICO: introducción teórica del tema a tratar.

DESCRIPCIÓN DEL MODELO: detallar la programación empleada dando especificaciones de cada bloque de Xilinx utilizado.

EXPERIMENTOS: aquí se tendrán dos tipos de experimentos, una que constará de la simulación del sistema, y la segunda parte la implementación física del mismo.

SIMULACIÓN

- Se especifica las modificaciones en cuanto a programación, además se detalla la forma de obtener una buena simulación con sus respectivos resultados usando Simulink/Matlab.

IMPLEMENTACIÓN

- Se especifica las modificaciones en cuanto a programación, además se detalla la forma de programar e introducir señales en la FPGA para luego obtener los resultados usando el software FUSE PROBE junto a la tarjeta XTREME DSP DEVELOPMENT KIT.

ANÁLISIS DE LOS RESULTADOS

1.6 Descripción del proyecto

Este proyecto surge de la necesidad de dar a conocer una de las varias tecnologías que se puede utilizar en sistemas de telecomunicación, como es el caso de la tecnología FPGA aplicada a tipos y sistemas de modulación, se quiere dar una guía detallada de estos temas usando este tipo de tarjetas para facilitar el aprendizaje del mismo.

Se realizará un manual de prácticas para el laboratorio de Telecomunicaciones, el cual tendrá 2 prácticas referentes a sistemas de telecomunicaciones basados en programación desarrollada en Simulink de Matlab, como herramienta de trabajo se utiliza un toolbox de Xilinx necesario para la compatibilidad de la FPGA Virtex 4 vsx35-10FF668 dentro de la tarjeta XTREME DSP DEVELOPMENT KIT. Las prácticas serán referentes al PLL, que es un elemento importante e indispensable en las comunicaciones, filtros digitales (IIR tipo butterworth, FIR tipo lsm), y una modulación importante utilizada mucho actualmente por las telefonías celulares como CDMA en el análisis de estas prácticas.

CAPÍTULO 2

MARCO TEÓRICO

2.1 Tarjeta “XTREME DSP DEVELOPMENT KIT”

2.1.1 Características del hardware

La tarjeta de desarrollo XtremeDSP consiste en una placa base con un módulo (tarjeta secundaria), la placa base se conoce como la "placa base benone-Kit" y el módulo se lo llama el "módulo de BenADDA DIME-II".

XtremeDSP Kit-IV es una plataforma de desarrollo ideal para la FPGA Virtex-4, contiene conversores analógicos – digitales, y digitales – analógicos, dos FPGA's (uno para programar la función

que se desea realizar y el otro para generar la señal de reloj de todo el sistema); éstos kits son ideales para implementar aplicaciones de procesamiento de señales de alto rendimiento como: Software Defined Radio, 3G Wireless, Redes, televisión de alta definición o imágenes de vídeo. [1]

En la Figura 2.1 se muestra la tarjeta XTREME DSP DEVELOPMENT KIT utilizada en el desarrollo del proyecto.

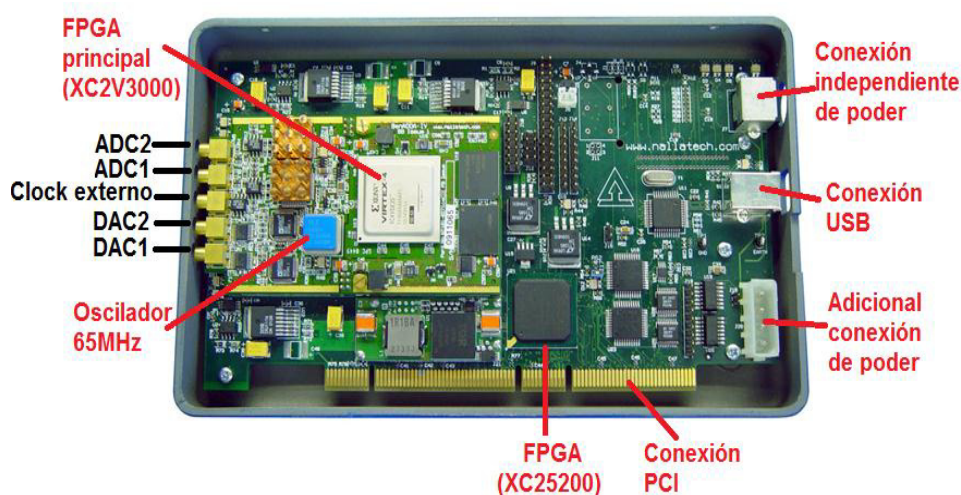


Figura 2.1 Tarjeta XTREME DSP DEVELOPMENT KIT. [2]

Las características técnicas más importantes de la tarjeta son:

- FPGA Virtex 4 vsx35-10FF668.
- canales ADC AD6644 (14 bits, 105Mps).

- canales DAC AD9772 (14 bits, 160Msps).
- Soporte para reloj externo, el oscilador a bordo y relojes programables.
- Dos bancos de memoria ZBT-SRAM (133 MHz, 512 K x 32 bits por banco).
- Interfaz PCI 32-bit/33 MHz o interfaz JTAG
- LEDs de estado

INTERFACE ENTRE UN ADC Y LA FPGA

Características más importantes:

- 14 bits en complemento a dos.
- Velocidad de muestreo máxima 105Msps.
- $Z_{in} = 50 \text{ ohm}$.
- Trabaja con reloj interno programable.
- SNR 74.5 dB.

INTERFACE ENTRE UN DAC Y LA FPGA

Características más importantes:

- 14 bits en complemento a dos.

- Velocidad de conversión 160Msps.
- $Z_{out} = 50 \text{ ohm}$.
- PLL interno.
- Trabaja con reloj interno programable.

RECURSOS DEL RELOJ

La tarjeta posee relojes internos con los cuales trabajar, además de un medio externo en el cual se puede introducir una señal de reloj a la tarjeta.

En la Figura 2.2 se muestra los diferentes relojes que se pueden configurar.

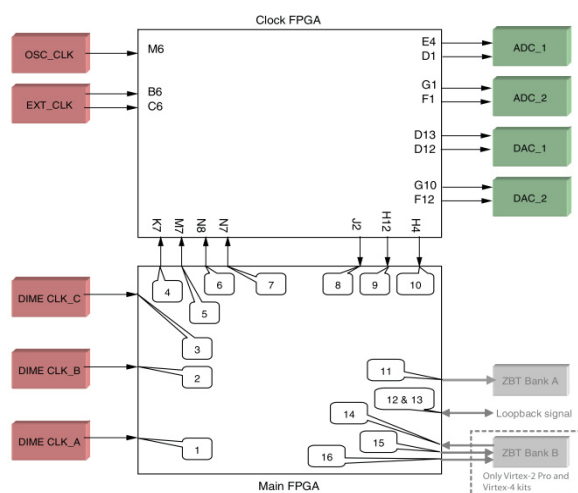


Figura 2.2. Diagrama de la tarjeta Xtreme DSP Development Kit,

Main y reloj de la FPGA. [3]

La tabla I muestra el pin del “Main FPGA” a configurar para utilizar los distintos tipos de reloj que se muestran, teniendo en cuenta que la versión de la FPGA es XC4VSX35-10FF668.

Número desde la figura	Nombre al pin asociado	XC4VSX35-10FF668
1	CLKA	AF12
2	CLKB	A16
3	CLKC	AF11
4	GEN_CLKA	B13
5	GEN_CLKB	C15
6	GEN_CLKC	B12
7	GEN_CLKD	C14
8	CLK1_FB	B14
9	CLK2_FB	A15
10	CLK3_FB	B15
11	ZBT_CLK	AB10
12	ZBT_FB_OUT	AC10
13	ZBT_FB_IN	AE12
14	ZBTB_CLK	AE14
15	ZBTB_CLK_FB_OUT	AB17
16	ZBTB_CLK_FB_IN	AC17

Tabla I. Relación de la numeración mostrada con los pines de la
FPGA.[3]

2.1.2 Especificaciones del software

Los modelos se desarrollan en Simulink de Matlab con el blockset de Xilinx, tomando en consideración la compatibilidad de Matlab, Xilinx y el sistema operativo, en este caso se trabaja con Matlab R2012a, Xilinx ISE 14.3 con Windows 7.

Dentro de la librería de Xilinx en Simulink se encuentra un bloque llamado System Generator, el cual es una herramienta de diseño de sistemas proporcionando abstracciones de alto nivel que se pueden compilar automáticamente en un FPGA. Sin embargo para que System Generator pueda implementar un sistema necesita la colaboración del paquete ISE de Xilinx.

Cabe mencionar que para la implementación de un sistema en una FPGA, System Generator emplea solo los bloques sintetizables, los no sintetizables solo sirven para la simulación del modelo. En la Figura 2.3 (a) y (b) se muestran ejemplos de bloques sintetizables y no sintetizables.

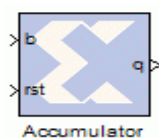


Figura 2.3 (a) Bloque sintetizable



Figura 2.3 (b) Bloque no sintetizable

El bloque de System Generator debe estar presente en cada diseño dentro del entorno de Simulink, se necesita compilar el modelo para obtener el archivo de configuración del FPGA, por medio de la integración de System Generator con Xilinx ISE. En la Figura 2.4 se muestra la ventana de configuración del System Generator, éste sintetiza el sistema a VHDL generando el archivo bitstream empleado para programar la FPGA.

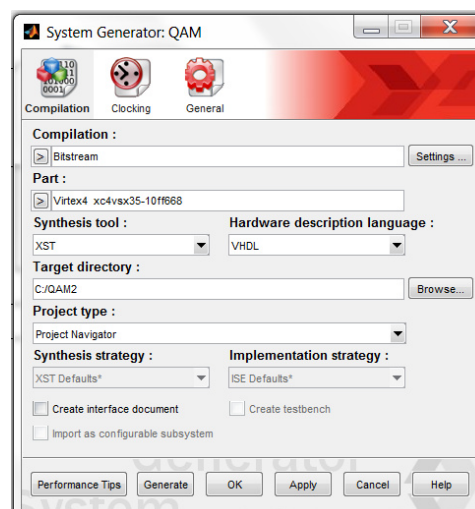


Figura 2.4 Ventana de configuración del System Generator

Para utilizar los convertidores ADC y DAC se necesita ubicar ciertos bloques en el sistema, que se encuentran en la librería de Simulink, propiamente en los blocksets de Xilinx (Ver Figura 2.5).

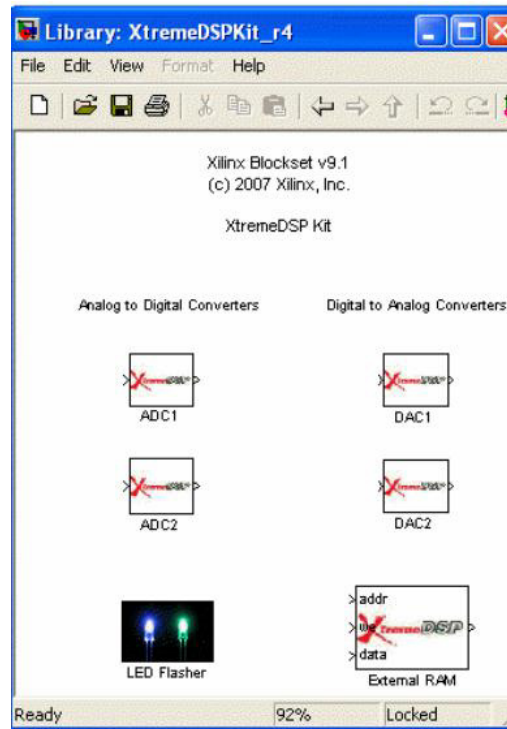


Figura 2.5 Blockset de Xilinx y convertidores ADC, DAC

En cuanto a la programación de las FPGA's, el fabricante proporciona un software específico denominado Fuse Probe, la programación es a través del bus PCI. La Figura 2.6 muestra la ventana del Fuse Probe en proceso de programación de la tarjeta.

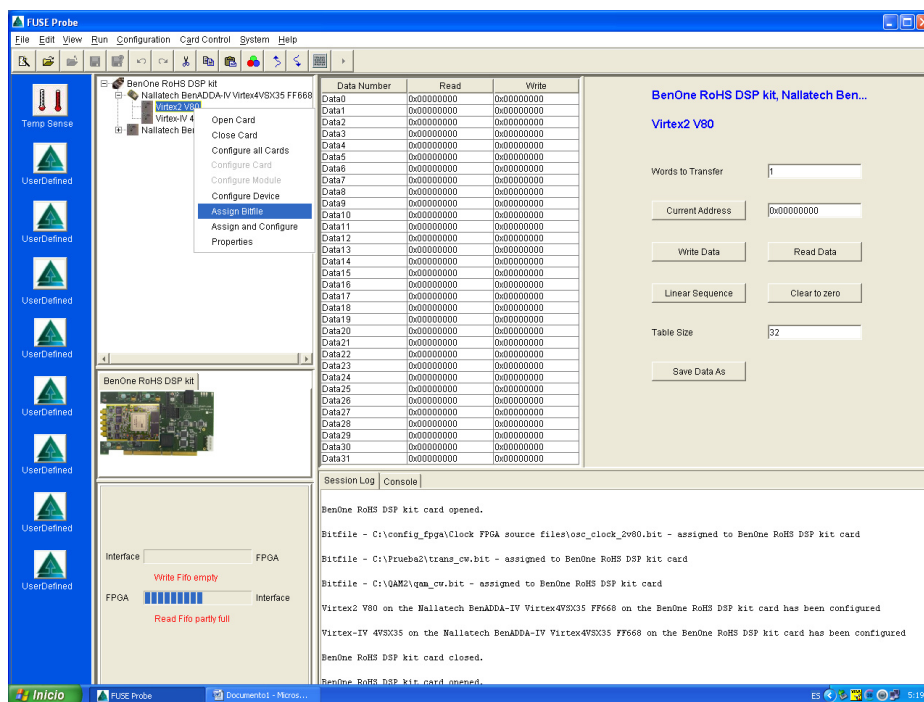


Figura 2.6 Ventana del Fuse Probe

A través de este software se ingresa a la FPGA el archivo bitstream generado a través del System Generator y así disponer del modelado del sistema desde la tarjeta, ya no se requiere de Matlab y se podrán obtener e introducir señales desde y hacia sus puertos de entrada y salida, facilitando la observación de la implementación de los sistemas.

2.2 Filtros

Un filtro es un “dispositivo” diseñado para dejar pasar ciertas partes de una señal y retener otras; pudiendo ser una determinada frecuencia o gama de frecuencias, consiguiendo modificar tanto la fase como la amplitud.

El filtrado “es el proceso de seleccionar, suprimir o atenuar ciertas componentes de una señal” [4]. El propósito es de separar componentes de una señal que la distorsionen.

2.2.1 Clasificación de los filtros

Una clasificación general de los filtros son analógicos y digitales, los primeros dedicados a las señales analógicas (valores dentro de un intervalo) y los segundos a las señales digitales (datos discretos). Los filtros digitales tienen diferentes clasificaciones: de acuerdo a su respuesta en frecuencia se clasifican en Pasa-bajos, Pasa-altos, Pasa-banda y Rechazo de banda; de acuerdo a su respuesta ante una entrada impulso se dividen en IIR (Respuesta Infinita al Impulso) y FIR (Respuesta Finita al Impulso); y finalmente si se los analiza de acuerdo a su estructura, se clasifican en cascada, serie y laticce.

Para filtrar señales analógicas, estas se las convierten a una señal digital a través de un convertidor Analógico-Digital, se realiza el

proceso de filtrado digital y finalmente se convierte nuevamente a una señal analógica (Ver Figura 2.7). Debido a este proceso es que surgen los filtros digitales, tanto en IIR como en FIR.[5]

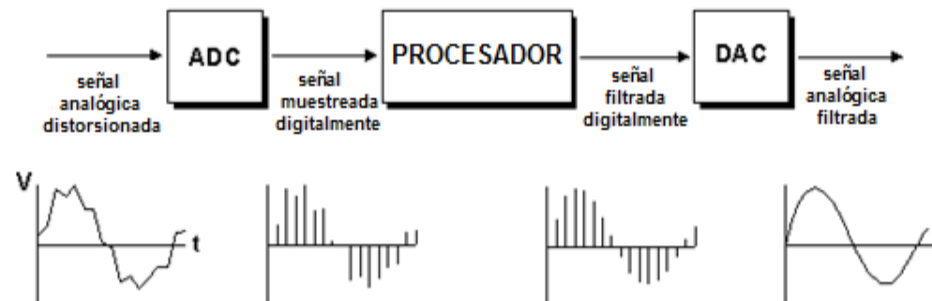


Figura 2.7 Proceso de filtrado digital de una señal analógica.[4]

Una característica fundamental de los filtros digitales es la de poder cambiar su comportamiento, es decir sus coeficientes, éstos cambian de valor a medida que se actualiza la información que disponen, siguiendo un procedimiento llamado algoritmo adaptativo. Cuando se diseña el filtro no se conoce el valor de los coeficientes, estos se calculan al implementarlo y se van actualizando en cada iteración mientras dura su etapa de aprendizaje. [6]

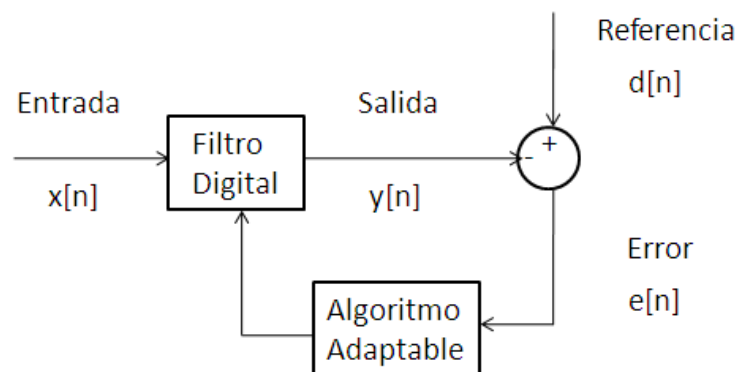


Figura 2.8 Componentes de un sistema de filtrado adaptativo

En la Figura 2.8 se muestra el proceso de un filtro adaptativo, donde $x(n)$ es la señal de entrada, $d(n)$ es la señal deseada (óptima), $y(n)$ es la salida del filtro y $e(n)$ es la señal de error, el cual se define como la diferencia entre la señal deseada y la señal de salida. **[6]**

El LMS (Algoritmo de mínimo cuadrado) es un algoritmo del filtro adaptativo, el cual se ha propuesto para adaptar el orden y los coeficientes del filtro simultáneamente. En este proyecto de tesis se desarrollarán los modelos del filtro IIR tipo Butterworth además del filtro FIR tipo LMS.

2.2.2 Filtro IIR tipo Butterworth

En los filtros IIR (Respuesta Infinita al Impulso) o respuesta infinita al impulso, como su nombre indica, si la entrada fuese una señal impulso, la salida tendría un número infinito de términos no nulos, es decir, nunca vuelve al reposo. Son llamados filtros recursivos, porque la salida del filtro depende de las entradas actuales y de las salidas en instantes anteriores, esto se logra a través de realimentación de la salida como se muestra en la Figura 2.9.

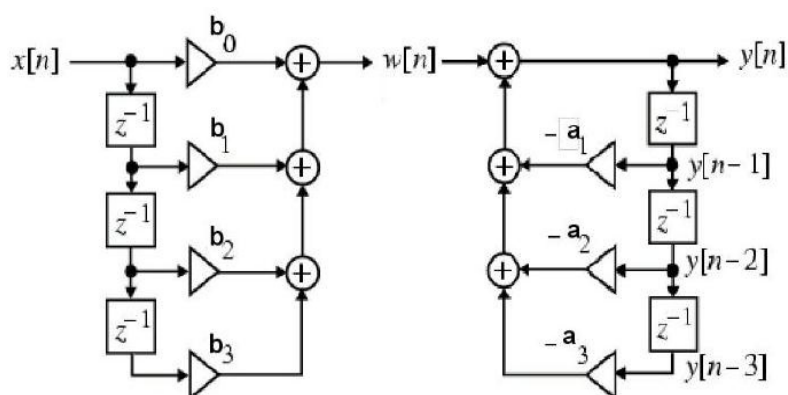


Figura 2.9 Esquema de implementación de un filtro IIR [7]

Donde a y b son los coeficientes del filtro. El orden es el máximo entre los valores de M y N respectivamente para a y b .

M y N son los términos que determinan la cantidad de polos y ceros en la función de transferencia (Ver ecuación 1).

$$H(z) = \frac{\sum_{k=0}^{N-1} b_k z^{-k}}{1 + \sum_{k=0}^{M-1} a_k z^{-k}} \quad (1)$$

Debido a la presencia de polos el filtro se puede volver inestable, además de que la implementación física es más compleja, no garantizan que su función de transferencia sea lineal.

Dentro de los filtros IIR está el filtro de Butterworth, denominado también filtro de máximo plano o plano-plano, es aquel cuya salida se mantiene constante casi hasta la frecuencia de corte.

En la Figura 2.10 se muestra la respuesta en frecuencia ideal (línea continua) y la práctica (líneas punteadas) para tres tipos de filtros Butterworth, conforme las pendientes se vuelven más pronunciadas se aproximan más al filtro ideal.

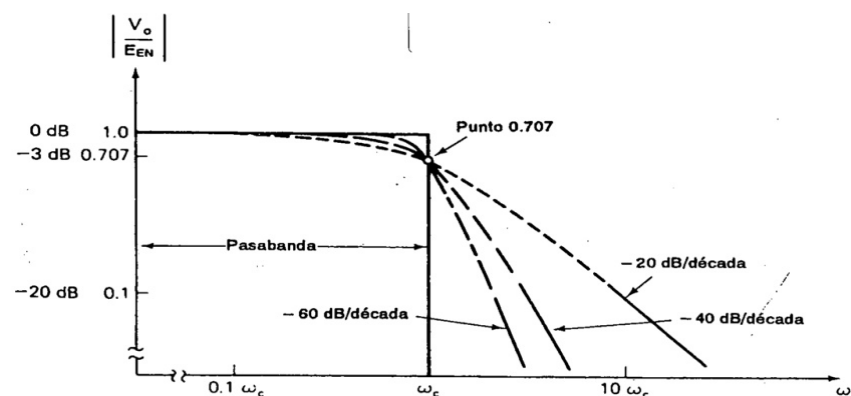


Figura 2.10 Gráfica de respuesta en frecuencia para tres tipos de filtros pasabajos Butterworth. [8]

Cabe destacar que un filtro Butterworth puede ser pasa bajos, pasa altos, pasa banda o rechaza banda; para este proyecto se desarrollará el filtro Butterworth pasa bajos.

2.2.3 Filtro FIR tipo LMS

Los filtros FIR (Finite impulse response) o respuesta finita al impulso, tienen la particularidad de que sus coeficientes son cero, lo que significa que la respuesta del filtro depende solamente de la entrada y no de valores pasados de la salida. Este tipo de filtros tiene una respuesta finita ya que no exhiben recursión (Ver Figura 2.11).

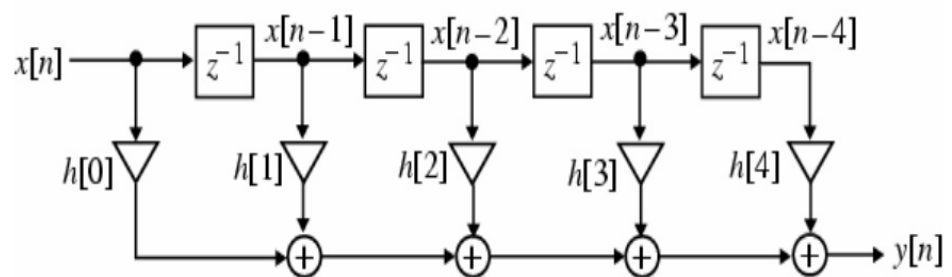


Figura 2.11 Esquema de implementación del filtro FIR.[7]

Una de las propiedades de este filtro es la simetría de sus coeficientes y también la ventaja de poder ser diseñados de tal forma de exhibir una respuesta de fase lineal siendo su función de transferencia. (Ver ecuación 2)

$$\mathbf{H}(z) = \sum_{n=0}^N \mathbf{b}_n z^{-n} \quad (2)$$

El filtro FIR es recurrentemente utilizado en sistemas adaptativos debido a su respuesta de fase y su estructura no recursiva por lo que la complejidad computacional se reduce.

Para poder minimizar el error de la señal de entrada se determina el concepto de superficie de error cuadrático medio J , la cual es una función de los coeficientes del filtro, determinado por:

$$J = E\{e^2[n]\} \quad (3)$$

Donde $E\{e^2[n]\}$ es el esperado del cuadrado del error, el algoritmo LMS determina el mínimo del cuadrado de la señal de error, por medio del método de descenso de gradiente, el cual ajusta los coeficientes a manera de pasos que minimice el error.

El vector de coeficientes (W) forma un punto en la superficie de error, como se observa en la Figura 2.12, y en cada iteración el punto se desplaza por la tangente de dicho punto.

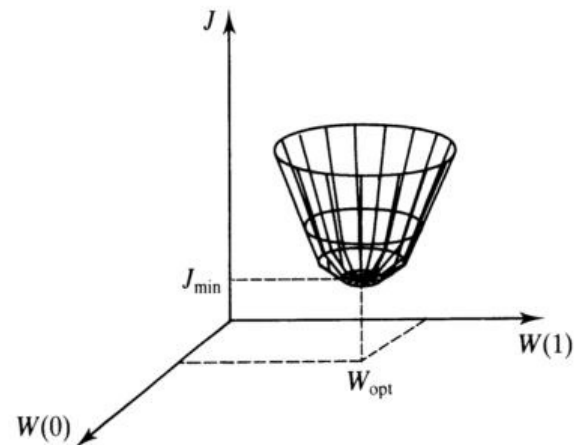


Figura 2.12 Gráfica de la superficie del error. [9]

Para este algoritmo se considera que las señales $d(n)$ y $x(n)$ de la figura 2.8 son estacionarias debido a que este tipo de señales tiene una superficie de error invariable y es más fácil que el algoritmo converja al punto mínimo de la superficie de error (J_{min}).

El concepto del descenso de gradiente se expresa por:

$$w(n+1) = w(n) - \frac{\mu}{2} \nabla \xi[n] \quad (4)$$

Donde μ es factor que controla la estabilidad y el ritmo de descenso al fondo de la superficie de error. El vector $\Delta\xi[n]$ especifica el gradiente de la función de error con respecto a $w[n]$.

$$\nabla\xi[n] = E\left[\frac{\partial e^2(n)}{\partial w_0}, \frac{\partial e^2(n)}{\partial w_1}, \dots, \frac{\partial e^2(n)}{\partial w_{n-1}}\right]^T \quad (5)$$

Obtener este gradiente se dificulta por el operador de expectación el cual requiere conocimiento de la probabilidad de la señal de entrada para poder ser calculado, se usa el error cuadrático instantáneo para estimar el error cuadrático medio (ecuación 6).

$$\xi[n] = e^2[n] \quad (6)$$

De la Figura 2.8 se obtiene:

$$e(n) = d(n) - W^T X(n) \quad (7)$$

Entonces:

$$\nabla e[n] = -x[n] \quad (8)$$

La estimación del gradiente se vuelve:

$$\nabla\xi[n] = -2x[n]e[n] \quad (9)$$

Por lo que la ecuación final es:

$$w(n+1) = w(n) + \mu e(n)x(n) \quad (10)$$

Donde $w(n)$ es el vector de peso, $x(n)$ es la entrada de referencia, $e(n)$ es la señal de error (Figura 2.13). El rango del valor de μ está dado en:

$$0 < \mu \leq \frac{1}{Nx^2(n)} \quad (11)$$

Siendo $x^2(n)$ el cuadrático medio de la potencia de $x(n)$ y N el número de coeficientes del filtro. Mientras más grande es μ la velocidad de convergencia y el error cuadrático medio aumenta.

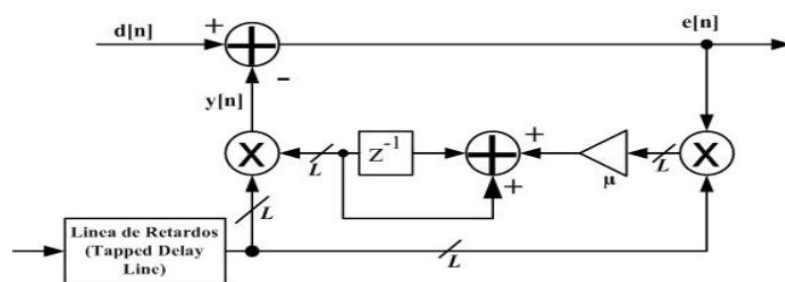


Figura 2.13 Modelo filtro FIR adaptativo LMS [10]

En la Figura 2.13 se observa la estructura básica de un filtro LMS donde L es el orden del filtro variable.

2.3 Lazo de enganche de fase

2.3.1 Características y componentes

El lazo de seguimiento de fase (PLL) es un circuito en el que un oscilador sigue la fase de una señal de entrada, a través de una retroalimentación que compara la fase de las dos señales y modifica la frecuencia de la oscilación generada, consta de tres partes importantes:

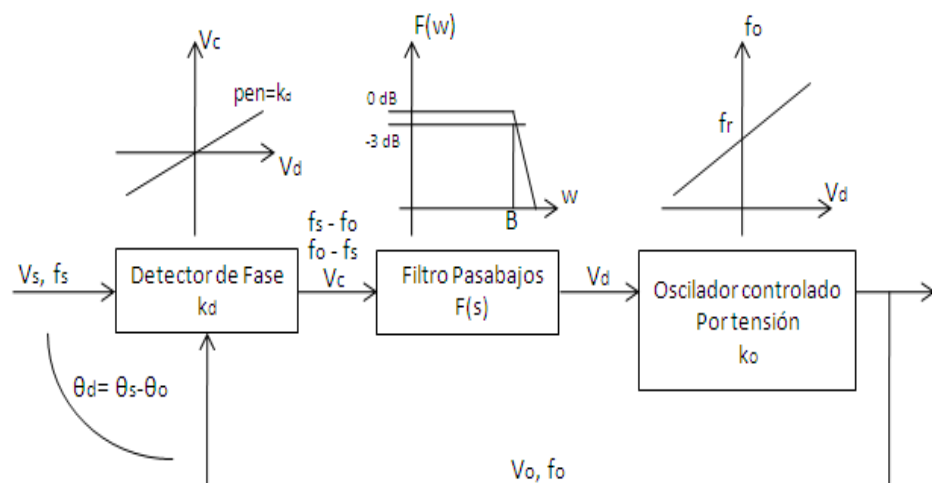


Figura 2.14 Diagrama de bloques del PLL [11]

- Comparador de fase

Dado que el PLL es un lazo de retroalimentación, en esta etapa se compara la fase y la frecuencia de la señal entrante con la

frecuencia del VCO. En el caso de que no hubiera señal entrante, el VCO oscilaría a una frecuencia f_o , que se la conoce como frecuencia de corrida libre o frecuencia libre de oscilación. En el caso de tener una frecuencia a la entrada f_e , el comparador de fases también funciona como un mezclador, dando como resultado una mezcla de ambas frecuencias (f_o-f_e , f_e-f_o , $2f_o$, $2f_e$, f_o+f_e). Siendo $f_o \neq f_e$.

- Filtro pasa bajo

Las mezclas de alta frecuencia, como las componentes sumas arrojadas por el detector de fase (f_o+f_e , $2f_o$, $2f_e$), son anuladas por el filtro pasa bajo por estar fuera de su ancho de banda, este es el trabajo de el filtro, eliminar las frecuencias fuera de banda que pueda arrojar el mezclador de frecuencias en la parte inicial y considerar las que están más cercanas a la frecuencia de oscilación f_o y solo deja pasar la componente DC. Otra función del filtro pasa bajos es de asegurar que el enganche se realice de una manera más rápida y eficiente.

- Oscilador controlado por voltaje

Las frecuencias que salen del filtro pasa bajos ahora ingresan al VCO, si estas frecuencias son las mismas o muy cercanas a la frecuencia libre de oscilación f_o , se da el enganche de fase y $f_o=f_e$, en el caso de que estas frecuencias aun fueran diferentes se repite el proceso hasta que ambas frecuencias se igualen y se dé el enganche.

2.3.2 Aplicaciones

La finalidad de estos tres componentes retroalimentados es la de poder igualar la fase del VCO con la fase de la señal entrante, se detalló el funcionamiento de cada componente para lograr el objetivo del enganche, el PLL es indispensable para los sistemas de telecomunicaciones y para la electrónica, a continuación se detallan algunas aplicaciones importantes del mismo.

- Filtros

A través del funcionamiento del PLL pueden desarrollarse filtros de fase, para poder reconstruir o recuperar señales que hayan sido perturbadas por ruido de fase o fluctuaciones de frecuencia.

- Moduladores/Demoduladores

La finalidad del PLL es enganchar un par de frecuencias, por lo que puede utilizarse este circuito para desarrollar moduladores y demoduladores de señales en frecuencia. También pueden desarrollarse moduladores en ángulo.

- Circuitos de sincronismo para barrido horizontal y vertical
- Generación de osciladores

2.4 CDMA (Acceso Múltiple por División de Código)

2.4.1 Breve historia

Se define como acceso múltiple por división de código CDMA (“Code Division Multiple Access”), es una técnica multiacceso utilizada por los terminales móviles para compartir los recursos comunes de la red, la cual fue desarrollada en la Segunda Guerra Mundial. Era usado en aplicaciones militares debido a que una señal ensanchada es difícil de bloquear, interferir e identificar, ya que la potencia de estas señales está distribuida en un gran ancho de banda y solo aparecen como un ruido ligero, en cambio ocurre lo contrario con el resto de tecnologías que concentran la potencia de la señal en un ancho de banda estrecho, fácilmente detectable.

La TIA (Asociación de Industria de Telecomunicaciones) aprobó el estándar CDMA IS-95 en julio de 1993, a partir de ahí se viene mejorando esta técnica multiacceso.

2.4.2 Características y desarrollo

CDMA otorga a cada usuario toda la anchura de banda, o espectro ensanchado, hay cuatro variantes de CDMA en función de la técnica utilizada para conseguir la expansión espectral [12]:

- 1) Saltos de frecuencia, FH (Frequency Hopping), donde la frecuencia varía en función del código.
- 2) Saltos de tiempo, TH (Time Hopping), donde se varía el intervalo temporal según el código.
- 3) Secuencia Directa, DS (Direct Sequence), en las que la señal de información se multiplica por el código de expansión.
- 4) Multiportadora, MC (Multicarrier), donde cada símbolo de información genera un conjunto de símbolos, según el código, que modulan distintas portadoras.

Cada transmisor ensancha en banda su señal de información utilizando una señal de código propia según la técnica utilizada, al

volver a multiplicar por la señal de código propia se produce el efecto contrario de compresión. Así en el receptor si se multiplica por un código incorrecto la señal no se comprime y es percibida por el receptor como una perturbación similar a un ruido blanco, estas señales con código incorrecto provienen de otros usuarios que utilizan el mismo espectro ensanchado.

Las dos familias más importantes de códigos son los ortogonales y los pseudoaleatorios.

Con los códigos ortogonales se puede separar los usuarios para eliminar la interferencia, el recurso en el que se busca la ortogonalidad es el código.

Un sistema CDMA con códigos ortogonales cumple:

$$\frac{1}{T} \int_0^T C_i(t) C_j(t) dt = \begin{cases} 1, & i = j \\ 0, & i \neq j \end{cases} \quad (12)$$

Donde C_i es el código del usuario dado y C_j es el código de los demás usuarios, se muestra el sistema CDMA con códigos ortogonales (Ver Figura 2.15)

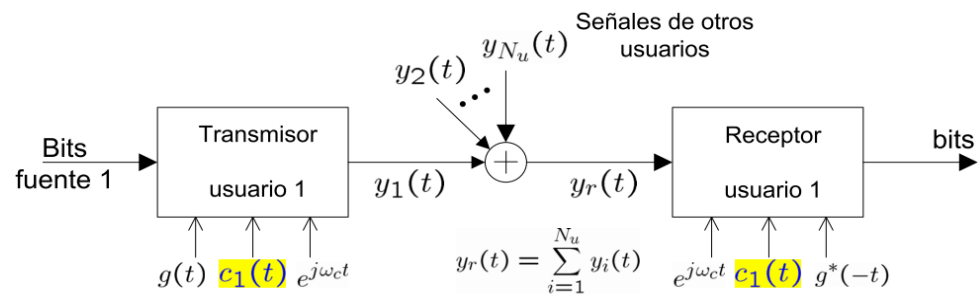


Figura 2.15 Sistema CDMA con códigos ortogonales [13]

Para los sistemas que tienen códigos pseudoaleatorios se utilizan códigos “largos”, de período 38400 chips; también existen códigos “cortos”, de período 256 chips.

