UNIVERSIDAD AUTÓNOMA DE QUERÉTARO

Facultad de Ingeniería

Metodología para el diseño y la implementación de filtros digitales IIR

TESIS

Para obtener el título de:

Ingeniero en Automatización

Presenta:

Márquez Gutiérrez Miguel Arturo

Asesor:

M. en C. Diana Carolina Toledo Pérez

Coasesor:

Dr. Miguel Ángel Martínez Prado

ÍNDICE GENERAL

1 CONTENIDO

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

ÍNDICE DE ECUACIONES

1 INTRODUCCIÓN

1.1 Justificación

Las fuentes de información que están centradas en el diseño de filtros, hasta el momento, no se muestran especializadas en el diseño de filtros digitales tipo IIR o contienen un desarrollo teórico demasiado extenso o muy reducido con lo cual no se obtiene una fundamentación suficiente para realizar su aplicación inmediata e implementación en proyectos donde se requiera filtrar una señal de manera digital. Este trabajo está orientado a la implementación de dichos filtros en FPGA (Matriz de puertas lógicas programable en campo, por sus siglas en inglés, *Field-Programmable Gate Array*) debido a que ofrece un método de bajo costo y eficiente para generar prototipos de manera más rápida de sistemas digitales amplios con tiempos cortos de regreso entre el desarrollo inicial de la estructura de un filtro y su implementación en un prototipo en tiempo real, ya que se pueden efectuar las adaptaciones necesarias en paralelo con su implementación. Esto es llamado "reconfigurabilidad", una ventaja frente a los sistemas ASIC (Circuitos de aplicación específica, por sus siglas en inglés *Application-specific Integrated Circuit*). De la misma manera, su funcionamiento en paralelo y la capacidad de flujo en línea secuencial (*Pipelining*) es una ventaja que presentan las FPGA frente a los DSP (Procesador de señales digitales, por sus siglas en inglés *Digital Signal Processor*).

La implementación de un determinado algoritmo en una FPGA tiene un menor consumo de energía y recursos que la ejecución del mismo en un procesador, particularmente cuando se trata de una computadora sobre la cual se corre un sistema operativo y dentro de él se ejecuta un programa que hace el cálculo para el tratamiento de la señal.

Existen en el mercado muchas opciones que probablemente satisfagan el objetivo, pero la mayoría de ellas son de alto costo o requieren mantenimiento obligado de su misma empresa para garantizar el rendimiento adecuado. Además, con la evolución tecnológica cada vez se requieren más sistemas que satisfagan las condiciones actuales, por lo que es necesario proponer una alternativa nacional que provea estos resultados y sobre todo que sea accesible, confiable y fácil de entender para el usuario final.

1.2 Descripción del problema

El diseño de filtros digitales hasta ahora ha sido limitado por los recursos teóricos existentes y el bajo nivel didáctico que poseen. Además, gracias al estudio de filtros analógicos y el uso de herramientas de software, se pueden generar algoritmos de procesamiento digital en lenguaje HDL (Lenguaje de descripción de hardware, por sus siglas en ingles *Hardware Description Language*), en corto tiempo y sin tanta complejidad. Aunado a esto, el rango de aplicaciones de los filtros digitales implementados en una FPGA se puede aumentar si las herramientas de diseño y la fundamentación teórica están al alcance de todos los estudiantes.

En el cuerpo humano se generan diferentes señales bio-eléctricas presentes en tejidos, células, nervios, glándulas, entre otros. Las señales eléctricas que provienen de los músculos son conocidas como mioeléctricas; las cuales, a pesar de presentar niveles de voltaje reducidos, pueden medirse con el equipo adecuado, para posteriormente, ser empleadas en una amplia variedad de aplicaciones, como el análisis de los movimientos orientados a la robótica para el manejo de prótesis. Una señal adquirida desde los músculos remanentes de un miembro amputado puede ser clasificada para realizar diferentes movimientos en una prótesis robótica. Sin embargo, debido a que estamos rodeados de ruido electromagnético, las señales eléctricas adquiridas deben ser limpiadas, es decir, filtradas para asegurar que los datos son certeros, lo más cercano a la realidad para tener una interpretación correcta y, por lo tanto, un mejor manejo de la información.

Así como éstas, existe una amplia variedad de señales de las cuales es necesario remover el ruido agregado por el medio ambiente, o realizar una selección específica de frecuencias con las que se desea trabajar; lo que hace necesario el desarrollo de filtros digitales que ofrezcan una mayor precisión que los analógicos, y que además, su uso permita su aplicación en tiempo real, por lo que al implementarse en una FPGA se obtenga una mayor rapidez que la implementación en un procesador secuencial.

1.3 Antecedentes

La tecnología FPGA ha sido introducida en diferentes áreas de la industria, tales como la industria automotriz, electrodomésticos y en áreas de aplicación donde se requiere un alto desempeño computacional. Esto se debe a que las FPGA ofrecen una alternativa efectiva en el diseño de sistemas digitales con respuesta de corto tiempo y su implementación en el prototipo en tiempo real (Madanayake *et al*., 2004).

El funcionamiento en paralelo y la capacidad de canalización de las señales se dan mejor en una FPGA que en un DPS; además, el bajo costo y la reconfigurabilidad son ventajas que tiene la primera comparadas contra un ASIC (Costa y Páez, 2015). En general, la implementación de un determinado algoritmo en FPGA tiene menor consumo de energía que la ejecución sobre un procesador, particularmente cuando se trata de una computadora en la que se está corriendo un sistema operativo y sobre éste, un programa que es el que ejecuta el cálculo. La descripción de hardware directa es la técnica que brinda la mayor flexibilidad, permite la portabilidad en los diseños y requiere menor cantidad de recursos, particularmente bloques DSP para las operaciones MAC. La desventaja de esta técnica reside en la dificultad del proceso de simulación. En otro sentido, la técnica de diseño empleando una herramienta de traducción de bloques de modelado resulta la más práctica y su desventaja reside en la falta de portabilidad y en la utilización de un poco más de recursos que en la otra técnica mencionada.

Podemos definir a un filtro digital como un sistema el cual realiza las operaciones matemáticas sobre una señal muestreada variante en el tiempo discreto para reducir o aumentar ciertos aspectos de dicha señal. El desempeño del diseño de filtros FIR e IIR se analiza a través del tiempo de respuesta, uso de recursos, etc, (Paul *et al*., 2015). En otros usos que se le da a la FPGA para filtrar información, Wang y Maruyama, (2016), emplean un filtro de caja (*Box Filter*) para el procesamiento de imágenes. Aunque, debido al empleo de un arreglo de memoria para almacenar la información, la cantidad de espacio necesario es proporcional al tamaño de la imagen, el desempeño está ligado a la tecnología donde se implemente. El método descrito en el documento demuestra cómo emplear menos memoria, al usar bloques RAM distribuidos y un escaneo en zigzag. Derivado de esto, se obtuvo un resultado efectivo para aplicaciones que calculan correlaciones cruzadas para encontrar la mejor coincidencia; sin embargo, se reduce el desempeño, aunque es lo suficientemente rápido para otras aplicaciones como también flujo óptico y reducción de ruido.

De igual forma, un ejemplo de procesamiento de imágenes con arquitectura completamente genérica y reprogramable se encuentra en Mukherjee *et al*., (2016), quienes muestran un algoritmo y su arquitectura de hardware para implementar operaciones morfológicas de imágenes en escala de grises. Las principales ventajas que obtuvieron fueron la baja latencia, bajos requerimientos de memoria y un alto rango de procesamiento, lo que lo hace apto para distintos tipos de aplicaciones y cuya desventaja radica en la falta de flexibilidad ante el procesamiento de distintos tipos de elementos de estructuración.

En aplicaciones como la adquisición de señales eléctricas por medio de transductores, es necesario filtrar la señal adquirida ya que el medio añade ruido a la información. Por ejemplo, la temperatura es uno de los parámetros de más interés en la producción industrial y agrícola, así como en experimentos científicos. Cuando se tiene una variación rápida de la temperatura, normalmente es porque se genera por procesos en condiciones hostiles. Además, en los alrededores se pueden tener otros factores medioambientales involucrados como altas presiones y alto impacto del aire. Dicho esto, un software tradicional aplicado para inhibir el ruido después de una prueba demuestra una precisión alta pero no alcanza los requerimientos para el procesamiento de pruebas intensivas en tiempo real. Una FPGA tiene una capacidad de procesamiento en paralelo, la cual le permite poseer filtros de alto rendimiento y operación rápida lo que, la hace una opción efectiva para limpiar una señal digitalizada con ruido añadido (Zhao y Zhang, 2016).

La implementación de filtros en FPGA para el pre-proceso de señales como las EEG (*Electroencefalograma,* señales provenientes del cerebro) se demuestra en Sundaram *et al*., (2016), donde los principales puntos de comparación son la potencia consumida, el tiempo de respuesta y el área usada para cada filtro. El pre-proceso involucra remover el ruido y artefactos de la señal EEG para incrementar la exactitud en la clasificación, los artefactos son potenciales indeseados originados por estímulos no cerebrales. En este trabajo, los filtros, promedio y mediana móvil son implementados en FPGA resultando mejor para el pre-procesado, el de mediana móvil que ocupa menos área y potencia, el de promedio móvil.

En Jayant *et al*., (2015) se muestra el diseño de un filtro Notch de segundo orden de tipo IIR, con el objetivo de suprimir el PLI (*Power Line Interference,* interferencia de la línea eléctrica). Para esta tarea, la técnica de optimización *Minimax* fue empleada para minimizar el RMSE (*Root Mean Square Error*, error cuadrático medio) definido como la

diferencia de magnitud entre la respuesta obtenida y la ideal deseada. Este trabajo explora la aplicación de dicha técnica para la optimización del diseño de tal filtro, donde en la simulación obtienen una atenuación de la señal a 50 [Hz] de -30 [dB] y en la aplicación real de -26 [dB] lo cual es muy acercado a lo esperado, sin embargo, dicho modelo es un diseño delimitado para esta única aplicación.

Por sí mismo, un filtro adaptativo tiene el objetivo de recuperar la señal que contiene información de la señal ruidosa ajustando el valor de sus coeficientes. Un filtro IIR adaptable se compone de dos elementos básicos: un filtro IIR de tiempo variable y un algoritmo que actualice los coeficientes del filtro para optimizar el error. D. Sharma y R. Kaur, (2015), cuyo objetivo principal es generar una señal estimada apropiada para la señal requerida. Este trabajo demuestra la efectividad de dicha técnica en la simulación.

Los sistemas de adquisición de datos, como el nombre lo indica, son productos y procesos usados para recolectar información a un documento o analizar algún fenómeno. Una FPGA provee una plataforma flexible para la computación en paralelo lo cual representa un beneficio para las características de un hardware re-configurable (Adhikary *et al*., 2012).

Cuando se trata de hablar sobre las implementaciones de la tarjeta de desarrollo BASYS-3 no se encuentran muchas opciones en cuanto al desarrollo tecnológico debido a que es una herramienta de hardware relativamente nueva, pero podemos apuntar las expectativas de aplicación de esta plataforma tomando en cuenta a Bhavanam *et al*., (2014), quien desarrolló un método alternativo para la detección de señales de doble tono en múltiples frecuencias, la cual es una aplicación muy importante en el equipo de telecomunicaciones. Existe mucha documentación acerca de la detección DTMF (sistema multifrecuencial, "Dual Tone Multi Frecuency", por sus siglas en inglés) pero ninguna emplea esta solución, debido a que se usa el algoritmo de Goertzel en combinación de la FPGA Zynq 7000, que en comparación con la técnica de FFT (transformada rápida de Fourier, "Fast Fourier Transform") para la aproximación de la detección de DTMF, la cual requiere demasiados recursos físicos y tiene un alto consumo de potencia, optimiza los recursos físicos usando una fuente de aproximación compartida que es el equivalente a tener bloques de operaciones físicos, pero aquí se llevan a cabo con una lista común de redes en conjunto con el algoritmo de Goertzel. En consecuencia, se puede prever la implementación para la detección de tonos de baja frecuencia.

Por otro lado, si se requiere implementar controladores para actuadores eléctricos, es necesario satisfacer ciertos requerimientos para obtener los mejores resultados. Lahoucine, en Idkhajine *et al*., 2009, menciona que dentro de los puntos más importantes del criterio para lograr los mejores resultados se encuentra: la integración de alto nivel y densidad del objetivo usado para la implementación de control, sistemas embebidos de bajo costo basados en controladores completamente integrados, lo cual asegura muchas tareas de control por el mismo dispositivo, y flexibilidad para modificar la estrategia y los parámetros de control; por lo que una FPGA ofrece una densidad significante debido a su arquitectura especifica de hardware digital por encima de soluciones de software.

Aloisio *et al*., (2007) explica como desarrollaron un sistema integrado en un solo chip (SoC) que posee la capacidad de adquirir señales de detectores, procesar la información y realizar pruebas de monitoreo. Todo logrado a partir de un procesador PicoBlaze de 8 bits de Xillinx®. En conjunto, el sistema integra un ADC que se probado con éxito en una FPGA. La descripción de VHDL es una tecnología completamente independiente que permite al usuario programar ya sea una FPGA o diseñar una célula ASIC. La máxima frecuencia de operación es 100[MHz] en la FPGA y 180[MHz] una célula estándar, el sistema trabaja en tiempo real y tiene una latencia de 5-6 ciclos de reloj para el IRQ y dos para las I/O.

En el diseño de sistemas de control no estandarizados, así como en los sistemas de control de alto rendimiento, control adaptable, o sistemas de control distribuido, necesitan una potencia de computación grande o acciones de control con bandas del orden de los kHz, y no son fácilmente alcanzables con los sistemas abiertos comerciales sin arquitecturas complejas y costosas. Entonces, para mejorar un sistema, el control digital basado en VME, puede ser reemplazado por un sistema de control digital híbrido. Por ejemplo, en una suspensión mecánica desarrollada para detectores *interferenciométricos* para ondas gravitacionales, manteniendo una frecuencia de muestreo de 10 [kHz] empleando en una tarjeta una FPGA, un ADC de 18 bits con bajo ruido y un DAC de 16 bits a 800 kHz con un protocolo estándar para la comunicación con la PC ya que al ser independiente del DAC y el ADC, el sistema se puede considerar síncrono y emplear un protocolo estándar como el EPP (puerto paralelo, Enhanced Parallel Port) (Garufi *et al*., 2008).

Es bien sabido que un software de aplicación puede ser transformado o adaptado a un hardware equivalente en una FPGA usando lenguajes de descripción de hardware

(HDL, Hardware Description Language). Por ejemplo, un segmento de software que añada elementos a un archivo puede ser convertido a un registro físico con un sumador y un acumulador usando HDLs. Estos módulos de HDL son llamados típicamente núcleos, los cuales pueden ser adquiridos o generados por el diseñador. Además, no sería necesario cambiar el hardware de un instrumento si se requiere añadir un núcleo con FFT o una transformada discreta del coseno (DTC) para una aplicación diferente. Además, los sistemas basados en FPGA tienen una capacidad de procesamiento mayor comparada con arquitecturas compuestas con comparadores basados en software (Abdallah *et al*., 2011).

Por otra parte, los convertidores de potencia ofrecen una alta capacidad para el manejo eficiente de flujos de energía eléctrica, y hasta hace algunos años, éstos se ocupaban en aplicaciones como alimentación de motores industriales y sistemas eléctricos de tracción, principalmente. Actualmente, en adición a este campo de aplicaciones, también son empleados en aquellas que implican bajo, medio y alto manejo de energía, incluyendo aplicaciones residenciales, sistemas de energía renovable, generación distribuida y automotriz. Debido a esto, los retos principales de un diseñador de convertidores de potencia son los siguientes: manejar la energía de entrada y salida, usualmente bidireccional, de tal manera que el proceso completo garantice la más alta eficiencia posible al mismo tiempo que satisfaga las operaciones esperadas; ofrecer un alto grado de precisión, flexibilidad, capacidad de comunicación y confiabilidad para el usuario final; así como disminuir los costos totales (Buccella *et al*., 2012).

Para situaciones de altas velocidades, los algoritmos de corrección usualmente son implementados fuera de línea, lo que representa un obstáculo para la aplicación de un TIADC (time-interleaved AD conversion, conversión de ADC intercalada en el tiempo). La digitalización de formas de onda a velocidades ultra-rápidas se puede alcanzar con la técnica TIADC, en contraste con los métodos basados en arreglos de capacitores conmutados (SCA), ya que se puede tener una captura continua más larga que puede ser ajustada para diferentes requerimientos; por ejemplo en los osciloscopios digitales modernos (Zhao *et al*., 2013). Debido a esto, la corrección en tiempo real dentro de la FPGA se convierte en la solución para altas velocidades y bajo consumo de energía.

Con el objetivo de alcanzar una resolución mayor que el periodo de muestreo de un ADC, se puede implementar en la FPGA una versión digital del método del discriminador fraccional constante (CFD, Constant Fraction Discriminator), eliminando la necesidad de un dispositivo externo de señal mezclada. La información que se puede obtener con la

FPGA incluye temporización de eventos, determinación-discriminación de energía y procesos de coincidencia. Ahora bien, el procesamiento en línea de eventos coincidentes reduce significativamente la cantidad de información que será transmitida a la computadora. De tal manera que, si se tiene una memoria temporal externa pero incluida en la tarjeta de desarrollo, los eventos coincidentes pueden ser escritos ahí y luego ser enviados por medio de Ethernet a una PC (Fysikopoulos *et al*., 2014).

Teniendo esto en mente, existen ciertas constantes que manejar para la implementación, particularmente para algoritmos complejos. Primeramente, la preservación del potencial inherente al paralelismo del algoritmo está fuertemente relacionado a los recursos disponibles de hardware. El tiempo de computación debe ser menor que el tiempo de simulación RT para evitar desbordamiento, además en adición a la latencia del algoritmo, el tiempo de computación depende de la frecuencia de reloj empleada. El diseñador debe entonces reducir los retrasos causados por la propagación de los operadores aritméticos, para incrementar la máxima frecuencia de reloj alcanzable y por ende reducir el tiempo de computación. Todo esto con el objetivo de reducir los recursos necesarios de hardware mediante la factorización de los operadores pesados como los multiplicadores (Dagbagi *et al*., 2016).

Veiga y Grunfeld, (2016) alcanzó una programación de alto nivel modular para aplicaciones en tiempo real de rayos gamma, empleando un ADC de 16 bits a 250 Mmps pero con una FPGA de gama media; lo que supone un buen balance entre tiempo y energía de resolución, comparado con los mejores instrumentos analógicos. La organización en línea de procesador digital puede garantizar un rendimiento de trabajo no accesible para otras tecnologías, además, el diseño modular no conlleva únicamente a un mejor entendimiento de la instrumentación de proceso, sino que también provee una organización que maneja eficientemente las cadenas de datos resultantes con FPGAs regulares.