En la tecnología CDMA EVDO se dividen los datos en paquetes individuales, cada paquete se envía de forma independiente de todos los otros paquetes como es el caso en saltos de frecuencia. Se utiliza modulación adaptativa a las condiciones de transmisión, quiere decir que si se opera con nivel de señal baja y altas interferencias se usa un tipo de modulación más robusto, entre los tipos de modulación que puede usar están: BPSK, QPSK, 8PSK Y 16 QAM.

La modulación QAM es importante ya que el mensaje no solo está en la variación de fase, sino también en la variación de amplitud; es decir, se logra una transmisión de dos mensajes

independientes por un solo canal, modulando una portadora desfasada 90 grados para cada canal, de esta forma se utiliza todo el ancho de banda posible de forma individual.

La modulación QAM es multinivel, es decir, depende del número de estados, el menor de ellos es 4 y se puede aumentar los estados tanto como se quiera, pero mientras mayor sea el estado, mayor es la probabilidad de error, por lo que se escogió la modulación 16QAM para el análisis de este proyecto, ya que esta se utiliza en los sistemas CDMA y es mucho más eficiente que las modulaciones PSK. Esta modulación se detalla con más precisión en los Capítulos 3 y 4.

2.4.3 Arquitectura de una red CDMA

La arquitectura consta del equipo de conmutación y del equipo de la estación base celular, estos interactúan con la red telefónica conmutada pública (PSTN) y con la estación móvil (MS) para proporcionar un sistema completo de comunicaciones celulares.

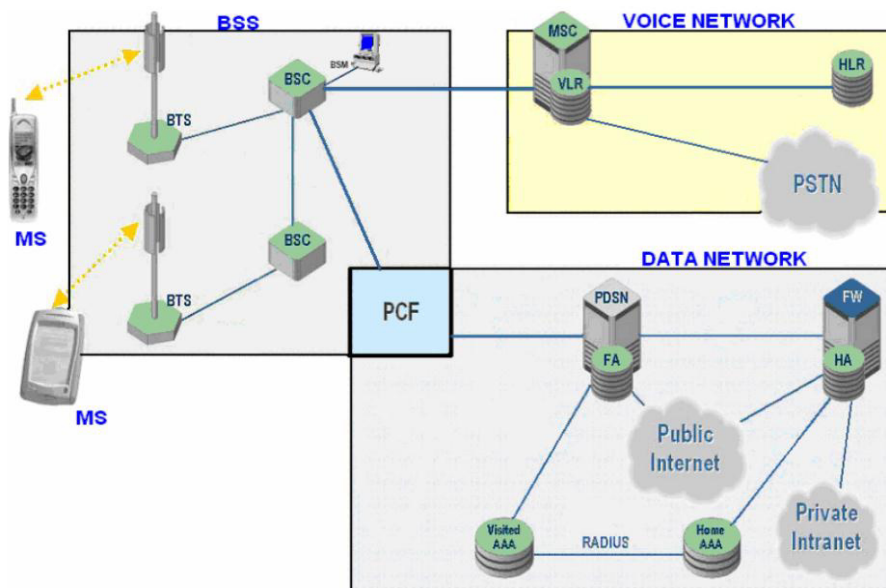


Figura 2.16 Arquitectura de una red CDMA [15]

Los principales subsistemas del sistema son:

- Estación Móvil (MS – Mobile Station).
- La Central de Telefonía Móvil (MTX - Mobile Telephone Exchange) ó Centro de Conmutación Móvil (MSC – Mobile Switching Center).
- El Controlador de Estación Base (BSC – Base Station Controller).
- El Subsistema de Estación Base Transceptora (BTS – Base Station Transceiver Subsystem).

- El Administrador de Estación Base (BSM – Base Station Manager).

- HLR (Home Location Register).- Base de datos con el registro de los suscriptores y sus respectivos perfiles de servicios.

Para realizar y recibir llamadas de voz y datos, la MS se debe registrar con el HLR.

La estación móvil en una llamada de datos funciona como un cliente móvil IP interactuando con la red de acceso para obtener un apropiado recurso de radio para el intercambio de paquetes.

El BSC controla el enrutamiento de mensajes y de señalización entre éste mismo, la MTX, el BSM y la BTS. También proporciona la codificación y decodificación de voz entre la estación móvil (a través de la BTS) y la MTX **[15]**.

CAPÍTULO 3

DISEÑO DE LOS MODELOS DE COMUNICACIÓN USANDO EL BLOCKSET DE XILINX EN SIMULINK

3.1 Diseño de filtros digitales

Una de las formas de eliminar las perturbaciones que sufren las señales digitales o análogas al momento de su transmisión es el filtrado de las mismas, en este caso se utilizan señales de audio que son afectadas por ruido blanco gaussiano, para luego aplicar el filtro FIR tipo LMS además del filtro IIR tipo butterworth y observar el comportamiento de ambos.

El ruido blanco gaussiano se define como un proceso estocástico con media cero, varianza constante y covarianza nula, este tipo de ruido es el más común en la transmisión de señales.

Generación de audio con ruido blanco gaussiano

Para generar una señal de audio con estas características, primero se introduce la señal original a Simulink por medio del bloque “From Multimedia File”, en el cual se elige como salida tipo frame, considerando que esta señal posee un solo canal con un tiempo de muestreo de 1/44100 segundos; que es la frecuencia estándar a la que se muestrean las señales de música. Para generar el ruido blanco con media cero y con varianza de 0.002, se utiliza el bloque “Gaussian Noise Generator” cuyo periodo es el mismo de la señal de audio y su tipo de salida es double. El objetivo es sumar ambas señales para obtener una señal de audio con ruido gaussiano, pero antes hay que filtrar el ruido blanco a través de un filtro pasa alto tal como se especifica en la Figura 3.1

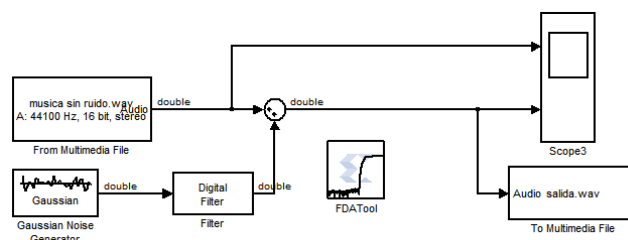


Figura 3.1 Generación de audio con ruido blanco gaussiano.

La razón por la cual se filtra el ruido blanco es para generar el ruido en las frecuencias altas, de tal forma que la señal de audio con ruido blanco que se genera tenga perturbaciones en alta frecuencia, esta señal es de tipo .wav y se la obtiene a través del bloque “To Multimedia File”. El filtro pasa alto es especificado a través de la herramienta FDATool, cuya configuración se muestra en la Figura 3.2.

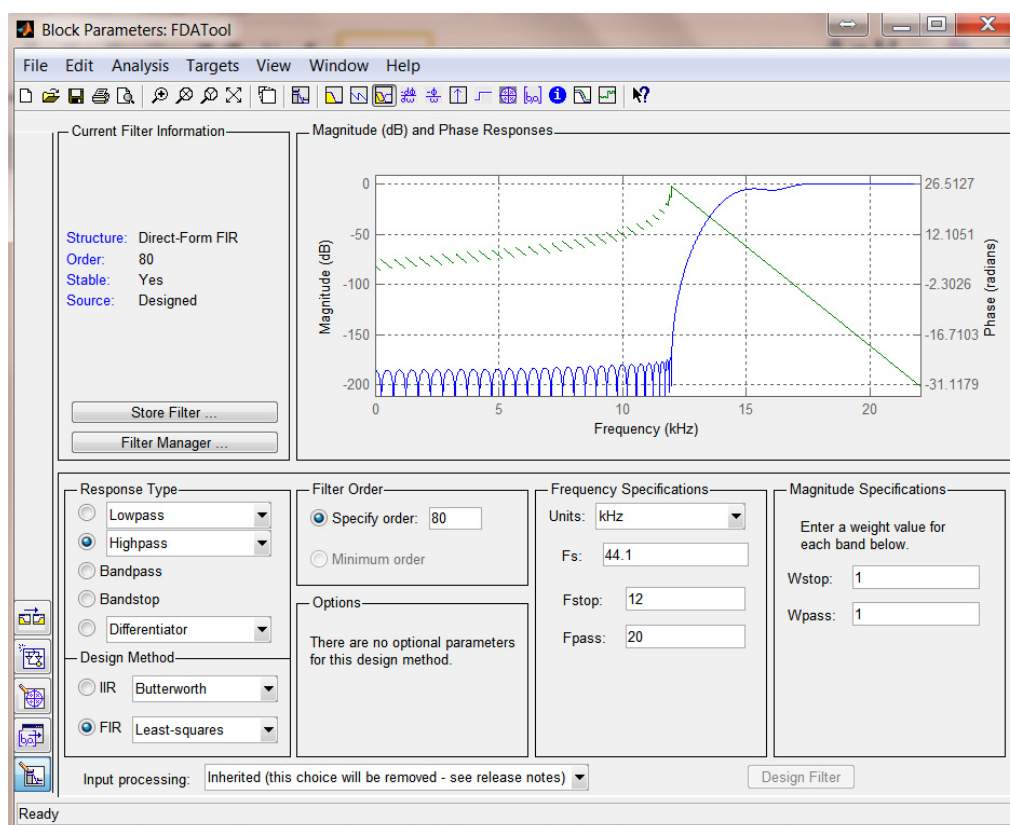


Figura 3.2 Configuración de parámetros del FDATool.

Esta herramienta es muy importante para la creación y generación de cualquier clase de filtro, en nuestro caso el diseño será de un filtro pasa bajos, para observar más detalles del FDATool ver Anexo D.

3.1.1 Diseño de filtro IIR tipo Butterworth

3.1.1.1 Descripción y Modelo

Mediante la herramienta FDATool se crea un filtro tratando de que el número de orden sea mínimo, de tipo pasa bajo con las especificaciones que se muestran en la siguiente tabla.

BLOQUE	PESTAÑA	PARÁMETRO	VALOR	
FDATool	Response Type	Lowpass		
	Design Method	IIR	Butterworth	
	Filter Order	Minimum order		
	Frecuency Specifications	Units		kHz
		Fs		1000
		Fpass		15
		Fstop		20

FDATool	Magnitude Specifications	Units	dB
		Apass	1
		Astop	60

Tabla II. Configuración de parámetros del FDATool para el diseño del filtro IIR.

Una vez diseñado el filtro pasa bajo se observa la respuesta de frecuencia del mismo, la amplitud en decibelios (azul) y la fase en radianes (verde), la fase debe ser lo más lineal posible para no afectar la frecuencia de la señal al momento del filtrado.

Luego se procede a exportar el filtro a Simulink con elementos básicos, como sumadores, multiplicadores y retardos, tal como se indica en la Figura 3.3.

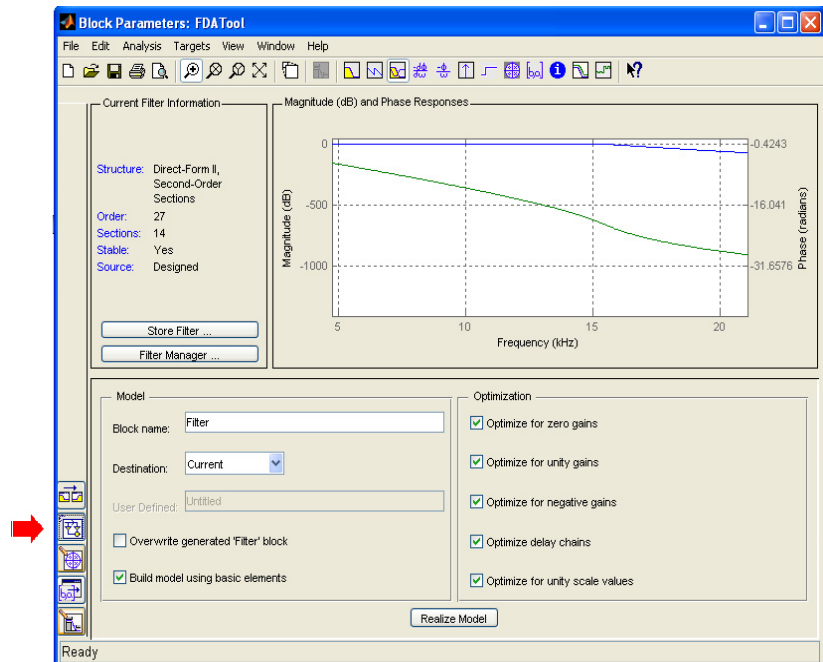


Figura 3.3 Exportación del filtro a Simulink con bloques básicos.

El filtro IIR tipo butterworth está conformado por 14 secciones, en la Figura 3.4 se muestra solo una de ellas, estas son iguales en su estructura pero diferentes en las configuraciones de los bloques que la conforman.

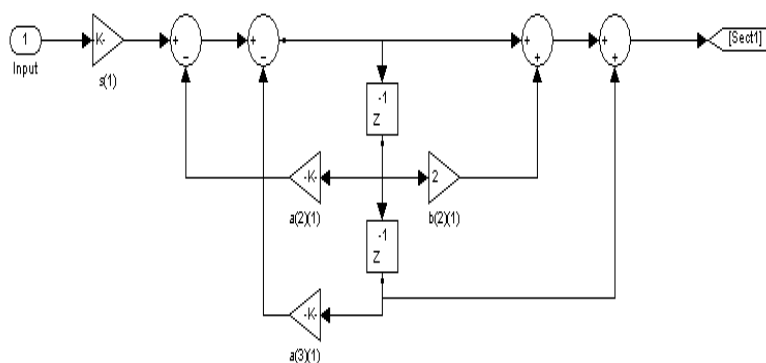


Figura 3.4 Primera sección de la estructura del filtro pasa bajos creado con bloques de Simulink.

El filtro que se creó está construido con bloques de Simulink, para la simulación se construirá la misma estructura pero con bloques del blockset de Xilinx en Simulink, como se muestra en la Figura 3.5.

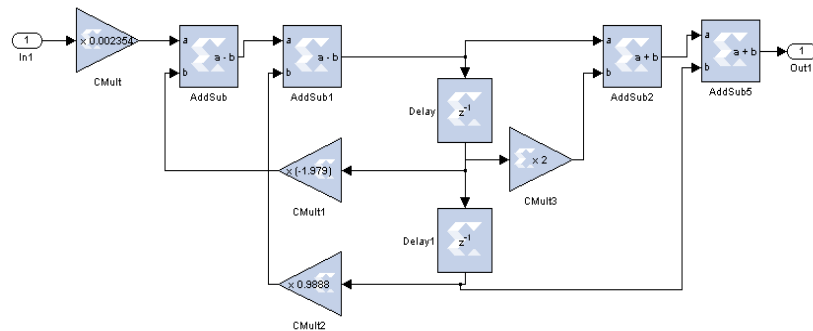


Figura 3.5 Primera sección de la estructura del filtro
pasa bajo creado con el blockset de Xilinx.

Para optimizar la visualización del diseño del filtro e identificar claramente cada una de las partes de la simulación, se formó un subsistema con las 14 secciones del filtro creado, en la Figura 3.6 se especifica este detalle, y además se muestra el esquema de la simulación con la incorporación de los bloques ADC y DAC, seteados a una frecuencia de muestreo de 2MHz. El archivo bitstream que genera el System Generator se crea a partir de los bloques entre los convertidores ADC y DAC.

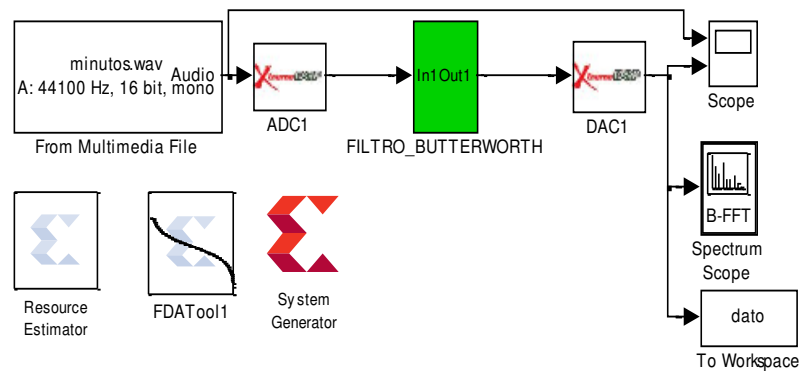


Figura 3.6 Diseño para la simulación del filtro IIR tipo butterworth.

La configuración de los bloques que se usaron para la simulación del filtro IIR tipo butterworth se muestran en la tabla III, se detalla también los bloques que se usaron en las 14 secciones del filtro creado con la herramienta del FDATool.

BLOQUE	PESTAÑA	PARÁMETRO	VALOR
Cmult, Cmult1, Cmult2, Cmult3	Basic	Constant Value	Variable
		Constant Type	Fixed-Point
		Number of bits	32
		Binary Point	29
	Output	Precision	User defined
		Arithmetic Type	Signed
		Number of bits	32
		Binary Point	29
		Quantization	Truncate
		Overflow	Saturate
Addsub, Addsub1	Basic	Operation	Subtraction
	Output	Arithmetic Type	Signed
		Number of bits	32
		Binary Point	29
		Quantization	Truncate
		Overflow	Saturate
Addsub2, Addsub5	Basic	Operation	Addition
	Output	Arithmetic Type	Signed

Addsub2, Addsub5	Output	Number of bits	32
		Binary Point	29
		Quantization	Truncate
		Overflow	Saturate
Delay	Basic	Latency	1
ADC1, DAC1		Sample Period	1/2000000
System Generator	Clocking	Simulink System Period	1/2000000

Tabla III. Configuración de parámetros del diseño del filtro IIR tipo butterworth.

3.1.1.2 Simulación del modelo

Una vez descrito el diseño del modelo del filtro se procede a simular el sistema con un tiempo de simulación de 10 segundos, para poder observar varios tramos de la señal, en la Figura 3.7 se muestra tanto la señal generada con ruido blanco gaussiano como la señal recuperada después del filtro.

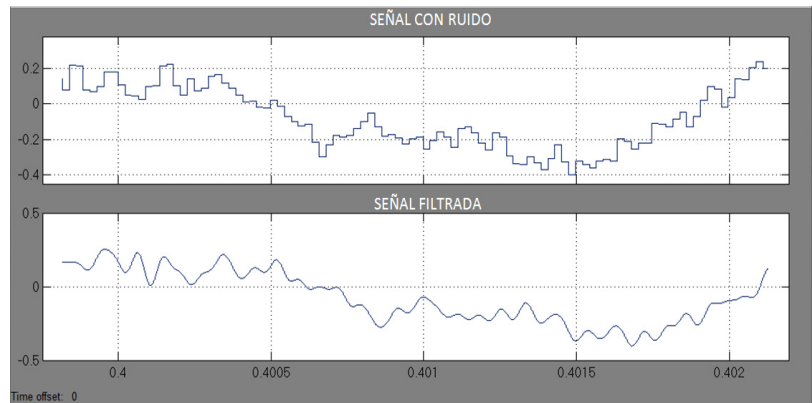


Figura 3.7 Simulación de la señal con ruido y señal filtrada.

En la Figura 3.8 se observa que el espectro de la señal de salida logra limitarse hasta los 20KHz con -25dB aproximadamente.

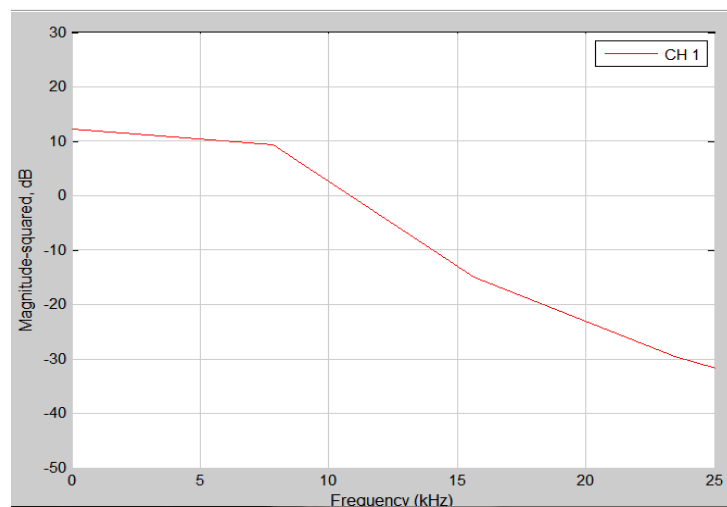


Figura 3.8 Espectro de la señal de salida.

3.1.2 Diseño de filtro FIR tipo LMS

3.1.2.1 Descripción y Modelo

La estructura del filtro FIR se detalla en la Figura 2.13 del capítulo 2, la cual se basa en la ecuación 9 y 10 de dicho capítulo, se utiliza la estructura de un filtro tal que los pesos serán modificados a través del algoritmo LMS, primero se diseña un filtro FIR con la herramienta FDATool, como se especifica en la tabla IV, en este filtro se utiliza el vector de coeficientes que genera, el cual es de igual número al vector que se genera en el algoritmo LMS, para realizar la renovación de coeficientes en cada interacción, se lo puede ver como un filtro “variable” en el que su vector de coeficientes irá cambiando de acuerdo al algoritmo LMS, durante la primera interacción se utilizan los coeficientes creados en el FDATool.

BLOQUE	PESTAÑA	PARÁMETRO	VALOR
FDATool	Response Type	Lowpass	
	Design Method	FIR	Least-squares
	Filter Order	Specify order	58

FDATool	Frequency Specifications	Units	kHz
		Fs	1000
		Fpass	4
		Fstop	15
	Magnitude Specifications	Wpass	1
		Wstop	1

Tabla IV. Parámetros del FDATool para el diseño del filtro de renovación de coeficientes.

Como se observa, el número de orden del filtro es bajo para que la renovación de coeficientes en cada interacción sea lo más rápido posible, debido que mientras mayor sea el número de orden del filtro, tendrá mayor grado de complejidad computacional, además le toma más tiempo al algoritmo de converger al punto mínimo de la superficie del error.

Para completar el sistema también es necesario definir otro filtro con la herramienta del FDATool, el objetivo de este filtro es generar la señal deseada $d(n)$ que se

requiere en la estructura. Los parámetros del filtro se detallan en la tabla V.

BLOQUE	PESTAÑA	PARÁMETRO	VALOR	
FDATool2	Response Type	Lowpass		
	Design Method	FIR	Least-squares	
	Filter Order	Specify order	200	
	Frecuency Specifications	Units		kHz
		Fs		1000
		Fpass		10
		Fstop		17
	Magnitude Specifications	Wpass		0.1
		Wstop		40

Tabla V. Parámetros del FDATool para el filtro que genera la señal deseada $d(n)$.

El algoritmo LMS no depende de este filtro, por lo que se puede tomar un mayor número de orden para obtener una buena respuesta de magnitud y fase.

Solo se han diseñado los coeficientes de los filtros, sin embargo es necesario crear la estructura con bloques adaptativos. El bloque “FIR Compiler” se encarga de crear ambos filtros dependiendo de las especificaciones dadas por el FDATool, el seteo de este bloque es detallado en la tabla VI, el cual muestra la configuración respectiva para el filtro que genera la señal deseada.

BLOQUE	PESTAÑA	PARÁMETRO	VALOR
FIR Compiler 5.0	Filter specifications	Coefficient Vector	Xlfa_numerator (‘FDATool2’)
		Number of coefficients set	1
		Filter Type	Single_rate
		Number of channels	1
		Select Format	Sample Period
		Sample Period	1

FIR Compiler 5.0	Implementation	Filter Architecture	Distributed_Arith metic
		Coefficient structure	Symmetric
		Coefficient type	Signed
		Quantization	Quantize_only
		Coefficient width	16
	Coefficient Options		Best precision fraction length

Tabla VI. Configuración de parámetros del FIR Compiler 5.0 del filtro que genera la señal deseada.

En la tabla VII se muestra también la configuración del bloque adaptativo FIR Compiler para generar el filtro variable.

BLOQUE	PESTAÑA	PARÁMETRO	VALOR
FIR Compiler 5.0	Filter specifications	Coefficient Vector	Xlfa_numerat or('FDATool')
		Number of coefficients set	1
		Filter Type	Single_rate
		Number of channels	1
		Select Format	Sample Period
	Implementation	Sample Period	1
		Filter Architecture	Distributed_Arit hmetic
		Coefficient structure	Symmetric
		Coefficient type	Signed
		Quantization	Quantize_only
		Coefficient width	24
		Coefficient fractional bits	22
	Coefficient options		Use reloadable coefficients

Tabla VII. Configuración de parámetros del FIR Compiler 5.0 del filtro variable.

Este bloque necesita un vector de coeficientes para crear la estructura el filtro, el cual lo obtiene desde el FDATool, utilizando el código `xlfd_numerator("FDATool")`, de esta forma se crea tanto el filtro que servirá para la renovación de coeficientes como el filtro que se utilizará para obtener la señal deseada $d(n)$. Es preciso indicar que en el primer filtro que se creó hay que seleccionar la opción de uso de coeficientes recargables y un ancho de bits suficiente como para definir de la mejor forma los coeficientes, se escogió un ancho de bits de 24.

El algoritmo se desarrolla en el subsistema descrito en la Figura 3.9, en el primer bloque se muestra la multiplicación de las señales $x(n)$ y $e(n)$, luego se multiplica por el factor μ obteniendo el siguiente coeficiente y reteniéndolo con un delay para la suma con el siguiente valor, el bloque mux sirve para recomenzar desde cero la suma de los coeficientes.

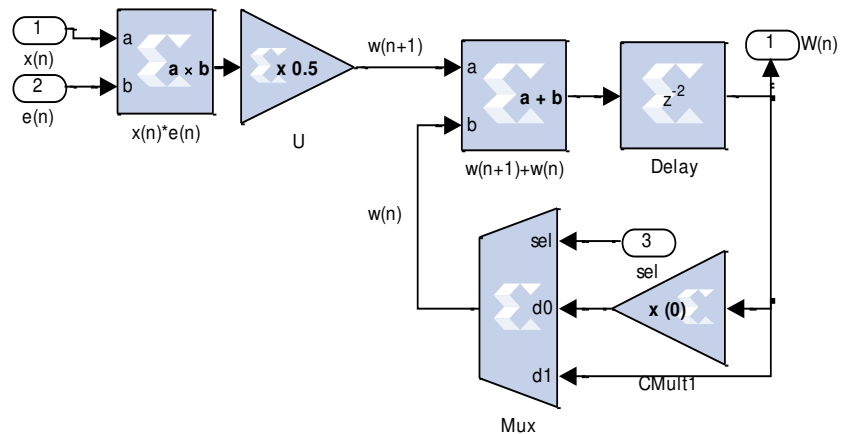


Figura 3.9 Algoritmo LMS.

La ecuación 10 del Capítulo 2, especifica un rango valor de μ :

$$0 < \mu \leq 0.4961$$

El valor de μ utilizado es de 0.25, el valor se escoge por pruebas en la simulación, mientras mayor μ , más rápido converge el filtro pero también incrementa el error en la salida, por el contrario si este valor es muy bajo la convergencia es lenta y el resultado no mostrará tantos errores.

El bloque FIFO posee la suficiente memoria para contener los N coeficientes que necesite el filtro “variable” hasta cuando este lo pueda recibir, teniendo la siguiente configuración. (Ver tabla VIII)

BLOQUE	PESTAÑA	PARÁMETRO	VALOR
FIFO	Basic	Memory type	Block RAM
		Performance options	First Word fall through
			Use embedded registers
		Depth	512
		Bits of precisión to use for %full signal	8

Tabla VIII. Configuración de parámetros del bloque FIFO.

Esta memoria nos permite almacenar los datos conforme se van generando, el primer coeficiente que entra es el primero que sale. El sistema completo se muestra en la Figura 3.10.

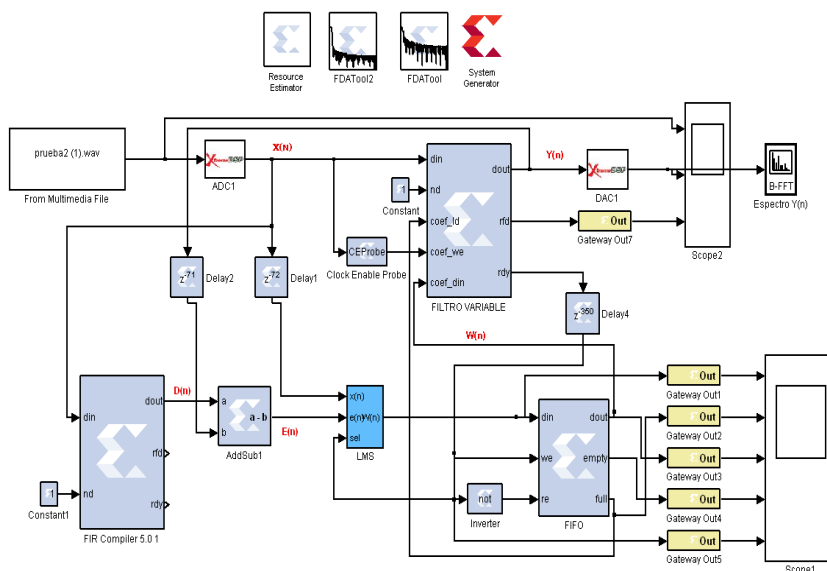


Figura 3.10 Diseño de la simulación del filtro FIR tipo LMS.

El bloque “FIR Compiler” permite obtener una señal deseada a la cual se le resta la salida para obtener el error y poder obtener los N coeficientes que serán almacenados en el bloque FIFO.

El filtro “variable” dispone de un puerto *rdy*, que indica cuando hay alguna salida válida en su puerto *dout*, cuando su salida es cero indica que está recibiendo un nuevo vector $W(n)$, el negado de esta señal nos sirve como habilitadora para la memoria FIFO, indicándole cuando debe almacenar datos y cuando enviarlos.

Mientras se coloca los coeficientes en la memoria FIFO, el filtro “variable” está operando con los coeficientes obtenidos en la anterior iteración, también se observan los bloques ADC y DAC seteados con una frecuencia de muestreo de 1MHz al igual que la frecuencia del sistema en el bloque “System Generator”.

3.1.2.2 Simulación del modelo

La Figura 3.11 corresponde al almacenamiento y envío de los coeficientes en el bloque FIFO, en donde se indican las gráficas de su entrada, memoria llena, salida, memoria vacía y habilitadora, respectivamente.

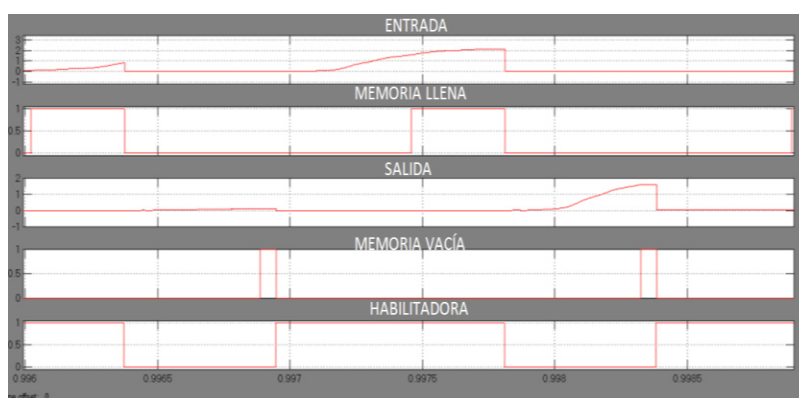


Figura 3.11 Simulaciones de las gráficas de entrada, memoria llena, salida, memoria vacía y habilitadora del bloque FIFO.

Cuando la memoria obtiene los N coeficientes su puerto lleno, manda una señal al filtro para que éste acepte el vector. El puerto vacío indica cuando la memoria ha quedado vacía; la memoria sólo vuelve a recibir los datos cuando recibe una señal del filtro y los envía cuando la misma señal es negada.

En la Figura 3.12, la primera señal corresponde a la entrada $X(n)$, la segunda a la salida del filtro variable $Y(n)$ y la tercera es del puerto *rfd*. En la gráfica se observa que hay un tiempo en el que su valor es de cero, esto se debe a que el filtro requiere de un tiempo para procesar la información.

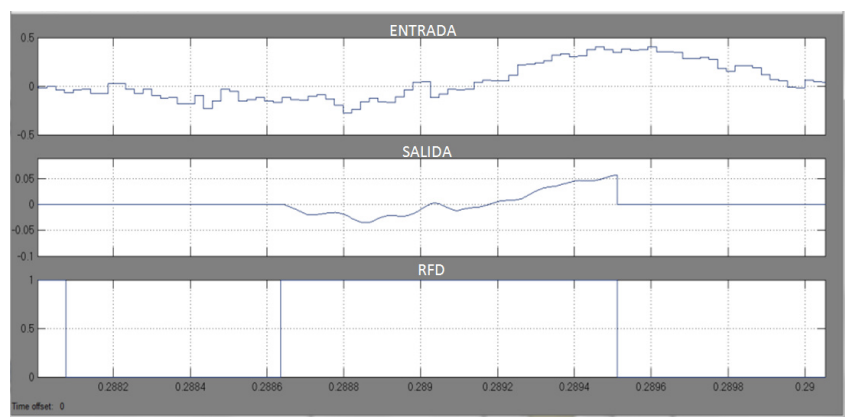


Figura 3.12 Simulaciones de las gráficas de entrada, salida y del puerto *rfd* del filtro “variable”.

El filtro no puede operar mientras recibe el nuevo vector, indicado por la señal *rdy*, por lo que en la gráfica $Y(n)$ se observan partes donde la señal es nula, también se muestra la salida del filtro de forma suavizada y disminuida en su amplitud con respecto a su entrada.

En la Figura 3.13 se detalla la respuesta del espectro de la señal $Y(n)$, se puede observar el correcto filtrado en las frecuencias requeridas, desde 15KHz hasta 25KHz hay una significativa disminución de la magnitud en dB, donde el ruido se hace presente en la señal.

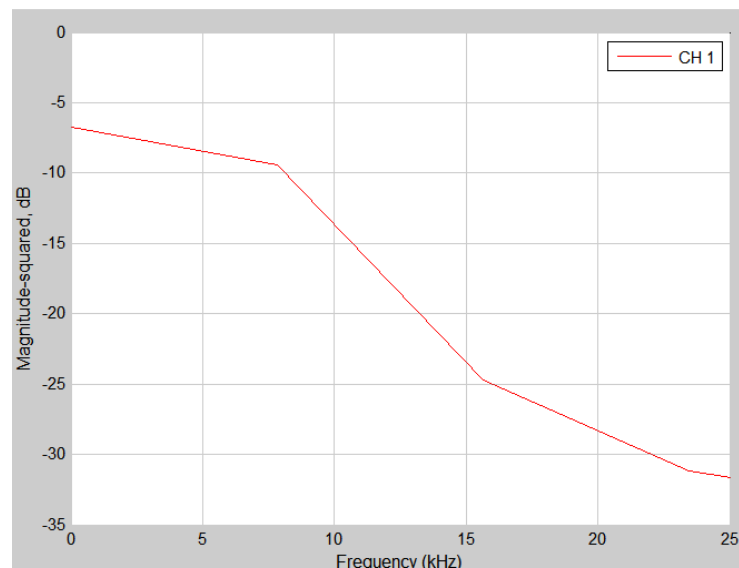


Figura 3.13 Espectro de la señal $Y(n)$.

3.2 Diseño de Lazo de enganche de fase

En la sección 2.3 se detalló la estructura del lazo de enganche de fase, uno de sus componentes principales es el filtro pasa bajos, en la primera sección de este capítulo se dieron dos formas de usar los filtros, una de ellas es construyendo el modelo del mismo con bloques básicos del blockset de Xilinx, y la otra opción es la de usar el bloque “FIR Compiler” para utilizar bloques adaptativos en el diseño del filtro, en esta sección se usará la segunda forma, ya que es la más simple de utilizar y facilita la visualización en la estructura del sistema.

3.2.1 Descripción y modelo

En este diseño se realiza un modelo linealizado, no se especifica una frecuencia para el VCO, pero el lazo de enganche de fase cumple su función y logra enganchar la señal entrante dependiendo de los parámetros descritos anteriormente, a continuación se detalla el diseño de los bloques usados del blockset de Xilinx en Simulink.

Primero se diseña el filtro FIR de tipo pasabajos con un mínimo de orden, a través de la herramienta del FDATool, con la configuración que se muestra en la tabla IX.

BLOQUE	PESTAÑA	PARÁMETRO	VALOR	
FDATool	Response Type	Lowpass		
	Design Method	FIR	Equiripple	
	Filter Order	Minimum Order	58	
	Options	Density factor	20	
	Frecuency Specifications	Units		kHz
		Fs		1000
		Fpass		14
		Fstop		24
	Magnitude Specifications	Units		dB
		Apass		1
		Astop		50

Tabla IX. Configuración del FDATool para el filtro FIR pasabajo.

Para la utilización del filtro se usará el bloque “FIR Compiler”, la configuración de este bloque se muestra en la tabla X, definiendo un ancho de bits de 11, ya que con este valor se definen completamente los coeficientes del filtro.

BLOQUE	PESTAÑA	PARÁMETRO	VALOR
FIR Compiler 5.0	Filter specifications	Coefficient Vector	Xlfda_numerator('FDATool 1')
		Number of coefficients set	1
		Filter Type	Single_rate
		Number of channels	1
		Select Format	Sample Period
		Sample Period	1
	Coefficient options	Filter Architecture	Distributed_Arithmetic
		Coefficient structure	Symmetric
		Coefficient type	Signed
		Quantization	Quantize_only
		Coefficient width	18
		Coefficient Fractional Bits	14

Tabla X. Configuración del FIR Compiler para el diseño del filtro.

Una vez diseñado el filtro a utilizar en el sistema, se procede a crear el VCO, consta de una ganancia que fue determinado de forma experimental para obtener más rango de frecuencias en el enganche, este valor es de 5, y del bloque integrador cuyo subsistema se detalla en la Figura 3.14.

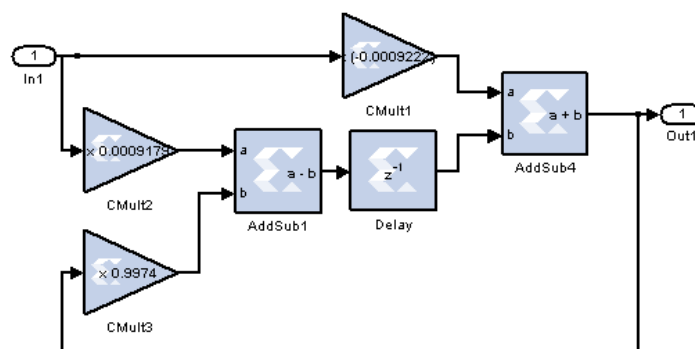


Figura 3.14. Diseño del integrador con bloques de Xilinx.

También se muestra el bloque “Slice”, cuya función es la de tomar cierto número de bits de la parte fraccionaria y ubicarlos en la parte real para poder observar la salida amplificada, la configuración se muestra en la tabla XI.

BLOQUE	PESTAÑA	PARÁMETRO	VALOR
Slice	Basic	Width of slice	5
		Specify range as	Upper bit location + width
		Relative to	MSB of input

Tabla XI. Configuración del bloque Slice.

El sistema completo del PLL se detalla en la Figura 3.15

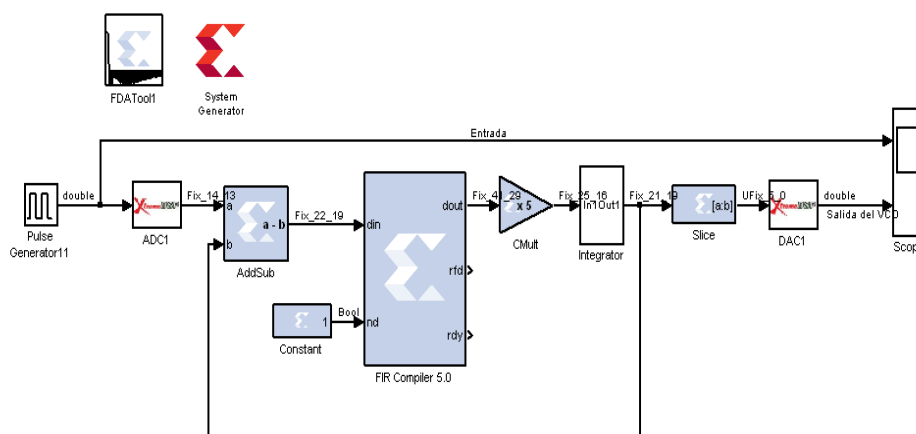


Figura 3.15 Diseño del PLL con bloques de Xilinx.

3.2.2 Simulación del modelo

En la Figura 3.16 se muestra la gráfica de la entrada y salida del VCO, el generador de pulsos está seteado a una frecuencia de 10KHz, se observa que el VCO mantiene la frecuencia.

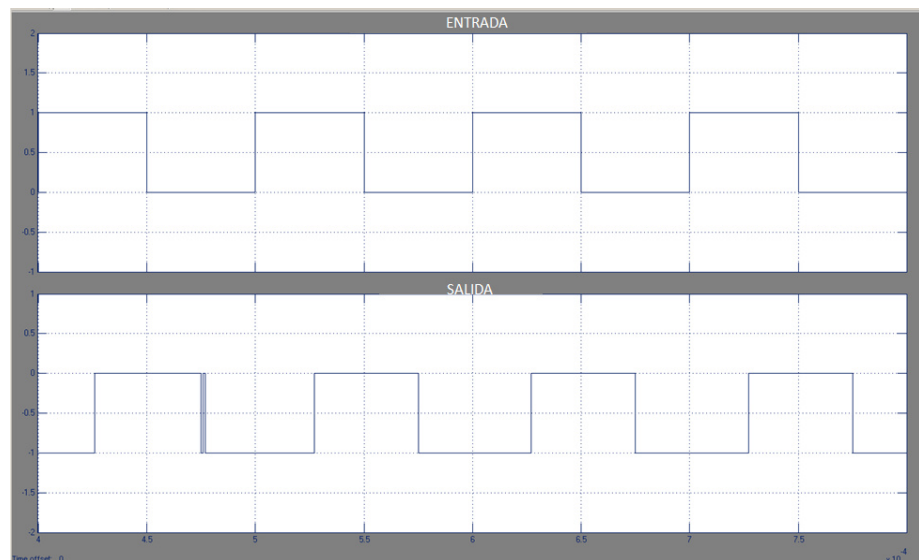


Figura 3.16 Simulación de la señal de entrada y salida del PLL.

3.3 Diseño de un modulador/demodulador 16QAM

Para indicar el funcionamiento de cada bloque primero se explicará el diagrama de bloques del modulador y posteriormente del demodulador, a continuación se presentan ambos diagramas en la figura 3.17 y 3.18 respectivamente.

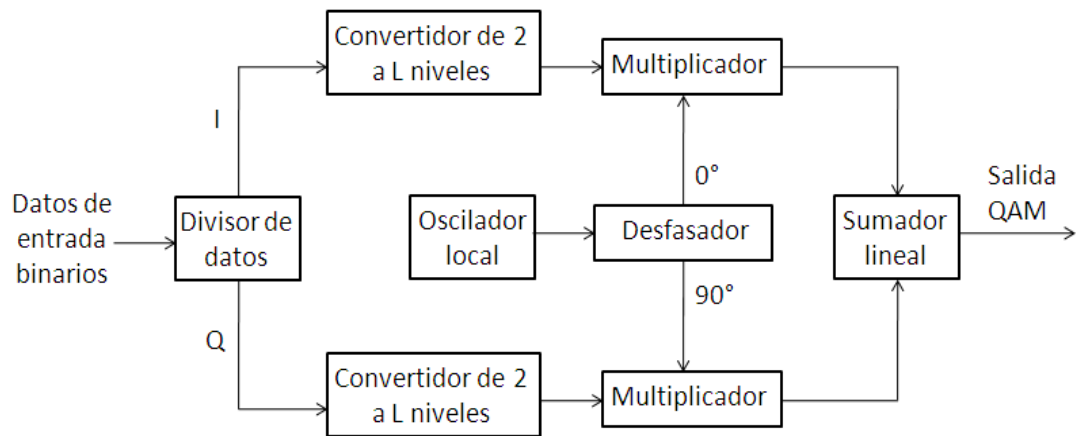


Figura 3.17 Diagrama de bloques de un modulador QAM.

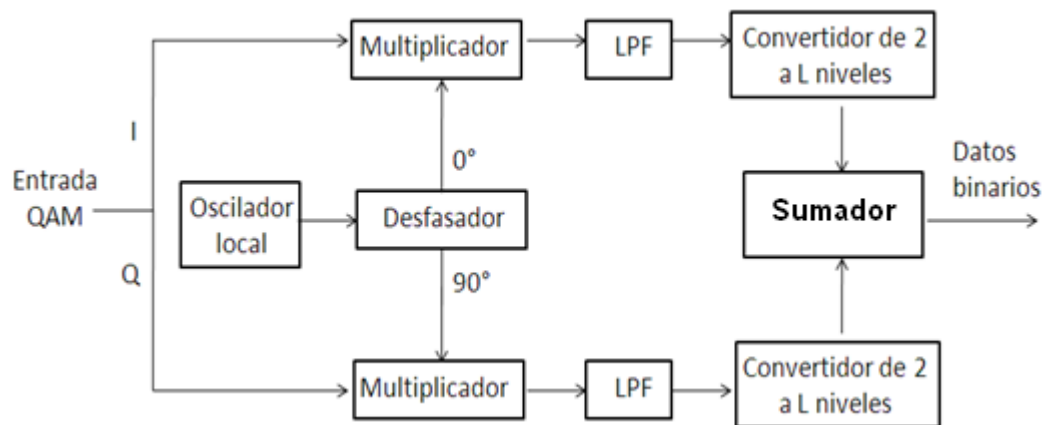


Figura 3.18 Diagrama de bloques de un demodulador QAM.

3.3.1 Descripción y modelo

DIVISOR DE DATOS

La entrada binaria primero pasa por el divisor de datos, o también llamado derivador, cuya función es el de agrupar N bits y posteriormente separarlos en N/2 bits hacia el canal Q (cuadratura) y el canal I (fase), ambos canales son ortogonales por lo que se puede aprovechar todo el ancho de banda en cada canal.

La agrupación de bits se realiza de acuerdo a la siguiente relación:

$$n = 2^N \quad (13)$$

Donde:

n= número de estados QAM.

N= número de bits.

Se realizará la modulación y demodulación 16QAM, por lo tanto n=16 y N=4, así que la entrada binaria se separa en 2 bits para el canal I y 2 bits para el canal Q, tal como se muestra en la tabla XII.

SECUENCIA DE BITS	CANAL I	CANAL Q
... Q ₄ Q ₃ I ₄ I ₃ Q ₂ Q ₁ I ₂ I ₁	..._ _ I ₄ I ₃ _ _ I ₂ I ₁	... Q ₄ Q ₃ _ _ Q ₂ Q ₁ _ _

Tabla XII. Agrupación de los bits en los canales I-Q.

Los subguiones representan el tiempo que el canal no recibe datos, es decir; mientras un canal recibe los bits de la entrada, el otro no recibe información y el multiplexor colocará un cero en dicha posición, y viceversa hasta completar la cadena binaria inicial. El funcionamiento del divisor de datos o derivador es equivalente a 2 multiplexores 2:1, cuya tabla de verdad para cada canal se muestra en la figura 3.19.

d0	d1	sel	salida
1	0	0	1
1	0	1	0

Figura 3.19 (a) Tabla de verdad del canal I.

d0	d1	sel	salida
0	Q	0	0
0	Q	1	Q

Figura 3.19 (b) Tabla de verdad del canal Q.

La conexión de los multiplexores y la señal de selección se detallan en la figura 3.20.

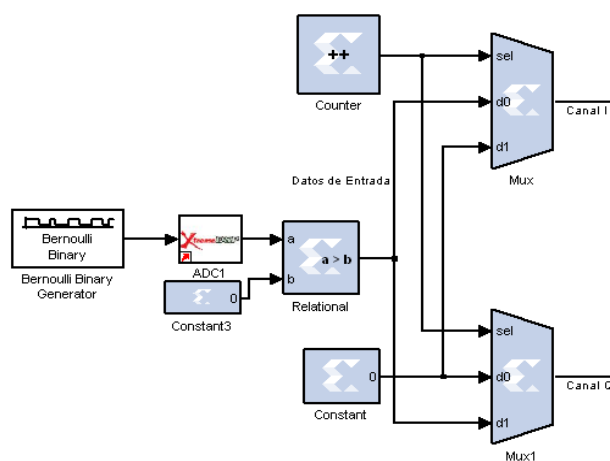


Figura 3.20 Conexión de los multiplexores para agrupar los canales I-Q.

Los datos ingresan hacia las 2 entradas d0 y d1 de acuerdo al multiplexor correspondiente, las otras entradas se conectan al valor constante de cero, y la señal de selección será una señal cuadrada, que se logra con la acción de un contador módulo 2, ésta señal validará los bits para cada canal respectivo definiendo correctamente el período de la señal. La entrada binaria se genera de forma aleatoria a través del bloque Bernoulli Binary Generator. La configuración de los parámetros de cada bloque se describe en la tabla XIII.

BLOQUE	PESTAÑA	PARÁMETRO	VALOR
Counter	Basic	Counter type	Free running
		Count direction	Up
		Initial value	0
		Step	1
		Output type	Unsigned
		Number of bits	1
		Binary point	0
		Explicit period	2/1228800
Mux, mux1	Basic	Number of inputs	2

Mux, mux1	Output	Precision	User defined
		Arithmetic type	Unsigned
		Number of bits	1
		Binary point	0
		Quantization	Truncate
		Overflow	Wrap
Relational	Basic	Comparision	a>b
ADC1	Parameters	Sample period	1/1228800
Bernoulli Binary	Parameters	Probability of a zero	0.5
		Initial seed	61
		Sample time	1/1228800
		Output data type	Double

Tabla XIII. Configuración de parámetros de los bloques que forman el bloque Divisor de datos.

CONVERTIDOR 2 A L NIVELES

Una vez separada la trama de bits en los canales I (fase) y Q (cuadratura), el bloque convertidor de 2 a L niveles genera una señal multinivel, la manipulación de los bits se muestra en la tabla XIV.

SECUENCIA DE BITS	CANAL I	CANAL Q
...Q ₄ Q ₃ I ₄ I ₃ Q ₂ Q ₁ I ₂ I ₁	..._I ₄ I ₃ _I ₂ I ₁	..._Q ₄ Q ₃ _Q ₂ Q ₁ _
	..._I ₃₋₄ _I ₁₋₂	..._Q ₃₋₄ _Q ₁₋₂ _
	...I ₃₋₄ I ₃₋₄ I ₁₋₂ I ₁₋₂	...Q ₃₋₄ Q ₃₋₄ Q ₁₋₂ Q ₁₋₂ _

Tabla XIV. Agrupación de bits para obtener la señal unipolar multinivel.

Los grupos de bits de cada canal serán representados por números decimales sin signo, de esta forma se obtiene la señal unipolar multinivel, ésta representación se detalla en la tabla XV.

ENTRADAS		SALIDAS
I_1/Q_1	I_2/Q_2	I_{1-2}/Q_{1-2}
0	0	0
0	1	1
1	0	2
1	1	3

Tabla XV. Tabla de verdad para los valores de la señal multinivel unipolar para 16QAM.

El bloque Serial to Parallel permite ir de I_2I_1 a I_{1-2} , requiere como parámetro el número de bits agrupados por muestra, y para ir de I_{1-2} a $I_{1-2}I_{1-2}$ se logra haciendo que la muestra permanezca el doble de su tiempo original mediante el bloque Down Sample. Para poder obtener la señal multinivel bipolar simétrica a partir de la señal unipolar, se debe restar un valor constante que es la mitad del nivel máximo de voltaje asignado para la señal multinivel unipolar, para 16QAM este máximo nivel de voltaje es 3, por lo que el valor a restarse de la señal unipolar es 1.5, de esa forma se obtiene la señal multinivel bipolar que se muestra en la tabla XVI.

ENTRADAS		SALIDAS
I_1/Q_1	I_2/Q_2	I_{1-2}/Q_{1-2}
0	0	-1.5
0	1	-0.5
1	0	0.5
1	1	1.5

Tabla XVI. Tabla de verdad para los valores de la señal multinivel bipolar para 16QAM.

La conexión de los bloques del convertidor 2 a L niveles se detalla en las figuras 3.21 y 3.22.

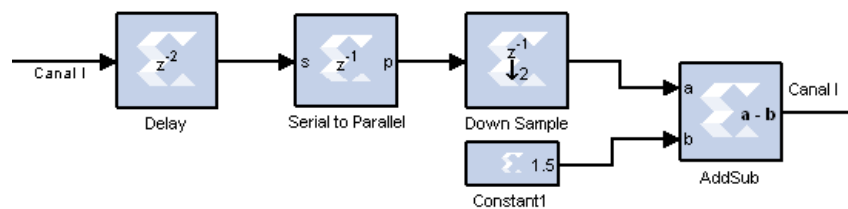


Figura 3.21 Conexión de los bloques que conforman el convertidor 2 a L niveles para el canal I.

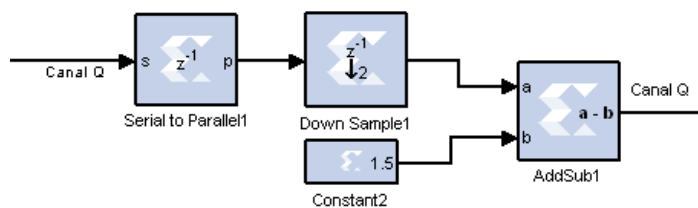


Figura 3.22 Conexión de los bloques que conforman el convertidor 2 a L niveles para el canal Q.

La presencia del bloque Delay es para sincronizar ambos canales en el proceso de la conversión, el detalle de la configuración de los bloques se muestra en la tabla XVII.

BLOQUE	PESTAÑA	PARÁMETRO	VALOR
Serial to Paralell, Serial to Paralell1	Basic	Input order	Most significant Word first
		Arithmetic type	Unsigned
		Number of bits	2
		Binary point	0
		Latency	1
Down Sample, Down Sample1	Basic	Sample rate	2
	Basic	Sample	First value of frame

		Latency	1
Constant1, Constant2	Basic	Constant value	1.5
		Output type	Fixed point
		Arithmetic type	Unsigned
		Number of bits	2
		Binary point	1
Addsub, Addsub1	Basic	Operation	Subtraction
	Output	Precision	Full
Delay	Basic	Latency	2

Tabla XVII. Configuración de parámetros de los bloques que conforman el convertidor 2 a L niveles.

OSCILADOR LOCAL

En estos bloques se genera la portadora, mientras mayor sea la frecuencia de la misma habrá mayor capacidad de transmisión y a su vez un mayor ancho de banda, para la construcción del oscilador se almacenarán muestras de una onda coseno a través de una memoria ROM, la cual será multiplicada directamente al canal I y la onda coseno desfasada se multiplicará con el canal Q.

Para definir el tamaño de la memoria es indispensable saber el número de muestras por periodo, que está relacionado con la frecuencia de muestreo y de la portadora mediante la siguiente ecuación:

$$x = \frac{f_m}{f_c} \quad (14)$$

Donde:

f_m = Frecuencia de muestreo.

f_c = Frecuencia de la portadora.

x = número de muestras por período.

Para nuestro caso se escoge una frecuencia de muestreo de 19660800 hz, y la frecuencia de la portadora en 4915200 hz, por lo cual se obtienen 40 muestras por período, la selección de frecuencias deben cumplir el teorema de muestreo.

Las líneas de direccionamiento de la memoria ROM se calculan a través de la siguiente relación:

$$x \leq 2^D \quad (15)$$

Considerando esta relación, la memoria debe tener 6 líneas de direccionamiento y 40 localidades de memoria. Para multiplicar la portadora con el canal Q, ésta debe desfasarse 90 grados, lo que es equivalente a que las muestras se retrasen $\frac{1}{4}$ del muestreo normal por período, esto se logra con el bloque Delay. Para el direccionamiento se usa el bloque Counter, el cual permite acceso secuencial a la memoria. La conexión de los bloques se muestra en la figura 3.23.

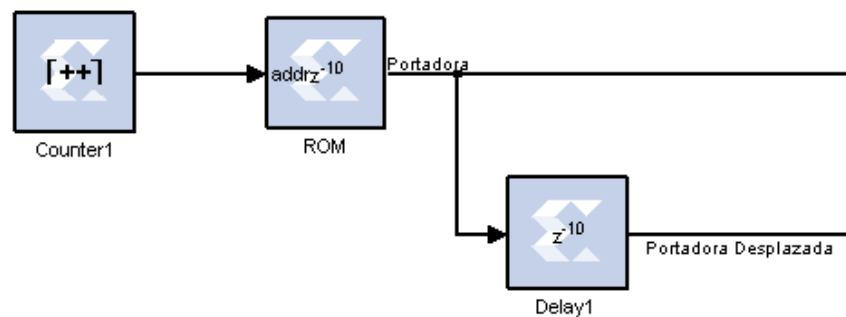


Figura 3.23 Conexión de los bloques del oscilador local y desfasador.

La configuración de cada bloque que conforma el oscilador local y el desfasador se muestra en la tabla XVIII.

BLOQUE	PESTAÑA	PARÁMETRO	VALOR
Counter1	Basic	Counter type	Count limit
		Count to value	39
		Count direction	Up
		Initial value	0
		Step	1
		Output type	Unsigned
		Number of bits	6
		Binary point	0
		Explicit period	1/196608000
ROM	Basic	Depth	40
		Initial value vector	$-2 \cdot \sin(2 \cdot \pi \cdot (0:39)/40)$
		Memory type	Block RAM
		Latency	10
	Output	Output type	Fixed-point
		Arithmetic type	Signed
		Number of bits	10
		Binary point	7

Delay1	Basic	Latency	10
--------	-------	---------	----

Tabla XVIII. Configuración de parámetros de los bloques que conforman el oscilador y desfasador.

MULTIPLICADOR

Una vez obtenida la portadora y su correspondiente onda desfasada, se procede a multiplicar con el canal en fase y cuadratura según corresponda, esta conexión se muestra en la figura 3.24.

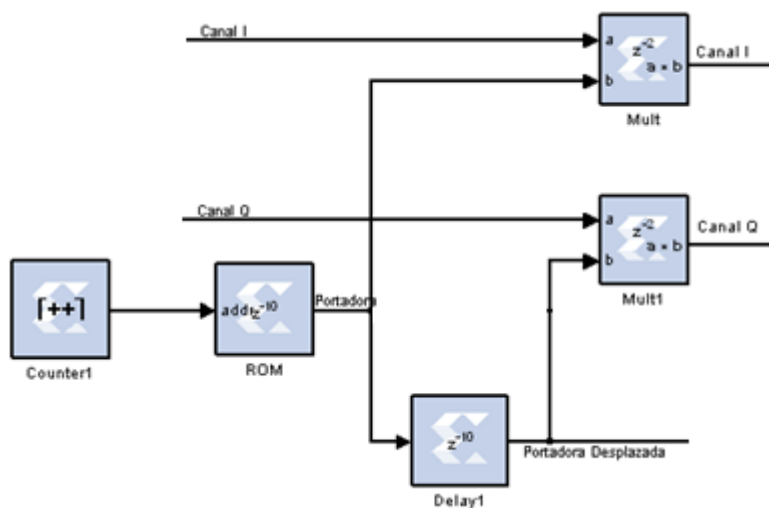


Figura 3.24 Conexión del bloque multiplicador del modulador.

La configuración del bloque multiplicador se detalla en la tabla XIX.

BLOQUE	PESTAÑA	PARÁMETRO	VALOR
Mult, Mult1	Basic	Precision	Full
		Latency	2

Tabla XIX. Configuración de parámetros del bloque multiplicador del modulador.

SUMADOR

Este es el último bloque para lograr la modulación 16QAM, en el que se mezclan ambos canales para formar la salida QAM, se muestra en la figura 3.25.

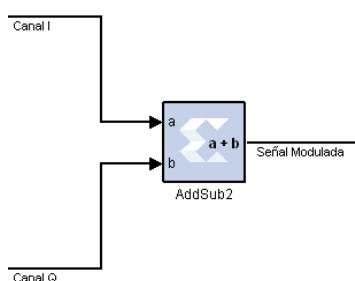


Figura 3.25 Conexión del bloque sumador del modulador.

La configuración del bloque sumador se detalla en la tabla XX.

BLOQUE	PESTAÑA	PARÁMETRO	VALOR
Addsub2	Basic	Operation	Addition
	Output	Precision	Full

Tabla XX. Configuración de parámetros del bloque sumador del modulador.

Se ha detallado cada uno de los bloques que conforman el modulador 16QAM, desde la entrada binaria hasta que se obtiene la señal modulada QAM, de la misma forma se detallará cada bloque del demodulador 16QAM para poder recuperar la señal binaria inicial.

MULTIPLICADOR

El primer paso para empezar la demodulación 16QAM es multiplicar la señal modulada por la portadora y por su respectivo desfase, de tal forma que se recuperará cada canal de forma individual, la conexión de los bloques se muestra en la figura 3.26.

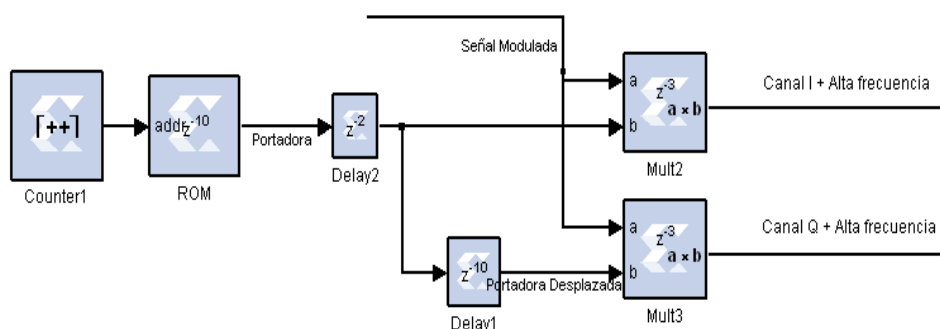


Figura 3.26 Conexión del bloque multiplicador del demodulador.

La función del bloque Delay2 es la de retrasar las muestras para que la señal modulada pueda separarse a través de los canales de fase y cuadratura, y obtenerlos de forma individual, la configuración de cada bloque se detalla en la tabla XXI.

BLOQUE	PESTAÑA	PARÁMETRO	VALOR
Counter1	Basic	Precision	User defined
		Arithmetic type	Signed
		Number of bits	12
		Binary point	7
		Quantization	Truncate
		Overflow	Wrap
		Latency	3

Delay2	Basic	Latency	2
--------	-------	---------	---

Tabla XXI. Configuración de parámetros del bloque multiplicador del demodulador.

FILTRO PASABAJO

A la salida del bloque multiplicador se obtiene una señal sinusoidal bipolar, esta señal tiene componentes que deben ser filtradas para obtener la señal multinivel bipolar, el objetivo es eliminar las armónicas que puedan aparecer por la portadora y obtener solo la componente constante. El diseño del filtro se lo realiza mediante la herramienta FDATool, se debe definir la frecuencia de paso (f_{pass}) y la frecuencia de parada (f_{stop}).

La frecuencia f_{pass} es el ancho de banda (AB) para propagar la señal modulada, este caso el ancho de banda es equivalente a la cuarta parte de la frecuencia de la portadora; es decir, 307.2 Khz y la frecuencia f_{stop} cumple la siguiente relación:

$$f_{stop} < 2f_c - AB \quad (16)$$

Una vez obtenidos los datos para el seteo del FDATool, en la tabla XXII se detalla la configuración del mismo.

BLOQUE	PESTAÑA	PARÁMETRO	VALOR	
FDATool1	Response Type	Lowpass		
	Design Method	FIR	Equiripple	
	Filter Order	Minimum Order		
	Options	Density factor	20	
	Frequency Specifications	Units		MHz
		Fs		100
		Fpass		0.1536
		Fstop		4
	Magnitude Specifications	Units		dB
		Apass		1
		Astop		60

Tabla XXII. Configuración de parámetros del bloque FDATool

Esto nos da que la frecuencia de parada (f_{stop}) debe ser menor a 9.5 Mhz, por lo que se escogió una frecuencia de 4Mhz, para utilizar el filtro se usará el bloque Dafir V9_0, que tiene la misma función del FIR Compiler pero además tiene la ventaja de poder trabajar con dos canales, y así recuperar el canal de fase y cuadratura, las señales de entrada al filtro operan a la frecuencia de muestreo, pero el filtro trabaja a la mitad de este valor, por ello se utiliza el bloque Down Sample a la entrada de filtro.

El objetivo es que a la salida del filtro se tenga la misma frecuencia de muestreo para cada canal, y esto se logra con el bloque Down Sample a la salida del filtro, que tomará el valor de la frecuencia de muestreo dividido para el doble del ancho de banda; es decir, el bloque tiene un valor de 320, la conexión del filtro y los acondicionadores de frecuencia se muestran en la figura 3.27.

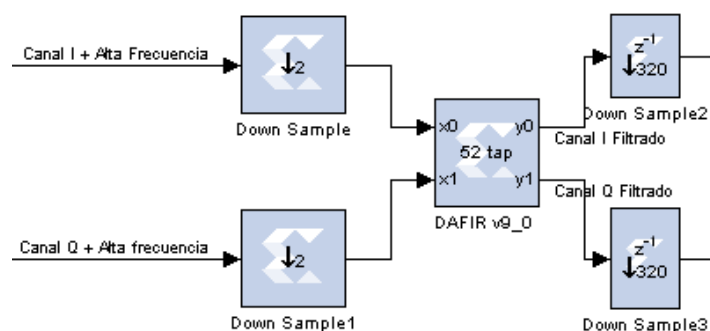


Figura 3.27 Conexión del filtro del demodulador.

La configuración de los bloques se muestra en la tabla XXIII.

BLOQUE	PESTAÑA	PARÁMETRO	VALOR
DAFIR V_9.0	Basic	Coefficients	xlfa_numerator('FDATool1')
		Structure	Inferred from Coefficients
		Number of bits	16
		Binary Point	14
		Hardware over-sampling rate	1
	Latency	33	
	Advanced	Number of channels	2
		Polyphase behavior	Single rate:
Down Sample, Down Sample1	Basic	Sample rate	2
		Sample	First value of frame
Down Sample2, Down Sample3	Basic	Sample rate	320
		Sample	Last value of frame
		Latency	1

Tabla XXIII. Configuración de parámetros del filtro del demodulador.

CONVERTIDOR DE L A 2 NIVELES

En el demodulador se realiza un proceso inverso al de la modulación, ahora se tiene una señal multinivel bipolar y se debe obtener la señal unipolar, anteriormente el proceso inverso se logró restando un valor constante, por lo que usando el mismo principio se obtiene la señal multinivel unipolar sumando dicha constante, que para 16QAM corresponde al valor de 1.5, debido a la acción del filtro la señal unipolar tendrá una parte decimal, por lo que se realiza un ajuste por redondeo con el bloque Convert.

Una vez obtenida la señal unipolar se debe obtener el equivalente binario del número decimal, que se detalló en la tabla 3.14; es decir, se desea realizar la conversión mostrada en la tabla XXIV.

CANAL I	CANAL Q	SECUENCIA DE BITS RECUPERADA
$\dots I_{3.4} I_{3.4} I_{1.2} I_{1.2}$ $\dots _ I_{3.4} _ I_{1.2}$ $\dots _ I_{4} I_{3} _ I_{2} I_{1}$	$\dots Q_{3.4} Q_{3.4} Q_{1.2} Q_{1.2} _ _$ $\dots _ _ Q_{3.4} _ _ Q_{1.2} _ _$ $\dots _ _ Q_{4} Q_{3} _ _ Q_{2} Q_{1} _ _$	$\dots Q_{4} Q_{3} I_{4} I_{3} Q_{2} Q_{1} I_{2} I_{1}$

Tabla XXIV. Separación de los bits de la señal multinivel unipolar.

El bloque que permite esta conversión es Up Sample, el cual incrementa la frecuencia de muestreo, y a la vez elimina una parte de ella colocando ceros, el último paso es conseguir separar los bits del canal de fase y cuadratura de forma independiente, para esto se usa el bloque Parallel to Serial, en la figura 3.28 se muestran los bloques que conforman el convertidor 2 a L niveles.

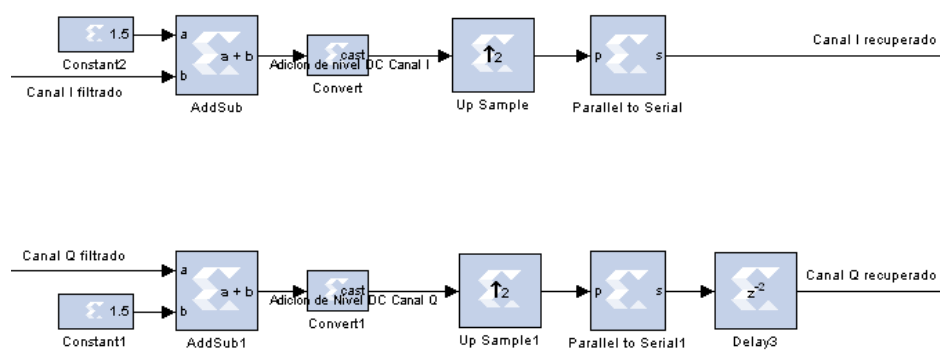


Figura 3.28 Conexión de los bloques del Convertidor L a 2 niveles para el canal I y Q respectivamente.

Como se observa en la figura 3.28, el canal Q presenta un retraso, cuyo objetivo es retrasar los datos de ese canal para que los bits se ubiquen de manera correcta. La configuración de cada uno de estos bloques se muestra en la tabla XXV.

BLOQUE	PESTAÑA	PARÁMETRO	VALOR
Constant1, Constan2	Basic	Constant value	1.5
		Output type	Fixed-point
		Arithmetic type	Unsigned
		Number of bits	2
		Binary Point	1
Addsub, Addsub1	Basic	Operation	Addition
	Output	Precision	User defined
		Arithmetic type	Unsigned
		Number of bits	4
		Binary Point	2
		Quantization	Truncate
		Overflow	Wrap
Convert, Convert1	Basic	Output type	Fixed-point
		Arithmetic type	Unsigned
		Number of bits	2
		Binary Point	0
		Quantization	Round
		Overflow	Wrap

Up sample, Up sample1	Basic	Sample rate	2
Parallel to Serial, Parallel to Serial1	Basic	Output order	Most significant word first
		Type	Unsigned
		Number of bits	1
		Binary Point	0
Delay3	Basic	Latency	2

Tabla XXV. Configuración de los bloques del convertidor L a 2 niveles.

SUMADOR

La señal binaria se la obtiene sumando los canales de fase y cuadratura, los cuales ya se obtuvieron de forma separada, por lo que con el bloque Addsub se tiene los datos binarios que se ingresaron en la etapa inicial, este detalle se observa en la figura 3.29.

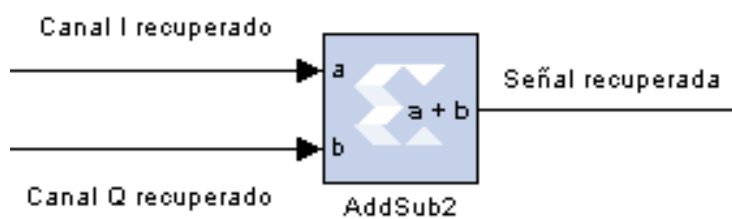


Figura 3.29 Conexión del sumador del demodulador.

La configuración del bloque sumador se muestra en la tabla XXVI.

BLOQUE	PESTAÑA	PARÁMETRO	VALOR
Addsub2	Basic	Operation	Addition
	Output	Precision	User defined
		Arithmetic type	Unsigned
		Number of bits	4
		Binary Point	2
		Quantization	Truncate
		Overflow	Wrap

Tabla XXVI. Configuración del bloque sumador del demodulador.

3.3.2 Simulación del modelo

En la figura 3.30 se muestra los datos binarios generados por el bloque Bernoulli Binary Generator, así como también se detallan los canales I y Q, la cuarta gráfica es la señal del reloj para observar la debida separación de los bits en cada canal respectivo.