La organización propuesta constituye una modernización viable para el equipo analógico de alta resolución como el que está en laboratorio de espectroscopia de aniquilación y TD-PAC en Mössbauer. Esto también provee una solución eficiente de alta densidad y bajo consumo de energía para aplicaciones que requieren un gran número de canales, como la tomografía del tiempo de vuelo de positrones donde simultáneamente se requiere la determinación del tiempo de impacto y la energía.

1.4 Hipótesis

A través de una metodología que emplee a la ganancia, la f_c, f_p, f_s, R_p y R_s como parámetros de diseño, se pueden diseñar filtros digitales IIR para ser implementados en una FPGA que provean atenuación al ruido añadido a la señal digitalizada en un menor tiempo que un procesador secuencial y sin tener alteraciones debido a la sensibilidad como las de un filtro analógico.

1.5 Objetivos

1.5.1 Objetivo General

Elaborar un método para el diseño de un filtro digital reconfigurable de tiempo discreto IIR basado en el modelo matemático de un filtro analógico, que sea más rápido al implementarse en un FPGA empleando VHDL que en una PC utilizando MATLAB®, sin tener alteraciones debido a la sensibilidad como las de un filtro analógico, proveyendo una atenuación en la amplitud a las frecuencias muy cercanas a las deseadas.

1.5.2 Objetivos Particulares

1. Definir el sistema de adquisición de señales de EMG a utilizar para prueba de los filtros digitales.
2. Describir el tipo y la cantidad de especificaciones del sistema necesarias para obtener los coeficientes del filtro al aplicar el modelo matemático del filtro analógico.
3. Diseñar e implementar un filtro digital IIR reconfigurable en una PC, utilizando el software de MATLAB®.
4. Diseñar e implementar un filtro digital IIR reconfigurable en una tarjeta BASYS 3, que cuenta con un FPGA de Xilinx Artix-7.

1.6 Metas

1.6.1 Metas tecnológicas

- Realizar la programación en FPGA de filtros que se pueda configurar según los requerimientos del usuario.
- Remover los componentes de voltaje a 60 [Hz] de una señal EMG previamente digitalizada usando el filtro implementado en FPGA.

1.6.2 Metas de formación de recursos humanos

Obtener el grado de licenciatura en ingeniería en automatización con línea terminal en electrónica.

2 FUNDAMENTACIÓN TEÓRICA

2.1 Filtros analógicos y digitales

Los filtros analógicos están presentes en casi todas las piezas de los equipos electrónicos. Dentro de los ejemplos más claros están los radios, las televisiones y los sistemas estéreos. Los equipos de medición como analizadores de espectro, así como los equipos generadores de señales también necesitan filtros; incluso donde las señales son convertidas en forma digital, usando convertidores analógico-digitales, los filtros se emplean para prevenir el efecto aliasing. Las computadoras usan filtros para reducir las emisiones EMI (interferencia electro-magnética) provenientes de la línea de alimentación, también para reducir la salida de la fuente conmutada para limitar el ancho de banda de las señales que se dirigen a la pantalla, entre otros ejemplos más.

Sin embargo, el uso de filtros digitales se está esparciendo cada vez más, reemplazando a los filtros analógicos en muchos sistemas electrónicos Comparado con los filtros analógicos, los filtros digitales procesan señales en el dominio del tiempo; para poder aplicarse, primero, las señales analógicas tienen que ser muestreadas y digitalizadas en intervalos discretos de tiempo (ciclos de reloj) usando un convertidor analógico digital.

Los filtros digitales son una clase particularmente importante de sistemas invariantes en el tiempo. Estrictamente hablando, el término *filtro de selección de frecuencia* sugiere un sistema que idealmente permite el paso a través de él de cierto intervalo de frecuencias definidas y rechaza completamente otro, sin embargo, en un contexto más amplio, cualquier sistema que modifica tanto la magnitud como la fase de las señales a ciertas frecuencias en relación a otras es también llamado filtro. En teoría de circuitos, un filtro es una red eléctrica que altera la amplitud y/o las características de la fase de una señal con respecto a la frecuencia. Idealmente, un filtro no debería añadir componentes de otras frecuencias a la señal eléctrica tratada, así como tampoco cambiar la frecuencia de los componentes de dicha señal, pero sí cambiar las amplitudes de los componentes de frecuencia seleccionados y/o las relaciones con su fase.

Ya que los filtros son definidos por sus efectos sobre el dominio en la frecuencia de las señales, tiene sentido, entonces, que la mayoría de las descripciones gráficas y analíticas de filtros también estén orientadas al dominio de la frecuencia. Debido a esto, las curvas de ganancia contra la frecuencia son

usadas comúnmente para ilustrar las características del filtro. Así mismo, las herramientas matemáticas están basadas en el dominio de la frecuencia.

El comportamiento en el dominio de la frecuencia de un filtro está descrito matemáticamente en términos de su función de transferencia o función de red. Esta es la relación de las transformadas de Laplace de las señales de entrada y de salida. La función de transferencia define la respuesta del filtro a cualquier señal arbitraria de entrada, su importancia radica especialmente en la magnitud de la misma en función de la frecuencia dado que este es el indicador del efecto del filtro sobre las amplitudes de las señales procesadas. Conocer la magnitud de la función de transferencia (ganancia) en cada intervalo de frecuencia permite determinar la manera en la que el filtro distinguirá entre señales a diferentes frecuencias.

Un filtro analógico puede ser descrito por su función

$$H_a(s) = \frac{B(s)}{A(s)} = \frac{\sum_{k=0}^{M} \beta_k s^k}{\sum_{k=0}^{N} \alpha_k s^k} \quad (1)$$

Donde $\{\alpha_k\}$ y $\{\beta_k\}$ son los coeficientes del filtro, o su respuesta al impulso la cual está relacionada a $H_a(s)$ por la transformada de Laplace.

$$H_a(s) = \int_{-\infty}^{\infty} h(t) e^{-st} dt \quad (2)$$

Alternativamente, el filtro analógico teniendo la función de transferencia racional $H_a(s)$ dada por (1), puede ser descrito de manera de ecuación lineal coeficientes-constantes.

$$\sum_{k=0}^{N} \alpha_k \frac{d^k y(t)}{dt^k} = \sum_{k=0}^{M} \beta_k \frac{d^k x(t)}{dt^k} \quad (3)$$

Donde $x(t)$ es la señal de entrada y $y(t)$ la señal de salida del filtro.

En el diseño de filtros selectivos a la frecuencia, las características del filtro deseado son especificadas en el dominio de la frecuencia en términos de la magnitud deseada y la respuesta de fase del filtro.

Si no existe el requerimiento de tener un comportamiento de fase lineal en la banda de paso, ya sea un filtro FIR o IIR puede ser empleado. De cualquier manera, como regla general, un filtro IIR causa crestas menores en la banda de paro que un filtro FIR teniendo el mismo número de parámetros. Por esta razón, si un poco de distorsión de fase es tolerable o sin importancia, es preferible usar un filtro IIR principalmente porque su implementación engloba menos parámetros, requiere menos memoria y tiene una complejidad computacional menor.

El diseño de cualquier filtro se divide en las siguientes tres etapas: 1) La especificación de las propiedades deseadas del sistema; 2) la aproximación de las especificaciones usando un sistema discreto causal y 3) la realización del sistema. A pesar de que estas tres etapas son de cierta forma independientes, la primera etapa es altamente dependiente de la aplicación y la segunda etapa de la tecnología empleada para la implementación.

El diseño de un filtro IIR está basado en el diseño de un filtro analógico. Las vías de la señal están arregladas de tal manera que la salida depende tanto de la señal de entrada como la propia señal de salida. Las señales de entrada son dirigidas a una línea de retraso, luego factores de multiplicación son usados de la misma manera que en un filtro FIR para proveer una señal de adición. Además, la señal de salida es llevada a una segunda línea de retraso e igualmente factores de multiplicación son usados para proveer una señal de retroalimentación. Estas dos señales son combinadas posteriormente, para producir la señal de salida. Finalmente, la respuesta analógica de frecuencia requerida es transformada al dominio discreto, usando ecuaciones simples, para generar los coeficientes de multiplicación del diseño.

Cuando un filtro de tiempo discreto será usado para procesar señales continuas, como se muestra en la configuración de la Figura 2.1, las especificaciones tanto para el filtro de tiempo discreto como el filtro efectivo en tiempo continuo son típicamente (pero no siempre) dadas en el dominio de la frecuencia. Esto es especialmente común para filtros selectivos de frecuencia como pasa-bajas, pasa-banda y pasa-alta. Si el ancho de banda de la entrada está limitado y la frecuencia de muestreo es lo suficientemente alta para evitar el efecto aliasing, entonces el sistema completo se comporta como un sistema lineal invariante y continuo en el tiempo con respuesta a la frecuencia:

$$H_{eff}(j\Omega) = \begin{cases} H(e^{j\Omega T}), & |\Omega| < \frac{\pi}{T}, \\ 0, & |\Omega| > \frac{\pi}{T}. \end{cases} \qquad (4)$$

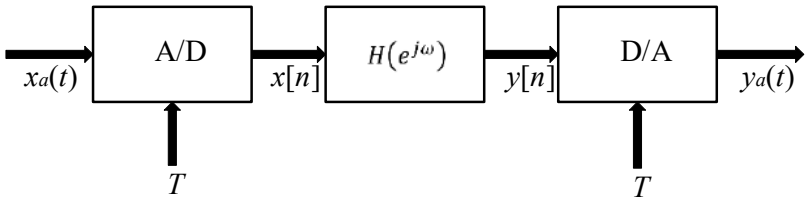

Figura 2.1. Sistema básico para filtrado en tiempo discreto de señales continuas en el tiempo (Winder, 2002).

En tales casos, es conveniente convertir las especificaciones efectivas para el filtro en el tiempo continuo al filtro en el tiempo discreto a través de la relación $\omega = \Omega T$. Que es, $H\left(e^{j\omega}\right)$ especificada sobre un periodo por la ecuación (2).

$$H(e^{j\omega}) = H_{eff}\left(j\frac{\omega}{T}\right), |\omega| < \pi \quad (5)$$

La causalidad tiene implicaciones realmente importantes en el diseño de filtros selectivos de frecuencia. Los cuales se pueden enlistar como: a) la respuesta a la frecuencia $H(\omega)$ no puede ser cero, excepto en un número finito de puntos en la frecuencia; b) la magnitud de $|H(\omega)|$ no puede ser constante en ningún rango finito de frecuencias y la transición de la banda de paso a la banda de paro no puede ser infinitamente recta (esto como consecuencia del fenómeno Gibbs el cual resulta de truncar $h(n)$ para alcanzar la causalidad); y c) la parte real e imaginaria de $H(\omega)$ son interdependientes y están relacionadas por la transformada discreta de Hilbert. Como consecuencia, la magnitud $|H(\omega)|$ y la fase $\Theta(\omega)$ de $H(\omega)$ no pueden ser escogidas arbitrariamente.

Entonces, conocidas las restricciones que la causalidad impone en las características de la respuesta a la frecuencia y el hecho de que los filtros ideales no son alcanzables en la práctica, limitaremos nuestra atención a la clase de sistemas lineales invariantes en el tiempo, descritos por la ecuación diferencial:

$$y(n) = -\sum_{k=1}^{N} a_k y(n-k) + \sum_{k=0}^{N} b_k x(n-k) \quad (6)$$

Estos sistemas tienen una respuesta a la frecuencia como se describe a continuación:

$$H(\omega) = \frac{\sum_{k=0}^{M} b_k e^{-j\omega k}}{1 + \sum_{k=1}^{N} a_k e^{-j\omega k}} \quad (7)$$

Hablando en términos prácticos, la no causalidad de un filtro se refiere a que la respuesta a la frecuencia $H(\omega)$ no puede ser cero, a excepción de un grupo finito de puntos dentro de un rango de frecuencias, Además, la función de salida no puede tener una transición de la banda de paso a la banda de paro absolutamente recta, es decir, no puede caer de manera abrupta la unidad a cero.

Sin embargo, aunque no sea posible producir un filtro con un comportamiento ideal, realizar uno de manera real que se aproxime y sea funcione en la aplicación deseada, puede lograrse si se relajan las condiciones características del filtro.

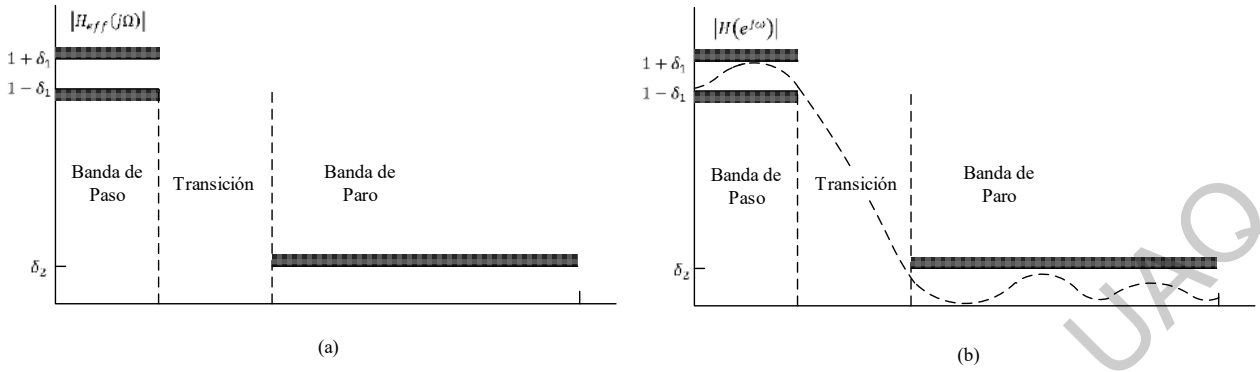

Figura 2.2. a) Especificaciones para la respuesta a la frecuencia efectiva del sistema de la Figura 2.1, en el caso de un filtro pasa-bajas; b) Especificaciones correspondientes para el sistema en tiempo discreto de la Figura 2.1 (Winder, 2002).

Los filtros físicamente realizables tienen una respuesta a la frecuencia que se puede generalizar como se muestra en la Figura 2.2, que es un ejemplo de filtro pasa-bajas donde se puede observar que la banda de paso, que va desde $\mathbf{0}$ hasta $\boldsymbol{\omega_p}$, el *ancho de banda del filtro*, presenta un rizo que, en la mayoría de las aplicaciones, es tolerable. De la misma manera, se produce un rizo en la *banda de paro* (desde $\boldsymbol{\omega_s}$ hasta $\boldsymbol{\pi}$) e incluso pueden no llegar estrictamente a cero pero si tener una atenuación suficiente en la señal de salida.

La sección de las frecuencias entre la banda de paso y la banda de paro se le conoce como banda de transición o *región de transición* que se puede definir como $\boldsymbol{\omega_s - \omega_p}$.

El rizo que exista en la banda de paro se denota con la literal $\boldsymbol{\delta_1}$, lo que nos indica que la magnitud que tenga la función $|H(\omega)|$ varía entre los límites $\mathbf{1} \pm \boldsymbol{\delta_1}$ y, de manera análoga, para la banda de paro con $\boldsymbol{\delta_2}$. Cabe resaltar que para representar esto en una tabla logarítmica, como generalmente se usa, es necesario multiplicar este factor por $20 \log_{10}$.

Para poder realizar cualquier filtro digital es necesario especificar cuatro cosas:

1) El máximo rizo tolerable en la banda de paso δ_1.
2) El máximo rizo tolerable en la banda de paro δ_2.
3) El límite de la banda de paso ω_p.
4) El límite de la banda de paro ω_s.

Entonces, basándose en estas especificaciones, se pueden seleccionar los parámetros $\{\boldsymbol{a_k}\}$ y $\{\boldsymbol{b_k}\}$ que van en la ecuación de la respuesta la frecuencia descrita en la ecuación (7). Y, hay que tener en cuenta,

que el grado de aproximación de la función de salida a las especificaciones depende en parte del criterio empleado en la selección de dichos coeficientes y a la cantidad M y N de coeficientes usados.

Como ya se mencionó, en este trabajo se empleará una técnica que se basa en el uso de un filtro analógico para diseñar un filtro digital tipo IIR. Un filtro analógico está descrito por su función de sistema.

$$H_a(s) = \frac{B(s)}{A(s)} = \frac{\sum_{k=0}^{M} \beta_k s^k}{\sum_{k=0}^{N} \alpha_k s^k} \quad (8)$$

Donde $\{\boldsymbol{\alpha_k}\}$ y $\{\boldsymbol{\beta_k}\}$ son los coeficientes del filtro. También se puede expresar por su respuesta al impulso, la cual está relacionada a $H_a(s)$ por la transformada de Laplace.

$$H_a(s) = \int_{-\infty}^{\infty} h(t) e^{-st} dt \quad (9)$$

Alternativamente, teniendo la función de transferencia racional $H_a(s)$ dada por (1), el filtro analógico puede ser descrito de manera de ecuación lineal con coeficientes-constantes:

$$\sum_{k=0}^{N} \alpha_k \frac{d^k y(t)}{dt^k} = \sum_{k=0}^{M} \beta_k \frac{d^k x(t)}{dt^k} \quad (10)$$

Donde $\boldsymbol{x(t)}$ es la señal de entrada y $\boldsymbol{y(t)}$ la señal de salida del filtro.

Como regla, se dice que un sistema invariante en el tiempo analógico con una función del sistema $H(s)$ es estable sí todos sus polos se encuentran en la mitad izquierda del plano s. Consecuentemente, para que la técnica de conversión sea efectiva, las siguientes características son deseables:

1. El eje $j\Omega$ del plano s debe trazar dentro del círculo unitario del plano z. De este modo, habrá una relación directa entre las dos variables de frecuencia en los dos dominios.
2. La mitad izquierda del plano s debe trazarse dentro del círculo unitario en el plano z. De esta manera, un filtro analógico estable será convertido a un filtro digital estable.

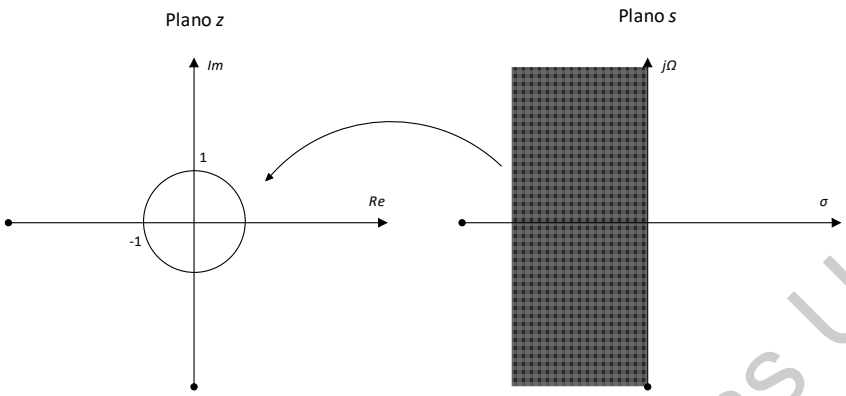

Figura 2.3 Mapeo debe llevar la mitad izquierda del plano s a puntos dentro del círculo unitario del plano z

Recordemos que un filtro de fase lineal debe tener una función del sistema que satisfaga la condición:

$$H(z) = \pm z^{-N} H(z^{-1}) \quad (11)$$

Donde z^{-1} representa un retraso en las unidades N en el tiempo, lo que significaría que el filtro poseería un polo en espejo reflejado fuera del círculo unitario por cada polo dentro del círculo unitario, lo que conlleva a la inestabilidad. Por lo tanto, un filtro causal y estable no puede tener fase lineal.

En el diseño de filtros IIR digitales, debemos especificar las características del filtro de acuerdo a la respuesta en la magnitud únicamente ya que el comportamiento de la fase es consecuencia del mismo.

La ecuación de la de la respuesta a la frecuencia lineal $H(\omega)$ puede ser convertida a su equivalente digital usando la transformación del impulso invariante, del escalón invariante o la transformación bilineal. Sin embargo, únicamente la transformada bilineal provee una conversión de propósito general que puede ser usada para generar respuestas de tipo pasa-bajas, pasa-banda, pasa-alta y alto de banda. Las conversiones de impulso y de escalón invariante son difíciles de aplicar y únicamente pueden ser usadas para generar filtros pasa-bajas (aunque también pasa-banda pero con cierto cuidado); estas funciones de conversión no pueden ser empleadas con respuestas pasa-alta o rechaza-banda. Por esta razón, únicamente la transformada bilineal será empleada.

La transformada Z juega el mismo papel en el análisis de señales de tiempo discreto y sistemas LTI como lo hace la transformada de Laplace en el análisis de señales de tiempo continuo.

La transformada z de una señal de tiempo discreto $x(n)$ se define como la sumatoria:

$$X(z) = \sum_{n=\infty}^{\infty} x(n)\, z^{-n} \tag{12}$$

Por conveniencia, la transformada z de una señal de tiempo discreto $x(n)$ se denota como:

$$X(z) \equiv Z\{x(n)\} \tag{13}$$

Tomando en cuenta la siguiente relación.

$$z = e^{\frac{sT}{2}} \tag{14}$$

La sustitución directa en un sistema digital conlleva a un comportamiento no-lineal. Por lo que para linealizar el sistema es necesario hacer la siguiente expansión.

$$e^{x} = \sum_{n=0}^{\infty} \frac{x^n}{x!} \tag{15}$$

Por lo que para $z = e^{\frac{sT}{2}}$ la serie de Taylor quedaría como sigue.

$$z = e^{\frac{sT}{2}} = 1 + \frac{sT}{1} + \frac{(sT)^2}{2} + \frac{(sT)^3}{6} + \cdots \tag{16}$$

Entonces, si únicamente tomamos los dos primeros términos, de orden más alto, se consigue una aproximación lineal de primer orden para z.

$$z \approx 1 + sT \tag{17}$$

Esta aproximación se puede mejorar si se consigue de la forma:

$$z = \frac{a + bs}{c + ds} \tag{18}$$

Observando la siguiente relación

$$2^1 = \frac{2^{\frac{1}{2}}}{2^{-\frac{1}{2}}} = 2^{\frac{1}{2}} \cdot 2^{\frac{1}{2}} = 2^1 \tag{19}$$

Podemos afirmar que:

$$e^{sT} = \left(e^{\frac{sT}{2}}\right)\left(e^{\frac{sT}{2}}\right) = \frac{e^{\frac{sT}{2}}}{e^{-\frac{sT}{2}}} = z \tag{20}$$

Expandiendo esta fracción de nuevo reemplazando tanto denominado como numerador con su serie de Taylor correspondiente:

$$z = \frac{1 + \frac{sT}{2} + \frac{(sT)^2}{8} + \cdots}{1 - \frac{sT}{2} + \frac{(sT)^2}{8} - \cdots} \tag{21}$$

Si únicamente nos quedamos, de igual forma, con los dos primeros términos:

$$z \approx \frac{1+\frac{sT}{2}}{1-\frac{sT}{2}} \quad (22)$$

Lo que se convierte en una aproximación de primer orden más exacta de $e^{\frac{sT}{2}}$.

Reacomodando:

$$s = \frac{1}{T}\ln(z) \quad (23)$$

Desarrollando:

$$s = \frac{2}{T}\left(\frac{z-1}{z+1} + \frac{1}{3}\left(\frac{z-1}{z+1}\right)^3 + \frac{1}{5}\left(\frac{z-1}{z+1}\right)^5 + \cdots\right) \quad (24)$$

Tomando el límite aproximándose a infinito:

$$s = \frac{2}{T}\left(\frac{z-1}{z+1}\right) \quad (25)$$

La transformación bilineal convierte la función de transferencia de un filtro analógico a una función del sistema digital haciendo la sustitución de la ecuación (25) con la ecuación (4).

$$s = \frac{2}{T}\left(\frac{1-z^{-1}}{1+z^{-1}}\right) \quad (26)$$

Si el filtro analógico es estable, la transformación bilineal resultará en un filtro digital estable. La función de transferencia proveniente del filtro analógico prototipo $H_a(s)$, en general, es el cociente de dos polinomios en el dominio *s*. Por lo tanto, la función del sistema $H(z)$ obtenida, generalmente contiene varias potencias de la división $1 - z^{-1}/1 + z^{-1}$ obtenidas por la potencia más alta de $1 + z^{-1}$ tanto en el denominador como en el numerador. Estos términos sirven para obtener $H(z)$ como el cociente de polinomios en z^{-1} de la forma (Ecuación (27)):

$$H(z) = \frac{\sum_{k=0}^{M} a_k z^{-k}}{\sum_{k=1}^{N} b_k z^{-k}} \quad (27)$$

De esta manera, a_k y b_k serán empleadas en la estructura de la Figura 2.5 para realizar el filtro digital. Dicho filtro procesa las señales en el dominio del tiempo como se muestra en la ecuación (6).

$$y[n] = \sum_{m=0}^{M} b_m x[k-m] - \sum_{n=1}^{N} a_n y[k-n] \quad (28)$$

Para profundizar un poco en las características de la transformación bilineal, establezca:

$$z = re^{j\omega} \quad (29)$$

$$s = \sigma + j\Omega \quad (30)$$

La ecuación (25) puede ser expresada como:

$$s = \frac{2}{T}\frac{z-1}{z+1} \quad (31)$$

$$s = \frac{2}{T}\frac{re^{j\omega}-1}{re^{j\omega}+1} \quad (32)$$

$$\sigma + j\Omega = \frac{2}{T}\left(\frac{r^2-1}{1+r^2+2r\cos\omega} + j\frac{2r\sin\omega}{1+r^2+2r\cos\omega}\right) \quad (33)$$

Por regla de la correspondencia:

$$\sigma = \frac{2}{T}\frac{r^2-1}{1+r^2+2r\cos\omega} \quad (34)$$

$$\Omega = \frac{2}{T}\frac{2r\sin\omega}{1+r^2+2r\cos\omega} \quad (35)$$

Si $r < 1$ entonces $\sigma < 0$, si $r > 1$ entonces $\sigma > 0$. De esta manera, la mitad izquierda del plano s corresponde al interior del círculo unitario del plano z, mientras que la mitad del plano derecho del plano s se mapea fuera de dicho círculo unitario. Cuando si $r = 1$, $\sigma = 0$ entonces:

$$\Omega = \frac{2}{T}\frac{2\sin\omega}{1+\cos\omega} \quad (36)$$

$$\Omega = \frac{2}{T}\tan\frac{\omega}{2} \quad (37)$$

Reacomodando:

$$\omega = \tan^{-1}\frac{\Omega T}{2} \quad (38)$$

La relación de la ecuación (38) entre las variables de frecuencia en los dos dominios se ilustra en la Figura 2.4. Se observa que el rango de Ω se extiende únicamente dentro del rango de $-\pi \leq \omega \leq \pi$. Sin embargo, el mapeo es altamente no lineal comportándose como una compresión de la gráfica o una deformación que es debida a la no linealidad de la función arco-tangente.

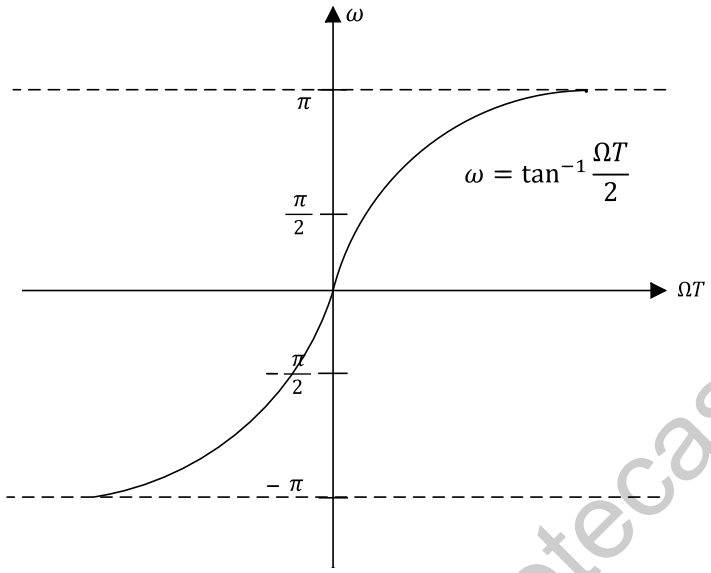

Figura 2.4. Mapeo de la variación de frecuencia ω y Ω resultado de la transformación bilineal.

Una característica de la transformación bilineal es que mapea el punto $s = \infty$ en $z = -1$. Por consecuencia, el filtro pasa-bajas, el cual tendría un cero en $s = \infty$, resultaría en un filtro digital con un cero en $z = -1$.

Usualmente el diseño de filtros digitales comienza con especificaciones en el dominio digital, las cuales involucran la variable de frecuencia ω. Usando la relación de la ecuación (37) se pueden convertir estas especificaciones al dominio analógico. A través de la transformada bilineal en la ecuación (26) el filtro digital que cumple estas especificaciones se puede convertir a un filtro digital. En este procedimiento, el parámetro T puede ser establecido en cualquier valor arbitrario.

Un filtro IIR básico se genera a partir de la estructura bicuadrada, la cual se muestra en la Figura 2.5. Estructura bicuadrada de segundo orden (Winder, 2002). Los elementos de retraso están denotados por $1/z$ en este diagrama. Dichos términos algunas veces son escritos como z^{-1}, especialmente en ecuaciones de función de transferencia.

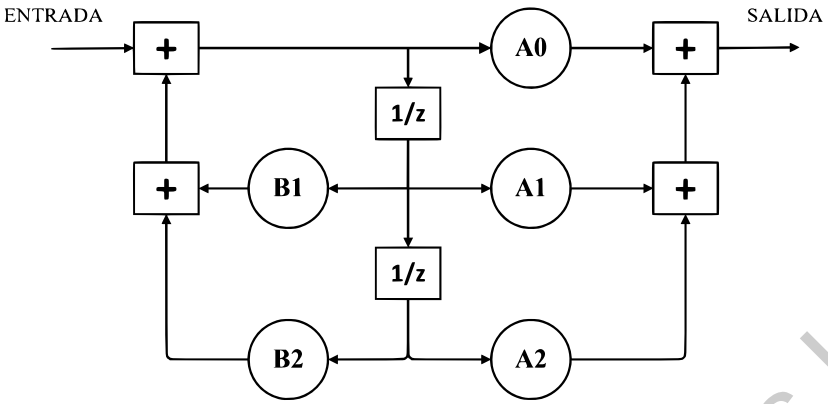

Figura 2.5. Estructura bicuadrada de segundo orden (Winder, 2002).

Dado que un filtro analógico bicuadrado puede desempeñar la función de un filtro pasa-bajas, pasa-banda, pasa-altas o rechaza-banda, también un filtro digital puede lograrlo. Un filtro bicuadrado usa cuatro bloques de suma, dos de retraso y cuatro multiplicadores. Los coeficientes de multiplicación son $A0, A1, A2, B1$ y $B2$. Dichos coeficientes son calculados durante el proceso de diseño. La función de transferencia para el filtro bicuadrado es como sigue a continuación (Ecuación (39)):

$$H(z) = \frac{Y(z)}{X(z)} = \frac{A0 + A1z^{-1} + A2z^{-2}}{1 - B1z^{-1} - B2z^{-2}} \quad (39)$$

El elemento de alimentación $A0$ otorga la ganancia de DC y es usualmente unitaria. No existe un elemento de retroalimentación $B0$, el cual es reemplazado por un elemento de valor unitario debido a que la línea de la señal a través de este elemento es hacia delante, no hacia atrás.

Los filtros de orden alto se diseñan colocando etapas bicuadradas. Cada etapa otorga una respuesta de segundo orden; entonces un filtro de cuarto orden emplearía dos etapas bicuadradas en serie. Por lo tanto, si se diera el caso en que un filtro de orden impar fuese requerido, se necesitarían una o más etapas de segundo orden seguidas por una etapa de primer orden. Una etapa de primer orden es simplemente un componente de retraso y un coeficiente de multiplicación, así como se muestra en la Figura 2.6:

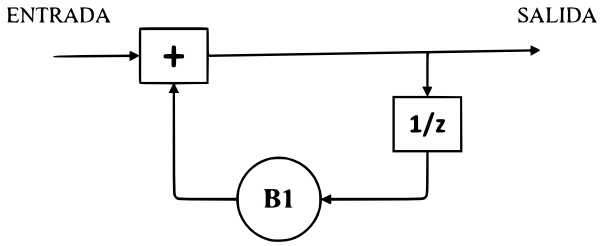

Figura 2.6. Estructura bicuadrada de primer orden.

También se puede ver la estructura bicuadrada de primer orden como una estructura bicuadrada normal de segundo orden con sus coeficientes $A1, A2$ y $B2$ establecidos en cero.

2.2 Señales mioeléctricas

La actividad eléctrica en un músculo se presenta debido a la transición de potenciales iónicos en las unidades motoras activadas, donde unidad motora es el agrupamiento funcional más pequeño en un músculo y se compone de una única neurona motora y muchas fibras de músculo asociadas. Las EMG pueden ser medidas en el músculo o de la superficie de piel que recubre al músculo, es un patrón de interferencia espacio-temporal de actividad eléctrica de las unidades motoras activadas localizadas cerca de las superficies de detección.

Los registros superficiales de las señales mioeléctricas, mejor conocidos como electromiograma (EMG), son señales importantes en el control de prótesis para personas con miembros amputados. Las prótesis mioeléctricas actuales requieren un par de músculos agonista-antagonista para el control de un grado de libertad (DOF: *degree of freedom*) y permitir únicamente un solo movimiento a la vez. Como resultado, dado un número limitado de músculos residuales disponibles posterior a la amputación, los métodos convencionales de control no son ideales para el control intuitivo de prótesis con múltiples DOF. Trabajos previos han demostrado que, usando una técnica de clasificación de patrones, un movimiento intencional puede ser predicho con características distinguibles de un patrón EMG, este nuevo método podría permitir a los usuarios operar una prótesis multifuncional de manera fácil e intuitiva. Algunos de los problemas más importantes al respecto, como el desempeño de varios algoritmos de reconocimiento entre otros ya han sido investigados. De cualquier manera, dos tópicos importantes, el acondicionamiento de la señal EMG y el rango de muestreo permanecen sin ser investigados.

2.2.1 La función de transferencia en el plano *s*

Los filtros tienen una respuesta dependiente de la frecuencia debido a que la impedancia de un capacitor o un inductor cambia con la frecuencia. Por consiguiente, las impedancias complejas son como sigue:

$$Z_L = sL \tag{40}$$

$$Z_C = \frac{1}{sC} \tag{41}$$

Las ecuaciones (40) y (41) son usadas para describir la impedancia de un inductor y un capacitor, respectivamente:

$$s = \sigma + j\omega \tag{42}$$

Donde σ is la frecuencia de Neper en [NP/s] y ω es la frecuencia angular en [rad/s].

$$N_P = \ln\left(\frac{V_1}{V_2}\right) \tag{43}$$

Usando técnicas de análisis de circuitos estándar, como la ley de Ohm, las leyes de Kirchhoff para voltaje y corriente y superposición, la ecuación de transferencia puede ser desarrollada.

Asumiendo un circuito RLC como el de la Figura 2.7, usando un divisor de voltaje puede ser demostrado que el voltaje a través de la resistencia es:

$$H(s) = \frac{Vo}{Vin} = \frac{RCs}{LCs^2 + RCs + 1} \tag{44}$$

El grado del denominador es el orden del filtro, cada polo proveerá una respuesta de -20[dB/Dec]. Para asegurar la estabilidad, todos los polos deben estar en el lado izquierdo del plano. Si se tiene un cero en el origen, eso es un cero en el denominador, el filtro no tendrá respuesta en corriente directa (pasa-altas o pasa-bandas).

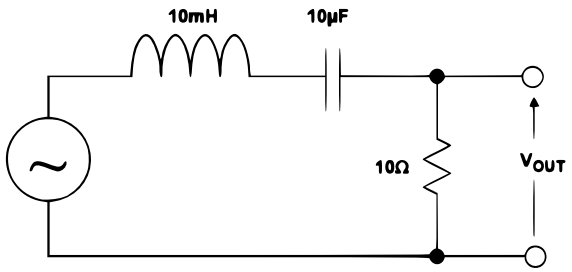

Figura 2.7. Circuito RLC.

Sustituyendo los valores de los componentes en la ecuación se obtiene:

$$H(s) = 10^3 \left(\frac{s}{s^2 + 10^3 s + 10^7}\right) \quad (45)$$

Factorizando la ecuación y normalizando:

$$H(s) = 10^3 \left(\frac{s}{[s - (-0.5 + j3.122)10^3][s - (-0.5 - j3.122)10^3]}\right) \quad (46)$$

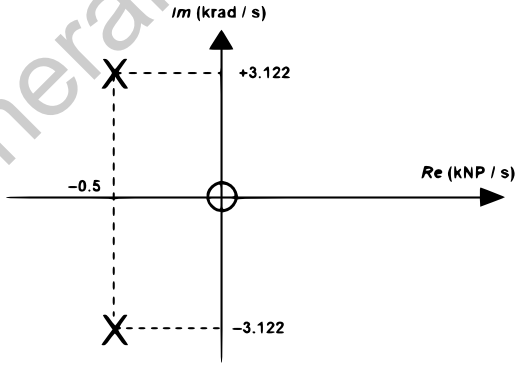

Figura 2.8. Polos y ceros graficados en el plano s.