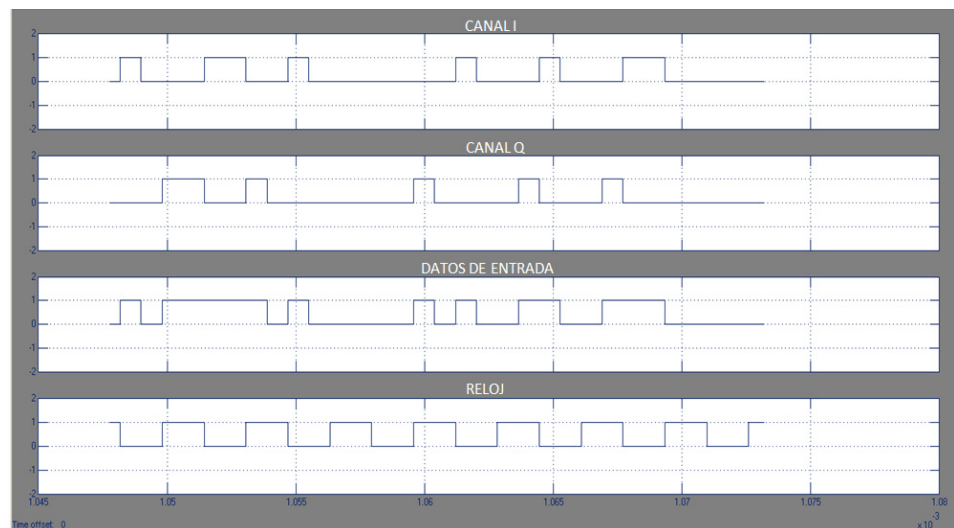


Figura 3.30 Gráfica del canal I, canal Q, señal de entrada y reloj.

Una vez separados los canales de fase y cuadratura, también se muestra en la figura 3.31 la señal multinivel bipolar obtenida después del bloque convertidor 2 a L niveles.

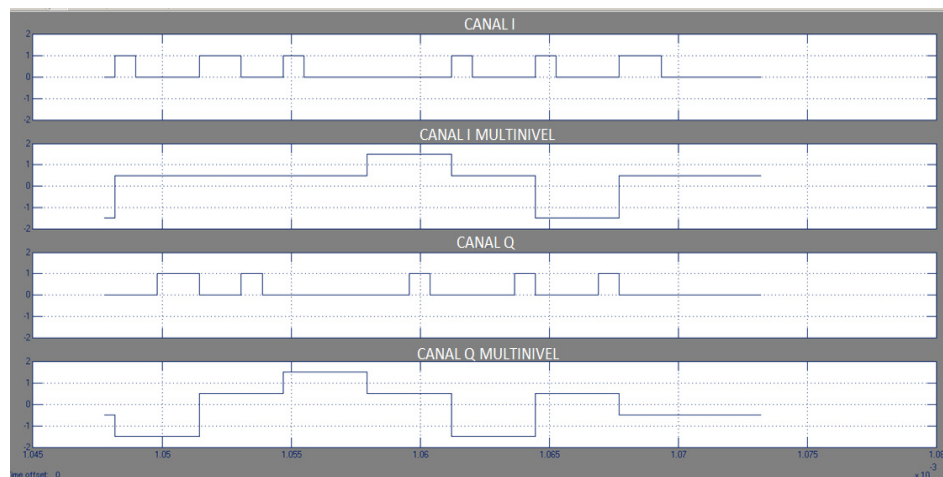


Figura 3.31 Gráfica del canal I, canal I multinivel, canal Q y canal Q multinivel.

La portadora y el efecto del bloque multiplicador en la modulación se muestran en la figura 3.32, tanto para el canal I y Q.

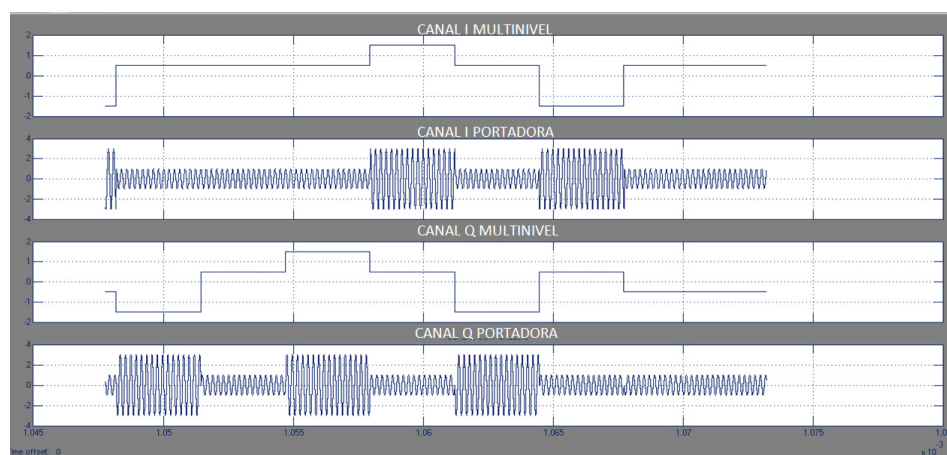


Figura 3.32 Gráfica del canal I multinivel, canal I con portadora, canal Q multinivel y canal Q con portadora

El paso final para obtener la señal modulada es sumar ambos canales, este detalle se observa en la figura 3.33.

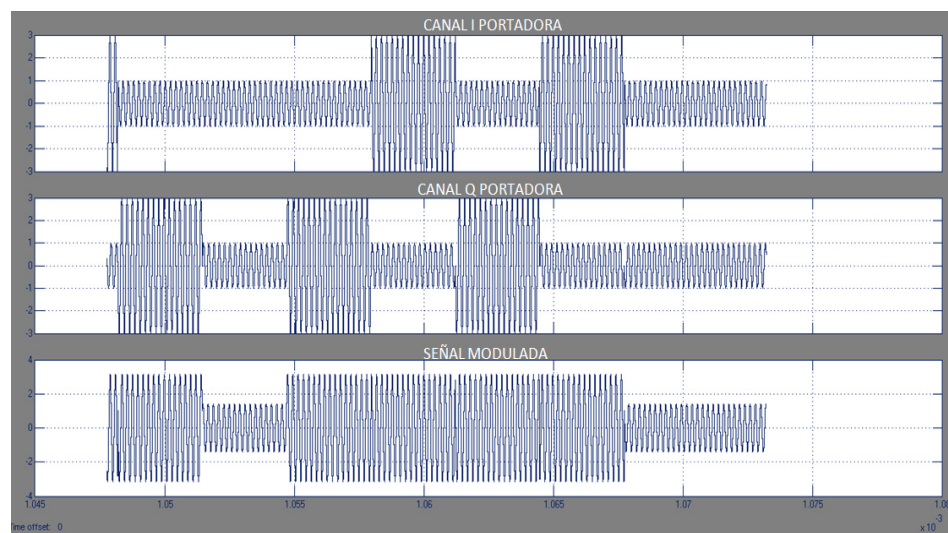


Figura 3.33 Gráfica del canal I-Q con portadora y señal modulada.

La constelación de la modulación se muestra en la figura 3.34

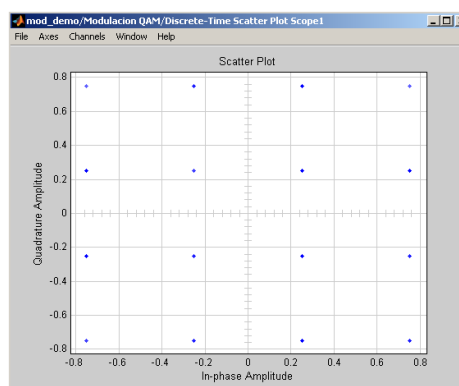


Figura 3.34 Constelación 16QAM en simulación.

Para recuperar el dato binario inicial a través de la demodulación, se realiza el proceso inverso, en la figura 3.35 y 3.36 se muestra la multiplicación de la señal QAM con la portadora y su respectivo desfase, para formar los canales I y Q de manera independiente, pero estas tendrán componentes armónicas creados por la acción de la portadora, por lo que la frecuencia que se obtiene es mucho mayor a la inicial.

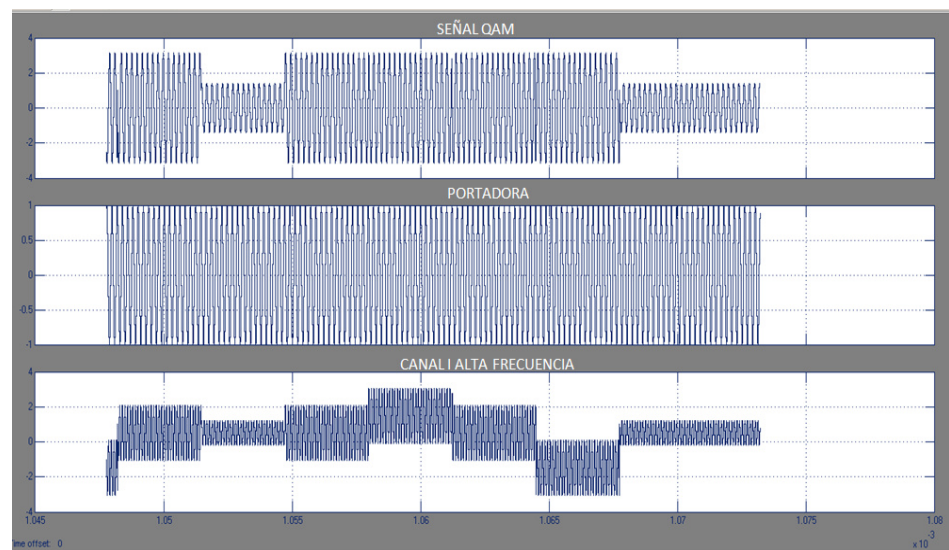


Figura 3.35 Gráfica de la señal modulada, portadora y el canal I en alta frecuencia.

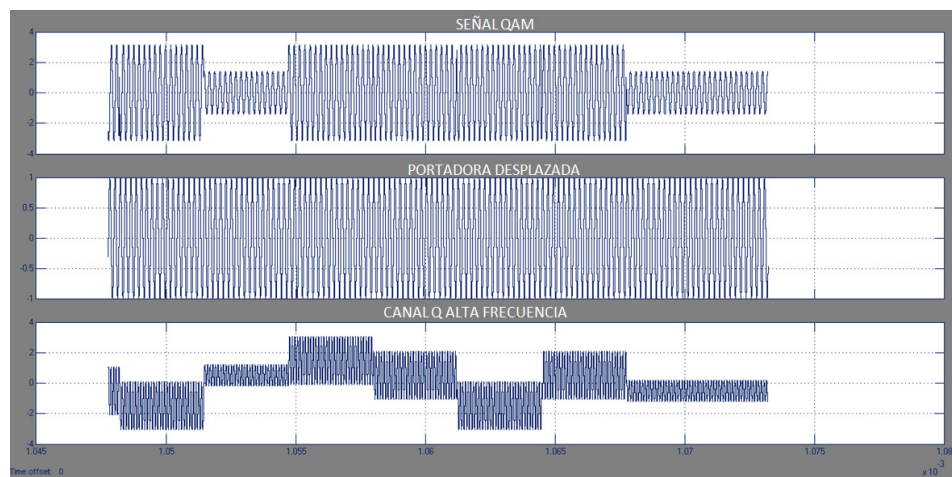


Figura 3.36 Gráfica de la señal modulada, portadora y el canal Q en alta frecuencia.

Las señales en el canal I y Q son filtradas para eliminar las frecuencias altas generadas por la portadora, se muestra en la figura 3.37.

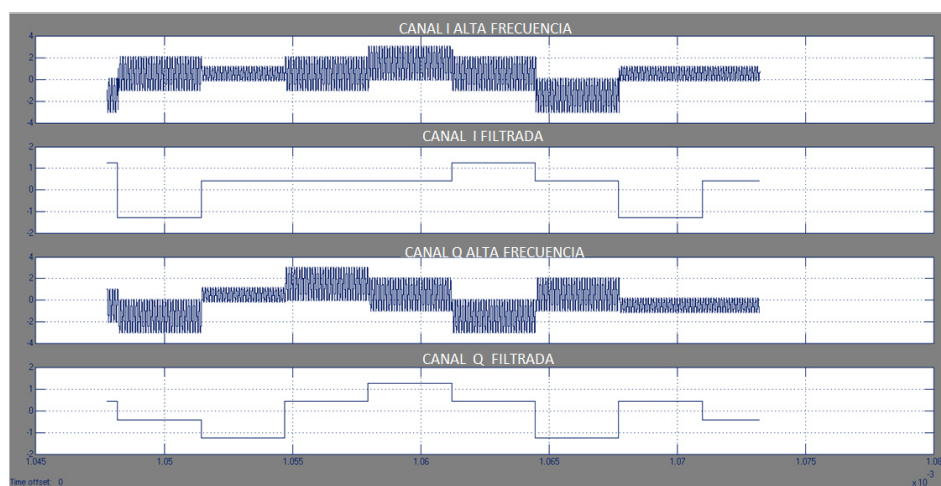


Figura 3.37 Gráfica del canal I-Q en alta frecuencia y del filtro.

En la figura 3.38 y 3.39 se muestra la señal unipolar, su respectivo redondeo de la parte decimal y los ajustes en frecuencia para poder obtener el canal I y Q respectivamente.

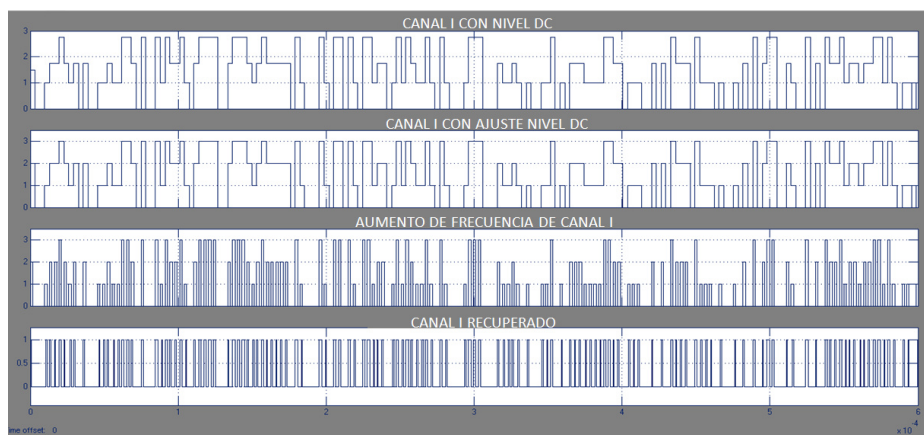


Figura 3.38 Gráfica de canal I con nivel DC, ajuste y recuperación de los bits I.

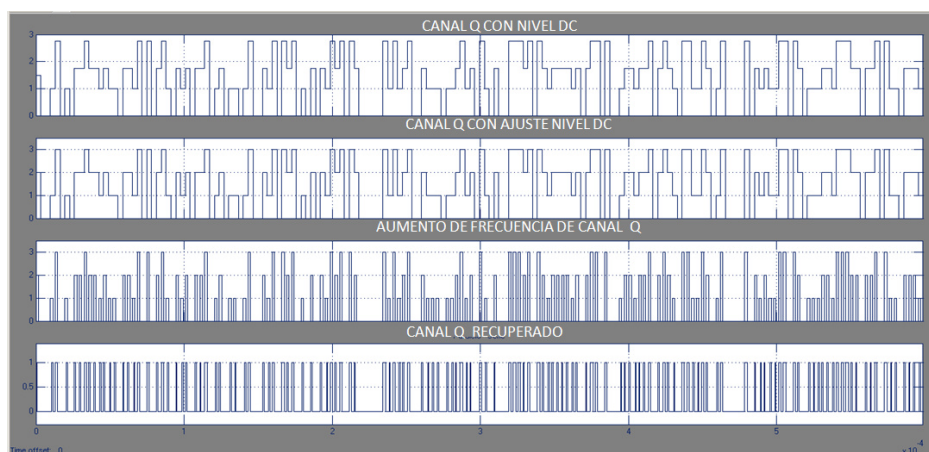


Figura 3.39 Gráfica del canal Q con nivel DC, ajuste y recuperación de los bits Q.

Como ya se tienen los bits I y Q, sumando ambos canales se recupera la señal binaria, en la figura 3.40 se muestra tanto la señal recuperada como la inicial.

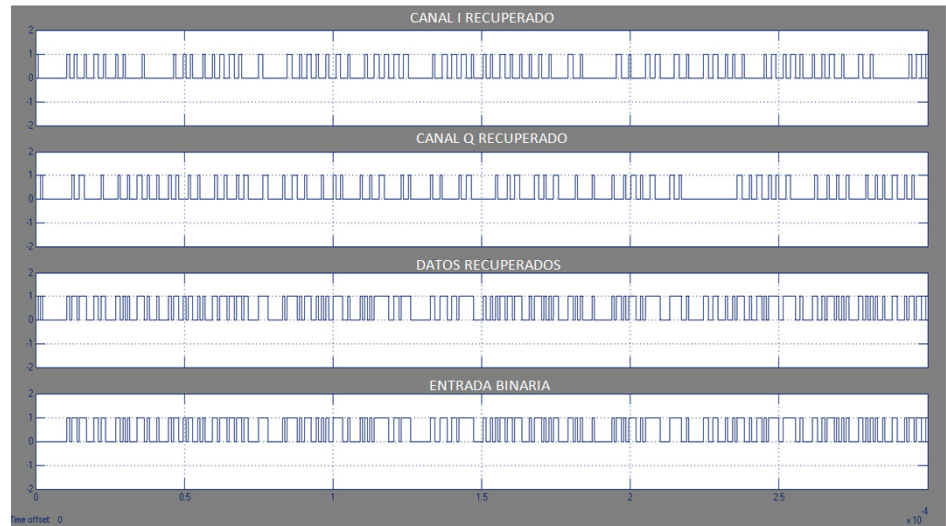


Figura 3.40 Gráfica del canal I recuperado, canal Q recuperado, la señal recuperada y la señal binaria inicial.

CAPÍTULO 4

IMPLEMENTACIÓN DE LOS MODELOS DE COMUNICACIÓN USANDO LA TARJETA “XTREME DSP DEVELOPMENT KIT”

4.1 Implementación de filtros digitales

4.1.1 Implementación de filtro IIR tipo Butterworth

En el Anexo H se detalla la generación del archivo bitstream que se ingresará en la FPGA para la implementación del modelo, una vez creado el archivo se procede a localizar las tarjetas de acuerdo al Anexo I, en la figura 4.1 se muestra la señal de audio a la cual se le aplicará la acción del filtro.

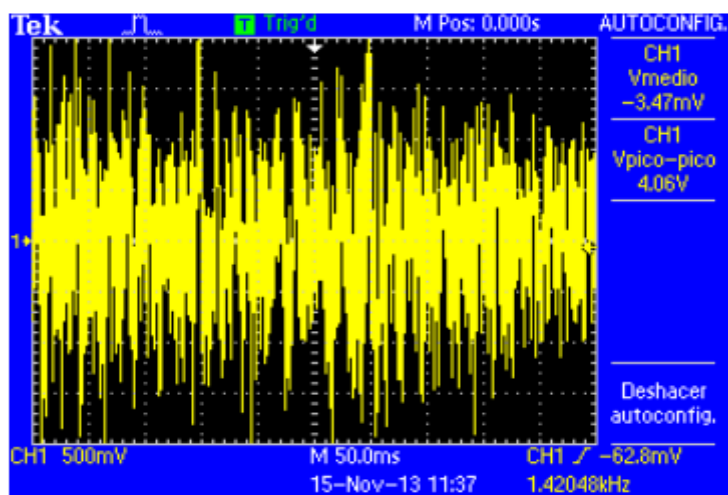


Figura 4.1 Señal de audio con ruido gaussiano.

La respuesta del filtro IIR tipo butterworth se muestra en la figura 4.2, se observa la eliminación de ciertos componentes de ruido.

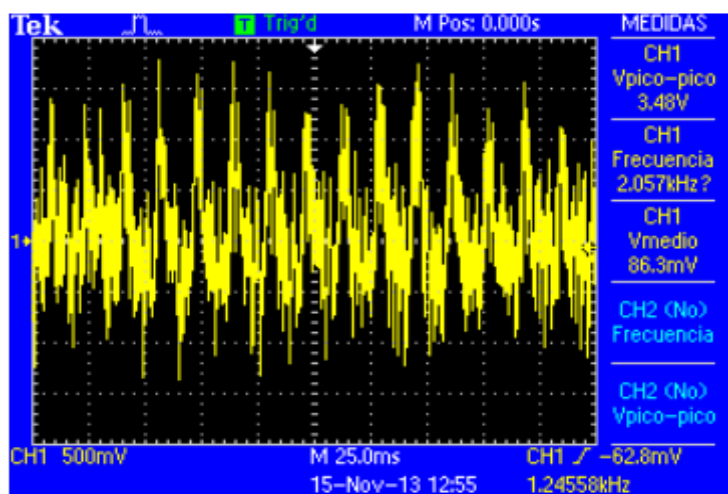


Figura 4.2 Señal con la acción del filtro IIR.

4.1.2 Implementación de filtro FIR tipo LMS

De la misma forma en la que se indicó en la sección 4.1, se genera el archivo .bit y se los carga en las tarjetas tal como se especifican en los anexos H e I, en la figura 4.3 se muestra la señal filtrada.

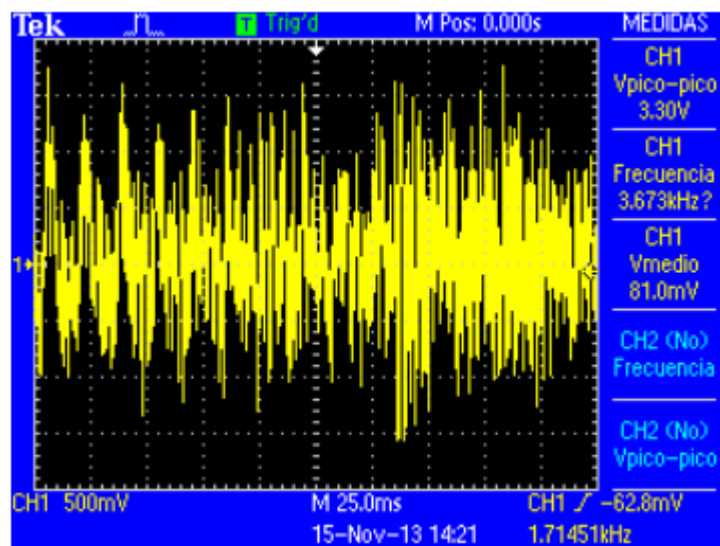


Figura 4.3 Señal con la acción del filtro FIR.

4.2 Implementación de lazo de enganche de fase

Una vez cargado el archivo .bit del modelo del lazo de enganche de fase (PLL), en la figura 4.4 y 4.5 se muestran las gráficas en el osciloscopio que muestran claramente el enganche de la fase.

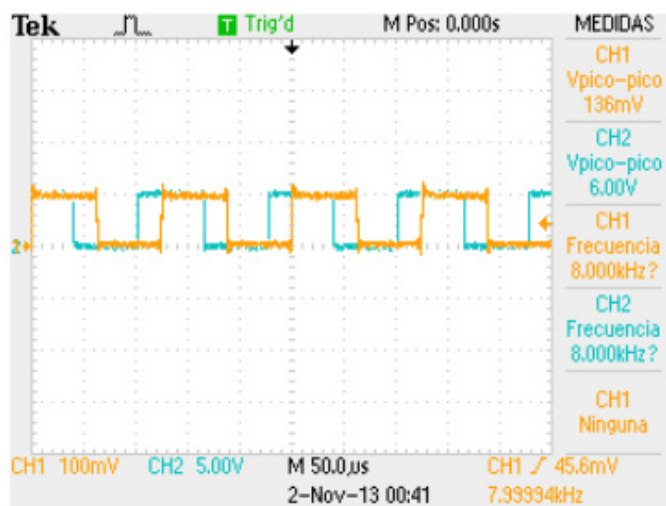


Figura 4.4 Señal referencia y VCO (8Khz).

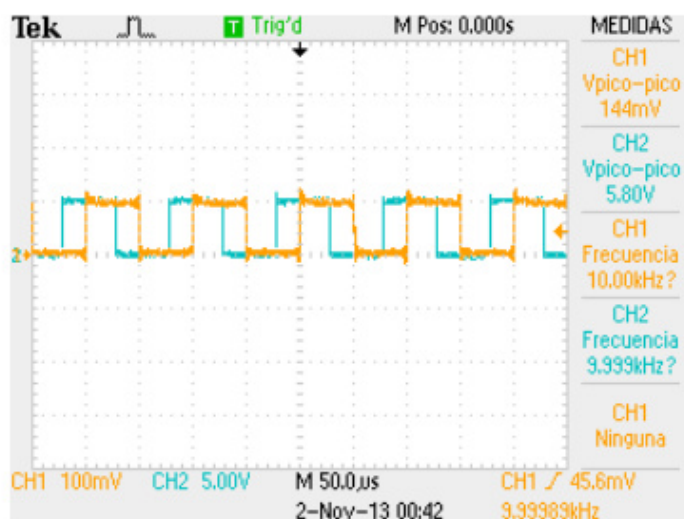


Figura 4.5 Señal referencia y VCO (10KHz)

4.3 Implementación del modulador/demodulador 16QAM

Así como los modelos anteriores, se carga archivo .bit del modelo modulador/demodulador 16QAM, en la figura 4.6 se detalla la señal modulada 16QAM.

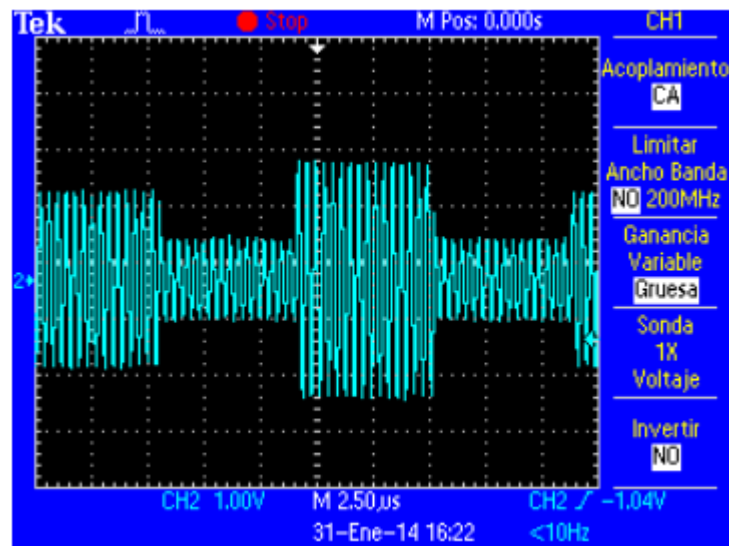


Figura 4.6 Señal modulada 16QAM

También se muestra la constelación que se obtiene al realizar la implementación de modulación en la figura 4.7.

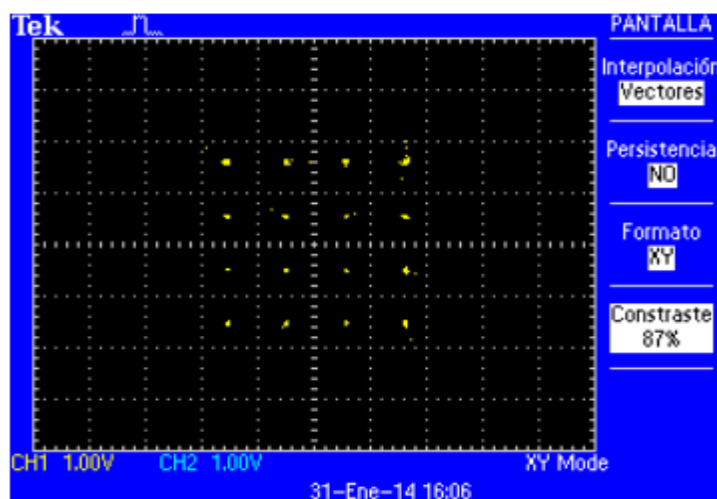


Figura 4.7 Constelación 16QAM en implementación

La recuperación de la señal de entrada binaria con el proceso de la demodulación se muestra en la figura 4.8

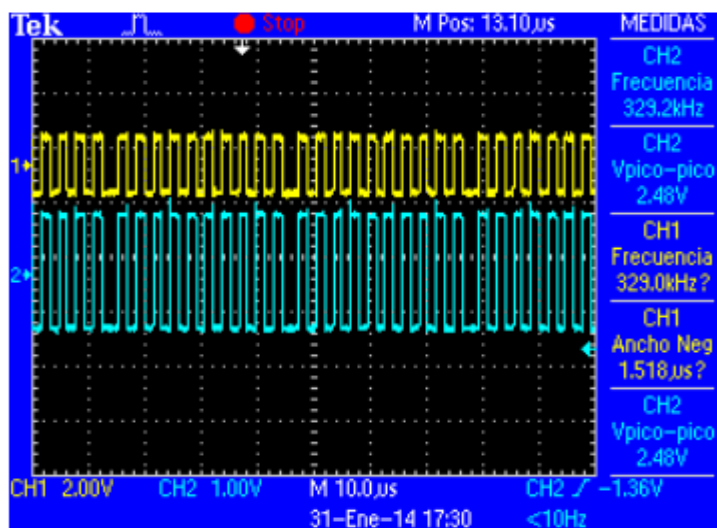


Figura 4.8 Señal demodulada

CAPÍTULO 5

PRUEBAS Y ANÁLISIS DE RESULTADOS

5.1 Filtros digitales

Tanto para la simulación y la implementación de los modelos realizados de filtros digitales se han determinado como característica principal de análisis la relación señal a ruido (SNR) y la tasa de error por bit (BER), la relación entre estos parámetros se detallan en el Anexo G.

En la simulación se calcula el SNR con la ecuación 16, donde P_s es la potencia de la señal y P_r la potencia del ruido, para el cálculo se utiliza un código en Matlab que se especifica en el Anexo E.

$$SNR = 10 \log_{10} \left[\frac{P_s}{P_r} \right] \quad (16)$$

En la implementación de los filtros el SNR se lo calcula con la ecuación 17, para lo cual se utiliza la captura de tablas de datos las cuales se indica en el Anexo F.

$$SNR = 10 \log_{10} \left[\frac{V_{rms_señal}}{V_{rms_ruido}} \right]^2 \quad (17)$$

5.1.1 Filtro IIR tipo Butterworth

Para este filtro se obtiene que la señal filtrada simulada tiene una relación señal a ruido de 11.06 dB y una tasa de error por bit de 0.049, de igual forma la relación señal a ruido de la señal filtrada implementada es 17.25 dB y una tasa de error por bit de 0.0195, como se observa los valores obtenidos para el SNR son distintos, esto se debe a que el código toma toda la señal para el cálculo del mismo, en cambio con el uso de las tablas de datos, solo se toman 4 muestras de 2500 datos cada una, exactamente a cada minuto, lo que hace que el código sea mucho más preciso y eficiente, es claro notar que mientras más muestras se tomen usando la tabla de datos la aproximación será mayor.

5.1.2 Filtro FIR tipo LMS

Usando la misma metodología para este filtro, se obtiene que la señal filtrada simulada tiene una relación señal a ruido de 9.52 dB y una tasa de error por bit de 0.062, y en la implementación se obtiene un SNR de 15.46 dB y un BER de 0.0245.

Es claro observar que tanto en simulación como en implementación, el SNR del filtro IIR es mayor al del FIR, lo cual indica que el filtro IIR elimina más componentes del ruido y por lo tanto es más eficiente en el proceso del filtrado que el filtro FIR.

5.2 Lazo de enganche de fase

Los parámetros que sirven para medir la aplicación y el alcance del lazo de enganche de fase, tanto en simulación como implementación, es su rango de captura y su rango de enganche, la ganancia del VCO seteada para nuestro diseño es de 5, el filtro pasabajos tiene su frecuencia de corte en 14dB, por lo que haciendo un barrido de frecuencias desde 18Khz hacia frecuencias menores, se observa en la simulación que el PLL se engancha aproximadamente en 0.022 segundos, que

corresponde a una frecuencia de 14Khz, en la figura 5.1 se observa el enganche de la señal con el VCO.

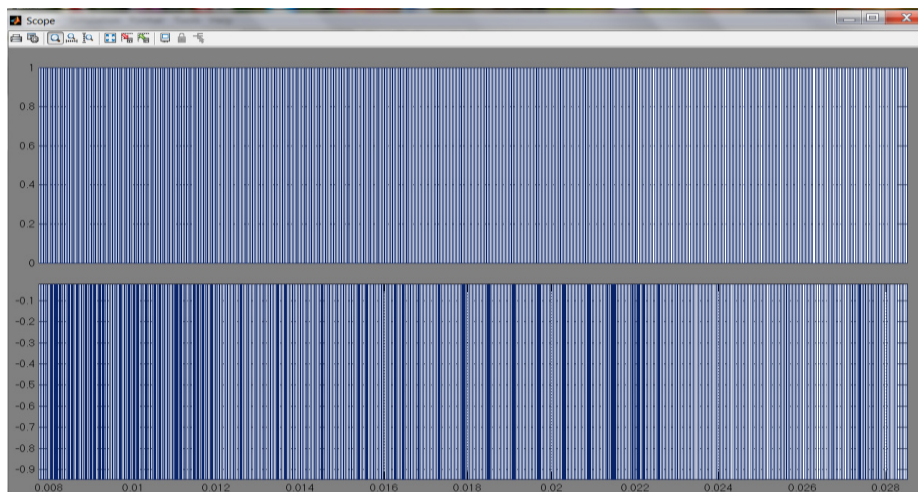


Figura 5.1 Enganche del PLL de altas a bajas frecuencias.

De igual forma realizando un barrido desde 6khz hacia frecuencias mayores, en la figura 5.2 se observa que el PLL se engancha aproximadamente 0.017 segundos, cuya frecuencia corresponde a 8khz.

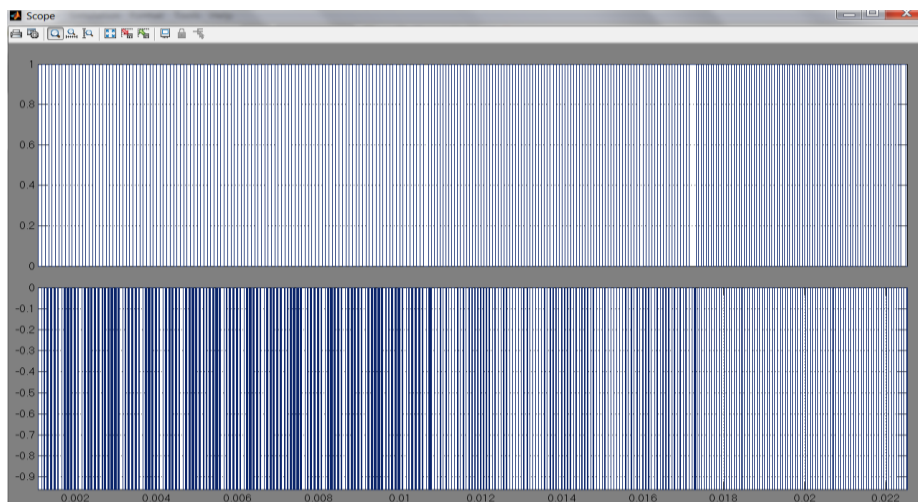


Figura 5.2 Enganche del PLL de bajas hacia altas frecuencias.

Por lo que el rango de captura se define desde 8khz hasta 14khz; es decir, para señales de entrada entre estas frecuencias el PLL cumplirá su función de enganchar la fase.

Una vez enganchado el PLL, al disminuir la frecuencia hasta el límite inferior se obtiene que el PLL se mantiene enganchado hasta 6khz aproximadamente en 0.031 segundos, tal como se muestra en la figura 5.3.

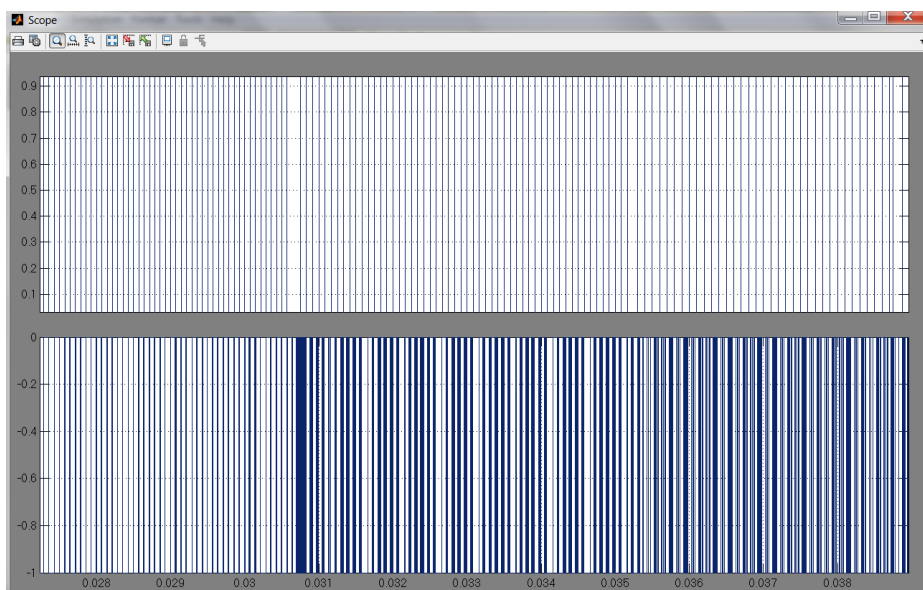


Figura 5.3 Determinación del rango de enganche para el límite de baja frecuencia en simulación.

De igual forma, una vez enganchado el PLL, al aumentar la frecuencia hasta el límite superior se observa que el PLL conserva el enganche hasta 17khz aproximadamente en 0.031 segundos, esto se detalla en la figura 5.4.

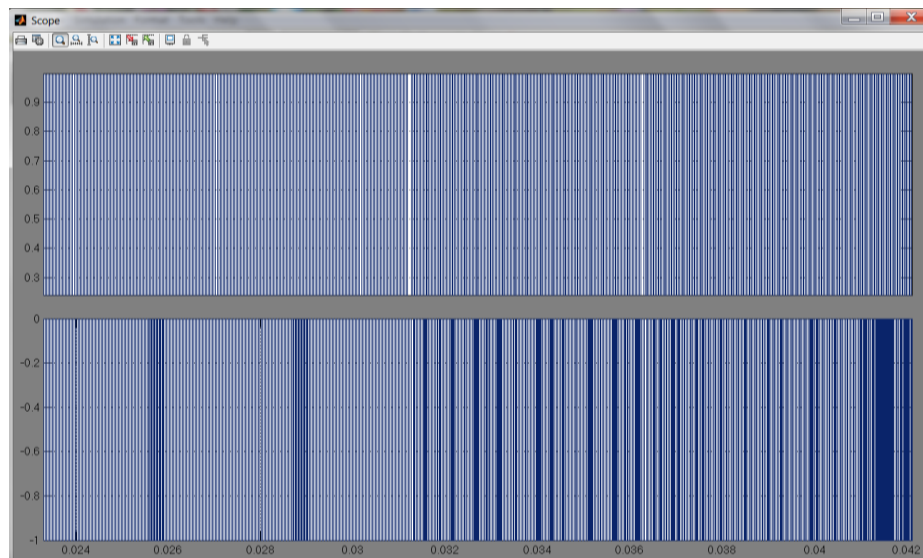


Figura 5.4 Determinación del rango de enganche para el límite en alta frecuencia en simulación.

El rango de enganche se define de 6khz hasta 14khz, se han determinado ambos rangos, por lo que se puede concluir que estos valores dependen directamente de la ganancia del VCO y de la frecuencia de corte del filtro pasabajos.

Para la implementación realizando el mismo barrido y análisis, se obtiene que el PLL se engancha desde la frecuencia inferior de 8Khz, tal como se muestra en la figura 5.5.

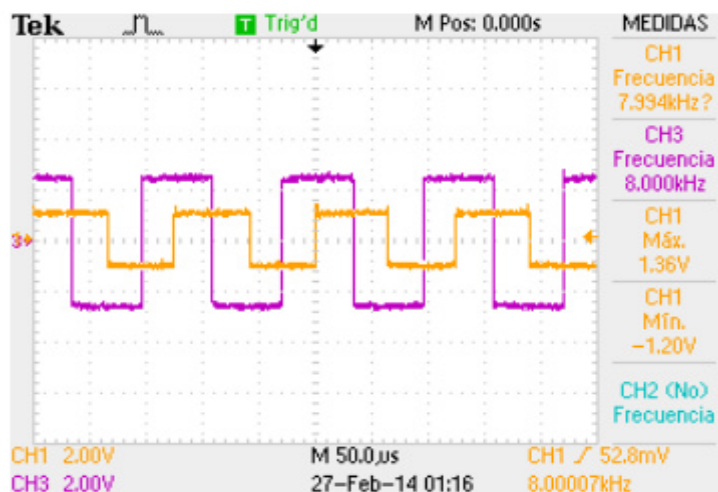


Figura 5.5 Enganche del PLL desde el límite inferior.

Realizando el barrido desde altas frecuencias se observa que el PLL se engancha en la frecuencia 17Khz, tal como se muestra en la figura 5.6.

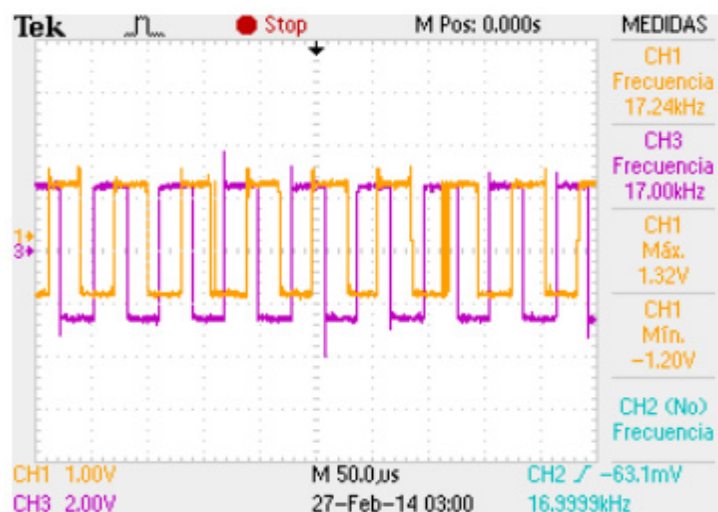


Figura 5.6 Enganche del PLL desde el límite superior.

Por lo que el rango de captura en la implementación se define de 8Khz hasta 17Khz, una vez enganchado el PLL, al disminuir la frecuencia se conserva el enganche hasta 7Khz, como se muestra en la figura 5.7.

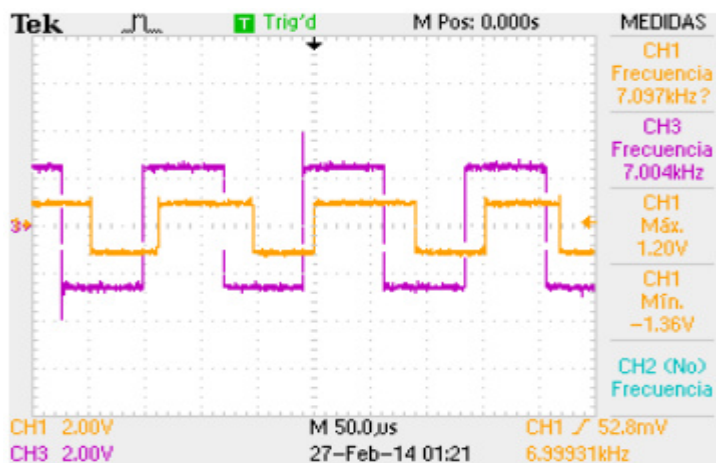


Figura 5.7 Determinación del rango de enganche para el límite de baja frecuencia en implementación.

De igual forma, una vez enganchado el PLL, al aumentar la frecuencia hasta el límite superior se observa que el PLL conserva el enganche hasta 19khz, tal como se detalla en la figura 5.8, por lo que el rango de captura en implementación se define de 7Khz hasta 19 Khz.

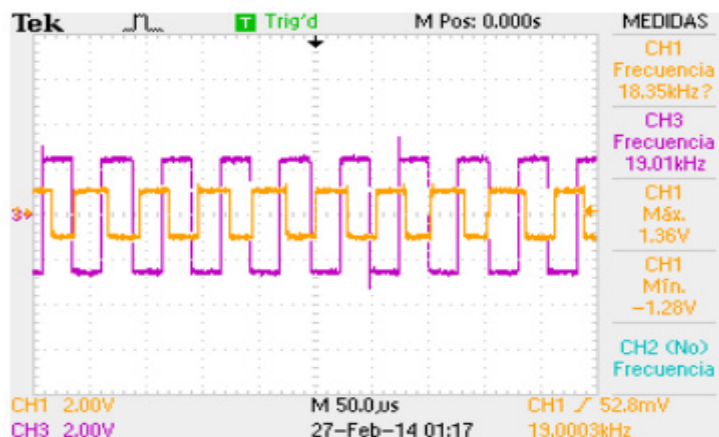


Figura 5.8 Determinación del rango de enganche para el límite de alta frecuencia en implementación.

5.3 Modulador/Demodulador 16QAM

Para el análisis del sistema modulador/demodulador 16QAM se añade un canal AWGN, para poder sacar varios valores de SNR variando la potencia del ruido, la esquematización del sistema general se muestra en la figura 5.5

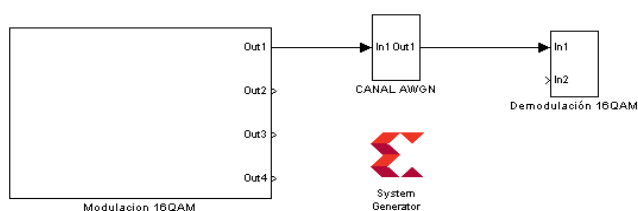


Figura 5.9 Sistema con canal AWGN.

El ruido en el canal AWGN se diseña de tal forma que la media sea cero, con esta premisa la potencia del ruido es equivalente al cuadrado de su desviación estándar y la variación de esta potencia está dada por un factor de ganancia, tal como se muestra en la figura 5.6

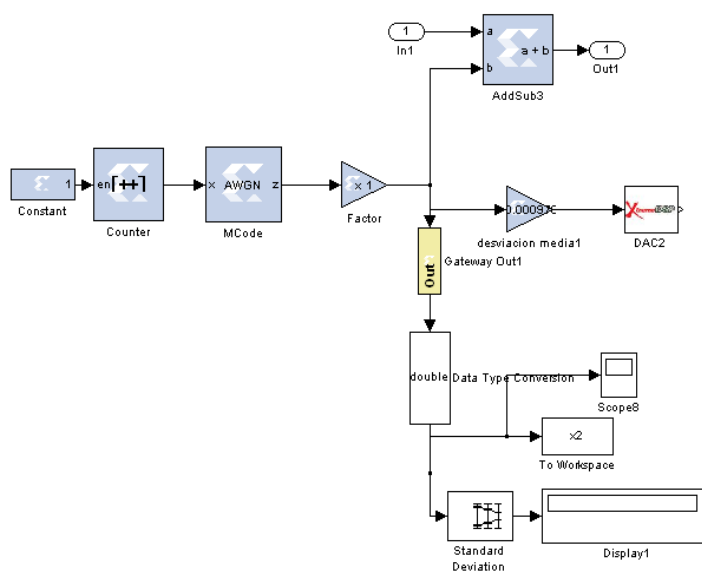


Figura 5.10 Esquema del canal AWGN.

Para el cálculo del SNR se usa la Ecuación 18. y el BER con la relación descrita en el Anexo G.

$$SNR = \frac{\text{Potencia de la señal}}{(\text{Desviación típica del ruido})^2} \quad (18)$$

En la simulación se usan las herramientas de Simulink y Matlab para extraer la desviación típica que representaría la potencia del ruido, al Command Window y trabajar con los códigos propios de Matlab, mientras que para la implementación se usa la manipulación de tablas para estimar la este valor. El cálculo de la potencia de la señal de acuerdo al anexo G se tiene que E_b es 0.625 y f_b corresponde al valor de 100000, por lo que la potencia de la señal es 62500. En la tabla XXVII se muestra los valores de SNR para varios valores de potencia del ruido.

FACTOR MULTIPLICACIÓN	POTENCIA DE SEÑAL MODULADA	DESVIACIÓN ESTÁNDAR DEL RUIDO	SNR
1	62500	0,89	49,00
10		8,87	29,00
50		44,35	15,02
100		88,71	9,00

Tabla XXVII. Valores de SNR simulados para diferentes potencias de ruido.

Tal como se observa, mientras aumenta la potencia del ruido, el SNR es cada vez menor, y es lo que se espera, ya que a mayor ruido en el sistema, la señal se perturba más y aumentan componentes indeseados en frecuencia alterando la misma.

Para la implementación, la potencia de la señal se la obtiene de la constelación generada en el osciloscopio, de acuerdo al anexo G y considerando el factor de escala en el osciloscopio se tiene E_b es 1250000, por lo que se tiene una potencia de la señal de 1250000000000, los valores de potencia que se obtienen mediante tablas y el SNR para cada variaciones de la potencia del ruido se muestran en la tabla XXVIII.

FACTOR MULTIPLICACIÓN	POTENCIA DE SEÑAL MODULADA	DESVIACIÓN ESTÁNDAR DEL RUIDO	SNR
1	1250000000000	1236,76	49,12
10		3067,72	41,23
50		35538,14	19,96
100		51218,19	16,78

Tabla XXVIII. Valores de SNR implementados para diferentes potencias de ruido

Se observa el efecto del aumento de la potencia del ruido sobre el SNR, está claro que la señal se afecta de gran manera cuando el ruido aumenta en el canal de comunicación, de tal forma que se comprueba lo realizado en la simulación del sistema con la congruencia de los datos en la implementación.

CONCLUSIONES

De acuerdo a la investigación, desarrollo e implementación de este proyecto, como a su vez de los distintos problemas que hubo que superar para la finalización del mismo, los autores de esta obra podemos concluir lo siguiente:

1. El filtro IIR realiza un mejor filtrado que el filtro FIR, tanto en simulación como en implementación y en parámetros que se hallaron para el análisis respectivo de ambos, esto se debe a que el algoritmo LMS tiene dos limitantes importantes como la tasa de convergencia y la sensibilidad a variaciones del ambiente, las cuales dependen directamente de la entrada, mientras la dimensión de la misma es mayor, la convergencia será más lenta y también puede ocurrir una

pequeña variación en la correlación de los coeficientes del filtro, otra de las razones por la que el filtro FIR es menos eficiente, radica en que se necesita de una gran carga computacional para poder renovar los coeficientes de una manera óptima.

2. El rango de enganche y el rango de captura del lazo de enganche de fase dependen de los parámetros del filtro pasabajos, específicamente de la frecuencia de corte del mismo, ya que este parámetro define la convergencia del enganche, la amplitud o variación de estos rangos también dependen de la ganancia del oscilador controlado por voltaje (VCO), mientras mayor sea esta ganancia mayor será la gama de frecuencias en los rangos.

3. De acuerdo a las pruebas realizadas, para frecuencias mayores de 100Khz a la entrada del modulador en el sistema 16QAM, no se logra recuperar la entrada binaria a través del demodulador, esto se debe al teorema de Nyquist, ya que la frecuencia de muestreo del sistema es de 1.23Mhz y debe cumplirse de que ésta debe ser mayor al doble de la frecuencia de entrada, es por ello que la demodulación para frecuencias a la entrada mayores al umbral de 100Khz falla.

4. La tarjeta de desarrollo XTREME DSP DEVELOPMENT KIT es óptima para desarrollar tecnologías, pruebas, proyectos, modelos e implementaciones complejas, el interfaz de simulación es Simulink, lo que lo hace más sencillo de manipular teniendo bases en el manejo de este software de Matlab.

RECOMENDACIONES

- 1.** Para realizar cualquier modelo es necesaria la presencia del bloque System Generator, ya que éste bloque es el que permite la simulación del sistema, y la creación del archivo bitstream con el cual se programará la tarjeta FPGA para su posterior implementación.

- 2.** Una correcta y eficiente simulación e implementación depende del correcto seteo de los parámetros de tiempos de sampling y frecuencias del sistema, los bloques de Xilinx que ameriten estos campos deben ser múltiplos de la frecuencia seteada en el System Generator.

BIBLIOGRAFÍA

[1] Nallatech, Interconnect Systems, INC. VIRTEX-4 XTREMEDSP DEVELOPMENT KIT, <http://www.nallatech.com/Development-Kits/virtex-4-xtremesp-development-kit.html>, Fecha de consulta Abril 2013.

[2] Nallatech, Interconnect Systems, INC. XTREMEDSP DEVELOPMENT KIT-IV USER GUIDE, http://www.es.ele.tue.nl/mininoc/doc/xdsp_ug.pdf

Fecha de publicación Marzo 2005, Fecha de consulta Abril 2013.

[3] Jacobus Naude, Xilinx Application Note: Virtex-4, Virtex-II Pro, Virtex-II Families,

http://www.xilinx.com/support/documentation/application_notes/xapp1005.pdf

Fecha de consulta Abril 2013.

[4] Gosh, Ranjan, IMPLEMENTATION OF DIGITAL FIR FILTER ON 8051 MICROCONTROLLER,

<http://iitkgp.vlab.co.in/?sub=39&brch=125&sim=637&cnt=1>

Fecha de consulta junio 2011.

[5] Espinoza Ronal, Coronel Mauro. "IMPLEMENTACIÓN, ANÁLISIS Y COMPARACIÓN DE MÉTODOS DE FILTRADO DE SEÑALES DE AUDIO

AFECTADAS POR RUIDO BLANCO GAUSSIANO ADITIVO Y RUIDO TIPO PULSOS”, Fecha de publicación Enero 2011, Fecha de consulta Abril 2013

[6] Criollo, Edgar Hernando, CANCELACIÓN DE RUIDO, MEDIANTE EL USO DE FILTROS ADAPTATIVOS IMPLEMENTADOS CON ALGORITMOS LMS Y RLS, <http://media.tripod.lycos.com/3168697/1713814.pdf>, Fecha de publicación Abril 2011, Fecha de consulta Julio 2011

[7] Marcelino Martínez, Antonio Serrano, Juan Gómez, INTRODUCCIÓN AL PROCESADO DIGITAL DE SEÑALES, <http://ocw.uv.es/ingenieria-y-arquitectura/1-1/1tema6.pdf> , Fecha de publicación 2009, Fecha de consulta Abril 2013

[8] Robert F. Coughlin, Frederick F. Driscoll, AMPLIFICADORES OPERACIONALES Y CIRCUITOS INTEGRADOS LINEALES,

Fecha de publicación 2000, Fecha de consulta Abril 2013

[9] Instituto Politécnico Nacional. ALGORITMO LMS CON ERROR CODIFICADO USANDO UN DSP.

https://www.google.com.ec/url?sa=t&rct=j&q=&esrc=s&source=web&cd=1&cad=rja&ved=0CCoQFjAA&url=http%3A%2F%2Fitzamna.bnct.ipn.mx%2Fdspace%2Fbitstream%2F123456789%2F3722%2F1%2FALGORITMO%2520LMS.pdf&ei=oOISUpS4BZP68QSCwYDgDQ&usg=AFQjCNEjaumO_UtVczGWb

ballTmfbV1udQ&bvm=bv.53537100,d.eWU. Fecha de publicación Junio 2008. Fecha de consulta Abril 2013

[10] Perez Barragán, FILTROS ADAPTATABLES.

<http://www.ptolomeo.unam.mx:8080/xmlui/bitstream/handle/132.248.52.100/872/A6.pdf?sequence=6>. Fecha de consulta Abril 2013

[11] Cesar. ELECTRÓNICA APLICADA. <http://ayudaelectronica.com/pll-lazos-enganchados-en-fase/>. Fecha de publicación Mayo 2011. Fecha de consulta Abril 2013.

[12] CDMA PRINCIPIOS BÁSICOS.

http://bibing.us.es/proyectos/abreproy/11244/fichero/Volumen+1%252F6_CDMA_PRINCIPIOS_BASICOS.pdf. Fecha de consulta Abril 2013

[13] Escuela Politécnica Superior. COMUNICACIONES DE ESPECTRO ENSANCHADO.

http://arantxa.ii.uam.es/~tac/Documentacion/Tema_III_Espectro_ensanchado_CDMA_ver0.pdf. Fecha de publicación Agosto 2007. Fecha de consulta Mayo 2013.

[14] Estudio Pro Nexus Radical

http://tecnologiahechapalabra.com/tecnologia/glosario_tecnico/articulo.asp?i=789

Fecha de publicación 13 de abril 2007. Fecha de consulta Mayo 2013

[15]Escuela Politécnica Nacional. INTRODUCCIÓN A LA ARQUITECTURA
DE UNA RED CELULAR CDMA.

<http://dspace.epn.edu.ec/bitstream/15000/8580/1/T10106CAP1.pdf>

Fecha de consulta Mayo 2013.

ANEXOS

ANEXO A

SIMULACIÓN E IMPLEMENTACIÓN DEL LAZO DE ENGANCHE DE FASE USANDO XILINX Y SYSTEM GENERATOR

1. OBJETIVOS

- Estudiar los principios del lazo de enganche de fase (PLL).
- Simular e implementar un lazo de enganche de fase (PLL) usando la librería de Xilinx.
- Analizar y determinar el rango de enganche y de captura.

2. FUNDAMENTOS TEÓRICOS.

2.1. CARACTERÍSTICAS Y COMPONENTES

El lazo de seguimiento de fase (PLL) es un circuito en el que un oscilador sigue la fase de una señal de entrada, a través de una retroalimentación que compara la fase de las dos señales y modifica la frecuencia de la oscilación generada, consta de tres partes importantes:

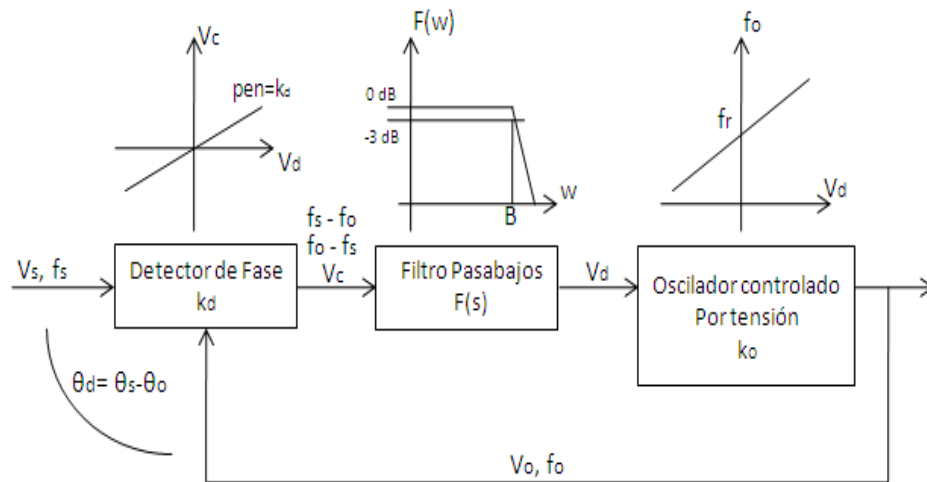


Figura 2.1 Diagrama de bloques del PLL

- Comparador de fase

Dado que el PLL es un lazo de retroalimentación, en esta etapa se compara la fase y la frecuencia de la señal entrante con la frecuencia del VCO. En el caso de que no hubiera señal entrante, el VCO oscilaría a una frecuencia f_0 , que se la conoce como frecuencia de corrida libre o frecuencia libre de oscilación. En el caso de tener una frecuencia a la entrada f_e , el comparador de fases también funciona como un mezclador, dando como resultado una mezcla de ambas frecuencias ($f_0 - f_e$, $f_e - f_0$, $2f_0$, $2f_e$, $f_0 + f_e$). Siendo $f_0 \neq f_e$.

- Filtro pasa bajo

Las mezclas de alta frecuencia, como las componentes sumas arrojadas por el detector de fase ($f_0 + f_e$, $2f_0$, $2f_e$), son anuladas por el filtro pasa bajo por estar fuera de su ancho de banda, este es el trabajo de el filtro, eliminar las frecuencias fuera de banda que pueda arrojar el mezclador de frecuencias en la parte inicial y considerar las que están más cercanas a la frecuencia de oscilación f_0 y solo deja pasar la componente DC. Otra función del filtro pasa bajos es de asegurar que el enganche se realice de una manera más rápida y eficiente.

- Oscilador controlado por voltaje

Las frecuencias que salen del filtro pasa bajos ahora ingresan al VCO, si estas frecuencias con las mismas o muy cercanas a la frecuencia libre de oscilación f_0 , se da el enganche de fase y $f_0 = f_e$, en el caso de que estas frecuencias aun fueran diferentes se repite el proceso hasta que ambas frecuencias se igualen y se dé el enganche.

2.2. APLICACIONES

La finalidad de estos tres componentes retroalimentados es la de poder igualar la fase del VCO con la fase de la señal entrante, se detalló el funcionamiento de cada componente para lograr el objetivo del enganche, el PLL es indispensable para los sistemas de telecomunicaciones y para la electrónica, a continuación se detallan algunas aplicaciones importantes del mismo.

- Filtros

A través del funcionamiento del PLL pueden desarrollarse filtros de fase, para poder reconstruir o recuperar señales que hayan sido perturbadas por ruido de fase o fluctuaciones de frecuencia.

- Moduladores/Demoduladores

La finalidad del PLL es enganchar un par de frecuencias, por lo que puede utilizarse este circuito para desarrollar moduladores y demoduladores de señales en frecuencia. También pueden desarrollarse moduladores en ángulo.

- Circuitos de sincronismo para barrido horizontal y vertical
- Generación de osciladores

3. DESCRIPCIÓN DEL MODELO

En la figura 3.1 se muestra el diseño que se empleará tanto para la simulación e implementación del mismo.

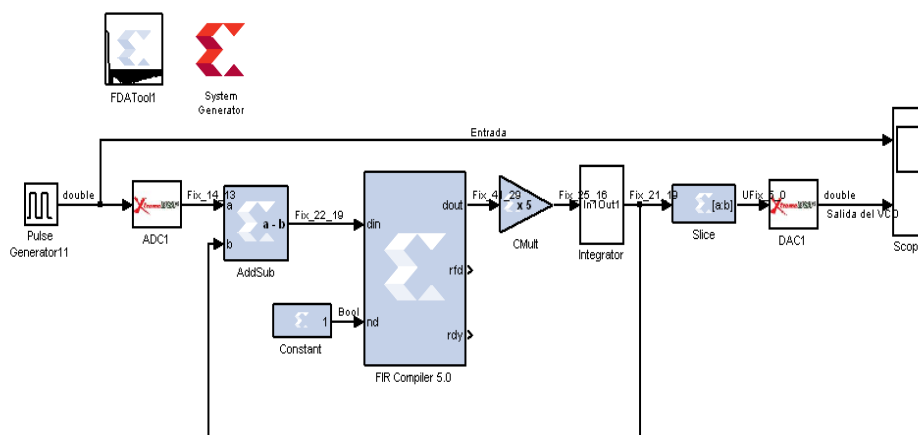


Figura 3.1 Diseño del modelo del lazo de enganche de fase.

La señal de entrada es un generador de pulsos, que será la señal de referencia del PLL para lograr el enganche, los bloques ADC y DAC permiten la conversión analógico/digital o digital/analógico de la señal entrante, la señal referencia y del VCO se la puede observar y comparar a través del Scope.

El bloque System Generator es necesario tanto para la simulación como para la implementación, ya que aquí se define el clock con el que se realizará la simulación y también genera un archivo .bit necesario para la implementación.

El bloque FDATool es donde se crea la estructura del filtro para posteriormente introducir el diseño en el bloque FIR Compiler.

El bloque AddSub realiza la función del comparador de fase entre la señal entrante y el VCO.

El bloque Slice nos permite hacer un ajuste de los bits para poder observar más claramente la salida del VCO.

El bloque Cmult y el subsistema Integrator realizan la función del VCO, el subsistema integrator se muestra en la figura 3.2.

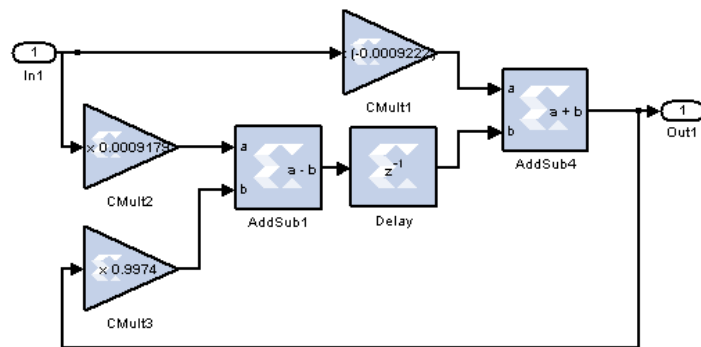


Figura 3.2 Diseño del subsistema Integrator.

4. EQUIPOS REQUERIDOS

- Cables
- Generador de funciones
- Osciloscopio

5. EXPERIMENTOS SOBRE FILTROS DIGITALES

Las tablas que se indican a continuación se encuentran en el archivo PLL.xls

Experimento 1: Simulación e Implementación del lazo de enganche de fase (PLL).

A. Diseño del modelo del lazo de enganche de fase (PLL)

1. Realice el modelo mostrado en la figura 3.1
2. Configure el bloque FDATool para formar un filtro tipo pasabajos con frecuencia de corte de 24kHz, frecuencia de paso de 14kHz y frecuencia de muestreo de 1000kHz., tal como se muestra en la figura 5.1

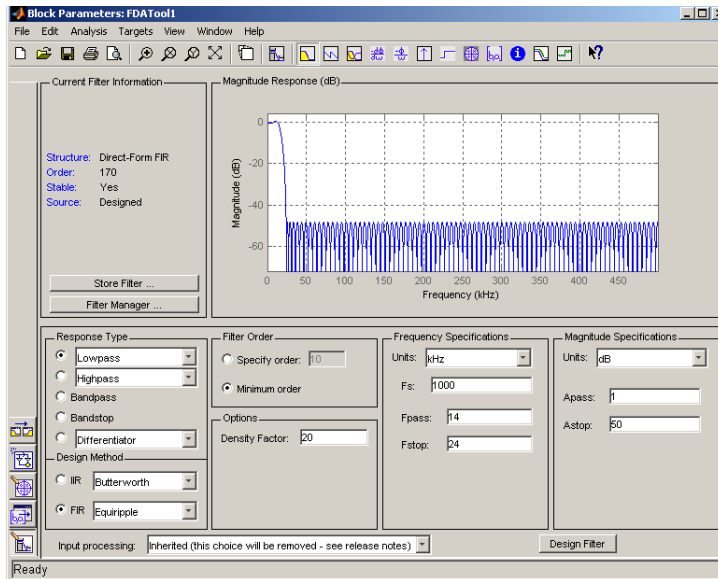


Figura 5.1 Configuración del FDATool.

3. Insertar el filtro creado en el FDATool al bloque FIR Compiler a través del código xlfda_numerator('FDATool1') en la pestaña Coefficient Vector
4. Configure el bloque Slice tal como se especifica en la figura 5.2

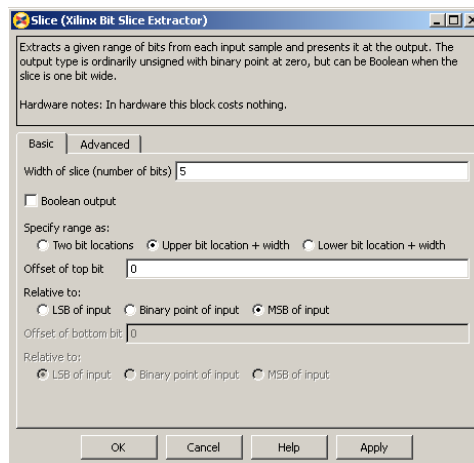


Figura 5.2 Configuración del bloque Slice.

5. Setee los valores de los convertidores ADC/DAC en 1/1000000.

6. Setee el clocking del bloque System Generator para que el periodo de la FPGA sea de 1000ns y la simulación a 1/1000000 segundos, tal como se muestra en la figura 5.3

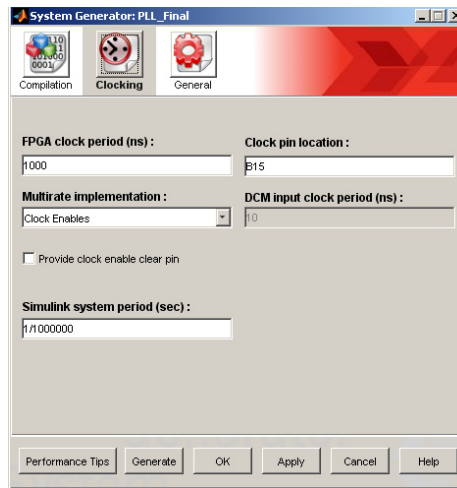


Figura 5.3 Configuración del Clock del System Generator.

7. Simule el modelo y observe el Scope.

B. Implementación del modelo del lazo de enganche de fase (PLL).

1. Genere el archivo bitstream haciendo click en la pestaña *Generate* del bloque System Generator tal como se indica en la figura 5.4

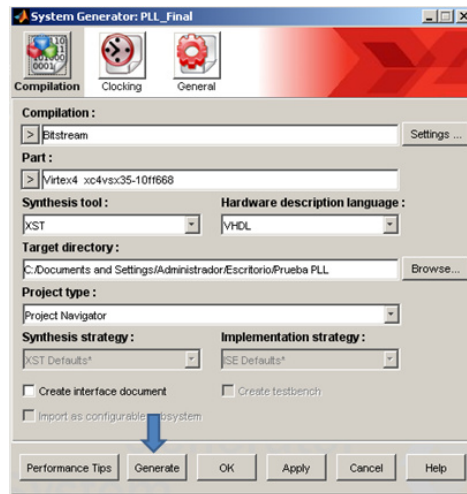


Figura 5.4 Generación del archivo bitstream.

2. Conecte los cables que están en la tarjeta de desarrollo DSP XTREME DEVELOPMENT KIT a la FPGA, tal como se muestra en la figura 5.5, para poder observar la función del archivo .bit del modelo simulado y observarlo en el osciloscopio.

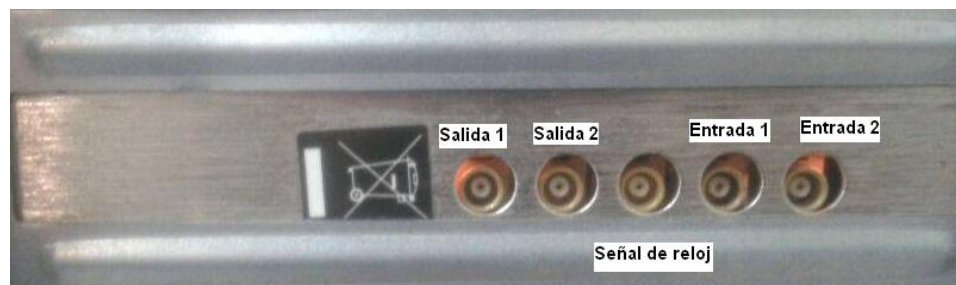


Figura 5.5 Configuración del System Generator.

3. Abra el programa Fuse Probe y localice las tarjetas de la FPGA a través del menú **Open Card>>Locate Cards**, tal como se muestra en la figura 5.6

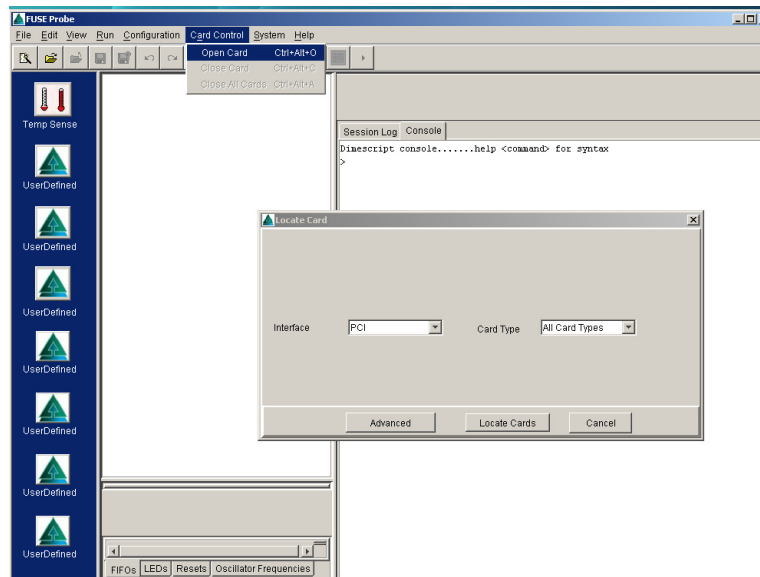


Figura 5.6 Localización de las tarjetas.

4. Coloque en cada tarjeta el archivo .bit correspondiente haciendo click derecho sobre ella y seleccionando la opción *Assign Bitfile*, en uno de ellos se carga el archivo para el clock y en el restante el archivo del modelo simulado.
5. Configure las tarjetas haciendo click en la pestaña **Configuration>>Configure all cards**
6. Con el generador de funciones ingrese una señal cuadrada en Entrada1, y observe la salida a través del osciloscopio.