Como se puede apreciar en la Figura 2.8, se obtuvieron un cero en el origen y un par de polos conjugados en:

$$s = (-0.5 \pm j3.122)10^3 \quad (47)$$

Existen cinco parámetros prácticos de un filtro que se encuentran definidos como sigue a continuación y se pueden observar en la Figura 2.9.

La frecuencia de corte f_c es la frecuencia a la cual la respuesta del filtro deja la banda de error (o el punto a -3[dB], por ejemplo en un filtro Butterworth). La frecuencia de la banda de paro f_s es la frecuencia a la cual la mínima atenuación en la banda de paro es alcanzada. El rizo de la banda de paso A_{max} es la variación (banda de error) en la respuesta de la banda de paso. La mínima atenuación de la banda de paso A_{min} define la mínima atenuación de la señal en la banda de paro. El orden M es también el número de polos en la función de transferencia.

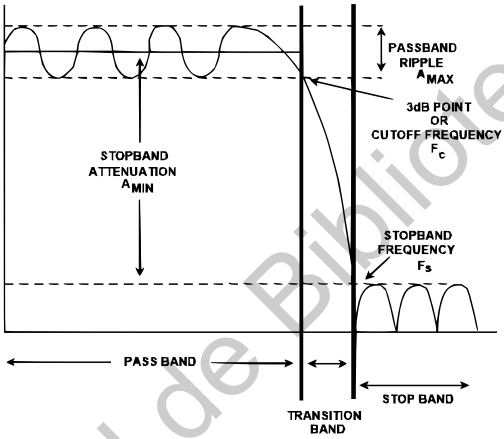

Figura 2.9. Parámetros clave de un filtro.

También tenemos un concepto llamado factor de calidad dado por Q, también expresado algunas veces como $\alpha = \frac{1}{Q}$; el índice de amortiguamiento ξ se puede ver como $\xi = 2\alpha$. Reescribiendo la función de transferencia H(s) en términos de ω_0 y Q:

$$H(s) = \frac{H_0}{s^2 + \frac{\omega_o}{Q} + {\omega_o}^2} \quad (48)$$

Donde H_0 es la ganancia de pasa banda y $\omega_o = 2\pi f_0$. Este es ahora el prototipo de pasa-baja que será empleado en el diseño de filtros.

2.2.1.1 Filtro pasa-altas

Cambiando el numerador de la ecuación de transferencia $H(s)$ del prototipo del pasa-bajas a $H_0 s^2$, transforma el filtro pasa-bajas en un filtro pasa altas. La respuesta del filtro pasa-altas es similar en forma

a un filtro pasa-bajas, simplemente invirtiendo la frecuencia. Entonces, la función de transferencia de un filtro pasa-alta es:

$$H(s) = \frac{H_0 s^2}{s^2 + \frac{\omega_o}{Q} + {\omega_o}^2} \quad (49)$$

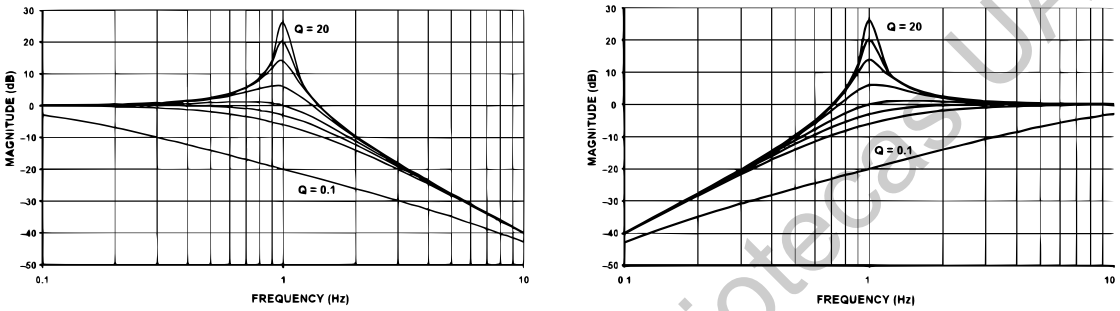

Figura 2.10. a) pico contra Q de un filtro pasa-bajas de orden dos; b) pico contra Q de un filtro pasa-altas de orden dos

La Figura 2.10 muestra que si Q es > que 0.707, habrá algunos picos en la respuesta del filtro. Si la Q es < que 0.707, el *rolloff* a la f_0 será más grande. También tendrá una caída más suave y empezará antes.

2.2.1.2 Filtro pasa-banda

Cambiando el numerador del prototipo de pasa-bajas a ${H_0\omega_0}^2$ convertirá al filtro a una función pasa-banda. De la cual la función de transferencia está dada por:

$$H(s) = \frac{{H_0\omega_0}^2}{s^2 + \frac{\omega_o}{Q} + {\omega_o}^2} \quad (50)$$

ω_0 aquí es la frecuencia ($f_0 = 2\pi\omega_0$) a la cual la ganancia del filtro tiene un pico.

H_0 es la ganancia del circuito definida como $H_0 = \frac{H}{Q}$

Q tiene un significado particular para la respuesta del filtro pasa-banda. Es la selectividad del filtro definida como:

$$Q = \frac{f_0}{f_H - f_L} \quad (51)$$

Donde f_H y f_L son las frecuencias donde la respuesta es -3[dB] el valor de la máxima.

El ancho de banda (BW) de un filtro de este tipo esta descrito como sigue y puede ser demostrado que la frecuencia de resonancia (f_0) se encuentra en el centro geométrico de f_L y f_H,

$$BW = f_H - f_L \tag{52}$$

Lo cual significa que f_0 aparecerá a medio camino entre f_L y f_H en una escala logarítmica.

$$f_0 = \sqrt{f_H f_L} \tag{53}$$

La Figura 2.11 muestra la respuesta en frecuencia del filtro pasa banda vs Q.

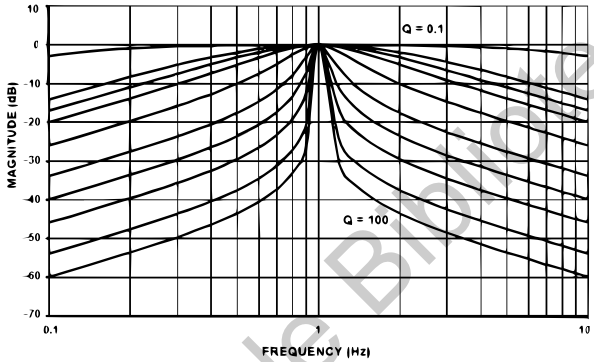

Figura 2.11. Filtro pasa-banda vs Q.

2.2.1.3 Filtro rechazo de banda

Cambiando el numerador a $s^2 + \omega_z^2$, se convierte el filtro en un rechaza-banda o *Notch.* Para facilitar los términos se hablará de un filtro Notch cuando se refiera a uno con un rechazo de banda muy estrecho, mientras que un rechaza-banda será uno con un espectro de rechazo muy largo. La ecuación (54) describe este filtro y la Figura 2.12 muestra su respuesta en frecuencia.

$$H(s) = \frac{H_0(s^2 + \omega_z^2)}{s^2 + \frac{\omega_o}{Q} + {\omega_o}^2} \tag{54}$$

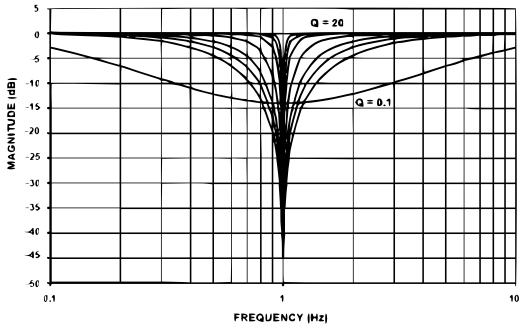

Figura 2.12. Amplitud del filtro notch comparado con varios valores de Q.

2.2.1.4 Twin T Notch

Este filtro es ampliamente usado como un circuito Notch de propósito general como se muestra en la Figura 2.13. La implementación pasiva del circuito (sin retro-alimentación) tiende a tener un Q que se acerque a 0.25. Esta consideración puede ser rectificada con la aplicación de retro-alimentación positiva al nodo de referencia. La cantidad de señal de retro-alimentación, establecida por el cociente de $\frac{R_4}{R_5}$, determinará el valor de Q del circuito, lo cual determina la profundidad del Notch. Para la máxima profundidad, las resistencias R_4 y R_5 y el op-amp asociado deberán ser eliminados. En este caso, la unión de C_3 y R_3 serán conectadas directamente a la salida.

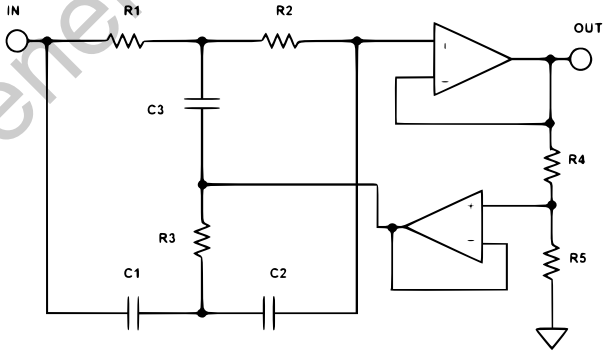

Figura 2.13. Filtro *Twin T Notch.*

La sintonización no se alcanza fácilmente. Usando componentes estándar de 1%, un Notch de 60 [dB] es tan bueno como se esperaría, con 40 [dB] a 50 [dB] siendo el más típico. La relación entre las variables del circuito se describe en la Tabla 2-1.

$$\frac{V_o}{V_{in}} = \frac{s^2 + \frac{1}{RC}}{s^2 + \frac{4}{RC}\left(1 - \frac{R_5}{R_4 + R_5}\right)s + \frac{1}{RC}} \quad (55)$$

Tabla 2-1. Condiciones de diseño para un filtro Twin T Notch.

C	R'
$k = 2\pi f_0 C$	$R_4 = (1-k)R'$
$R = \frac{1}{k}$	$R_5 = KR'$
$R = R_1 = R_2 = 2R_3$	$k = 1 - \frac{1}{4Q}$
$C = C_1 = C_2\frac{C_3}{2}$	Para K = 1, eliminar R_4 y R_5.
$f_0 = \frac{1}{2\pi RC}$	Para $R \gg R_4$, eliminar el búfer.

2.2.1.5 Bainter Notch

Un filtro Notch simple es el circuito Bainter como se muestra en la Figura 2.14. La Q de un notch no está basada en el conjunto de componentes como en todas las otras configuraciones, en lugar de eso es dependientes de las ganancias de los amplificadores. Por lo tanto, la profundidad del Notch no varía con la temperatura, humedad y otros factores medioambientales. La frecuencia del Notch puede desplazarse, pero no la profundidad. La ecuación (56) describe su comportamiento y la Tabla 2-2 describe las condiciones de diseño del filtro Bainter Notch.

Un amplificador con lazo de ganancia abierto de 10^4 ofrecerá una $Q_Z > 200$. Es capaz de sintonización ortogonal con una mínima interacción. R_6 Sintoniza a Q y R_1 a ω_Z. Variando R_3 establece el cociente de $\frac{\omega_0}{\omega_Z}$ que produce un notch pasa-baja ($R_4 > R_3$), notch ($R_4 = R_3$) o notch pasa-altas ($R_4 < R_3$).

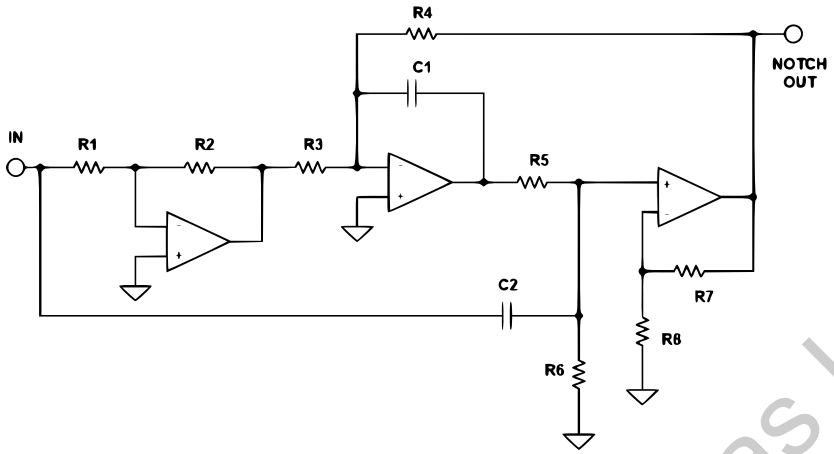

Figura 2.14. Filtro Bainter Notch.

$$\frac{V_{out}}{V_{in}} = \frac{K_2\left[s^2 + \frac{k_1}{R_3R_5C_1C_2}\right]}{s^2 + \left(\frac{R_5 + R_6}{R_5R_6C_2}\right)s + \frac{k_2}{R_4R_5C_1C_2}} \quad (56)$$

Tabla 2-2. Condiciones de diseño para un filtro Bainter Notch.

$\boldsymbol{C_1, R_1, R_7, k_1, k_2}$	
$k = 2\pi f_0 C$	$R_4 = \frac{k_2}{2Qk}$
$C_2 = C_1 = C$	$R_5 = R_6 = \frac{2Q}{k}$
$R_2 = k_1 R_1$	$R_5 = (k_2 - 1)R_7$
$Z = \left(\frac{\omega_Z}{\omega_0}\right)^2$	
$R_3 = \frac{k_1}{2ZQk}$	

2.2.1.6 Butterworth

El filtro *Butterworth* tiene el mejor compromiso entre atenuación y respuesta en fase. No tiene rizos en la banda de paso o en la de paro, y debido a esto, a veces es llamado filtro plano. El filtro alcanza esta propiedad a expensas de una región de transición relativamente grande de la banda de paso a la banda de paro, con características de transitorios.

Los polos normalizados del filtro *Butterworth* en el círculo unitario (en el plano s). Las posiciones de los polos están dadas por:

$$-sin\frac{(2K-1)\pi}{2n}+jcos\frac{(2K-1)\pi}{2n},\quad K=1,2,\dots,n$$

Donde K es el número de pares de polos y n es el número de polos.

Los polos están equidistantemente espaciados entre sí en el círculo unitario, lo cual significa que los ángulos entre los polos son iguales.

Dadas las localizaciones de los polos ω_0 y α (o Q) pueden ser determinados. Entonces, estos valores pueden usarse para determinar los componentes del filtro. Las tabas de diseño para filtros pasivos usan filtros normalizados en frecuencia a 1 [rad/s] e impedancia a 1 [Ω]. Estos filtros pueden ser des-normalizados para determinar el valor real de sus componentes. Esto último permite la comparación de las respuestas en el dominio de la frecuencia y/o tiempo de varios filtros en la misma posición.

2.2.1.7 Chebyshev

El filtro *Chebyshev* tiene una región de transición más pequeña que un *Butterworth* del mismo orden, a expensas de rizos en su banda de paso. Estos filtros tienen 0 [dB] de atenuación relativa en C.D. Filtros de orden impar tienen una banda de atenuación que se extiende de 0 [dB] al valor del rizo. Filtros de orden par tienen una ganancia igual al rizo de la banda de paso. El número de ciclos de rizos en la panda de paso es igual al orden del filtro.

Los polos de un filtro *Chebyshev* pueden ser determinados moviendo los polos de un *Butterworth* a la derecha, formando una elipse. Esto se consigue multiplicando la parte real del polo por k_r y la parte imaginaria por k_I.

$$k_r=\sinh(A)$$
$$k_I=\cosh(A)$$
$$A=\frac{1}{n}sinh^{-1}\left(\frac{1}{\varepsilon}\right)$$

Donde n es el orden del filtro y

$$\varepsilon=\sqrt{10^R-1}$$

Donde:

$$R=\frac{R_{dB}}{10},\quad R=\text{rizo en banda de paso en [dB]}$$

Los filtros *Chebyshev* típicamente están normalizados de tal manera que el borde de la banda de rizo está en $\omega_0 = 1$. El ancho de banda de 3 [dB] está dado por:

$$A_{3dB} = \frac{1}{n} cosh^{-1}\left(\frac{1}{\varepsilon}\right)$$

Tabla 2-3. Ancho de banda al ancho de banda de rizo para filtros Chebyshev.

Orden	0.01 [dB]	0.1 [dB]	0.25 [dB]	0.5 [dB]	1 [dB]
2	3.30362	1.93432	1.59814	1.38974	1.21763
3	1.87718	1.38899	1.25289	1.16749	1.09487
4	1.46690	1.21310	1.13977	1.09310	1.05300
5	1.29122	1.13472	1.08872	1.05926	1.03381
6	1.19941	1.09293	1.06134	1.04103	1.02344
7	1.14527	1.06800	1.04495	1.03009	1.01721
8	1.11061	1.05193	1.03435	1.02301	1.01316
9	1.08706	1.04095	1.02711	1.01817	1.01040
10	1.07033	1.03313	1.02194	1.01471	1.00842

2.2.1.8 Bessel

Los filtros *Butterworth* tienen un comportamiento de amplitud y transición bastante bueno. Los filtros *Chebyshev* mejoran en la amplitud de respuesta a expensas de un comportamiento transitorio. El filtro *Bessel* está optimizado para obtener una respuesta transitoria mejorada debido a la linealidad de fase (esto es un retraso constante) en la banda de paso. Esto significa que habrá una respuesta a la frecuencia más pobre (menor discriminación de amplitud).

Los polos de este filtro pueden ser determinados al colocar todos los polos en un círculo y separando su parte imaginaria por $^2/_n$; donde n es el número de polos. Nótese que los polos superior e inferior están distanciados donde el círculo cruza el eje $j\omega$ por $^1/_n$.

2.2.2Comparación de respuestas

Las respuestas de varios filtros, ya sea Bessel, Butterworth y Chebyshev (en este caso para 0.5 [dB] de rizo) serán comparadas. Un filtro de orden 8 es usado como la base de la comparación. Las respuestas han sido normalizadas para una frecuencia de corte de 1[Hz]. Comparando las Figuras Figura 2.15 yFigura

2.16, es fácil observar la compensación en los tipos de respuesta. Moviéndose de Bessel, pasando por Butterworth y terminando en Chebyshev, nótese que la discriminación de amplitud mejora mientras el comportamiento transitorio se vuelve progresivamente pobre.