ANEXO B

SIMULACIÓN E IMPLEMENTACIÓN DE FILTROS DIGITALES USANDO XILINX Y SYSTEM GENERATOR

1. OBJETIVOS

- d) Estudiar los principios de los filtros digitales IIR y FIR.
- e) Simular e implementar un filtro digital IIR tipo RLS usando la librería de Xilinx.
- f) Simular e implementar un filtro digital FIR tipo LMS usando la librería de Xilinx.
- g) Analizar y comparar ambas clases de filtros digitales.

2. FUNDAMENTOS TEÓRICOS.

2.1. FILTROS

Un filtro es un “dispositivo” diseñado para dejar pasar ciertas partes de una señal y retener otras; pudiendo ser una determinada frecuencia o gama de frecuencias, consiguiendo modificar tanto la fase como la amplitud.

El filtrado “es el proceso de seleccionar, suprimir o atenuar ciertas componentes de una señal”. El propósito es de separar componentes de una señal que la distorsionen.

2.2. CLASIFICACIÓN DE FILTROS

Una clasificación general de los filtros son analógicos y digitales, los primeros dedicados a las señales analógicas (valores dentro de un intervalo) y los segundos a las señales digitales (datos discretos). Los filtros digitales tienen diferentes clasificaciones: de acuerdo a su respuesta en frecuencia se clasifican en Pasa-bajos, Pasa-altos, Pasa-banda y Rechazo de banda; de acuerdo a su respuesta ante una entrada impulso se dividen en IIR (Respuesta Infinita al Impulso) y FIR (Respuesta Finita al Impulso); y finalmente si se los analiza de acuerdo a su estructura, se clasifican en cascada, serie y latice.

Para filtrar señales analógicas esta se la hace digital a través de un convertidor Analógico-Digital, se le hace el proceso de filtrado digital y finalmente se convierte esa señal en analógica (Ver Figura 2.1). Debido a este proceso es que se surgen filtros digitales, tanto en IIR como en FIR.



Figura 2.1 Proceso de filtrado digital de una señal analógica.

Una característica fundamental de los filtros digitales es la de poder cambiar su comportamiento, es decir sus coeficientes, éstos cambian de valor a medida que se actualiza la información que disponen, siguiendo un procedimiento llamado algoritmo adaptativo. Cuando se diseña el filtro no se conoce el valor de los coeficientes, estos se calculan al implementarlo y se van actualizando en cada iteración mientras dura su etapa de aprendizaje.

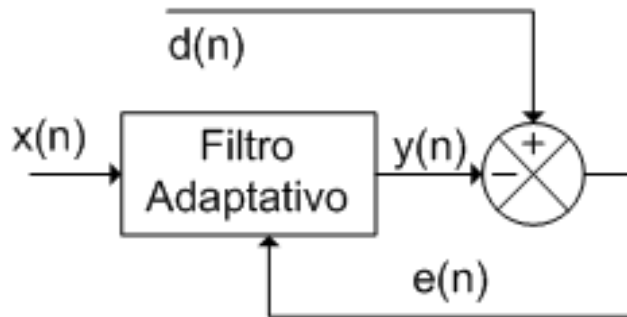


Figura 2.2 Componentes de un sistema de filtrado adaptativo

En la Figura 2.2 se muestra el proceso de un filtro adaptativo, donde $x(n)$ es la señal de entrada, $d(n)$ es la señal deseada (óptima), $y(n)$ es la salida del filtro, $e(n)$ es la señal de error, el cual se define como la diferencia entre la señal deseada y la señal de salida.

El LMS es un algoritmo del filtro adaptativo, el cual se ha propuesto para adaptar el orden y los coeficientes del filtro simultáneamente, se desarrollarán los modelos del filtro IIR tipo Butterworth además del filtro FIR tipo LSM.

2.2.1 FILTRO IIR TIPO BUTTERWORTH

En los filtros IIR (Infinite Impulse Response) o respuesta infinita al impulso, como su nombre indica, si la entrada fuese una señal impulso, la salida tendría un número infinito de términos no nulos, es decir, nunca vuelve al reposo. Son llamados filtros recursivos, porque la salida del filtro depende de las entradas actuales y de las salidas en instantes anteriores, esto se consigue mediante el uso de realimentación de la salida como se muestra en la Figura 2.3.

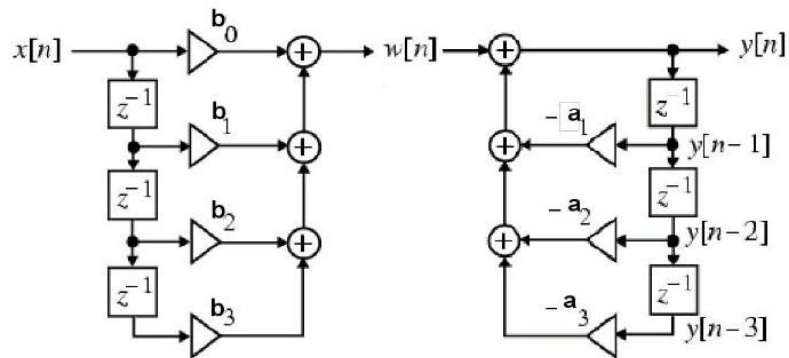


Figura 2.3 Esquema de implementación de un filtro IIR

Donde a y b son los coeficientes del filtro. El orden es el máximo entre los valores de M y N respectivamente para a y b.

M y N son los términos que determinan la cantidad de polos y ceros en la función de transferencia.

$$H(z) = \frac{\sum_{k=0}^{N-1} b_k z^{-k}}{1 + \sum_{k=0}^{M-1} a_k z^{-k}}$$

Debido a la presencia de polos el filtro se puede volver inestable, además no garantizan que su función de transferencia sea lineal y la implementación física de es más compleja.

Dentro de los filtros IIR está el filtro de Butterworth, denominado también filtro de máximo plano o plano-plano, es aquel cuya salida se mantiene constante casi hasta la frecuencia de corte.

En la Figura 2.4 se muestra la respuesta en frecuencia ideal (línea continua) y la práctica (líneas punteadas) para tres tipos de filtros Butterworth, conforme las pendientes se vuelven más pronunciadas se aproximan más al filtro ideal.

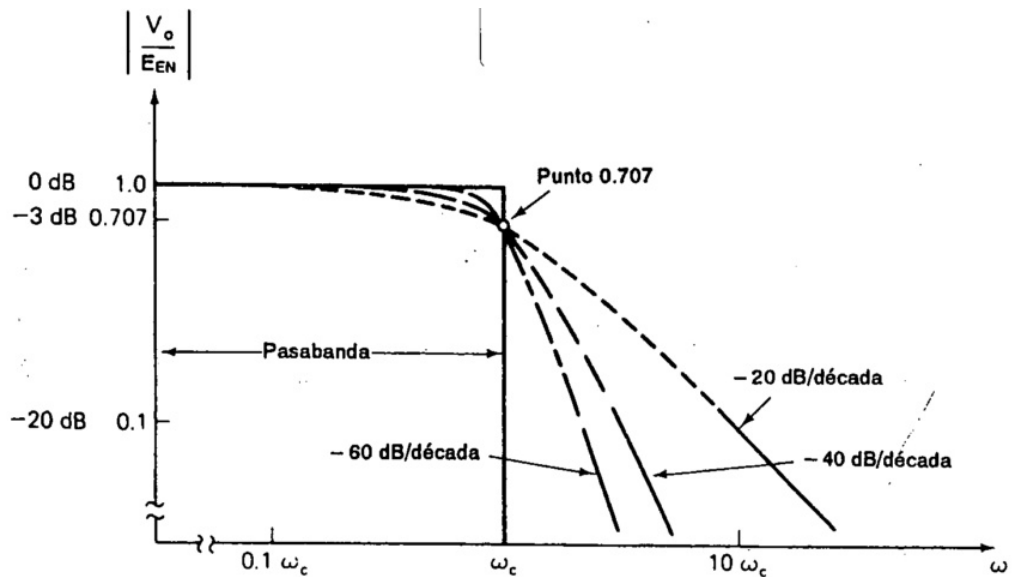


Figura 2.4 Gráfica de respuesta en frecuencia para tres tipos pasabajas Butterworth.

Cabe destacar que un filtro Butterworth puede ser pasa bajas, pasa altos, pasa banda o rechaza banda; para esta práctica se desarrollará el filtro Butterworth pasa baja.

2.2.2 FILTRO FIR TIPO LMS

Los filtros FIR (Finite impulse response) o respuesta finita al impulso, tienen la particularidad de que sus coeficientes son cero, lo que significa que la respuesta del filtro depende solamente de la entrada y no de valores pasados de la salida. Este tipo de filtros tiene una respuesta finita ya que no exhiben recursión (Ver Figura 2.5).

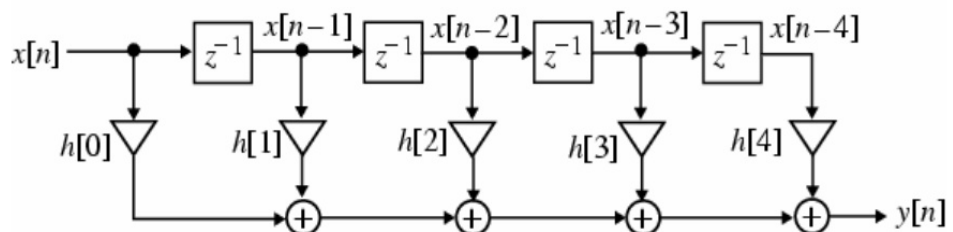


Figura 2.5 Esquema de implementación del filtro FIR.

Una de las propiedades de este filtro es la simetría de sus coeficientes y también la ventaja de poder ser diseñados de tal forma de exhibir una respuesta de fase lineal siendo su función de transferencia:

$$H(z) = \sum_{n=0}^N b_n z^{-n}$$

El algoritmo LMS determina el mínimo del cuadrado de la señal de error, por medio del método de descenso de gradiente, el cual ajusta los coeficientes a manera de pasos de manera que minimice el error.

Cada grupo de coeficientes W forma un punto en la superficie de error, como se observa en la Figura 2.6, y en cada iteración el punto se desplaza por la tangente de dicho punto.

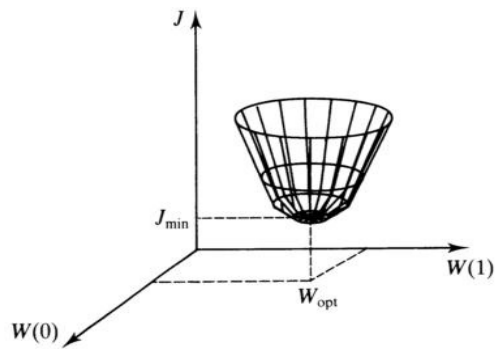


Figura 2.6 Gráfica de la superficie del error.

Para este algoritmo se considera que las señales $d(n)$ y $x(n)$ de la figura 2.2 son estacionarias debido a que este tipo de señales tiene una superficie de error invariable y es más fácil que el algoritmo converja al punto mínimo del error.

El concepto del descenso de gradiente se expresa por:

$$w(n+1) = w(n) - \frac{\mu}{2} \nabla \xi[n]$$

Donde μ es factor que controla la estabilidad y el ritmo de descenso al fondo de la superficie de error. El vector $\nabla \xi[n]$ especifica el gradiente de la función de error con respecto a $w[n]$.

$$\nabla \xi[n] = E \left[\frac{\partial e^2(n)}{\partial w_0}, \frac{\partial e^2(n)}{\partial w_1}, \dots, \frac{\partial e^2(n)}{\partial w_{n-1}} \right]^T$$

Obtener este gradiente se dificulta por el operador de expectación el cual requiere conocimiento de la probabilidad de la señal de entrada para poder ser calculado, por lo que, se usa el error cuadrático instantáneo para estimar el error cuadrático medio.

$$\xi[n] = e^2[n]$$

De la Figura 2.2 se obtiene

$$e(n) = d(n) - W^T X(n)$$

Entonces:

$$\nabla e[n] = -x[n]$$

La estimación del gradiente se vuelve:

$$\nabla \xi[n] = -2x[n]e[n]$$

Por lo que la ecuación final es:

$$w(n+1) = w(n) + \mu e(n)x(n)$$

Donde $w(n)$ es el vector de peso, $x(n)$ es la entrada de referencia, $e(n)$ es la señal de error (Ver Figura 2.7).

El rango del valor de μ esta dado en:

$$0 < \mu \leq \frac{1}{N\overline{x^2(n)}}$$

Siendo $\overline{x^2(n)}$ el cuadrático medio de la potencia de $x(n)$ y N el número de coeficientes del filtro. Mientras más grande es μ la velocidad de convergencia y el error cuadrático medio aumenta.

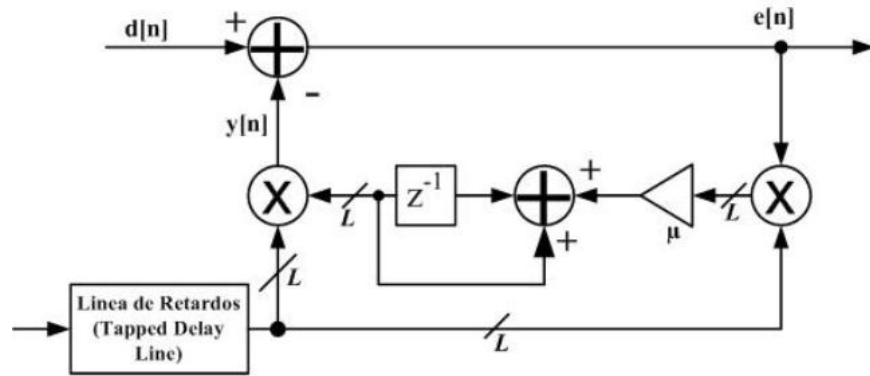


Figura 2.7 Modelo filtro FIR adaptativo LMS

En la Figura 2.7 se observa la estructura básica de un filtro LMS donde L es el orden del filtro variable.

3. DESCRIPCIÓN DEL MODELO

3.1 FILTRO IIR TIPO BUTTERWORTH

En la figura 2.8 se muestra el diseño que se empleará tanto para la simulación e implementación del mismo.

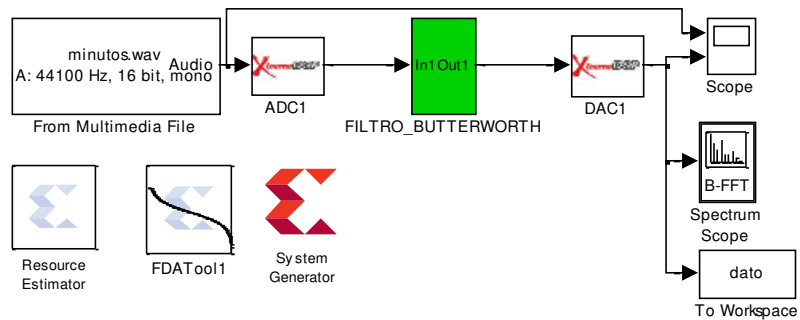


Figura 2.8 Diseño del modelo del filtro IIR tipo RLS

La señal de entrada es una música a la que se le añadió ruido blanco gaussiano, la cual será filtrada para poder recuperar la señal original, los bloques ADC y DAC permiten la conversión analógico/digital o digital/analógico de la señal entrante y la filtrada, la respuesta del filtro en frecuencia se la puede analizar a través del Spectrum Scope, también se puede visualizar y comparar la señal entrante con la filtrada a través del Scope.

El bloque System Generator es necesario tanto para la simulación como para la implementación, ya que aquí se define el clock con el que se realizará la simulación y también genera un archivo .bit necesario para la implementación.

El bloque Resource Estimator sirve para optimizar los recursos de la tarjeta cuando se realice la implementación del mismo.

El bloque FDATool es de donde se copia la estructura del filtro para poder crearlo con bloques básicos de la librería de Xilinx en Simulink.

3.2 FILTRO FIR TIPO LMS

En la figura 2.9 se muestra el diseño que se empleará tanto para la simulación e implementación del mismo.

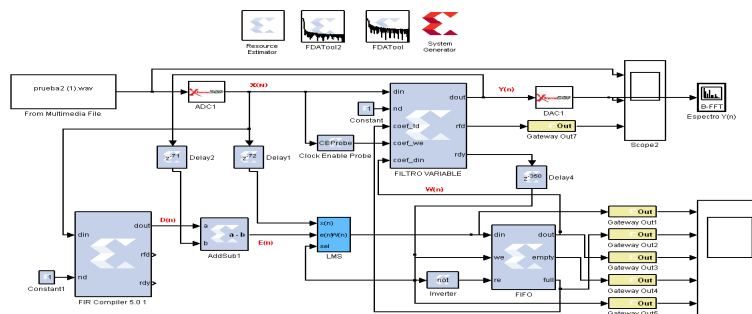


Figura 2.9 Diseño del modelo del filtro FIR tipo LMS

La señal de entrada es una música a la que se le añadió ruido blanco gaussiano, la cual será filtrada para poder recuperar la señal original, los bloques ADC y DAC permiten la conversión analógico/digital o digital/analógico de la señal entrante y la filtrada, la respuesta del filtro en frecuencia se la puede analizar a través del Spectrum Scope, también se puede visualizar y comparar la señal entrante con la filtrada a través del Scope.

El bloque System Generator es necesario tanto para la simulación como para la implementación, ya que aquí se define el clock con el que se realizará la simulación y también genera un archivo .bit necesario para la implementación.

El bloque Resource Estimator sirve para optimizar los recursos de la tarjeta cuando se realice la implementación del mismo.

El bloque FDATool es de donde se copia la estructura del filtro para poder crearlo con bloques básicos de la librería de Xilinx en Simulink, para este caso se tienen 2 bloques de este tipo, ya que uno se usará para el filtro “variable” y el restante servirá para obtener la señal deseada $d(n)$.

El bloque CEProbe sirve para extraer la señal de clock de otra señal.

El bloque FIFO almacenará los nuevos coeficientes creados para la renovación de los mismos en el filtro “variable” para cada interacción.

El bloque LMS es el que nos permitirá crear los nuevos coeficientes del filtro “variable”, su estructura recursiva se la detalla en la figura 2.10.

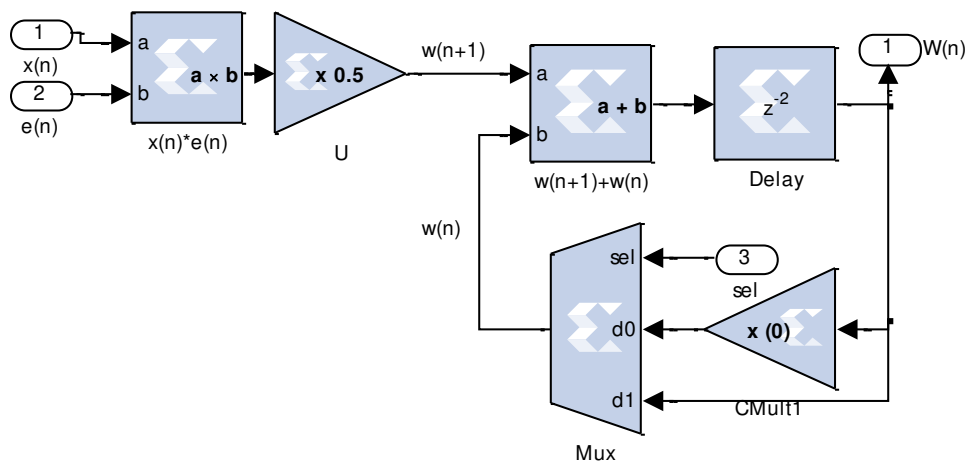


Figura 2.10 Algoritmo LMS

La señal de error $e(n)$ se la obtiene con el bloque AddSub, el cual restará la señal deseada $d(n)$ con la salida $y(n)$.

4. EQUIPOS REQUERIDOS

- Cables
- Cable de audio
- Generador de funciones
- Osciloscopio

5. EXPERIMENTOS SOBRE FILTROS DIGITALES

Las tablas que se indican a continuación se encuentran en el archivo *FILTROS.xls*

Experimento 1: Simulación e Implementación de un sistema usando el filtro digital IIR

A. Diseño del modelo del filtro IIR tipo butterworth

8. Realice el modelo del filtro mostrado en la figura 2.8
9. Configure el bloque FDATool para formar un filtro tipo pasabajos con frecuencia de corte en baja de 15kHz y frecuencia de corte en alta de 20kHz, tal como se muestra en la figura 4.1,

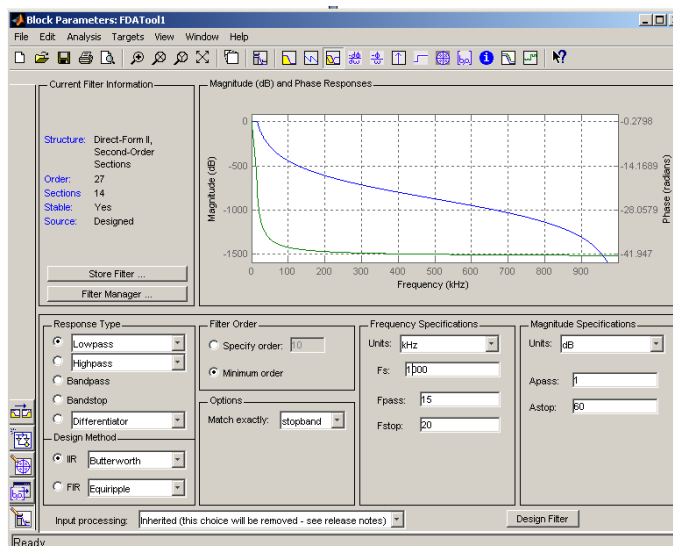


Figura 4.1 Configuración del FDATool.

10. Exporte el filtro a Simulink con bloques básicos, tal como se indica en la Figura 4.2.

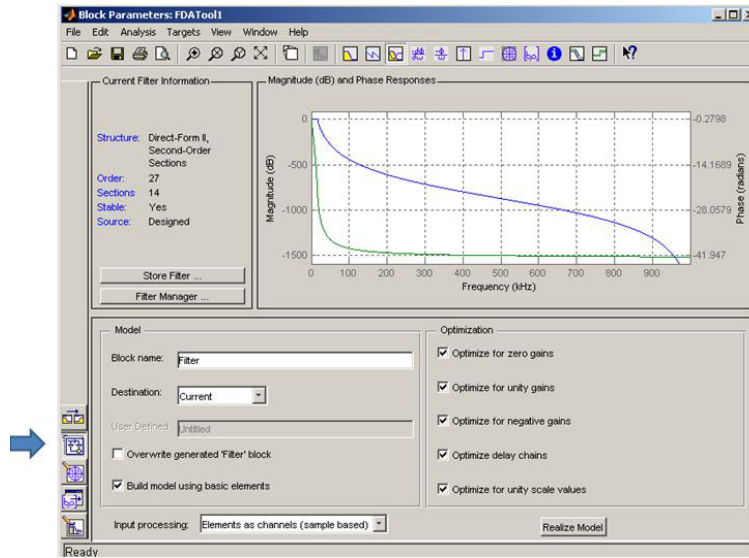


Figura 4.2 Exportación del filtro usando el FDATool.

11. Realice el modelo del filtro creado en Simulink con el blockset de Xilinx, una sección del filtro se muestra en la figura 4.3, el filtro está constituido por 14 secciones.

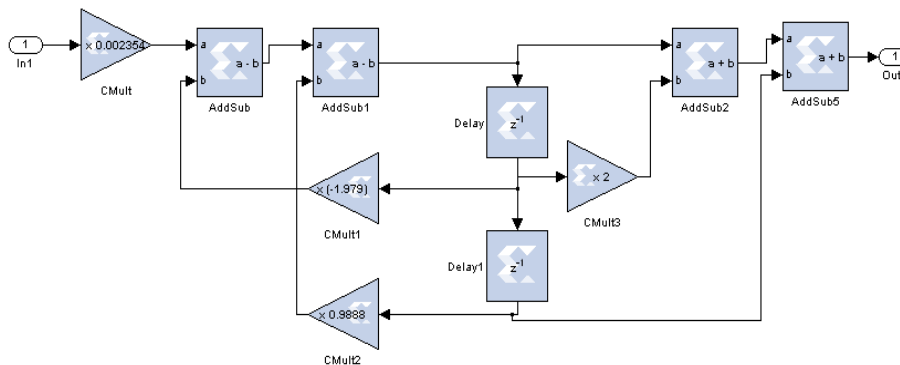


Figura 4.3 Primera sección del filtro con el blockset de Xilinx.

12. Setee los valores de los convertidores ADC/DAC en 1/1000000 e importe al modelo la música respectiva con el bloque "From Multimedia File" para observar el comportamiento del filtro.

13. Setee el clocking del bloque System Generator para que el periodo de la FPGA sea de 500ns y la simulación a 1/1000000 segundos, tal como se muestra en la figura 4.4

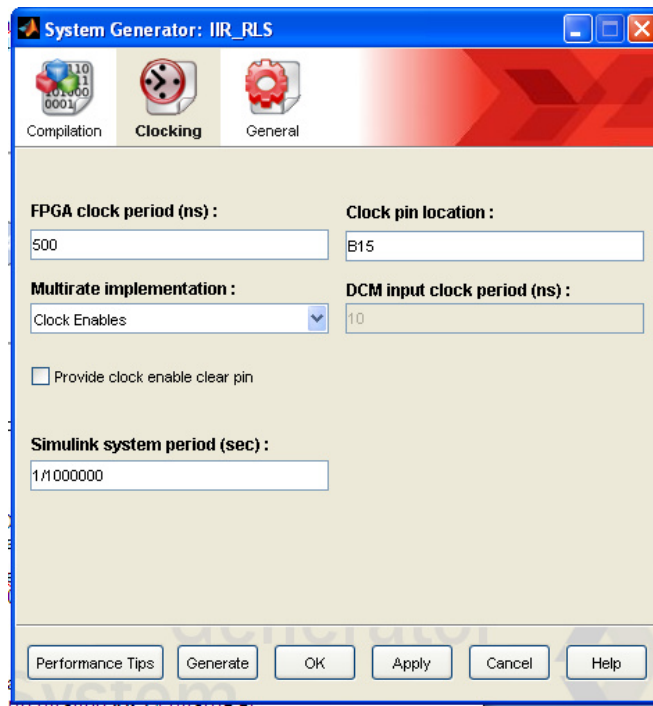


Figura 4.4 Configuración del System Generator.

B. Implementación del modelo del filtro IIR tipo butterworth

7. Conecte los cables que están en la tarjeta de desarrollo DSP XTREME DEVELOPMENT KIT a la FPGA, tal como se muestra en la figura 4.5, para poder ingresar el archivo .bit del modelo simulado a la tarjeta y observarlo en el osciloscopio.

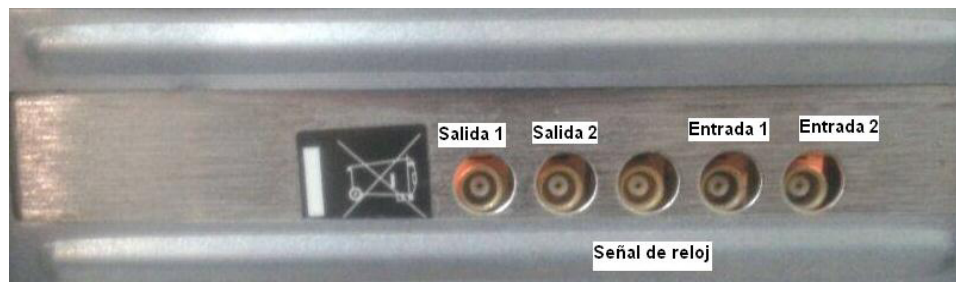


Figura 4.5 Configuración del System Generator.

- Abra el programa Fuse Probe y localice las tarjetas de la FPGA a través del menú **Open Card>>Locate Cards**, tal como se muestra en la figura 4.6

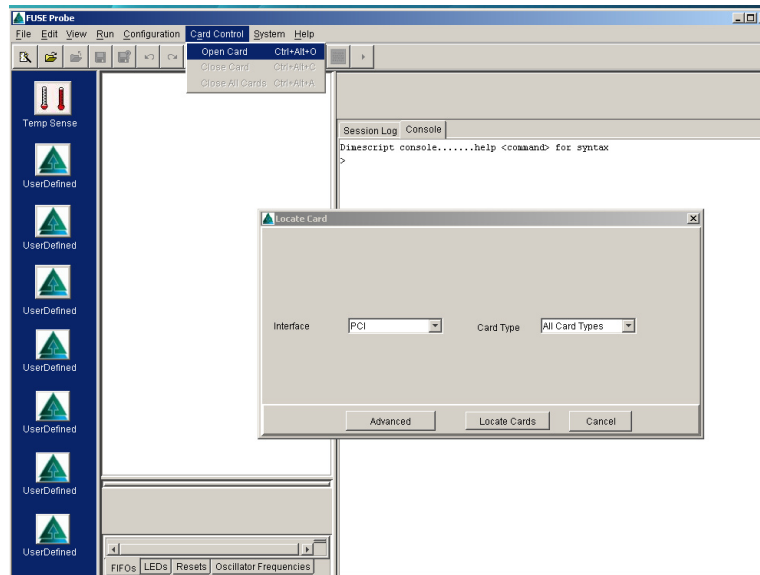


Figura 4.6 Localización de las tarjetas.

- Coloque en cada tarjeta el archivo .bit correspondiente, en uno de ellos se carga el archivo para el clock y en el restante el archivo del modelo simulado.
- Asigne los archivos haciendo click derecho en la tarjeta y seleccionando la opción Assign Bitfile y observe el osciloscopio.
- Capture la señal mostrada en el osciloscopio a cada minuto, a través del *OpenChoice Desktop*, con la opción *Get Data*, guárdela en una carpeta en el escritorio.
- Con los archivos .csv guardados, calcule el SNR de la señal capturada.

Experimento 2: Simulación e Implementación de un sistema usando el filtro digital FIR

A. Diseño del modelo del filtro FIR tipo LMS

- Abra el modelo ubicado en el escritorio.
- Observe el Spectrum Scope para analizar la respuesta de frecuencia del filtro

B. Implementación del modelo del filtro FIR tipo LMS

1. Conecte los cables que están en la tarjeta de desarrollo DSP XTREME DEVELOPMENT KIT a la FPGA, tal como se mostró en la figura 4.5, para poder ingresar el archivo .bit del modelo simulado a la tarjeta y observarlo en el osciloscopio.
2. Abra el programa Fuse Probe y localice las tarjetas de la FPGA a través del menú **Open Card>>Locate Cards**, tal como se mostró en la figura 4.6
3. Coloque en cada tarjeta el archivo .bit correspondiente, en uno de ellos se carga el archivo para el clock y en el restante el archivo del modelo simulado.
4. Asigne los archivos haciendo click derecho en la tarjeta y seleccionando la opción Assign Bitfile y observe el osciloscopio.
5. Capture la señal mostrada en el osciloscopio a cada minuto, a través del *OpenChoice Desktop*, con la opción *Get Data*, guárdela en una carpeta en el escritorio.
6. Con los archivos .csv guardados, calcule el SNR de la señal capturada.

ANEXO C

SIMULACIÓN DEL CANAL DE TRÁFICO EN CDMA E IMPLEMENTACIÓN DE MODULACIÓN Y DEMODULACIÓN 16 QAM

1. OBJETIVOS

- a) Estudiar los fundamentos del canal de tráfico en el enlace forward CDMA.
- b) Simular e implementar la modulación y demodulación 16 QAM.
- c) Realizar el análisis del SNR y BER en el sistema 16 QAM.
- d) Simular la configuración básica del canal de tráfico del enlace forward en CDMA.
- e) Realizar el análisis de los bits de CRC para el enlace forward CDMA.

2. FUNDAMENTOS TEÓRICOS.

2.1 CDMA

Es un esquema de acceso múltiple por división de código (*Code Division Multiple Access*). CDMA otorga a cada usuario toda la anchura de banda, o espectro ensanchado, hay cuatro variantes en función de la técnica utilizada para conseguir la expansión espectral:

- 1) Saltos de frecuencia, FH (Frequency Hopping) .La frecuencia varía en función del código.
- 2) Saltos de tiempo, TH (Time Hopping), donde se varía el intervalo temporal según el código.
- 3) Secuencia Directa, DS (Direct Sequence), en las que la señal de información se multiplica por el código de expansión.
- 4) Multiportadora, MC (Multicarrier), donde cada símbolo se información genera un conjunto de símbolos, según el código, que modulan distintas portadoras.

2.2 ARQUITECTURA DE UNA RED CDMA

La arquitectura consta del equipo de conmutación y del equipo de la estación base celular, estos interactúan con la red telefónica conmutada pública (PSTN) y con la estación móvil (MS) para proporcionar un sistema completo de comunicaciones celulares.

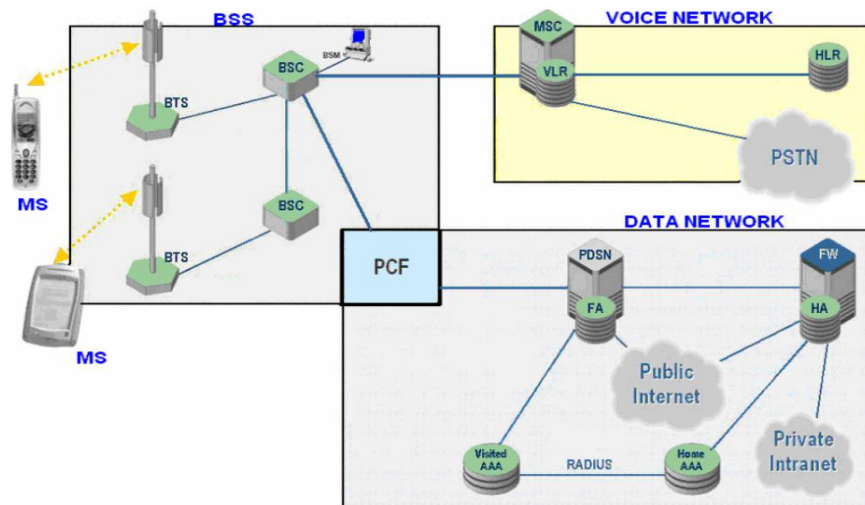


Figura 2.1 Arquitectura de una red CDMA

Los principales subsistemas son:

- 1) Estación Móvil (MS – Mobile Station).
- 2) La Central de Telefonía Móvil (MTX - Mobile Telephone Exchange) ó Centro de Conmutación Móvil (MSC – Mobile Switching Center).
- 3) El Controlador de Estación Base (BSC – Base Station Controller).
- 4) El Subsistema de Estación Base Transceptora (BTS – Base Station Transceiver Subsystem).
- 5) El Administrador de Estación Base (BSM – Base Station Manager).
- 6) HLR (Home Location Register).- Base de datos con el registro de los suscriptores y sus respectivos perfiles de servicios

Para realizar y recibir llamadas de voz y datos, la MS se debe registrar con el HLR.

La estación móvil en una llamada de datos funciona como un cliente móvil IP interactuando con la red de acceso para obtener un apropiado recurso de radio para el intercambio de paquetes.

El BSC controla el enrutamiento de mensajes y de señalización entre éste mismo, la MTX, el BSM y la BTS. También proporciona la codificación y decodificación de voz entre la estación móvil (a través de la BTS) y la MTX.

Para poder realizar la llamada el móvil se comunica con la antena (enlace reverse) y la antena se comunica con el móvil (enlace forward) en bandas específicas:

- Forward Channel Frequency → 869 – 894 Mhz
- Reverse Channel Frequency → 824 – 849 Mhz
- Tx/Rx Frequency Spacing → 45 Mhz

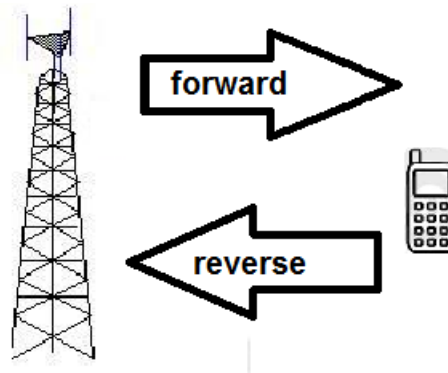


Figura 2.2 Enlaces de una red CDMA

Cada enlace posee diferentes tipos de canales que hace posible la efectuación de una llamada.

- Enlace Forward
 - Piloto
 - Paginado
 - Sincronización
 - Tráfico
- Enlace Reverse
 - Acceso
 - Tráfico

El canal piloto sirve para controlar la potencia del móvil, debido a las atenuaciones de la señal la estación base podría requerir aumento o disminución de la potencia del móvil.

El canal de sincronización proporciona al móvil la temporización desde la base, el sistema y la velocidad del canal de paginado, transporta información necesaria para decodificar el canal de Paginado como los códigos a utilizarse, la identidad del sistema y de la red.

El canal de paginado es utilizado para localizar y registrar a un usuario y procesar una llamada entrante. El mensaje es utilizado para registro del sistema; para informar de una llamada entrante; para informar de un mensaje en el voice-mail; como servicio de mensajes.

Esta práctica se enfoca en el canal de tráfico debido a que en este se maneja la señal de voz que pasa por la estación base para llegar al móvil.

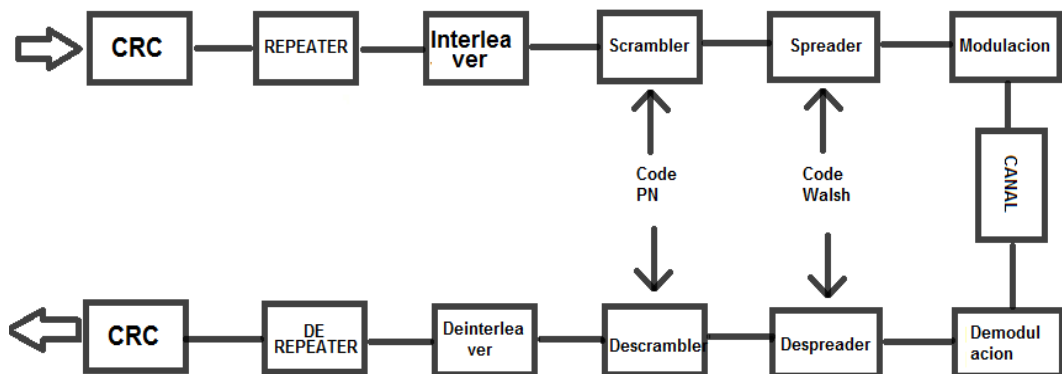


Figura 2.3 Proceso del canal de tráfico del enlace forward en CDMA

CRC: La señal de voz se codifica y se digitaliza antes de llegar al bloque CRC. El bloque de chequeo de redundancia cíclica (cyclic redundancy check) toma un conjunto de 40 bits y realiza una división para 16, el valor que se obtiene en el residuo se envían serialmente después los 40 bits indicados formando así una trama de 48 bits, se repite el proceso al final del sistema para los mismos 40 bits para poner a comparar el residuo actual con el anterior y así poder determinar si en la trama hubo algún error.

REPEATER: Repetir los bits permite corregir errores y corregirlos, el número de veces que repite cada bit depende de la velocidad inicial del bloque, según:

CANAL	TRÁFICO				
Velocidad de datos inicial	1200	2400	4800	9600	bps
Factor de repetición	8	4	2	1	

Tabla 1. Repetición del código

INTERLEVER: El entrelazador se encarga de alterar el orden de una secuencia de bits de entrada. El propósito de usar el interleaver es el de aleatorizar la posición en que se localizan los errores en la transmisión de una señal y así aumentar la eficiencia del FEC dispersando los errores de ráfaga introducidos en el canal de comunicaciones.

Existen varias implementaciones del interleaver entre las que se destacan el interleaver convolucional de Forney y el interleaver por bloque, del cual nos vamos a enfocar.

El interleaver de bloque funciona guardando un bloque de datos de entrada dado en una matriz $m \times n$. La escritura se realiza fila por fila, y una vez guardado todos los datos del bloque se realiza la lectura pero esta vez columna por columna.

SCRAMBLER: Encripta la señal con un código pseudoaleatorio dándole seguridad a la señal. El código se genera mediante 42 registros de desplazamiento lineal retroalimentados (LFSR), el LFSR es un registro de desplazamiento que de manera sincrónica hace avanzar la señal a través de los registros comenzando con el bit más significativo. Algunas salidas son combinadas usando una puerta XOR en forma de retroalimentación.

El código largo PN permite la ortogonalización del usuario. Cada uno lleva una máscara distinta.

SPREADER: Aquí se efectúa el esparcimiento de la señal mediante secuencias ortogonales, códigos Walsh, el cual es único para cada usuario y así evitar las interferencias entre usuarios.

Un conjunto de códigos Walsh de longitud n consiste en una matriz n por n donde cada fila o columna de la matriz es un código Walsh. La matriz para generar códigos se define de manera recursiva:

$$W_1 = (0) \quad W_{2n} = \begin{Bmatrix} W_n & W_n \\ W_n & \overline{W_n} \end{Bmatrix}$$

Donde n es una potencia de 2 así un ejemplo de la matriz.

$$W_2 = \begin{pmatrix} 0 & 0 \\ 0 & 1 \end{pmatrix} \quad W_4 = \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 \\ 0 & 1 & 1 & 0 \end{pmatrix}$$

En la figura 2.4 se muestra un ejemplo de esparcimiento de la señal donde se observa que la señal del código Walsh posee una mayor frecuencia que el dato, de la misma forma se realiza el encriptado en el proceso anterior, scrambler, con la diferencia que esta señal posee la misma frecuencia en que el dato.

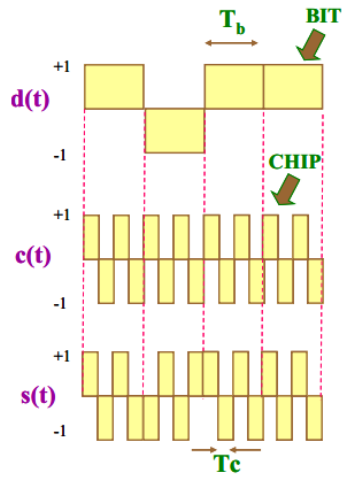


Figura 2.4 Ejemplo de espaciamiento con chip de código Walsh

MODULACIÓN: En CDMA se utiliza modulación adaptativa a las condiciones de transmisión, quiere decir que si se opera con nivel de señal baja y altas interferencias se usa un tipo de modulación más robusto, entre los tipos de modulación que puede usar están, BPSK, QPSK, 8PSK Y 16 QAM. En esta práctica se desarrolla la modulación 16 QAM

El mensaje de la técnica QAM no esta conenido unicamente en la variacion de fase, tambien contiene variaciones en la amplitud, por lo tanto su naturaleza es multinivel siendo 4 su menor numero de niveles y 256 el mayor, en teoria puede realizarse mas niveles pero aumentaria su probabilidad de error

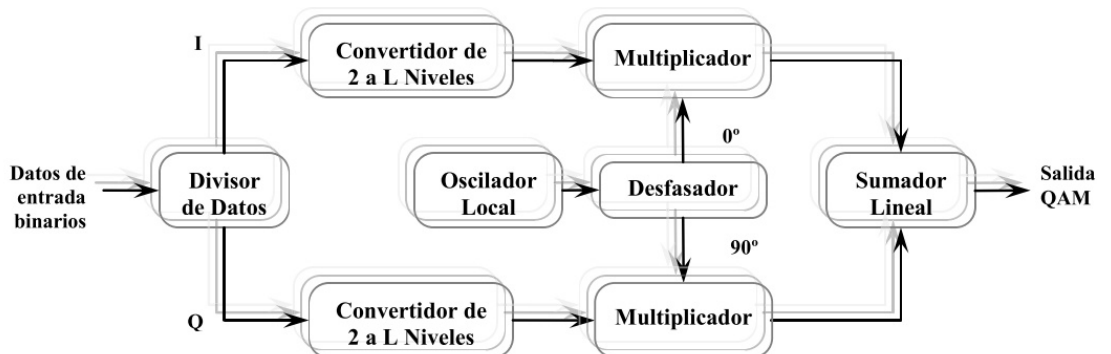


Figura 2.5 Esquema de modulación 16QAM

El SNR, relación señal ruido, y el BER, error por bit, son parámetros para medir la calidad de la señal que ha sido transmitida; siendo sus fórmulas:

$$SNR = \frac{S}{N} = \frac{\text{Potencia de la señal}}{\text{Potencia del ruido}} = \frac{(\text{Amplitud de la señal})^2}{(\text{Desviación típica del ruido})^2}$$

$$BER = \frac{\text{número de bits erróneos}}{\text{número de bits transmitidos}}$$

$$BER = \frac{1}{2} \left[1 - \operatorname{erf} \left(\frac{\sqrt{SNR}}{2\sqrt{2}} \right) \right]$$

ERF es conocido como la función de error gaussiano definido por la fórmula:

$$\operatorname{erf}(x) = \frac{1}{\sqrt{\pi}} \int_0^x e^{-x^2} dx$$

Para poder hallar el valor de SNR y por consiguiente del BER se necesita especificar ciertas definiciones.

Se debe analizar el número de bits por símbolo necesarios para derivar hacia cada canal, en fase(I) y en cuadratura(Q), determinado por:

$$N = \log_2 n,$$

Donde n determina el número de modulación en nuestro caso 16. Dado el número de bits por símbolo, el ancho de banda para los canales I, Q se establece:

$$AB_{IQ} = \frac{f_b}{N}$$

Siendo f_b la frecuencia de bit la misma frecuencia del dato de entrada. Otro parámetro importante es la energía promedio por símbolo dada por la fórmula:

$$E_{simbolo} = \sum_{m=1}^M p_m E_m$$

Donde p_m es la probabilidad de que ocurra un símbolo, sin embargo, se considera que los símbolos son equiprobables, siendo M el número de símbolos.

$$E_{\text{simbolo}} = \frac{1}{M} \sum_{m=1}^M E_m$$

Siendo E_m la energía de un punto en la constelación, se lo calcula:

$$E_m = \|\vec{S}_m\|^2 = \sum_{m=1}^M c_{m,i}^2$$

Tal como se muestra en la figura 2.6, C_m es la distancia del origen hasta la coordenada x ó del punto a tratar. A partir de la energía del símbolo se obtiene la energía del bit.

$$E_m = \frac{E_b}{\log_2 M}$$

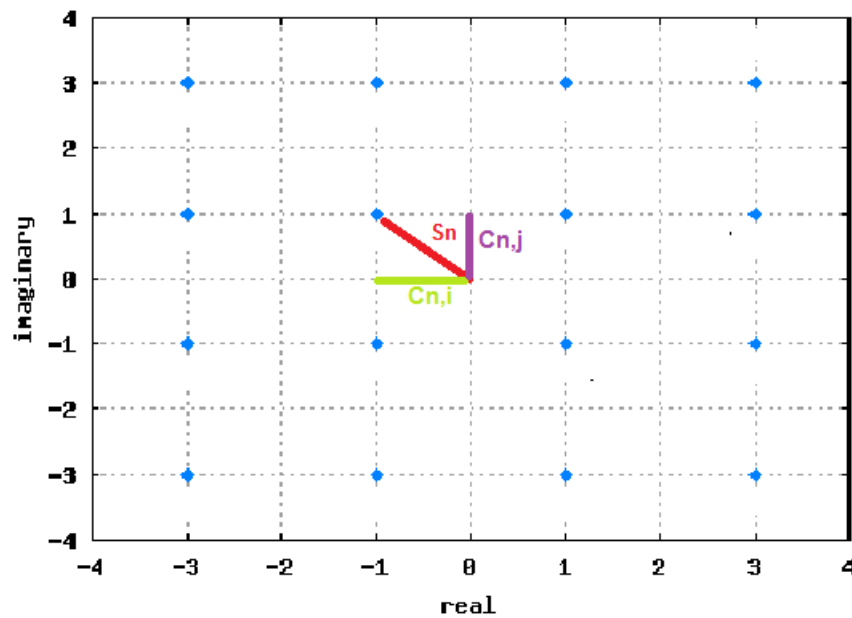


Figura 2.6 Constelación de modulación 16QAM. Vector S_n del símbolo n .

$$E_b = \frac{E_{\text{simbolo}}}{\log_2 M}$$

$$P = E_b R_b$$

Teniendo la energía del bit se puede calcular la potencia de la señal para hallar la relación señal ruido (SNR) y la tasa de error por bit (BER).

DEMODULACIÓN: es el proceso a través de cual se recupera la señal modulada para esto se deben seguir las etapas que se muestran en la figura 2.7.

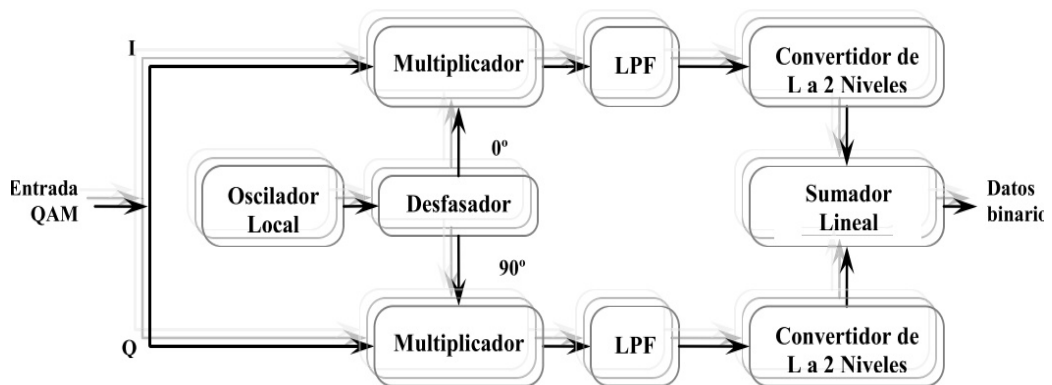


Figura 2.7 Esquema de demodulación 16QAM

Como se nota en la figura 6 es el proceso contrario al de la modulación pero añadiendo un filtro pasa bajo para eliminar las altas frecuencias que se adquiere al multiplicar la señal modulada con la portadora.

Como se demuestra en la figura existe un oscilador local idéntico para la modulación y demodulación, el cual tiene una frecuencia invariable determinada por la frecuencia (f_a) que se programa en la FPGA.

La frecuencia del oscilador f_c consta de x ciclos del reloj programada en la FPGA para poder definir la señal senoidal, mientras mayor ciclos de reloj se disponga mejor será la precisión de la senoidal.

$$f_c = x * f_a$$

Al multiplicar por la portadora se producen frecuencias altas por lo que se añade un filtro paso bajo, se presentan las siguientes fórmulas para poder configurar correctamente el filtro.

$$f_s < 2f_c - AB_{IQ}$$

$$f_p = \frac{AB_{IQ}}{2}$$

$$f_m = \frac{f_a}{2}$$

Donde f_p es la frecuencia de paso, f_s la frecuencia de parada y la f_m frecuencia de muestreo.

Los demás bloques realizan el proceso contrario, es decir, desencapsula el dato de la trama en el enlace forward. Entonces en el despreader lo que se hace es combinar por medio de un xor la señal por el mismo código walsh, en la figura 2.8 se muestra un ejemplo.

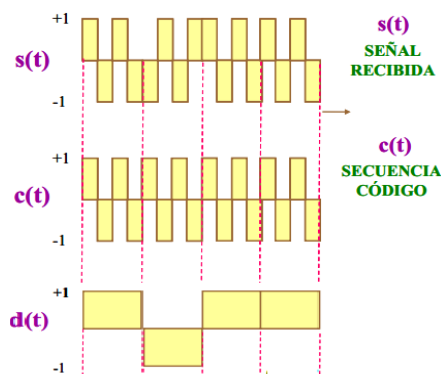


Figura 2.8 Ejemplo de despreading con chip de código walsh

De la misma manera el código PN se combina en el descrambler con la señal para decodificarla pasando al bloque deinterleaver que realiza el mismo proceso de forma contraria al interleaver, ésto es, ubicando los bits a manera de columnas para después sacarlos a manera de filas, luego se quitan los bits repetidos reduciendo la frecuencia de la señal; por último se obtiene el CRC y se compara el residuo inicial y actual para determinar si la trama de llegada tiene error.

3. DESCRIPCIÓN DEL MODELO

Se expone el funcionamiento de los bloques que conforman el modelo del sistema enlace forward en CDMA. En la figura 3.1 se muestra el sistema.

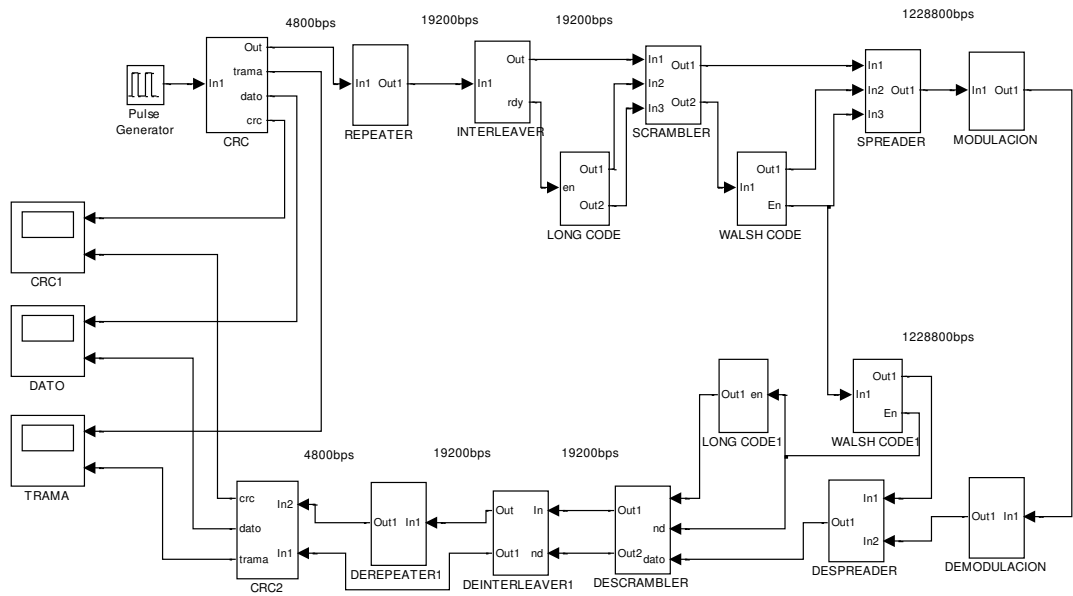


Figura 3.1 Modelo del canal de tráfico del enlace forward en CDMA.

El funcionamiento de los subsistemas que conforma el enlace forward se especifica a continuación.

1. Chequeo de redundancia cíclica

En la figura 3.2 se muestra el modelo que modifica el dato de entrada agrupándolo en 40 bits, estos ingresan en el puerto *dividendo* del bloque *Divider Generator* en donde es dividido para 16, el recíproco conformado por 8 bits es añadido a los bits del dato de entrada formando así la trama.

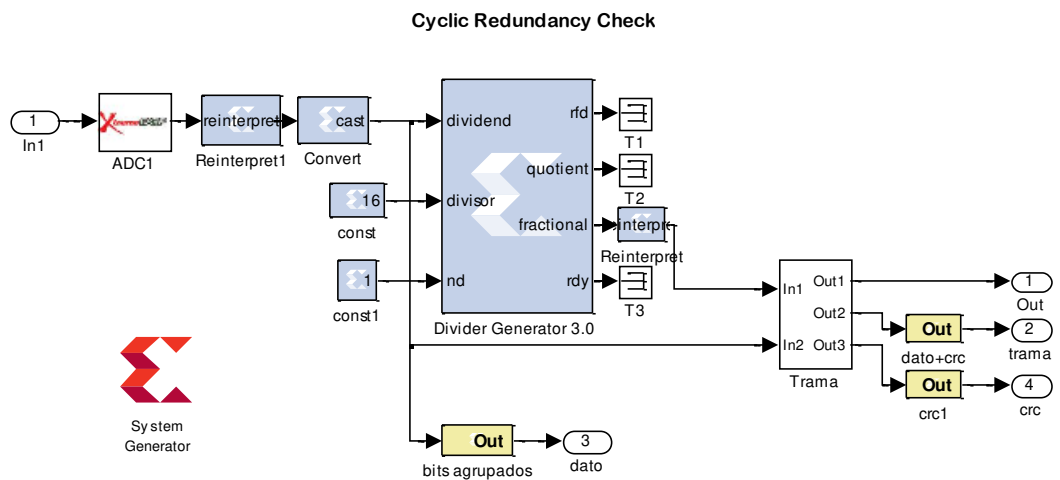


Figura 3.2 Formación de la trama con los bits de entrada y CRC.

Al finalizar el recorrido por el enlace forward se realiza un chequeo a los últimos bits de la trama, si éstos son iguales significa que la trama esta correcta, sino la trama presenta algún error en sus bits. Este proceso lo realiza el modelo de la figura 3.3.

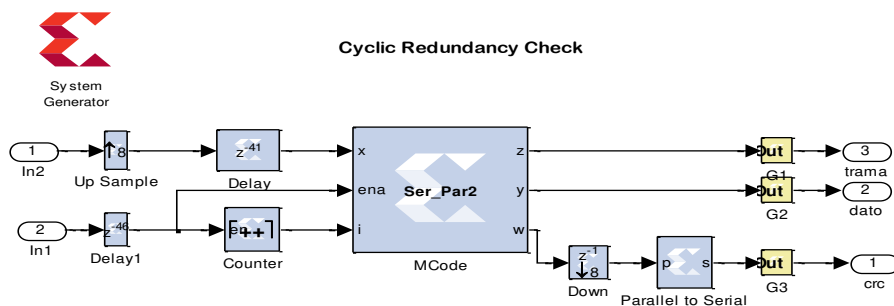


Figura 3.3 Chequeo de redundancia cíclica.

Este modelo presenta a su salida la trama, el dato formado por 40 bits y los 8 bits que sirven para el chequeo.

Los 40 bits se vuelven a dividir para 16 y su cociente se compara con los 8 bits de chequeo. En el workspace de Matlab se presenta un mensaje si los bits de chequeo fueron o no iguales.

2. Repetidora y De-repetidora

Se toma cada bit y se repite, 4 veces por las especificaciones de la tabla 1, teniendo en dato con los 4 bits paralelos en la salida del bloque repetidor de la figura 3.4, con el siguiente bloque se envía los bits serialmente.

Se tiene en cuenta que al repetir bits se requiere una frecuencia mayor para procesar la información así que la señal de salida es n veces más rápida según el número de bits que se repitan.

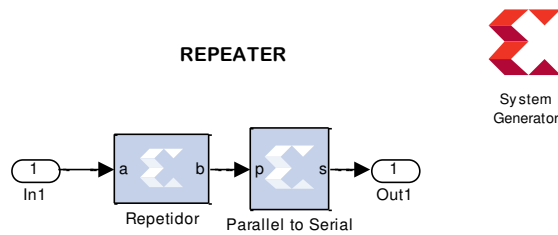


Figura 3.4 Repetidor de bits.

El bloque de la figura 3.5 toma un bit de 4 que ingresan de manera consecutiva reduciendo así el número de bits repetidos.

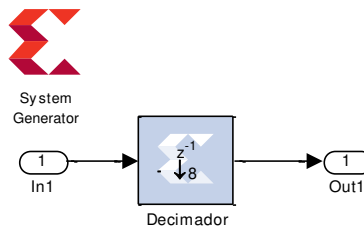


Figura 3.5 De-Repetidor de bits.

Al eliminar los bits repetidos la señal de salida reduce su velocidad ya que el bit que queda persistente el mismo tiempo que los cuatro bits de entrada.

3. Interleaver y De-interleaver

Se trabaja con un bloque especial de xilinx para poder desarrollar el intercalador y el de-intercalador. En el intercalador de la figura 3.6 se observa un subsistema llamado fifo, este es necesario debido a que el interleaver tarda mas en procesar la información anterior que el tiempo que demora en llegar la nueva trama por lo que guarda la información en un sistema fifo (first in, first out).

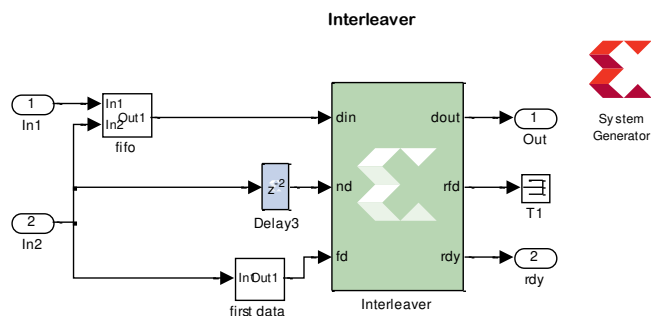


Figura 3.6 Intercalador de bits.

Aunque estos bloques, interleaver y de-interleaver, realizan toda la operación de intercalado y de-intercalador le son necesarias señales de control como la habilitadora en el puerto *nd* y la otra señal que notifique el primer dato de la trama, esta ultima funcion de control es del subsistema *first data*.

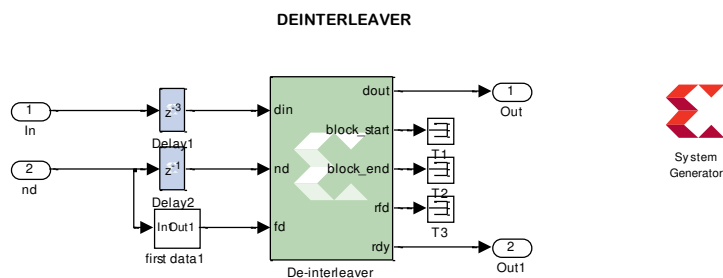


Figura 3.7 De-Intercalador de bits

Estos bloques entregan señales de control como la señal en el puerto *rdy* que me indica cuando hay un dato valido en el puerto *dout*.

4. Códigos

Para encriptar los datos y esparcir su potencia se necesitan de códigos, códigos largos PN y códigos Walsh.

En la figura 3.8 se muestra el bloque generador del código que le agrega encriptacion al dato, se conoce a este como pseudo ruido.

Este bloque trabaja con una señal de control habilitadora en el puerto *en*.

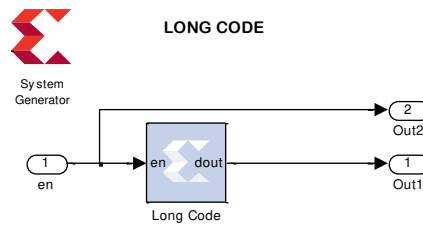


Figura 3.8 Generador de código largo

Los códigos walsh son utilizados para esparcir la señal ya que ayuda a la señal a esparcir su potencia en un mayor espectro. Tienen una frecuencia mucho mayor al dato en este caso trabajan a 1.2288MHz.

Este bloque trabaja con un archivo de programación “*codigo.m*” en donde se guarda el código para acceder a el mediante un vector. El contador permite acceder al vector del código dentro del bloque Mcode.

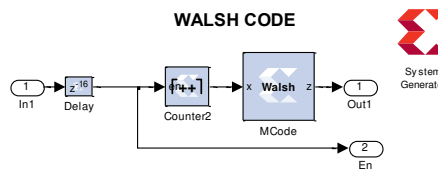


Figura 3.9 Generador de código Walsh

Estos códigos se mezclan con la señal por medio de un bloque exor, que por simplicidad no se muestra una figura, para poder realizar la encriptación y el esparcimiento de la señal.

5. Modulación 16QAM

En la figura 3.10 se muestra el diseño que se empleará tanto para la simulación e implementación del mismo.

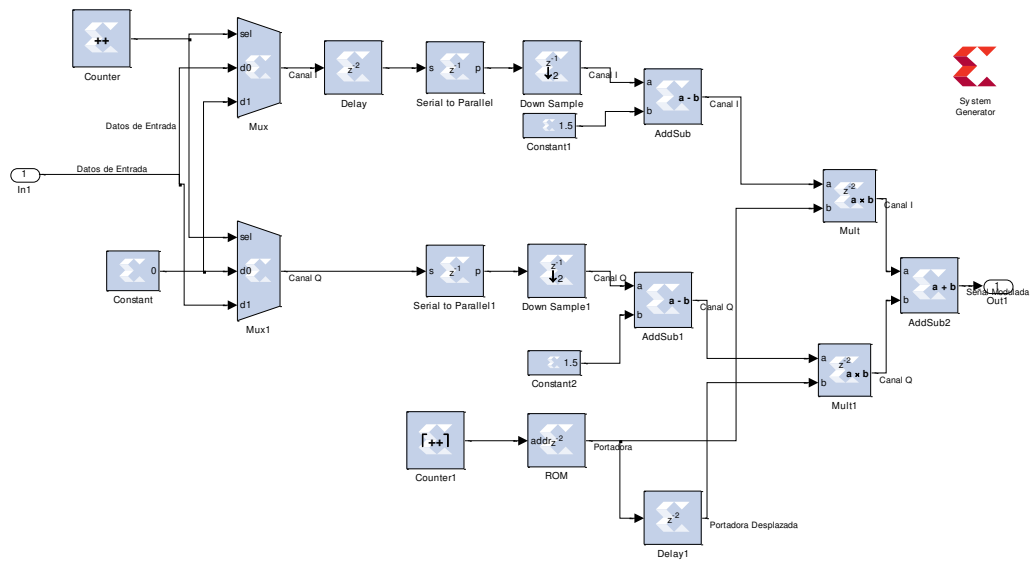


Figura 3.10 Diseño de la modulación 16QAM

La señal de entrada es una señal binaria aleatoria, los bloques ADC y DAC permiten la conversión analógico/digital o digital/analógico de la señal entrante y la modulada y de los canales en fase y cuadratura para observar posteriormente la constelación.

Los datos de entrada se separan cada dos bits en los canales de fase y cuadratura para efectuar los multiples niveles característicos de 16QAM, es aquí donde se puede apreciar la constelación y verificar su correcto funcionamiento.

Luego los multiniveles son multiplicados por la portadora que posee una frecuencia mayor a la señal de entrada.

El bloque "contador1" y el bloque "ROM" sirven para definir el oscilador local, la cual se desplaza con un delay para enviar la portadora en cuadratura. El contador tiene una cuenta limitada hasta el valor x-1 a la misma frecuencia del "system generator", en el bloque "ROM" se define la señal senoidal definida en x ciclos del reloj (Depth) como se muestra en la figura 3.11.

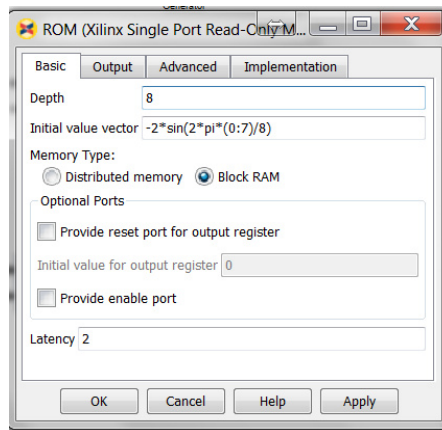


Figura 3.11 Configuración del bloque ROM

El bloque System Generator es necesario tanto para la simulación como para la implementación, ya que aquí se define el clock con el que se realizará la simulación y también genera un archivo .bit necesario para la implementación.

El bloque Resource Estimator sirve para optimizar los recursos de la tarjeta cuando se realice la implementación del mismo.

6. Demodulación 16QAM

En la figura 3.12 se muestra el diseño que se empleará tanto para la simulación e implementación del mismo.

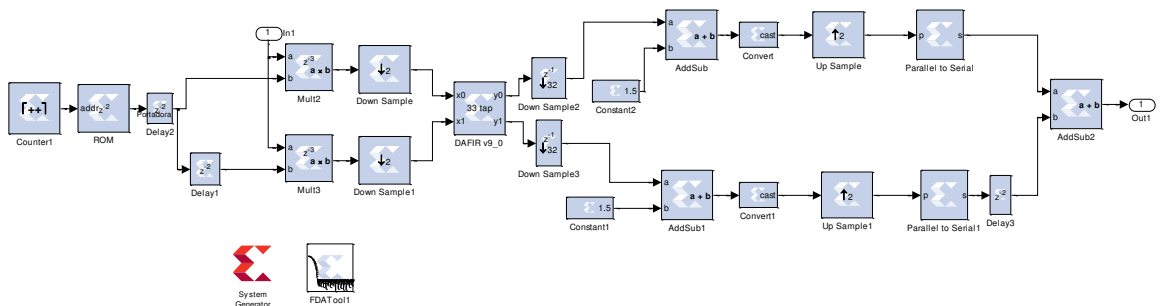


Figura 3.12 Diseño de la demodulación 16QAM

La señal de entrada es la señal modulada a la que se le añade ruido blanco gaussiano para simular una señal interfiriente y así analizar el SNR y el BER, los bloques ADC y DAC permiten la conversión analógico/digital o digital/analógico de la señal entrante modulada y la salida demodulada, el ancho de banda que ocupa la señal se la puede analizar a través del Spectrum Scope, también se puede visualizar y comparar la señal modulada con la demodulada a través del Scope.

El oscilador local y el convertidor de niveles son similares tanto en el modulador como el demodulador, mientras el filtro paso bajo se define en el bloque DAFIR y su estructura especificada en el FDATool.

El bloque System Generator es necesario tanto para la simulación como para la implementación, ya que aquí se define el clock con el que se realizará la simulación y también genera un archivo .bit necesario para la implementación.

El bloque Resource Estimator sirve para optimizar los recursos de la tarjeta cuando se realice la implementación del mismo.

7. Canal Ruido Gaussiano

Para el desarrollo de la práctica se realizará pruebas en la modulación y demodulación además del sistema del enlace forward por lo que es necesario agregarle ruido al sistema, por medio del canal ruido gaussiano figura 3.13, para comprobar como afecta a la señal.

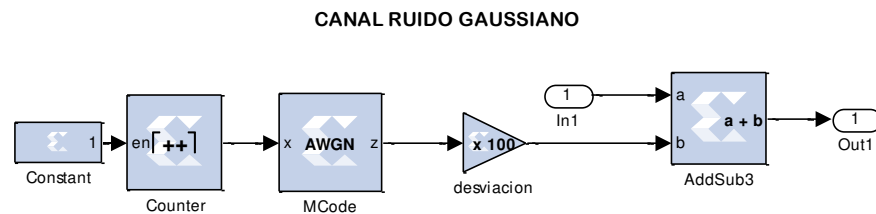


Figura 3.13. Diseño de la demodulación 16QAM

Cuando el ruido gaussiano posee media cero podemos cambiar su potencia cambiando su varianza, debido a esto se encuentra el bloque desviación que multiplica la señal proveniente del bloque MCode, que es el ruido con desviación estándar de valor 1, por un factor de ganancia para cambiar su potencia.

4. EQUIPOS REQUERIDOS

- Cables
- Generador de funciones
- Osciloscopio

5. EXPERIMENTOS

Las tablas que se indican a continuación se encuentran en el archivo CDMA.xls

Experimento 1: Simulación e Implementación de la modulación y demodulación 16QAM.

A. Diseño del modelo de la simulación

1. Abra en simulink el modelo "Mod_16QAM.mdl "
2. Setee la frecuencia de la señal de entrada a 100KHz. Determine la frecuencia de la portadora mediante la fórmula descrita en la teoría.
3. Corra la simulación y observe el espectro de la señal modulada mediante el bloque "Spectrum Scope", compruebe que la portadora está correcta. Guarde la imagen mostrada.
4. Observe el gráfico obtenido por el bloque "Discrete time scatter plot", guarde la imagen y determine la energía por bit mediante los símbolos presentes en la constelación. Halle la potencia de la señal.
5. Observe el gráfico obtenido por el bloque scope en la salida del modulador. Guarde la imagen y justifique la forma de onda que se presenta.
6. Abra en simulink el modelo "Mod_Demo_16QAM"
7. Ubique un bloque "spectrum scope" antes y después del filtro, corra la simulación; analice los espectros de frecuencia obtenidos y guarde las gráficas.
8. Compruebe que la salida del demodulador se obtiene la misma señal a la entrada del modulador. Obtenga gráficas mediante el scope.
9. Abra en simulink el modelo "Canal AWGN" copie el modelo y ubique la entrada de este en la salida del modulador, y la salida del modelo en la entrada del demodulador.
10. Setee a 1 el bloque "des_est" (desviación estándar) ubicado dentro del "Canal AWGN". Corra la simulación.
11. Cuando termine la simulación se crea una variable que es la amplitud del ruido, halle su desviación estándar ($\text{std}(x)$) y determine los valores de SNR y BER.
12. Repita los pasos 10 y 11 cambiando a 10, 50 y 100 el valor del bloque "des_est". Realice el análisis de los resultados obtenidos.