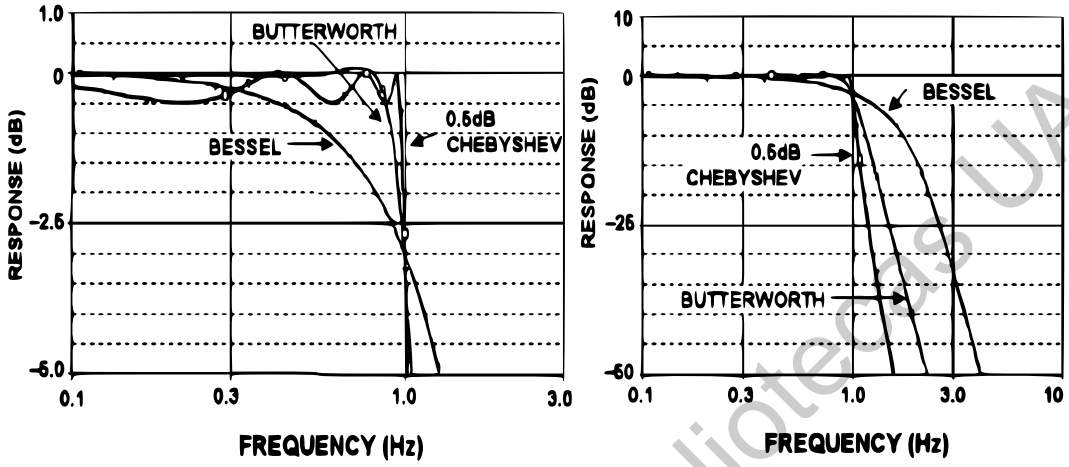

Figura 2.15. Comparación de respuesta a la amplitud de los filtros Bessel, Butterworth y Chebyshev.

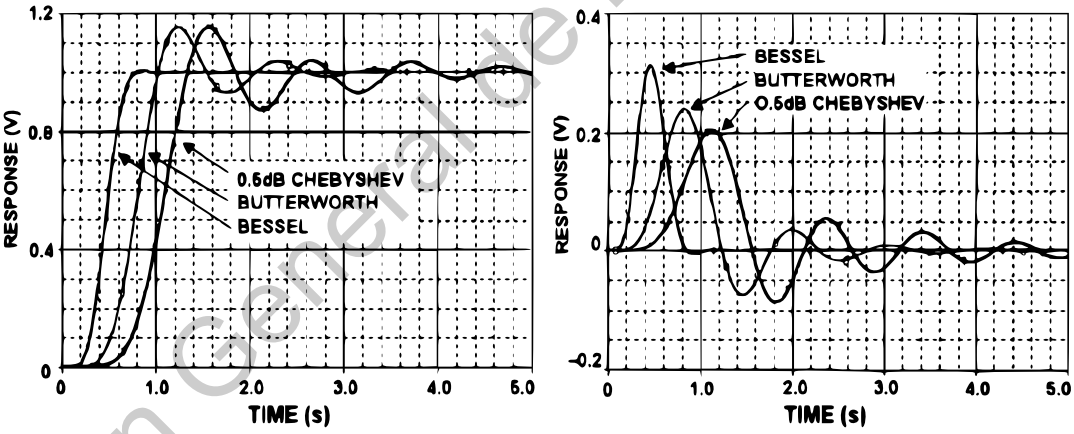

Figura 2.16. Comparación de las respuestas al escalón y al impulso de un filtro Bessel, Butterworth y Chebyshev.

2.2.2.1 Sallen-Key

La configuración Sallen-Key también conocida como fuente de voltaje controlada por voltaje (VCVS), es una de las topologías para filtros más usadas. La configuración del filtro pasa-bajas se muestra en la Figura 2.17, la ecuación (57) describe su comportamiento y la Tabla 2-4 describe las condiciones de diseño del filtro. Una de las razones de su popularidad es que esta configuración muestra la mínima dependencia de comportamiento de filtro en el desempeño de un op-amp. Esto es debido a que el

amplificador operacional está configurado como amplificador, como opuesto de integrador, lo cual minimiza los requerimientos de ganancia-ancho de banda del op-amp. Se infiere que para un op-amp dado, se podrá diseñar un filtro de más alta frecuencia que con otras topologías debido a que el producto de ganancia con el ancho de banda del op-amp no limitará el desempeño del filtro como lo haría si fuera configurado como integrador. La fase de señal a través del filtro se mantiene (configuración no inversora).

Otra ventaja de esta configuración es que la proporción del valor de la resistencia más grande al de la más pequeña con la del capacitor más grande al más pequeño es baja, lo cual es bueno para la posibilidad de manufactura. Las condiciones de frecuencia y de Q son independientes, pero tienen mucha sensibilidad al parámetro de ganancia.

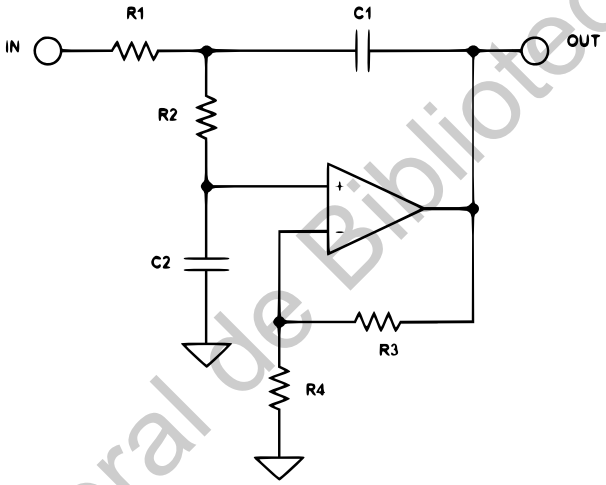

Figura 2.17. Filtro pasa-bajas Sallen-Key.

$$\frac{V_o}{V_{in}} = \frac{H\dfrac{1}{R_1R_2C_1C_2}}{s^2 + \left[\left(\dfrac{1}{R_1}\dfrac{1}{R_2}\right)\dfrac{1}{C_1} + \dfrac{(1-H)}{R_2C_2}\right]s + \dfrac{1}{R_1R_2C_1C_2}} \qquad (57)$$

Tabla 2-4. Condiciones de diseño para un filtro pasa-bajas Sallen-Key.

$\boldsymbol{C_1}$	$\boldsymbol{R_2}$
$k = 2\pi f_0 C_1$	$R_4 = \dfrac{R_3}{H-1}$
$m = \dfrac{\alpha^2}{4} + H - 1$	
$C_2 = mC_1$	

$R_1 = \frac{2}{ak}$	
$R_2 = \frac{\alpha}{2mk}$	

Aunque el filtro pasa-bajas tipo Sallen-Key es ampliamente usado, una desventaja importante es que el filtro no es de fácil sintonización, debido a la interacción de los valores de f_0 y Q.

Por otra parte, también existe la configuración del filtro para el paso de frecuencias altas; la configuración del filtro pasa-altas se muestra en la Figura 2.18, la ecuación (58) describe su comportamiento y la Tabla 2-5 describe las condiciones de diseño del filtro

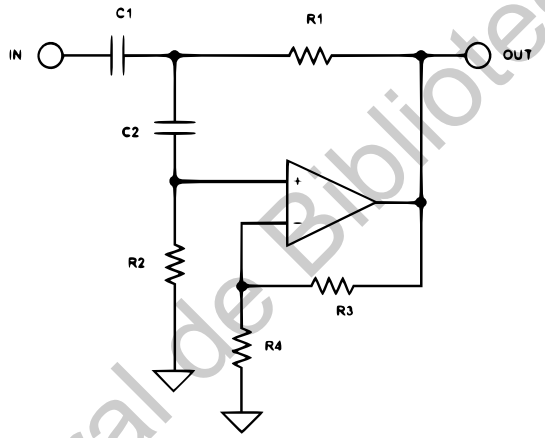

Figura 2.18. Filtro pasa-altas Sallen-Key.

Para transformar el pasa-bajas en pasa-altas simplemente se cambian los capacitores y las resistencias en la red de determinación de la frecuencia (esto es la resistencia de ganancia).

$$\frac{V_o}{V_{in}} = \frac{Hs^2}{s^2 + \left[\frac{\frac{C_2+C_1}{R_2} + (1-H)\frac{C_2}{R_1}}{C_1C_2}\right]s + \frac{1}{R_1R_2C_1C_2}} \quad (58)$$

Tabla 2-5. Condiciones de diseño para un filtro pasa-altas Sallen-Key.

$\boldsymbol{C_1}$	$\boldsymbol{R_3}$
$k = 2\pi f_0 C_1$	$R_4 = \frac{R_3}{H-1}$

$R_1 = \frac{\alpha + \sqrt{\alpha^2 + h - 1}}{4k}$	
$R_2 = \frac{4}{\alpha + \sqrt{\alpha^2 + h - 1}} \frac{1}{k}$	

3 METODOLOGÍA

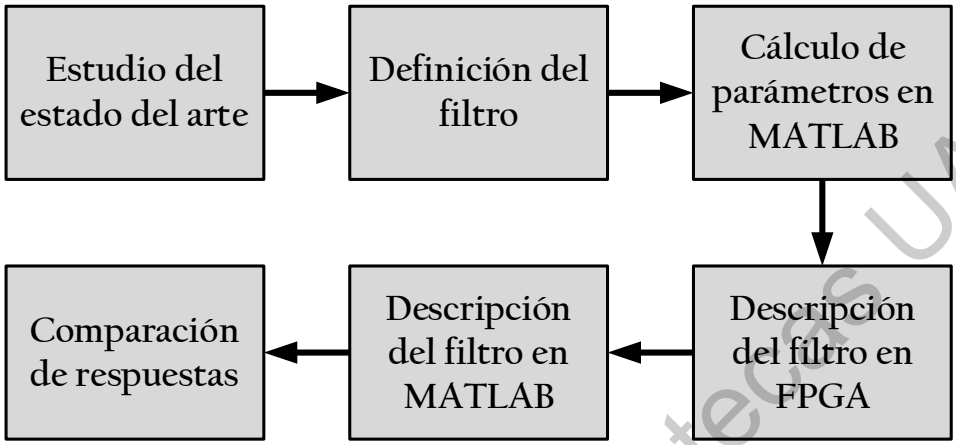

Figura 3.1. Diagrama descriptivo de la metodología utilizada.

La Figura 3.1 describe a grandes rasgos la metodología utilizada para el desarrollo de la tesis. A continuación, se hace una descripción más amplia de la metodología.

El diseño e implementación de filtros digitales, implica una serie de pasos repetitivos, que son la base para la descripción del método propuesto. Para la obtención de los parámetros primero será necesario definir el tipo de respuesta requerido, puede ser una respuesta de tipo pasa-bajas, pasa-altas, pasa-banda, rechazo de banda o Notch. Una vez que se tenga establecido el tipo de respuesta, las especificaciones de diseño efectivas para un filtro en tiempo continuo serán transformadas para un filtro de tiempo discreto a través de las relaciones matemáticas detalladas anteriormente en la sección de fundamentación teórica. La ecuación que establece la respuesta a la frecuencia lineal del filtro será convertida a su equivalente digital usando la transformación bilineal, la cual tiene la cualidad de proveer una función de propósito general para poder ser empleada para los cinco tipos de respuesta posibles.

Para la adquisición de las señales EMG será necesario emplear sensores superficiales los cuales estarán conectados a una tarjeta para una etapa de pre-amplificación y conversión analógica digital. Esta conversión será llevada a cabo en un ADC comunicado directamente con una tarjeta BASYS, la cual suministrará los valores obtenidos muestra por muestra al arreglo de filtrado interno en la FPGA para, posteriormente, ser transferidas a la computadora.

El filtro reconfigurable consiste en un modelo gráfico llamado estructura bicuadrada para el cual se tiene una descripción de hardware en VHDL. Esta descripción es genérica ya que para aumentar el orden

del filtro sólo hace falta poner en cascada ciertos bloques y para aumentar la precisión, aumentar el tamaño de la señal digital (número de bits enteros y decimales). Además, su comportamiento lo dictan los coeficientes del arreglo que se suman y se multiplican con la señal de entrada y la que es retroalimentada al filtro. La programación del algoritmo será más fácil con la ayuda de los programas realizados en MATLAB®, que es la herramienta de software para soporte principal.

Una vez que el diseño de hardware para el tratamiento de la señal deseado sea realizado, se comunicará la parte de la FPGA, que contiene la tarjeta de desarrollo BASYS.

Durante la implementación en la PC, se describirá el programa que tomará los valores provenientes de la FPGA de la señal EMG digitalizada y filtrada. Los resultados serán medidos de acuerdo a la precisión y la velocidad obtenida desde la adquisición de la señal hasta la obtención de los valores digitales filtrados adquiridos.

Las señales serán comparadas contra el resultado obtenido después del filtrado con la FPGA y el arrojado por el acondicionamiento por la PC, con un algoritmo en MATLAB® directamente en la computadora. Esta última comparación incluye también la velocidad de respuesta, ya que la rapidez con la que se realice el filtrado también influye en el resultado deseado que es lograr clasificar las señales EMG adquiridas en tiempo real.

Finalmente, una vez realizado un filtro, si se requieren cambios en el mismo o generar uno nuevo o ser añadido uno nuevo, el procedimiento para dicha tarea será descrito usando las herramientas de software necesarias para que sea lo más rápido y práctico posible. Así el diseñador no tendrá inconvenientes en adaptar su sistema a nuevos requerimientos.

La siguiente lista describe el flujo del proceso de diseño del filtro:

1. Especificación de los parámetros, de acuerdo a la Figura 2.2. a) Especificaciones para la respuesta a la frecuencia efectiva del sistema de la Figura 2.1, en el caso de un filtro pasa-bajas; b) Especificaciones correspondientes para el sistema en tiempo discreto de la Figura 2.1 (Winder, 2002)..
2. Encontrar los coeficientes correspondientes a a_k y b_k del filtro especificado (ecuación (31)), empleando un algoritmo en Matlab®.
3. Simulación del filtro usando una base de datos y un análisis de espectro.

4. Adaptación del diseño en VHDL de filtros digitales para el filtro especificado.
5. Co-Simulación del diseño en VHDL con Matlab® en conjunto con Modelsim®.
6. Implementación del filtro en FPGA.

4 RESULTADOS

4.1 Diseño del filtro

El filtro se diseñó siguiendo los pasos propuestos en la metodología. El filtro elegido es rechaza-banda, de tipo elíptico. Los parámetros calculados para este son:

$f_{p1} = 45[Hz]$, frecuencia de paso 1.

$f_{p2} = 55[Hz]$, frecuencia de *paso* 2.

$f_{s1} = 48[Hz]$, *frecuencia* de paro 1.

$f_{s2} = 52[Hz]$, *frecuencia* de paro 2.

$R_p = 0.5[dB]$, rizo de banda de paso.

$R_s = 50[dB]$, *atenuación* de banda de paro.

Los coeficientes del filtro elíptico, obtenidos con el software de MATLAB®, de acuerdo al *listado 2* del apéndice, se muestran en la Tabla 4-1.

Tabla 4-1. Coeficientes del filtro.

i	**a_i**	**b_i**
1	1	0.8067
2	-6.2237	-5.2292
3	18.2184	15.9365
4	-32.3191	-29.4156
5	37.8494	35.8196
6	-29.9151	-29.4156
7	15.6103	15.9365
8	-4.9374	-5.2292

i	a_i	b_i
9	0.7347	0.8067

Con el listado 1 y 2, de la sección de apéndices, las siguientes gráficas (Figura 4.1Figura 4.2 yFigura 4.3) se obtienen las cuales muestran el comportamiento del filtro, además de que con la función *PrintROM*, del código descrito en dichos listados, se obtiene directamente el código correspondiente al bloque de ROM mismo que contiene los valores de los coeficientes de a_k y b_k para el filtro digital.

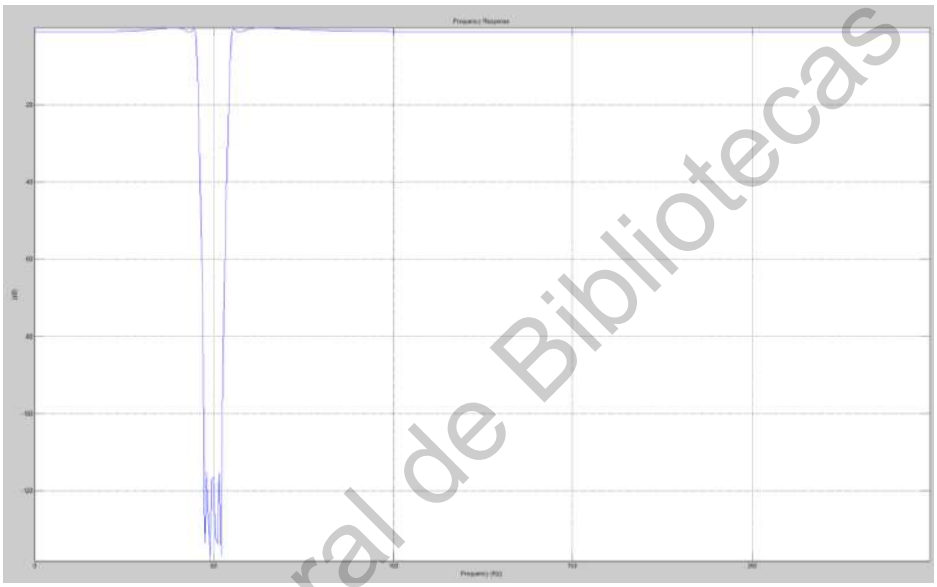

Figura 4.1. Respuesta a la frecuencia de un filtro de orden cuatro elíptico de rechazo de banda en 50 [Hz].

Para corroborar que la respuesta del filtro en el dominio del tiempo seleccionado es correcta para nuestra aplicación, se simula una señal de entrada a una frecuencia relativamente baja con una señal montada de menor magnitud y de 50 [Hz] de frecuencia. Como se muestra en la ecuación (59).

$$x(t) = 10\sin(10\pi t) + 2\sin(100\pi t) \quad (59)$$

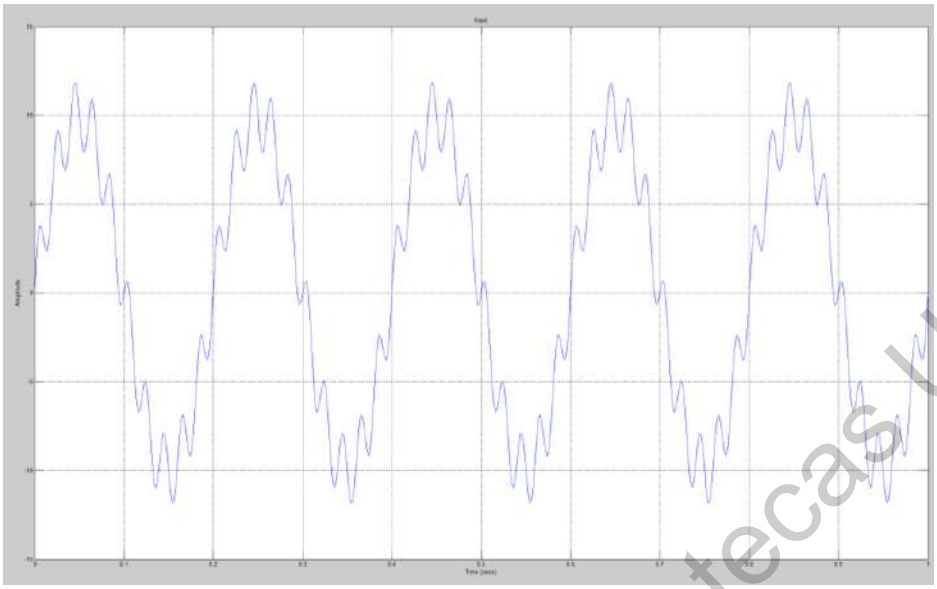

Figura 4.2. Señal de entrada simulada para la ecuación (43).

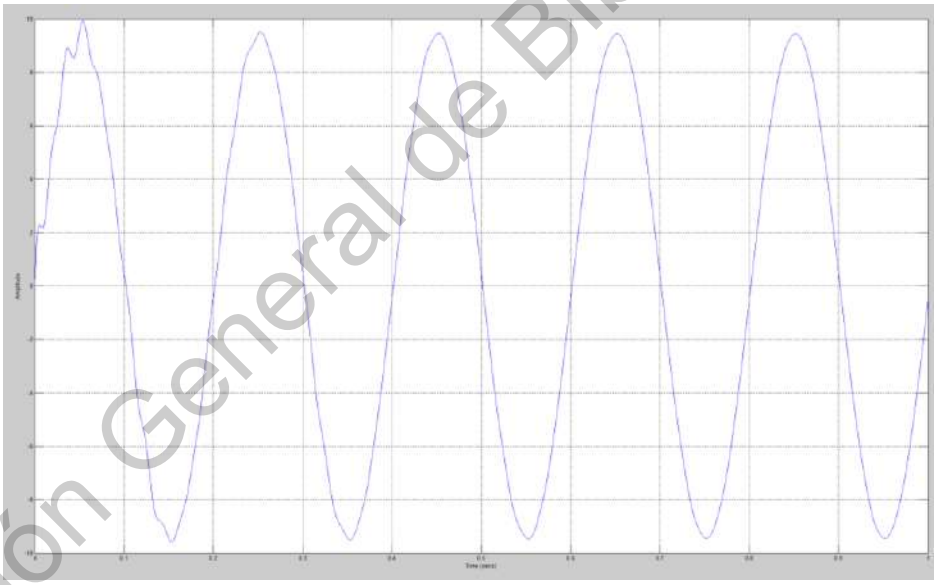

Figura 4.3. Salida obtenida del filtro, usando los coeficientes obtenidos.

Para la simulación del filtro se utilizó una base de datos y un análisis de espectro. Usando el listado 3 del apéndice, las gráficas del análisis de espectro muestran que el filtro es capaz de disminuir la amplitud de los componentes de la señal filtrada alrededor de 50 [Hz]. Una vez ajustado el comportamiento deseado, se procede al diseño en VHDL del filtro digital correspondiente.

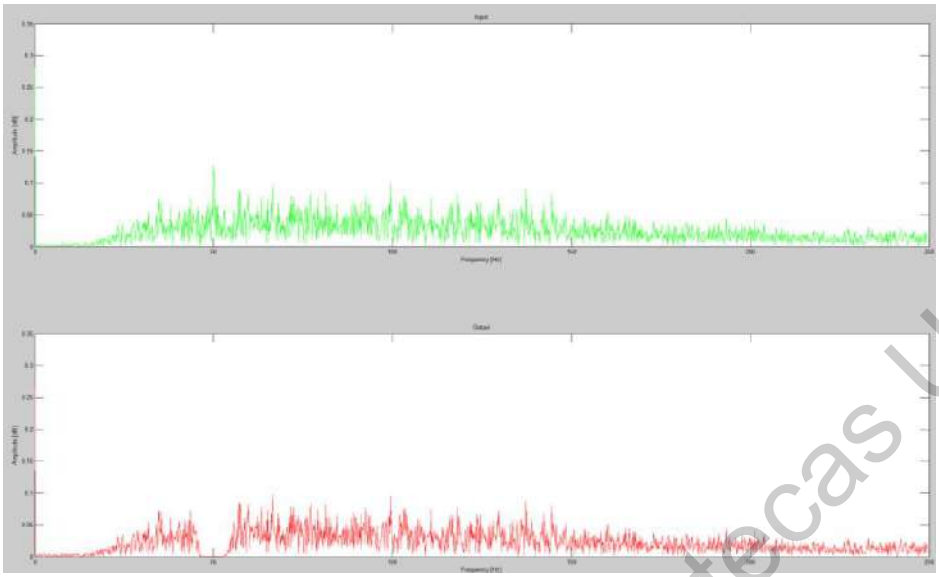

Figura 4.4. Comparación del análisis espectral de la señal de entrada (arriba) con respecto a la señal de salida (abajo). Se puede observar que la atenuación ocurre alrededor de los 50 [Hz].

En la etapa de adaptación del diseño en VHDL para el filtro diseñado se obtuvo el diagrama de la Figura 4.4, muestra la estructura de un filtro IIR usando como base la estructura Bi-Cuadrada descrita en la Figura 4.5. El bus **YK0** es la señal de retroalimentación al propio sistema. Los bloques de retraso están hechos usando Registros de Carga, los cuales almacenan los valores de entrada de la señal (**XIN**) cada vez que la señal **EN** está activa. De esta manera, activando y desactivando dicha señal de manera periódica, se convierte en el controlador de tiempo de muestreo. El número de bloques de Registros de Carga y los estados finitos del Multiplexador depende del número de valores constantes dentro del bloque llamado "ROM_Q" que contiene todos los coeficientes previamente obtenidos de la ecuación (28). Entonces, para el caso del filtro elíptico que contiene diecisiete, el valor forzado en el bloque "Bus Target Counter" o Contador de Bus Objetivo es "10000" en base binaria. Esto es debido a que dicho contador funge como manejador de los estados del Multiplexor y del bloque "ROM_Q" comenzando con el estado cero o elemento cero hasta llegar al dieciseisavo. De esta manera, todos los valores almacenados en los Registros de Carga "Load_Register" que pertenecen tanto a los retardos para la entrada **XIN** como para los valores almacenados de la señal de retroalimentación **YK0** son multiplicados en orden por su coeficiente correspondiente almacenado en el bloque "ROM_Q" siguiendo la función de transferencia para el filtro Bi-Cuadrado de la ecuación (39).

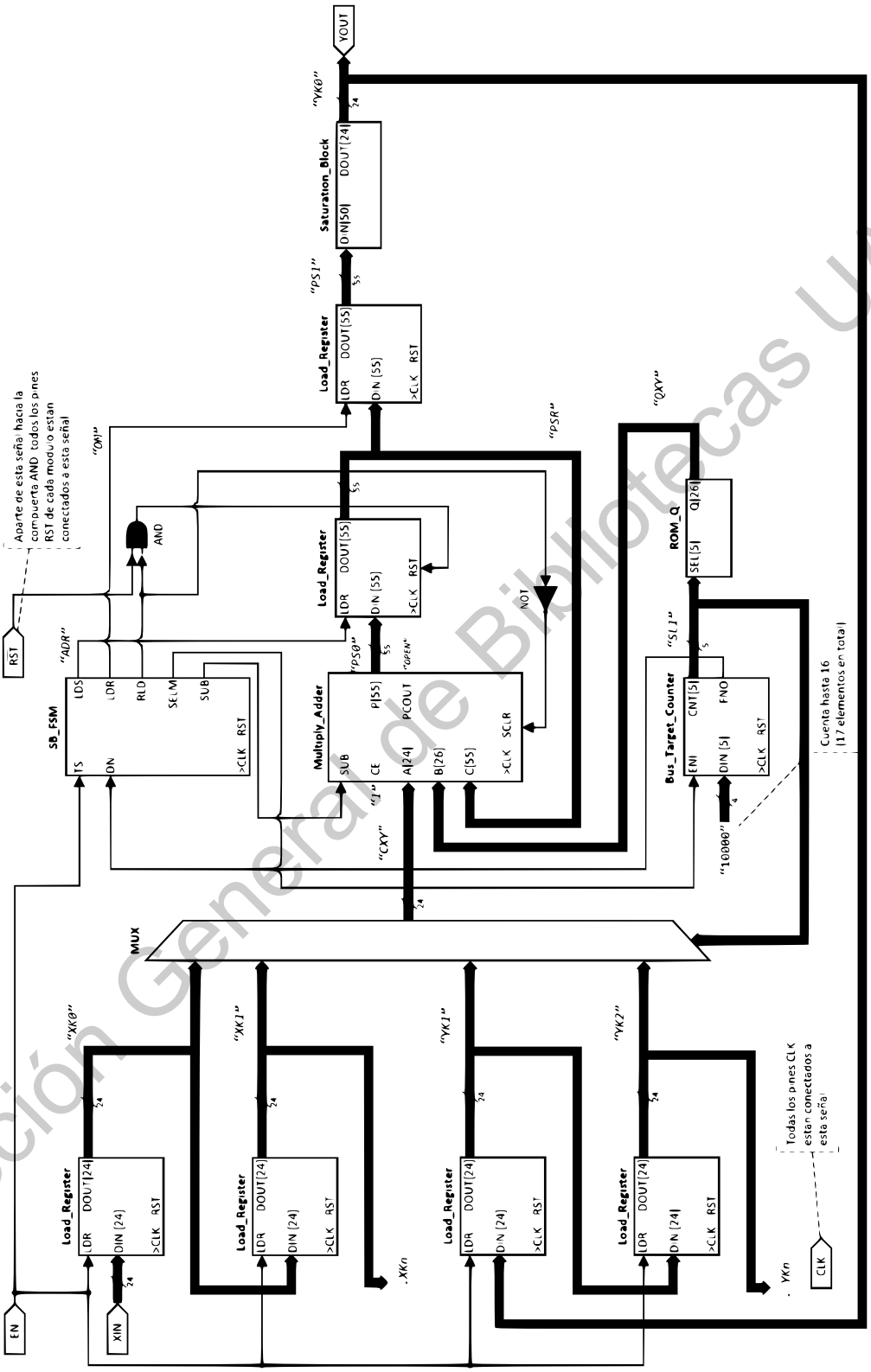

Figura 4.5. Diagrama de más alto nivel del filtro digital IIR.

4.1.1 El Bloque de Multiplicación – Adición

El bloque de multiplicación – adición se muestra en la Figura 4.6, y su diagrama Pipeline la Figura 4.7; que es en la herramienta de diseño Vivado® toma en consideración dos diferentes tiempos de latencia; uno para la operación de multiplicación de las entradas **A** y **B** y su resultado puesto en **P** y otro para **C**/**PCIN** y su salida en **P**. Dichas latencias, son definidas como "A : B – P Latency" y "C- P Latency" como se puede observar en la Figura 4.7.

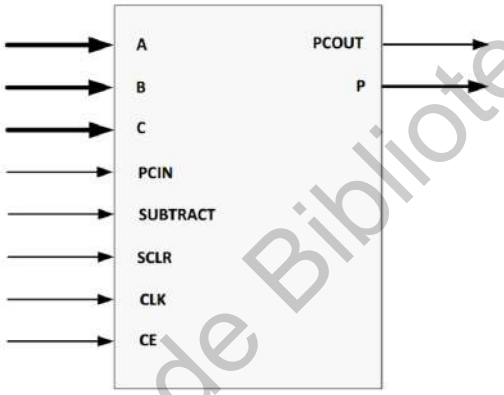

Figura 4.6. Símbolo del bloque de Multiplicación-Adición.

La Tabla 4-2 enlista y define cada una de las señales de entrada y salida que componen el bloque de Multiplicación – adición de la Figura 4.6.

Tabla 4-2. Definición de las señales para la Figura 4.6.

Nombre	Dirección	Descripción
A[N:0]	Entrada	Bus de entrada A (operando multiplicador 1).
B[M:0]	Entrada	Bus de entrada B (operando multiplicador 1).
C[L:0]	Entrada	Bus de entrada C (operando 1 de adición/substracción)
PCIN	Entrada	Entrada en cascada
SUBTRACT	Entrada	Control de Adición/Substracción (Alto = Substracción, Bajo = Adición),
CE	Entrada	Habilitador del reloj (Activo en alto)
CLK	Entrada	Reloj (flanco de subida)
SCLR	Entrada	Limpieza síncrona (activa en alto)
PCOUT (2)	Salida	Salida en cascada

P[Q:0]	Salida	Bus de salida

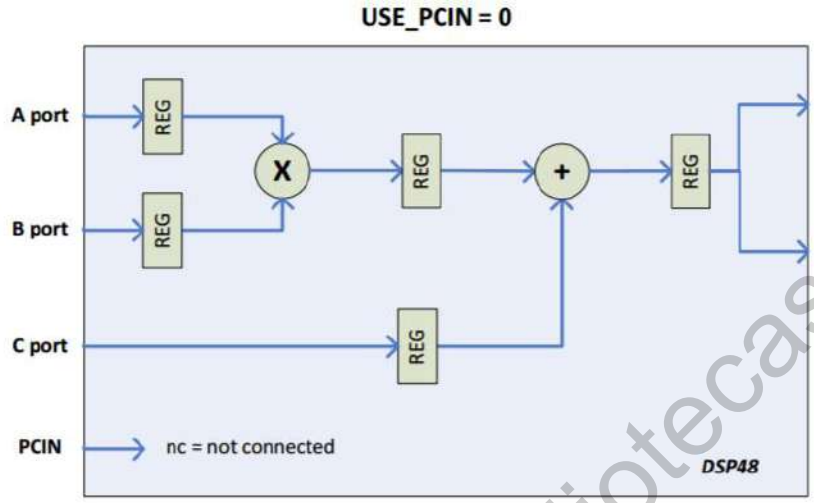

Figura 4.7. Diagrama "Pipeline" del bloque de Multiplicación-Adición.

Todas las entradas son justificadas cuando son pasadas a los operadores dentro del módulo. El usuario debe colocar apropiadamente el LSB (por sus siglas en inglés, Bit Menos Significativo) o MSB (por sus siglas en inglés Bit Más Significativo) relleno o signo extendiendo la entrada (relativo al punto binario) del módulo. En el bloque de Multiplicación-Adición, no hay truncamiento o redondeo a la salida del multiplicador; es un resultado de precisión completa. La entrada **C** es sumada al producto LSB a LSB. El siguiente ejemplo muestra como toman lugar las operaciones dentro del módulo. Las posiciones del MSB y el LSB pueden ser escogidas para extraer la cantidad deseada de datos en la salida. El extracto mostrado en el ejemplo toma el LSB = 0 a el MSB = 11. La configuración utilizada se muestra en la Figura 4.8.

Ejemplo: A * B + C = P

Donde

- El amaño de A es de 6 bits.
- El tamaño de B es de 8 bits.
- El tamaño de C es de 8 bits.

```
      XXXXXX
    * XXXXXXXX
  XXXXXXXXXXXXXX
+       XXXXXXXX
  XXXXXXXXXXXXXX
    |         |
   MSB       LSB
```

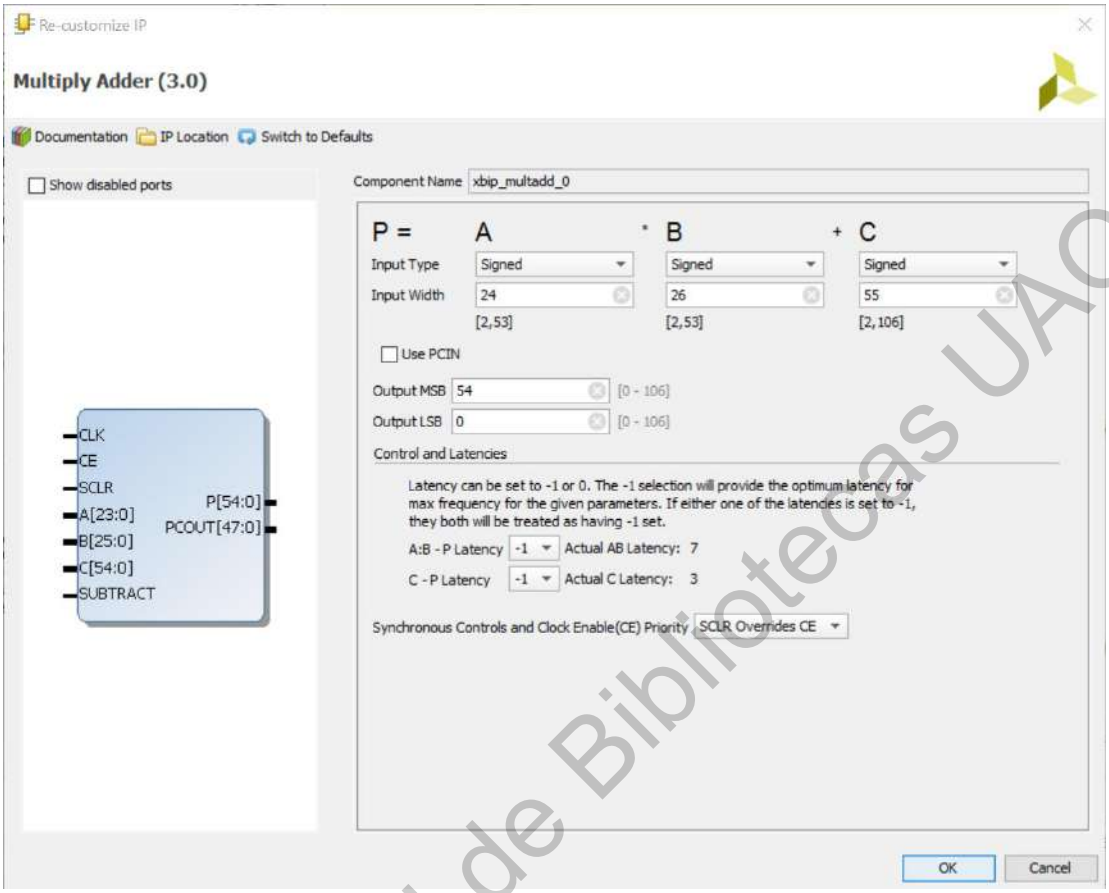

Figura 4.8. Ventana de configuración del bloque de Multiplicación-Adición "Multiply Adder" con los valores usados para este diseño de filtro.