B. Implementación de la modulación/demodulación 16QAM

1. Conecte el canal ADC de la tarjeta a la entrada del generador de ondas con el cable
2. Conecte los canales DAC de la tarjeta a los canales del osciloscopio con el cable

3. Mediante el programa FUSE configure las FPGA con los archivos "16QAM_constelacion.bit" y "osc_clock_2v80.bit"
4. Seleccione una señal cuadrada de 100KHz en el generador de ondas. Configure el osciloscopio a el formato XY(t) para observar la constelación.
5. Observe la distancia entre símbolos, halle la energía por bit y la potencia de la señal, guarde la gráfica de la constelación.
6. La distancia entre símbolos está dada por un factor de 4 debido a cuantización en el convertido digital a analógico, multiplicar este factor por la distancia observada. La potencia de la señal obtenida servirá en el paso 7.
7. Ahora se realizara la prueba del SNR y BER cambiando la potencia del ruido en el canal. Mediante el programa FUSE configure las FPGA con los archivos "Canal_1.bit" y "osc_clock_2v80.bit". El archivo configurado corresponde al canal con ruido cuando la desviación estándar es de 1.
8. La señal que se observa en el osciloscopio corresponde al ruido capturar 4 muestras y procesarlas en Excel para obtener la desviación estándar del ruido, calcular además el SNR y BER.
9. Configure la FPGA con el archivo "Canal_10.bit" " y "osc_clock_2v80.bit". El archivo configurado corresponde al canal con ruido cuando la desviación estándar es de 10.
10. Repita el paso 7 para obtener los resultados.
11. Configure la FPGA con el archivo "Canal_50.bit" " y "osc_clock_2v80.bit". El archivo configurado corresponde al canal con ruido cuando la desviación estándar es de 50. Repita además el paso 7.
12. Configure nuevamente la FPGA con el archivo "Canal_100.bit" " y "osc_clock_2v80.bit". El archivo configurado corresponde al canal con ruido cuando la desviación estándar es de 100. Repita el paso 7.
13. Obtenido todos los resultados realice una tabla entre los snr y ver simulados e implementados en la tarjeta. Expresé sus conclusiones.

Experimento 2: Simulación del canal delantero (forward) del sistema CDMA

1. Abra el archivo sistema CDMA en Simulink.
2. Setee la onda de entrada al sistema en 100KHZ y corra la simulación.
3. Observe el funcionamiento de cada una de los bloques del sistema.
4. Guarde los resultados del bloque CRC y especifique de cuántos bits está conformado la trama de salida.
5. Especifique cuántas veces se repite un bit en el bloque repeater.

6. Obtenga gráficas de los datos de entrada y salida al bloque interleaver, confirme que función cumple.
7. Obtenga la gráfica a la salida del bloque scrambler, estipule el funcionamiento de este bloque.
8. Determine cómo se produce el esparcimiento de la señal en el bloque spreader y obtenga gráficas.
9. Compruebe que el proceso contrario se da en el bloque desreader, descrambler, deinterleaver mediante observación de las gráficas.
10. En la trama aún no se ha agregado ruido, por lo tanto, cuál debe ser el resultado en bloque CRC de regreso, testifique su conclusión mediante gráfica. Además en el Workspace de Matlab se indica si la trama ha llegado correctamente a su destino mediante la verificación de los bits del CRC.
11. Ubique el canal AWGN entre el modulador y demodulador para agregar ruido al sistema. Corra la simulación.
12. Habiendo ruido en el canal, compruebe si los bits CRC de salida y llegada de la trama concuerdan, certifique su conclusión con las gráficas y las indicaciones que se presentan en el Workspace.

ANEXO D

DISEÑO DE FILTROS EN FDATOOL

FDATool es una herramienta muy práctica y eficiente para poder diseñar filtros conforme a nuestros requerimientos. Lo podemos encontrar en la librería de Simulink/Matlab como la figura 1

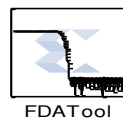


Figura 1. Bloque FDATool

El FDATool presenta el diseño de un filtro por default pero tenemos varias opciones para cambiar el diseño del filtro tales como tipo de respuesta, orden del filtro y especificaciones de las frecuencias como frecuencia de muestreo (f_s), frecuencia de paso (f_{pass}) y frecuencia de parada (f_{stop}).

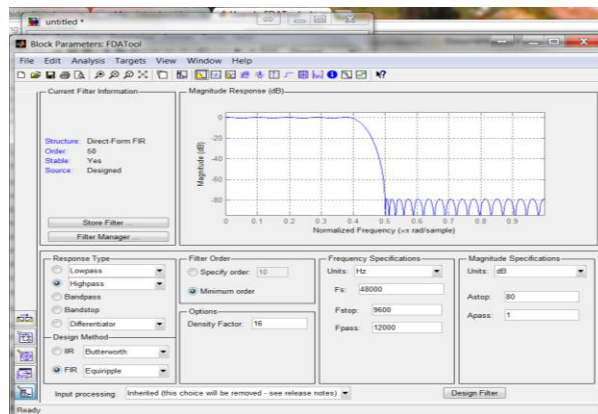


Figura 2. Ventana de configuración FDATool

Se puede elegir entre varios tipos de repuesta.

- Paso bajo
- Highpass
- Bandpass
- Parabanda
- Diferenciador

Se puede además seleccionar un método de diseño, se refiere a la estructura del filtro.

- FIR
- IIR

Otras opciones que podemos variar es el orden del filtro el cual podemos escoger entre el mínimo necesario para una correcta respuesta o uno especificado manualmente que puede ser inferior al mínimo requerido.

Existen opciones especiales que aparecen dependiendo de la respuesta del filtro que elijamos tales como atenuación, valores alfa y beta y factor de densidad.

Las opciones más importantes que se deben setear son las especificaciones de frecuencia (*Frequency Specification*), estas varían dependiendo del filtro y de los requerimientos que necesitemos.

Para un filtro pasa alto o pasa bajo se requieren tres frecuencias, frecuencia de muestreo, frecuencia de paso y frecuencia de parada, mientras que para un filtro pasa banda o banda de parada se requieren 5 especificaciones del filtro: frecuencia de muestreo, frecuencia de paso 1 y 2, frecuencia de parada 1 y 2.

Se especifica también la respuesta de magnitud (*Magnitude Specification*), pudiendo ser en dB o lineal en la cual elegimos la atenuación de la banda de supresión y la ondulación de la banda de paso.

Después de haber seteado los valores necesarios para el diseño del filtro se calcula los coeficientes del mismo haciendo click en Design Filter. Luego de esto se muestra la gráfica de la respuesta de magnitud aunque también se disponen otras graficas como la respuesta de fase, respuesta impulso y la gráfica de ceros y polos.

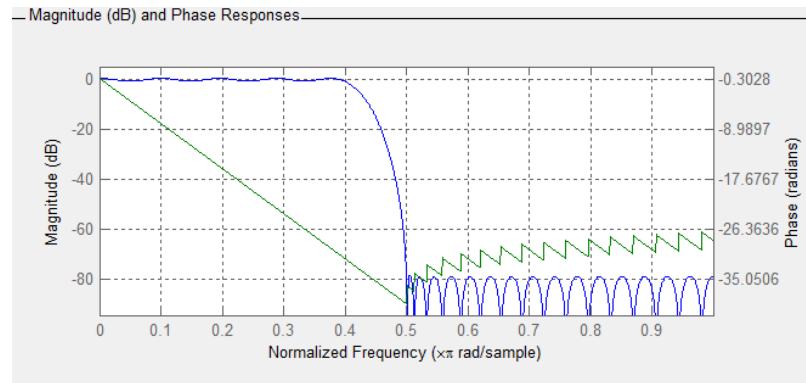


Figura 3. Respuesta en Magnitud y Fase

Se nota en la figura 3 que se tiene una respuesta en magnitud normalizada y que la respuesta en fase es lineal.

Después de realizado el diseño del filtro se puede exportar su estructura con sus coeficientes, para la exportación a Simulink se lo realiza de la siguiente manera:

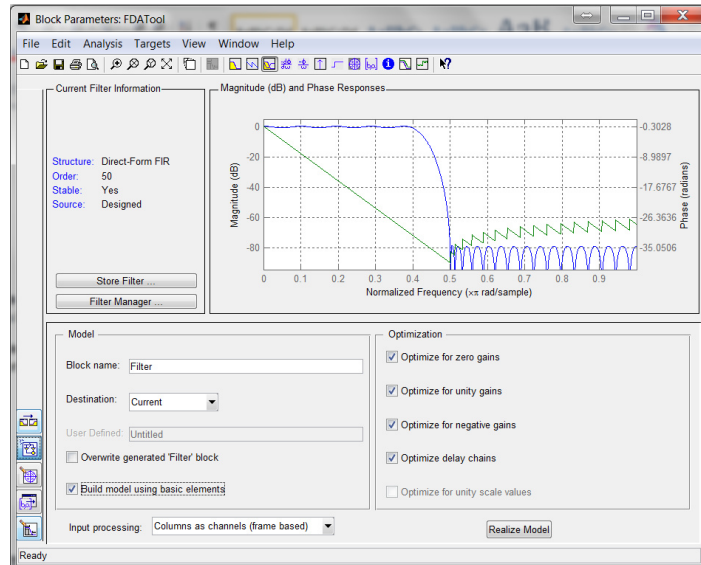


Figura 4. Exportación del modelo a Simulink

Se puede elegir entre varias optimizaciones que nos ofrecer esta herramienta, al momento de realizar el modelo del filtro se puede elegir si crearlo mediante elementos básicos como bloques de suma, multiplicación, delays, esto es útil si se requiere construir un filtro con otros bloques no propios de Simulink tales como los de xilinx.

Hacer click en *Realize Model* y obtenemos un bloques con el modelo del filtro requerido en simulink.

Disponemos de otra ventana en la cual se puede exportar los coeficientes del filtro.

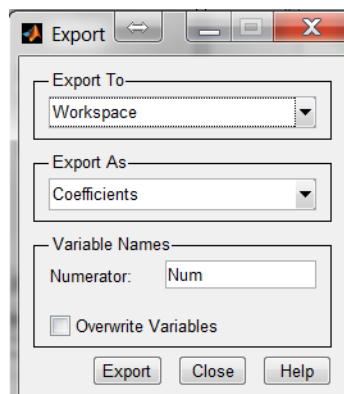


Figura 5. Exportación del modelo a workspace de Matlab

Esta ventana nos muestra opciones al momento de exportar dándonos la facilidad de elegir el formato con que requerimos los coeficientes y hacia donde queremos enviarlo.

Podemos elegir exportar los coeficientes como vector coeficiente o como objeto, además podemos exportarlo hacia el workspace de Matlab o como un archivo.

ANEXO E

CÓDIGO PARA HALLAR EL SNR A TRAVÉS DE MATLAB

Para poder analizar el SNR de los filtros se ha utilizado el siguiente código:

```
>>P_ruido=(sum(ruido.^2))/length(ruido);
```

```
>>P_musica_iir=(sum(musica_iir.^2))/length(musica_iir);
```

```
>>P_musica_fir=(sum(musica_fir.^2))/length(musica_fir);
```

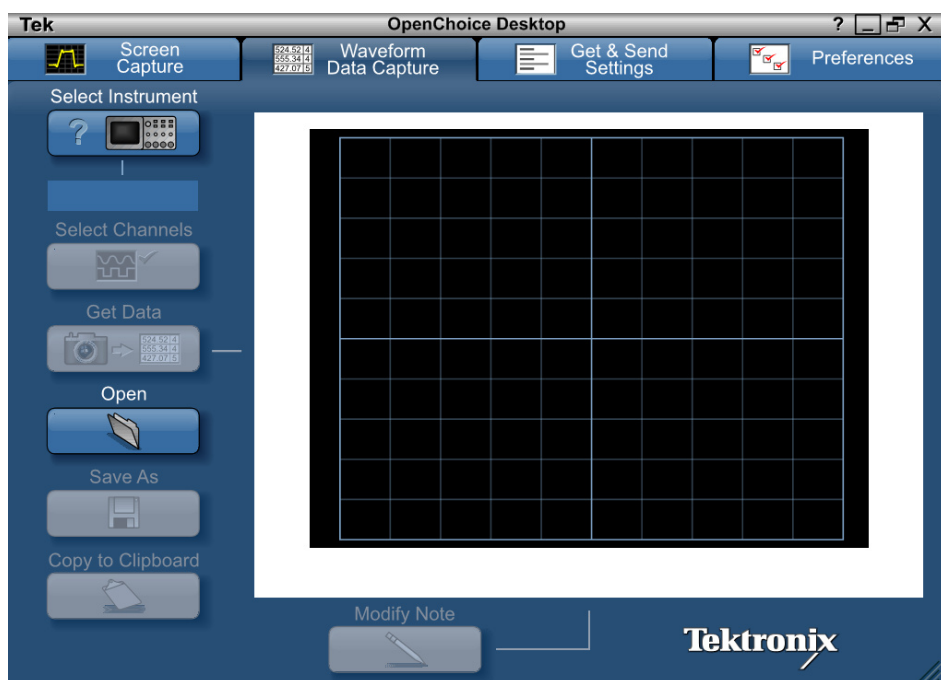
```
>>snr_iir=10*log10(P_musica_iir/P_ruido);
```

```
>>snr_fir=10*log10(P_musica_fir/P_ruido);
```

ANEXO F

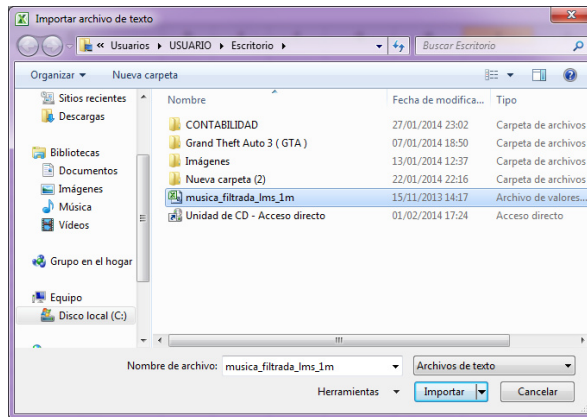
OBTENCIÓN DE TABLAS DE DATOS A PARTIR DE UNA SEÑAL MOSTRADA EN EL OSCILOSCOPIO

Una vez generada la señal en el osciloscopio, abrir el software *Open Choice Desktop*, luego hacer click en la pestaña *Waveform Data Capture*, la opción *Get Data* toma muestras de la señal del osciloscopio cada vez que se la seleccione, para tener muestras equidistantes para el análisis tomar muestras a cada minuto de la señal.

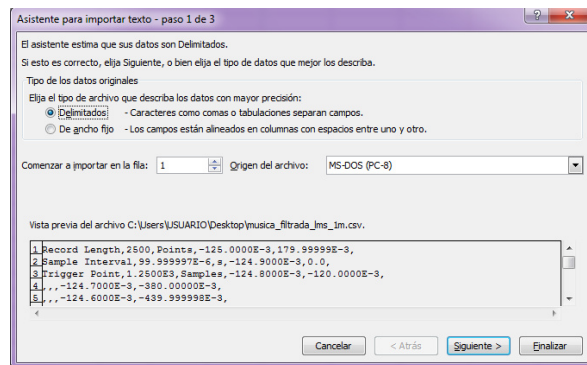


Guardar los datos obtenidos de la señal en el directorio que se desee haciendo click en la opción *Save As*.

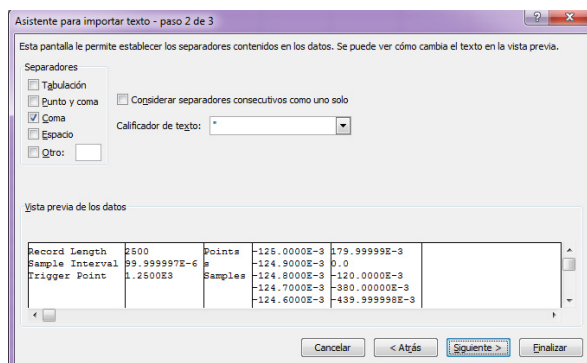
Para manipular los datos abrir Excel y hacer click en la pestaña *Datos*>>Desde texto en esta ventana se selecciona el dato que ha sido guardado en el directorio que se escogió.



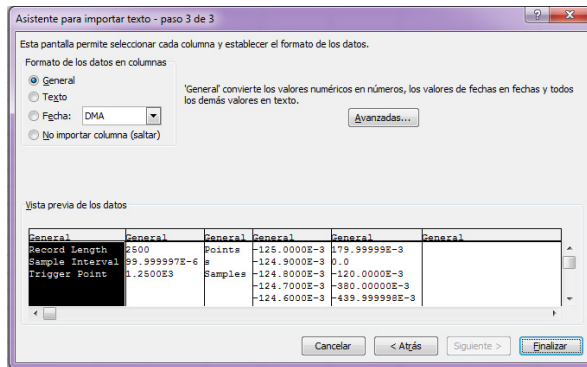
Se abrirá la ventana del Asistente para importar texto, seleccionar la opción *Delimitados* y hacer click en *Siguiente*.



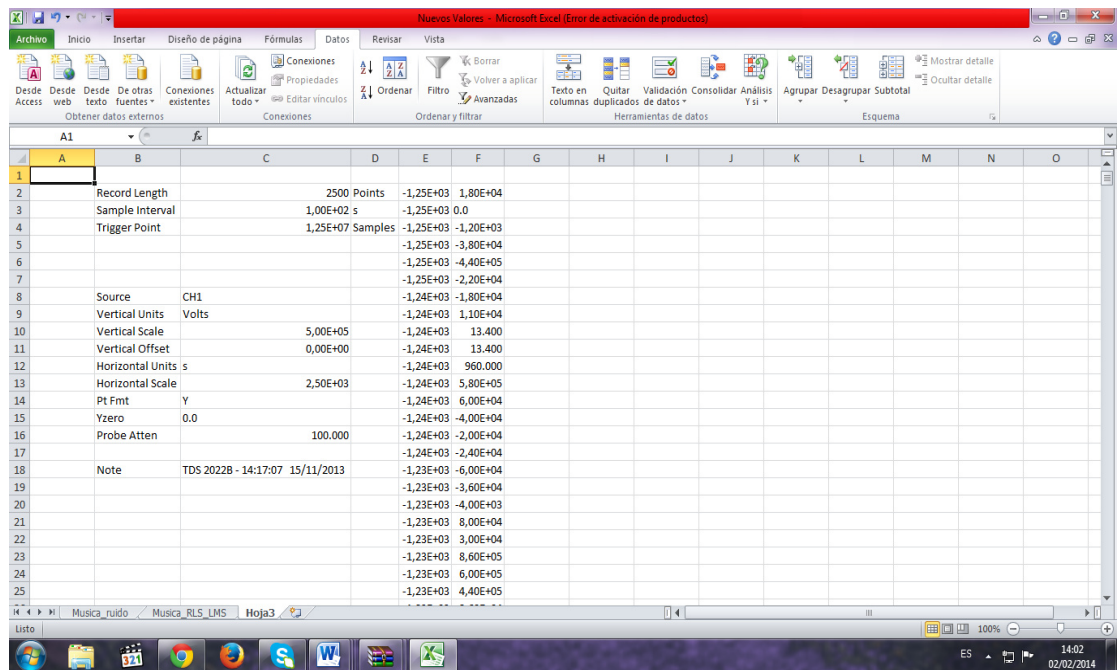
En el siguiente paso seleccionar la opción de separador por coma y hacer click en *Siguiente*.



En el último paso del asistente seleccionar la opción *Finalizar*.



Por último seleccionar la celda en la que se quiere insertar la tabla de datos, hacer click en Aceptar y la tabla aparecerá en la hoja de Excel.



Este mismo procedimiento se realiza tanto para las 4 muestras que se tomarán de la señal filtrada del osciloscopio, como para el ruido gaussiano, los datos obtenidos son lecturas de los picos voltaje de la señal en ese tiempo, las cuales se las transforma a voltaje RMS para poder hallar la relación señal a ruido, lo cual ya es una manipulación netamente matemática.

ANEXO G

RELACIÓN SNR/BER

La relación señal ruido y la tasa de error por bit son parámetros que se han escogido para analizar los filtros digitales y también la modulación/demodulación 16QAM, hay una ecuación que relaciona ambos parámetros que se detallan a continuación.

- Filtros digitales

Para el análisis de los filtros digitales simulados e implementados se tiene que la relación señal a ruido es la siguiente.

$$SNR = \frac{S}{N} = \frac{\text{Potencia de la señal}}{\text{Potencia del ruido}}$$

Mientras que el error por bit se define:

$$BER = \frac{\text{número de bits erróneos}}{\text{número de bits transmitidos}}$$

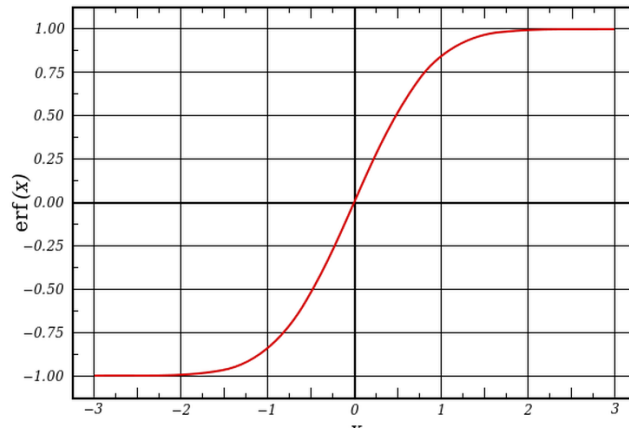
Sin embargo entre los ambos se tiene una relación que nos facilita el análisis:

$$BER = \frac{1}{2} \left[1 - \operatorname{erf} \left(\frac{\sqrt{SNR}}{2\sqrt{2}} \right) \right]$$

Donde *erf* representa la función de error gaussiano, el cual está definido por la ecuación.

$$\operatorname{erf}(x) = \frac{1}{\sqrt{\pi}} \int_0^x e^{-x^2} dx$$

Siendo la gráfica de la función de error gaussiana:



- Modulación/Demodulación 16QAM

En este caso la potencia del ruido se define como el cuadrado de la desviación típica del ruido blanco gaussiano cuando la media es cero.

$$SNR = 10 \log_{10} \frac{\text{Potencia de la señal}}{\text{Potencia del ruido}}$$

$$SNR = 10 \log_{10} \frac{\text{Potencia de la señal}}{(\text{Desviación típica del ruido})^2}$$

Para el análisis del SNR y BER en esta modulación es preciso detallar ciertos conceptos.

Para determinar la potencia de la señal debe analizarse el número de bits por símbolo necesarios para derivar hacia cada canal, en fase(I) y en cuadratura(Q), determinado por:

$$N = \log_2 n,$$

Donde n determina el número de modulación en nuestro caso 16. Dado el número de bits por símbolo, el ancho de banda para los canales I, Q se establece:

$$AB_{IQ} = \frac{f_b}{N}$$

Siendo f_b la frecuencia de bit la misma frecuencia del dato de entrada. Otro parametro importante es la energía promedio por símbolo dada por la fórmula:

$$E_{simbolo} = \sum_{m=1}^M p_m E_m$$

Donde p_m es la probabilidad de que ocurra un símbolo, sin embargo, se considera que los símbolos son equiprobables, siendo M el número de símbolos.

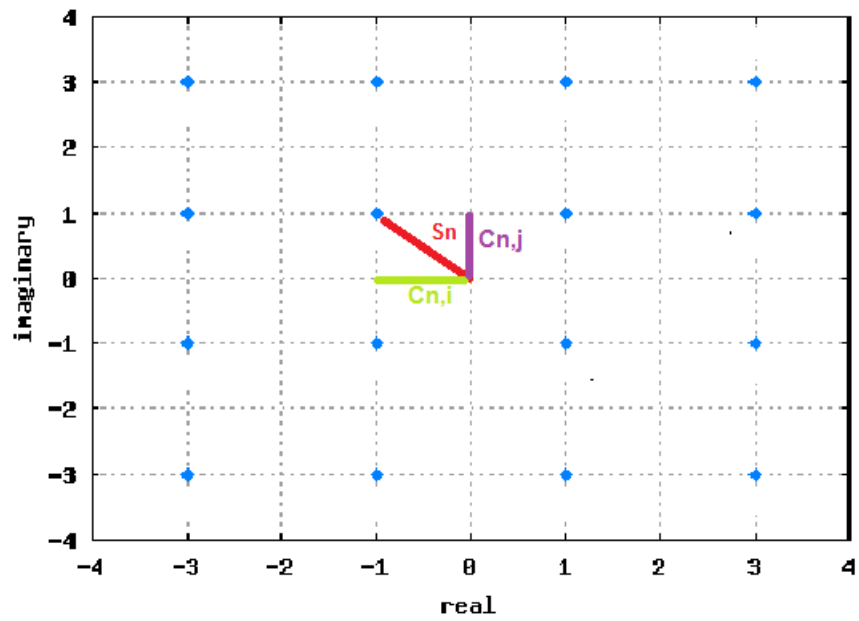
$$E_{simbolo} = \frac{1}{M} \sum_{m=1}^M E_m$$

Siendo E_m la energía de un punto en la constelación, se lo calcula:

$$E_m = \|\vec{S}_m\|^2 = \sum_{m=1}^M c_{m,i}^2$$

C_m es la distancia del origen hasta la coordenada (x,y) del punto a tratar. A partir de la energía del simbolo se obtiene la energía del bit.

$$E_m = \frac{E_b}{\log_2 M}$$



$$E_b = \frac{E_{simbolo}}{\log_2 M}$$

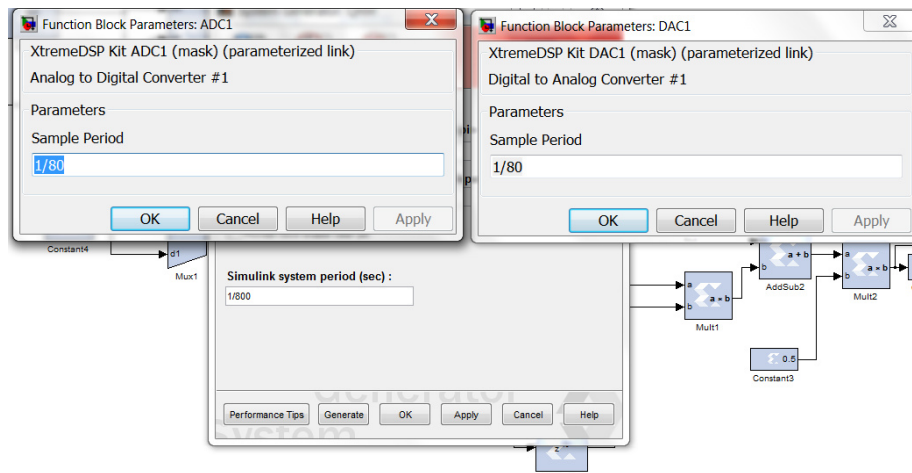
$$P_{señal} = E_b R_b$$

Teniendo la energía del bit se puede calcular la potencia de la señal para hallar la relación señal ruido (SNR) y la tasa de error por bit (BER).

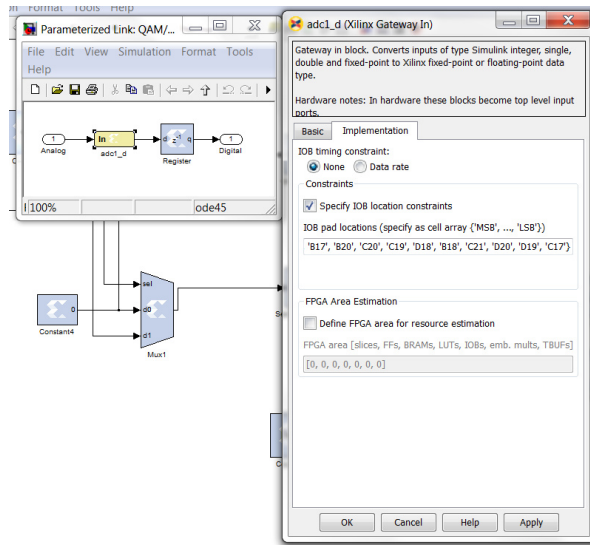
ANEXO H

ANTES DE GENERAR ARCHIVO .BIT

- 1.- Abrimos el archivo del programa hecho en Simulink
- 2.- Se deben tener en cuenta que los periodos de muestreo de las entradas y salidas del sistema deben ser múltiplo del período general del sistema especificado en System Generator.

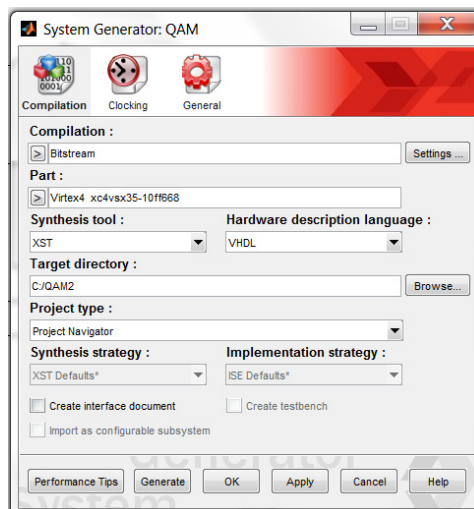


- 3.- Si se va a utilizar los periféricos adc's y dac's se debe tener en cuenta de que estén ubicados en orden desde el bit más significativo al menos significativo, siendo estos 14 bits tanto para la entrada y salida de datos.
Para esto nos ubicamos una máscara inferior a las dac's y adc's, haciendo click derecho en los bloques periféricos y eligiendo "look undermask".



GENERANDO EL ARCHIVO .BIT

Este tipo de archivo es necesario para poder programar la fpga de la tarjeta, para lo cual se debe seguir los siguientes pasos.



1.- Doble clic en el icono del System Generator del programa realizado en Simulink del que se desea generar el archivo .bit

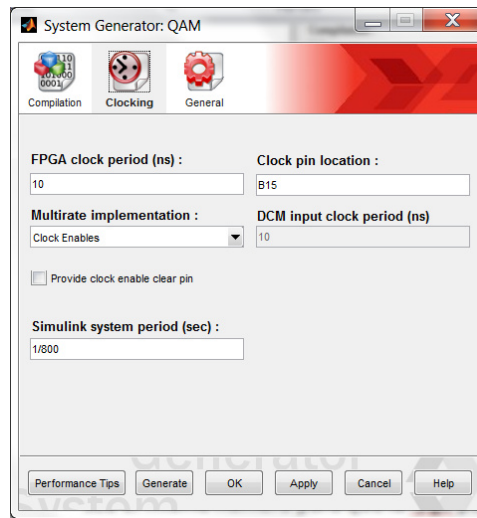
En nuestro caso es un archivo llamado ion que se encuentra grabado en el disco local c de las máquinas.

2.- En “compilation” elegir bitstream, siendo el tipo de archivo que necesitamos generar.

3.- Elegimos la tarjeta siendo esta: virtex4 xc4vsx35-10ff668, es sumamente necesario que sea el mismo número para que la fpga pueda reconocer el archivo .bit

4.- En “synthesistool” y “hardware description language” se deja con los valores que vienen por default.

5.- En target directory se elige la carpeta donde se quiere ubicar el .bit, es recomendable que la carpeta quede ubicada en (:c)



6.- Clic en la pestaña de clocking.

7.-En “*fpga clock period*” se especifica el valor del periodo con que el sistema requiere que trabaje el reloj

8.- En “*clock pin location*” debe ir b15 este es el lugar en el main de la fpga donde se va a colocar el clock interno

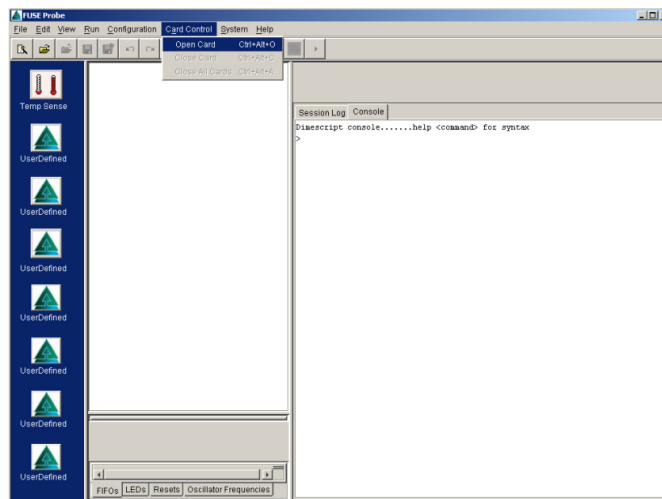
9.- En “*multi rate implementation*” se deja el valor que viene por default.

10.- Generamos .bit haciendo click en “generate”.

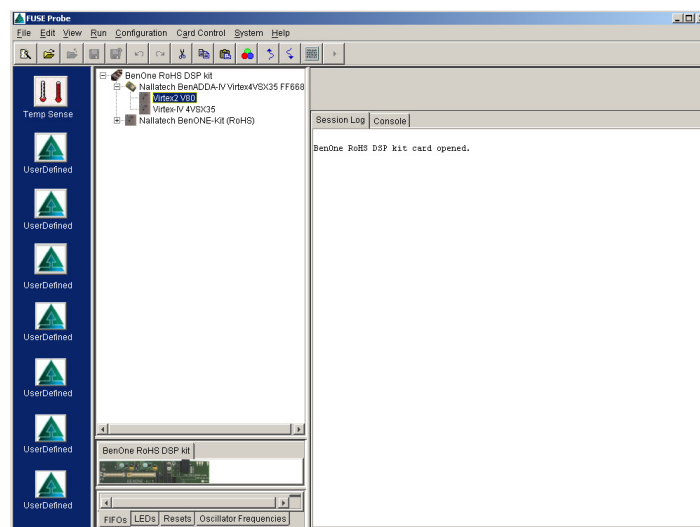
ANEXO I

LOCALIZACIÓN DE LAS TARJETAS FPGA

Una vez generado el archivo .bit se proceden a localizar las tarjetas en las que se ensamblará el archivo generado y también el archivo del clock, primero se abre el software fuse probe y hacemos click en open>>card control.



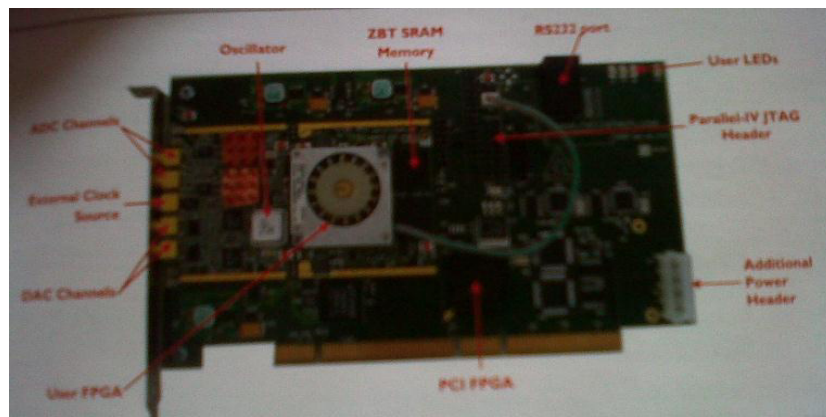
Luego se hace click en locate cards>>open card y se abren las tarjetas en las cuales se ingresara el archivo de clock y el .bit del modelo generado.



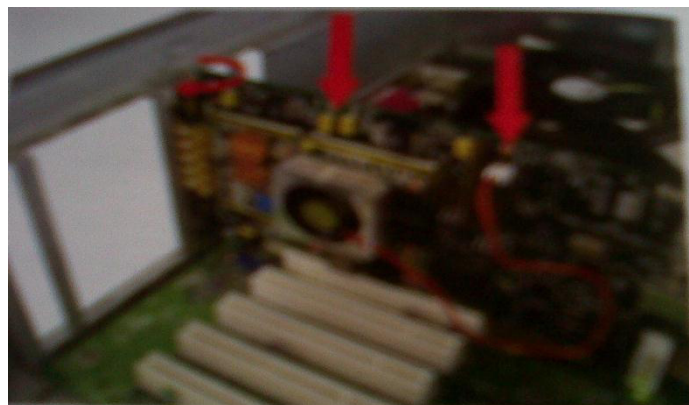
En la tarjeta virtex2 v80 se carga el archivo del clock que viene el cd de instalación del software fuse probe, y en la tarjeta virtex-iv 4vsx35 se carga el archivo .bit del modelo.

INSTALACIÓN DE LA TARJETA FPGA-VIRTEX IV EN LA PC

En esta sección instalaremos el hardware (la tarjeta fpga-virtex iv) a la pc siguiendo los pasos del manual del kit XtremeDevelopment kit IV.



- 1.- Asegurarse de que usa los procedimientos de manipulación de leds durante la instalación.
- 2 -. Asegúrese de que la alimentación del pc está apaga
- 3.-Quite la cubierta de la PC, busque una ranura PCI libre para el kit.
- 4 -. Colocar firmemente la tarjeta en la ranura PCI.
- 5 -. Utilice el tornillo de fijación para asegurar la placa posterior de la tarjeta en el chasis de pc.
- 6 -. Encender la fuente de alimentación del pc `s.



INSTALACIÓN DEL PROGRAMA XtremeDSP Development kit IV