El diseño del filtro mostrado en la Figura 4.5 funciona de la siguiente manera:

1. La máquina de estados finita detecta la señal de habilitación **EN** y comienza a dejar pasar 10 ciclos de reloj, ya que es el tiempo de latencia del bloque de Multiplicación-Adición para terminar su proceso, como se muestra en la Figura 14.
2. Cuando el bloque de Multiplicación-Adición coloca el resultado en la señal de salida **P**, se almacena en un Registro de Carga del cual su bus de salida se convierte en la entrada **C** de dicho bloque de Multiplicación-Adición.
3. La máquina de estados finita "SB_FSM" produce un impulso para el contador "Bus_Target_Counter" de tal manera que, dicho contador, puede incrementar la cuenta producida en su señal de salida **SL1** con lo que cambian los estados actuales del Multiplexor que maneja los retrasos en los registros de carga y del valor actual en ROM que a su vez son

multiplicados por el bloque de Multiplicación-Adición. Por ejemplo, en el caso del primer sub-ciclo, los valores que son multiplicados son el coeficiente b_1 y el valor de entrada expresado como $x(k-1)$. Así, se multiplican y se suman todos los valores almacenados de las señales de entrada como de las de retro-alimentación y usus respectivos coeficientes como lo muestra la ecuación (31).

4. Al término de este proceso, el valor resultante final es almacenado, de igual forma, en un Registro de Carga produciendo la señal **PS1** la cual entra a un bloque de saturación cuyo objetivo es limitar, hasta un cierto valor especificado, la salida. En operación normal, dicho bloque "Saturation_Block" corta el bus en su entrada para producirlo como salida **YOUT** en el formato deseado, que en este caso son 24 bits.
5. Finalmente, el valor de salida es retornado al sistema por medio de **YK0** como retro-alimentación hacia cada uno de los Registros de Carga correspondientes para cada elemento $a_n y(k-n)$.

4.1.2 Co-Simulación del diseño en VHDL con Matlab® en conjunto con Modelsim®.

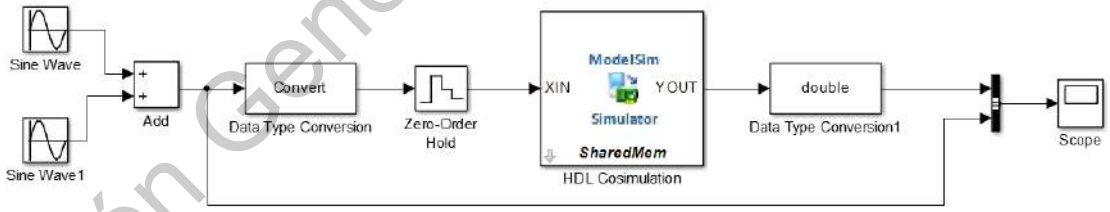

Figura 4.9. Diagrama de Simulink para la co-simulación del HDL usando Modelsim®.

La Figura 4.9 muestra el diagrama usado para la simulación en conjunto del diseño en VHDL usando Modelsim® y Matlab®. Para poder realizar la simulación es necesaria la instalación de la herramienta Model Sim® y una licencia de estudiante para ejecutarlo, para más información acceder al sitio oficial de Mentor Graphics®.

Desde MatLab se accede a ModelSim con el comando “vsim”, luego se añade una nueva biblioteca con todas las opciones por default. Después se añaden los archivos de VHDL, previamente depurados, para ser compilados y para inicializar la simulación el comando es el siguiente:

```
simulink Top_IIR_Filter
```

A continuación, se muestran los comandos para configurar los valores de cada señal de entrada para el Módulo de VHDL:

```
force RST 0 0, 1 20 ns
force CLK 1 0, 0 10 ns -repeat 20 ns
force EN  1 0, 0 20 ns -repeat 2  ms
```

El retenedor de orden cero tiene un tiempo de muestreo de 0.002 [s]. En el buscador de la librería de Simulink se busca el bloque de verificador de HDL y se agrega el nombrado como “HDL Cosimulation”. En Auto relleno, se escribe el nombre del Nivel de más alto nivel del diseño de VHDL. Posteriormente, se seleccionan las señales de interés que, en este caso, son la entrada XIN y la salida YOUT. Es necesario asegurarse de que la señal de salida tenga el mismo período de muestreo y sea con signo. Además, en la sección de Escala de Tiempo, que un segundo en Simulink corresponda a un segundo en el simulador de HDL.

El sistema se tiene que alimentar con el tipo de dato adecuado, por eso se tienen los bloques de conversión de información de la entrada a una conversión de entero de 24 bits con punto fijo y 16 bits para la parte fraccionaria.

Las señales de entrada usadas para confirmar el comportamiento esperado del filtro constan de una señal base con una frecuencia de 5 [Hz] y una señal de menor amplitud montada a 50 [Hz]. Se espera que la señal de entrada que contiene dos elementos de frecuencia, sea limpiada y solo se observe la señal de mayor amplitud a 5 [Hz], como se ve en la Figura 4.10.

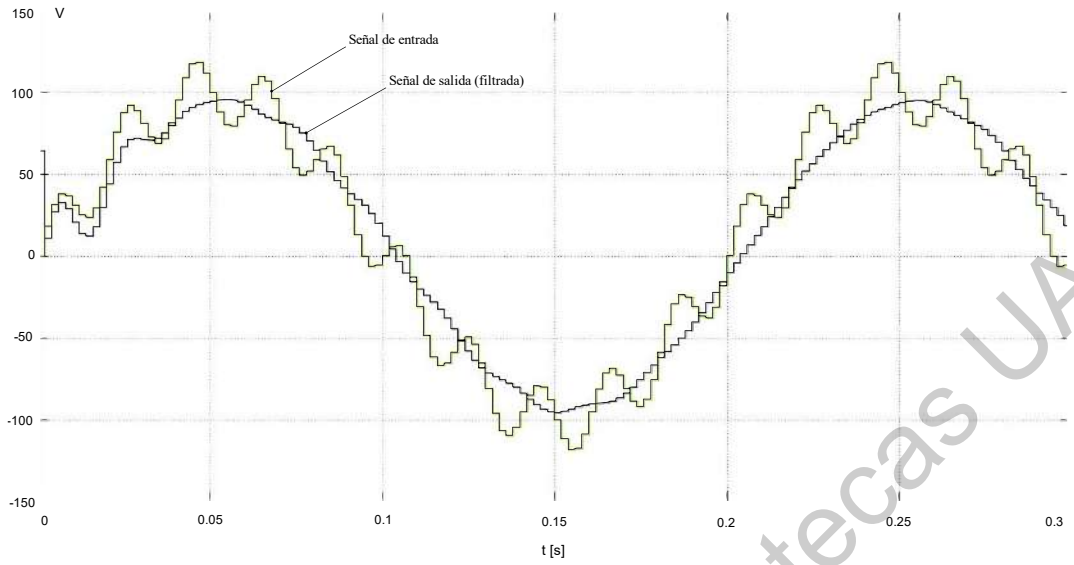

Figura 4.10. Salida del visor de señales al final de la simulación. Se puede observar que la señal de entrada (con la señal montada, como rizos) es filtrada correctamente.

4.1.3 Implementación del filtro en FPGA

Las pruebas de implementación del diseño del filtro en una FPGA necesitan aislar el filtro IIR de la parte de adquisición de señales además de extraer los valores del bus de salida resultantes en la salida del filtro directamente en la computadora para poder analizar su desempeño. Para realizar esto, se propone un sistema cerrado de transmisión-recepción USB, como se muestra en el siguiente diagrama (Figura 4.11), cuyo funcionamiento es meramente el de transmitir los valores de una base de datos de una señal discretizada desde la computadora al filtro y, posteriormente, los valores filtrados resultantes para cada valor de entrada devueltos de la misma manera a la computadora para su posterior análisis en Matlab®.

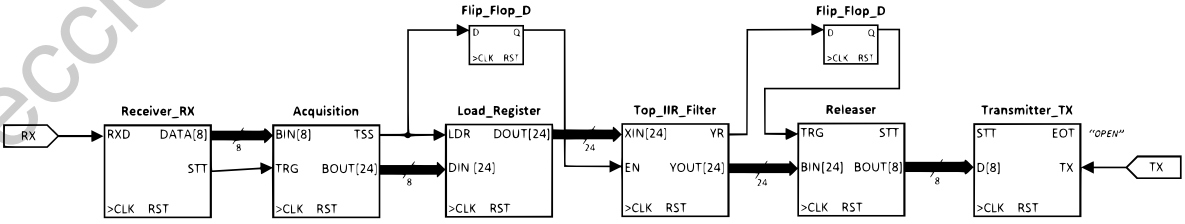

Figura 4.11. Diagrama de nivel más alto del sistema propuesto para el análisis del diseño del filtro implementado en una FPGA.

Debido a que solamente es posible enviar 8 bits de información a la vez a través del puerto USB desde Matlab® hacia la FPGA, es necesario que el algoritmo en Matlab transforme el valor discreto de la señal en un número binario de 24 bits para, luego, diseccionar la palabra completa en 3 bytes (8 bits cada uno) y entonces ser enviado por partes a la FPGA. Posteriormente, de la misma manera, los valores de respuesta serán enviados por partes (3 bytes para cada valor resultante) hacia la computadora. En los listados 4, 5 y 6 del apéndice se describe un algoritmo para Matlab® que, primero, extrae la información discretizada de una base de datos especificada valor por valor, luego, formatea el valor discreto en su forma binaria de punto fijo con ocho bits para la parte entera con signo y dieciséis bits para la parte fraccional. Como ya se mencionó, cada valor discreto es enviado en tres bytes. Cuando el programa en Matlab® recibe información desde el puerto USB enviada por la FPGA, los bytes son organizados en un arreglo de tres y luego unidos en una palabra de 24 bits que son el valor resultante a la salida del filtro.

Dentro de la FPGA el bloque de adquisición (Figura 4.12) recibe tres bytes y lo ensambla en una palabra de 24 bits y habilita la señal que maneja el periodo de muestro **EN**, entonces el bloque IIR comienza el proceso de filtrado y cuando el resultado está listo, alerta al bloque Liberador. Dicho bloque, disecciona el bus de entrada de veinticuatro bits en tres palabras de ocho bits, o tres bytes, y las envía una por una a través del puerto USB. Así el programa en Matlab® las reorganiza para representar los valores de respuesta.

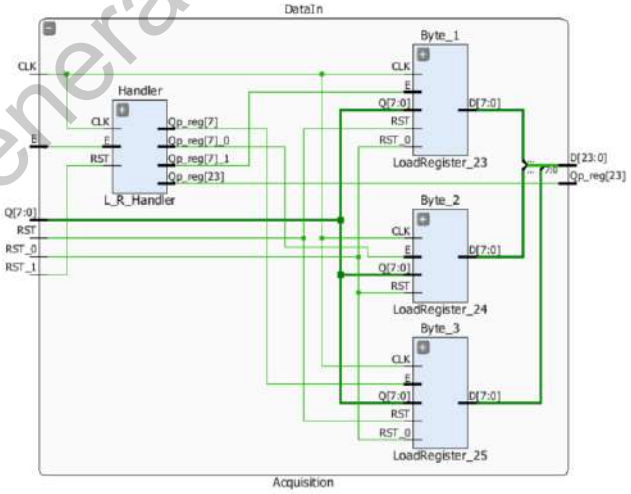

Figura 4.12. Diagrama del bloque de adquisición.

Este bloque percibe cuando el bloque de recepción tiene un valor listo en su salida. De esta forma, la primera vez atrapa un valor en alto, almacena los primeros ocho bits en un registro de carga, la segunda

vez almacena la información en un segundo registro de carga de ocho bits igualmente y la tercera vez en un tercer registro de carga. Entonces, esos tres buses mencionados son ensamblados en un único bus de veinticuatro bits y puestos en el bus de entrada del filtro IIR.

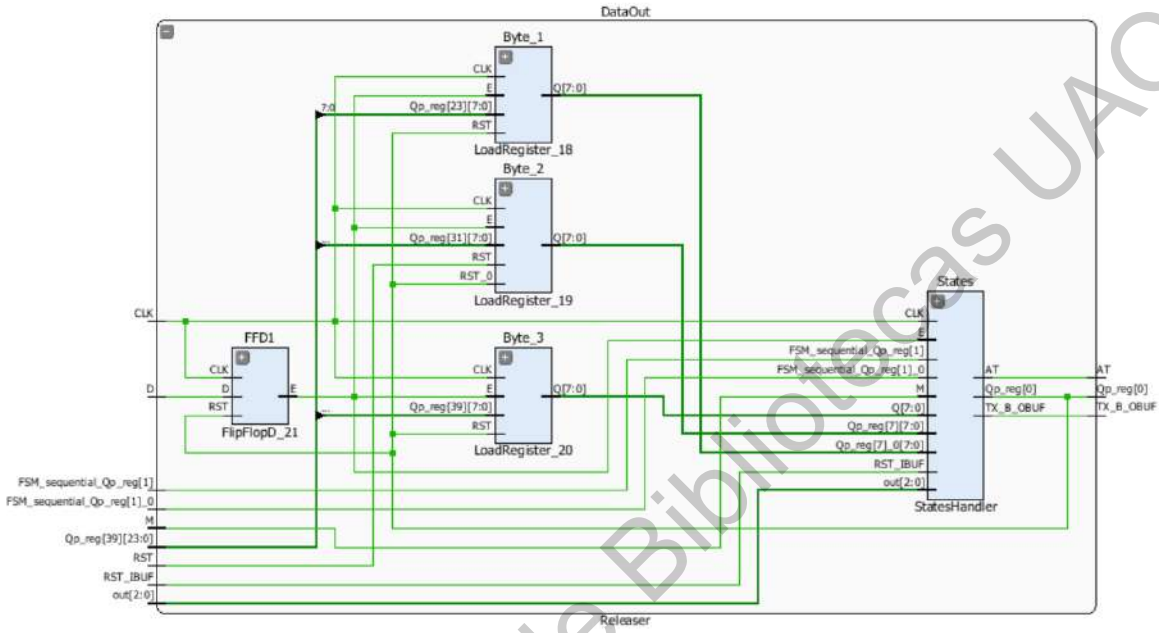

Figura 4.13. Diagrama del bloque liberador.

En la primera etapa, este bloque (Figura 4.13) tiene un Flip-Flop de tipo D, de tal forma que la señal de habilitación es retrasada un ciclo de reloj ayudando así a la sincronización de cuando el valor llega y a cuando el valor está listo para ser transmitido.

A continuación de muestran los resultados de la síntesis (Figura 4.14) e implementación (Figura 4.15) dentro de una FPGA BASYS III usando Vivado 2015®.

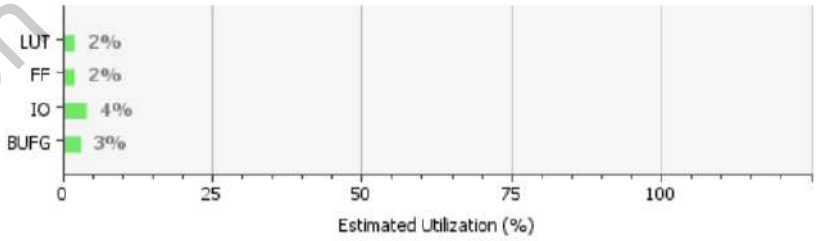

Figura 4.14. Reporte posterior a la Síntesis.

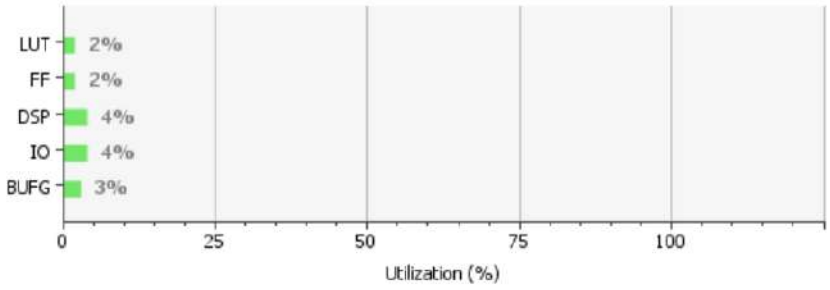

Figura 4.15. Reporte posterior a la implementación.

4.1.4 Tarjeta de adquisición de señales

La tarjeta de adquisición de señales consta de dos etapas, la primera consiste en la adquisición de señales EMG, cuyo elemento principal es el INA114, conectado según se sugiere en la hoja de datos (Figura 4.16). Los valores de los componentes se ajustaron para una ganancia de 1000.

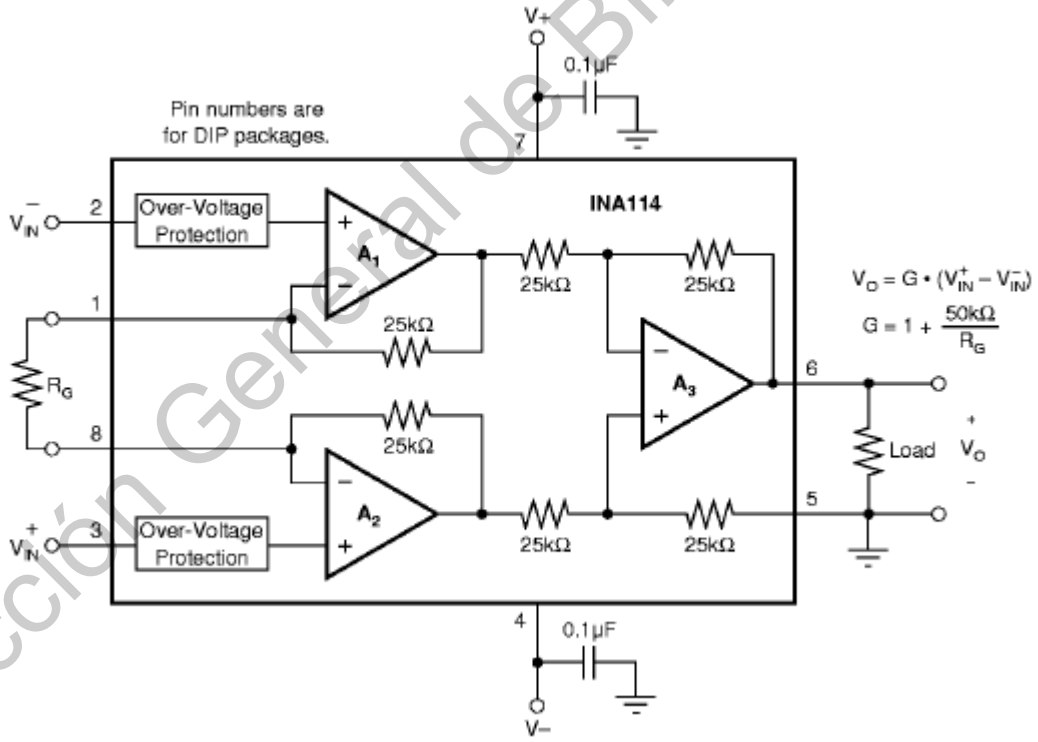

Figura 4.16. Diagrama de conexión INA114, según la hoja de datos.

Tabla 4-3. Valores propuestos en la hoja de datos del INA114, según la ganancia deseada.

Ganancia deseada	$R_G(\Omega)$	Más cercano 1%
1	No conexión	No conexión
2	50.00 k	49.9 k
5	12.50 k	12.4 k
10	5.556 k	5.62 k
20	2.632 k	2.60 k
50	1.02 k	1.02 k
100	505.1	511
200	251.3	249
500	100.2	100
1000	50.05	49.9
2000	25.01	24.9
5000	10.00	10
10000	5.001	4.99

En la segunda etapa mueve la referencia de la señal, que se encuentra en 0 V, a un rango positivo de 2.5 V, para que pueda ser correctamente interpretada por el ADC. El diagrama de conexión de muestra en la Figura 4.17, donde TP4 y TP3 corresponden a puntas de prueba que se colocaron para revisar el comportamiento de la señal en el osciloscopio.

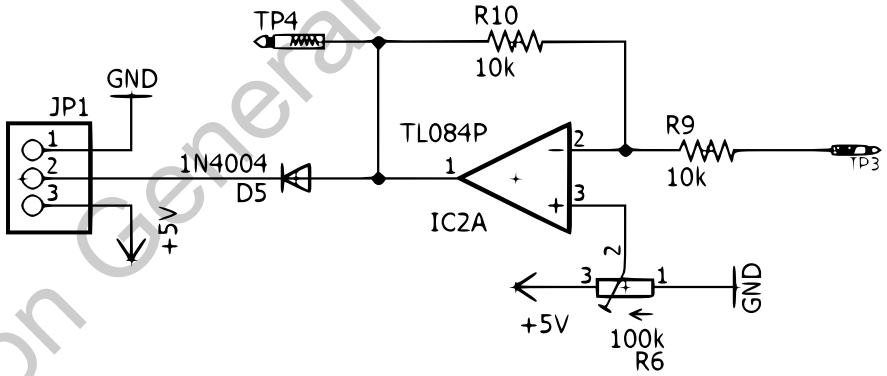

Figura 4.17. Diagrama para el cambio de referencia de la señal.

El electrodo utilizado para las pruebas fue de la marca Kendal MediTrace 200 (Electrodo circular bipolar Ag/AgCl con un diámetro de 10 mm y con un gel adhesivo conductor) y se muestra en la Figura 4.18.

Figura 4.18. Electrodo Kendal MediTrace 200.

El cable utilizado para los electrodos es de tres líneas, terminal positiva, terminal negativa y tierra física (Figura 4.19).

Figura 4.19. Cables para electrodos.

4.1.5 Sistema de filtrado

Añadido al sistema de adquisición de señales se incorporó el sistema de filtrado que consiste en un convertidor analógico-digital ADS7816 de Burr-Brown. De 12 bits a 200 [kHz] con un voltaje de referencia que va desde 100 [mV] hasta 5 [V] con una resolución correspondiente desde 24 [μV] hasta 1.22 [mV].

Dicho integrado requiere una referencia externa, una señal de reloj externa y una única alimentación de voltaje a +5 [V]. La referencia externa puede ser cualquier voltaje entre 100 [mV] y Vcc. La frecuencia de la señal de reloj externa puede variar desde 10 [kHz] (625 [Hz] por el tiempo de procesamiento) y 3.2 [MHz] (200 [kHz] por el tiempo de procesamiento). La conversión digital resultante es dispuesta por la

entrada DCLOCK y producida de manera serial empezando por el Bit Más Significativo (MSB) en el pin de salida DOUT, cabe resaltar que no hay retraso por la conducción en línea.

El mejor uso para la entrada negativa "–In" es para detectar una señal remota de tierra que fuese propensa a desviarse ligeramente con respecto al potencial de tierra loca. Para mantener la linealidad del convertidor, la entrada negativa "– In" no debe exceder GND ±200 [mV]. Finalmente, la referencia externa establece el rango analógico de entrada, por lo tanto, el integrado operará dentro del rango de 100 [mV] hasta Vcc.

Para comprender su implementación en la FPGA es necesario conocer los diagramas de tiempo como se muestra a continuación (Figura 4.20).

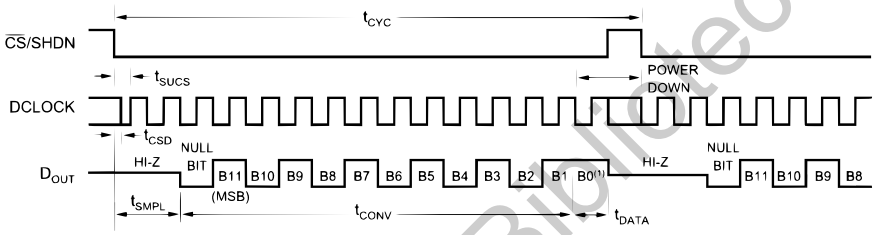

Figura 4.20. Diagrama de tiempo 1 de la hoja de datos ADS7816.

Después de completar la transferencia de información, si se aplican ciclos de reloj adicionales con la señal $\overline{CS}$ en Bajo, el ADC colocará la información del primer Bit Menos Significativo seguida de ceros indefinidamente (Figura 4.21, Figura 4.22 y Figura 4.23).

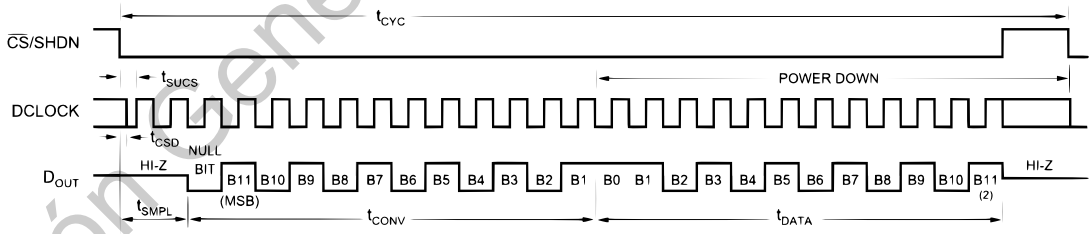

Figura 4.21. Diagrama de tiempo 1 de la hoja de datos ADS7816.

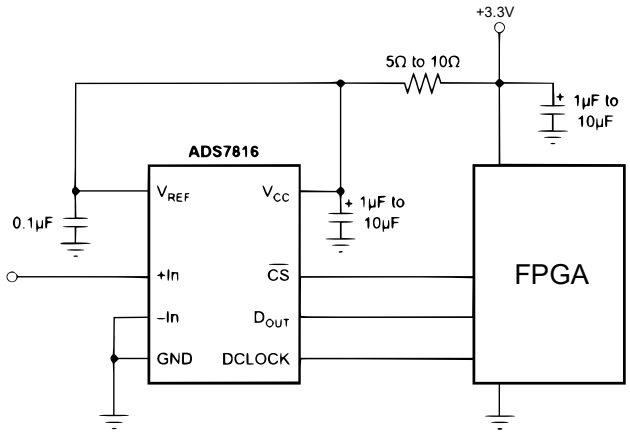

Figura 4.22. Esquemático de aplicación.

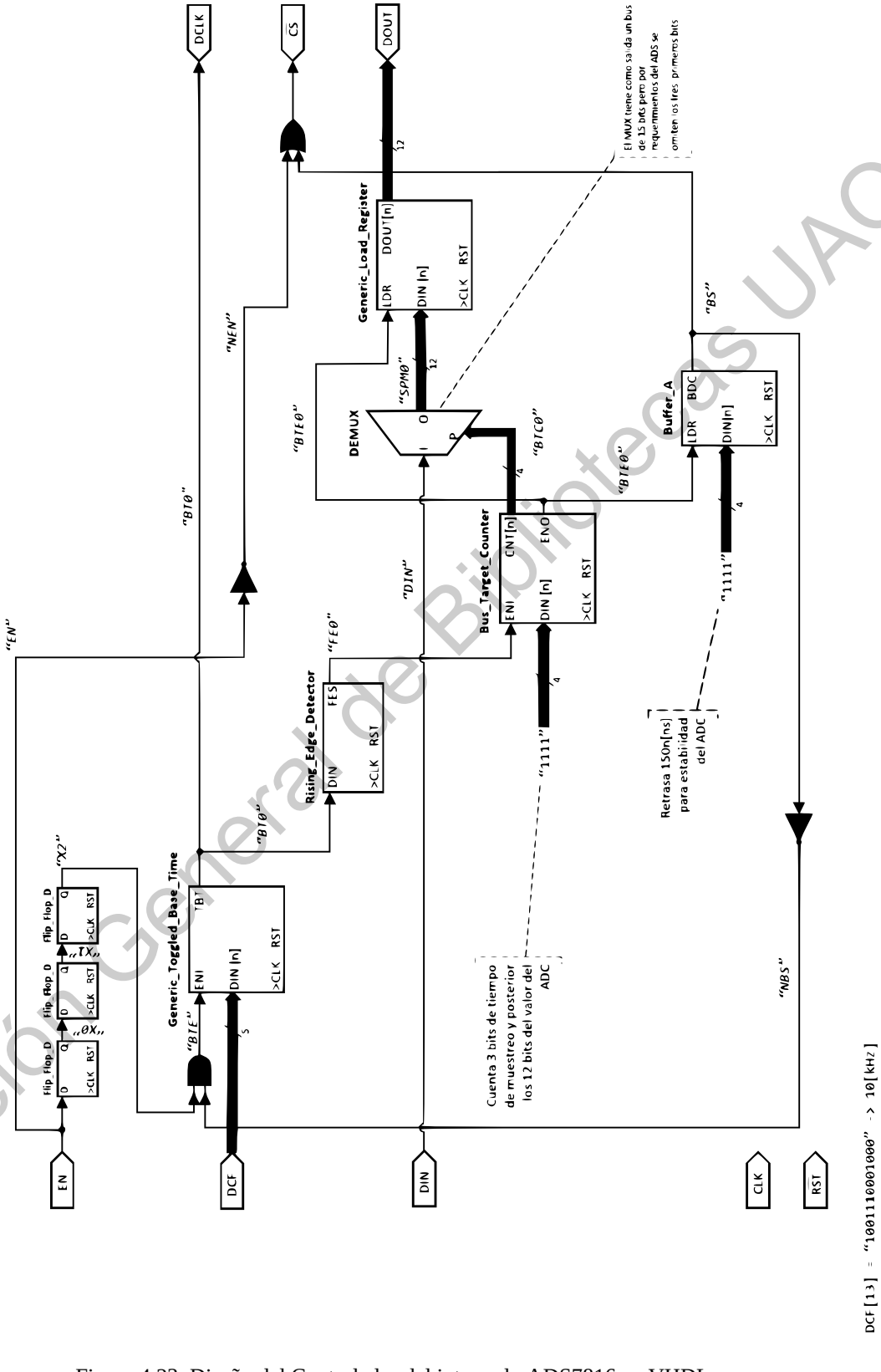

Figura 4.23. Diseño del Controlador del integrado ADS7816 en VHDL.

4.2 Tiempo de procesamiento

La latencia o tiempo de procesamiento se considera desde que el módulo Filtro IIR recolecta el valor en el bus de entrada **XIN**, Figura 4.5, hasta que la respuesta es entregada en el bus de salida **YOUT**. Este tiempo de procesamiento está directamente relacionado con el módulo interno de Multiplicación-Adición, como se puede apreciar en la Figura 4.23, y a la cantidad de coeficientes necesarios para el filtro. Eso quiere decir, que mientras más alto sea el orden del filtro o más bits se empleen para aumentar la exactitud de las operaciones, más tiempo le tomará el procesamiento. Este tiempo de procesamiento se puede obtener al sumar los pulsos de reloj de principio a fin y multiplicarlos por el período del reloj maestro de la FPGA.

Estrato	**Pulsos de reloj**
RTS (Retraso señales de entrada y retroalimentación)	1
MAB (Bloque de Multiplicación-Suma)	10
ADR (Almacenamiento de adición)	1
BTC (Incremento Contador MUX-ROM)	1
Se repite RTS, MAB y BTC n - 1 veces más, donde n es el total de estados de "BP_FSM"	
OM (Retenedor de salida)	2

En el primer pulso de reloj, los valores de retraso de la señal de entrada y de la señal de retroalimentación son guardados en los Registros de Carga, en los siguientes diez pulsos de reloj se realiza la multiplicación. Después, se necesita un pulso para guardar el valor resultante de la primera operación y otro más para incrementar el contador que controla el estado del Multiplexor y del bloque ROM. Estos últimos tres estados se repiten *n* veces, donde *n* es el total de estados que es igual al total de coeficientes del filtro y, finalmente, se le añaden dos pulsos más en el retenedor de salida. Por lo tanto, para un filtro de dieciséis estados se necesitarían 195 pulsos de reloj que es igual a 1.95 [µs] con un reloj maestro a 100 [MHz].

El tiempo de procesamiento del filtro implementado en MATLAB®, según el *listado 2* del apéndice, se calculó utilizando la función *timeit*. Se realizaron 10 pruebas, colocando el electrodo en la pierna, sobre el músculo *Vastus lateralis* (VL), al 66% de la longitud del músculo en la línea desde la espina iliaca superior al lateral de la rótula. Durante la medición de la señal, se mantuvo el reposo sentado.

Tabla 4-4. Pruebas de tiempo en MATLAB®.

Prueba	**Tiempo (seg)**
1	0.000282
2	0.000230
3	0.000242
4	0.000364
5	0.000216
6	0.000217
7	0.000225
8	0.000358
9	0.000217
10	0.000239
Promedio	**0.000259**

Los resultados de las pruebas de MATLAB® se muestran en la Tabla 4-4; el promedio de tiempo de ejecución del filtro es de 259 µs por cada muestra, un tiempo muy superior al obtenido utilizando la FPGA, lo que demuestra las ventajas de filtrar la señal utilizando esta tecnología.

En la Figura 4.24 se muestran la señal de entrada, que es aquella que no está filtrada y la señal de salida, que corresponde a la que sí está filtrada. A simple vista, no se aprecia gran cambio, sin embargo, al realizar un análisis de la relación entre la señal y el ruido, utilizando la función *snr* de MATLAB®. Para la señal de entrada el valor obtenido fue de -15.1763 dB y para la señal de salida fue de -14.8943 dB. El cambio en este valor de *snr*, significa la disminución del ruido en la señal.

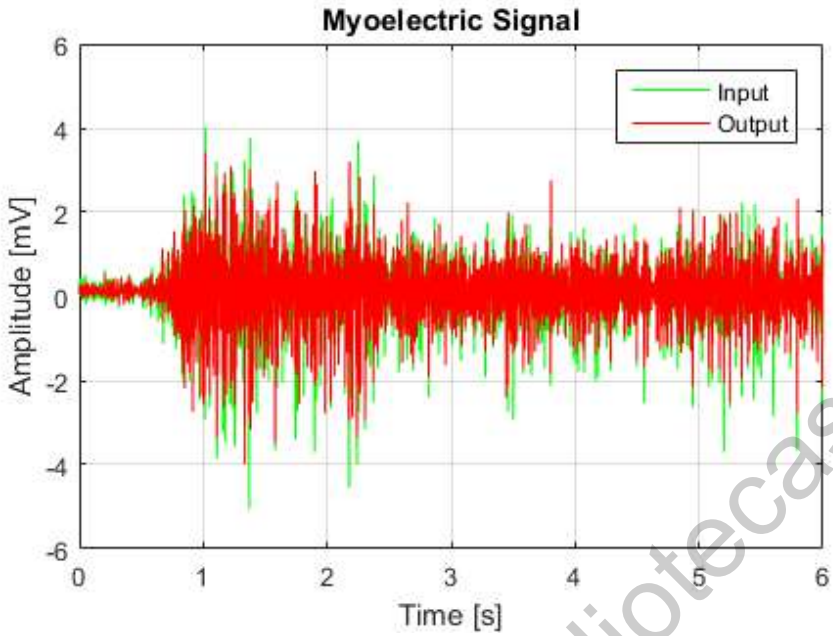

Figura 4.24. Señal obtenida VS señal filtrada.

5 CONCLUSIONES

El sistema de adquisición de señales permitió recuperar las señales desde los músculos de sujetos de prueba y guardarlos de forma digital para después ser usados en Matlab®, como se muestra en la sección 4.1.4.

Los coeficientes necesarios para el diseño del filtro digital IIR que se muestran en la tabla 4-1 se obtuvieron usando las especificaciones de frecuencias de paso y de paro así como la atenuación en la banda de paro y el rizo de banda permitido, descritas en la sección 4.1 Diseño del Filtro.

Con el modelo del filtro digital obtenido previamente, se logró implementar en Matlab y se ejecutó obteniendo las pruebas de tiempo de la Tabla 4-4, donde se aprecia el tiempo de procesamiento. Dicho tiempo de ejecución tuvo un promedio de 259 [µs].

Posteriormente, el mismo modelo del filtro usado para filtrar la señal usando Matlab® se implementó en la FPGA con el diseño en VHDL del diagrama en la Figura 4.5 y descrito en la sección 4.1.3 Implementación del filtro en FPGA. El cual tiene un tiempo de ejecución de 1.95 [µs] el cual es 132 veces más pequeño que el tiempo de ejecución en una PC, por lo tanto se comprueba que el procesamiento de un filtro digital tipo IIR es más rápido en una FPGA, como la que se usó en este trabajo de Xilinx, así

como también se comprueba que no se añaden componentes a otras frecuencias sobre la señal que está siendo procesada como muestra el análisis espectral de la Figura 4.24.

REFERENCIAS BIBLIOGRÁFICAS

Abdallah, M., O. Elkeelany, y A. T. Alouani. 2011. A Low-Cost Stand-Alone Multichannel Data Acquisition, Monitoring, and Archival System With On-Chip Signal Preprocessing. IEEE Transactions on Instrumentation and Measurement. 60:2813–2827. doi:10.1109/TIM.2009.2036402.

Adhikary, R., M. Chakraborty, B. Dara, A. Bharti, and S. Mukherjee. 2012. FPGA based data acquisition and data monitoring system. In: 2012 International Conference on Radar, Communication and Computing (ICRCC). p. 250–253.

Aloisio, A., F. Cevenini, R. Giordano, y V. Izzo. 2007. A VLSI-FPGA system-on-Chip for detectors monitoring. In: 2007 IEEE Nuclear Science Symposium Conference Record. Vol. 1. p. 468–473.

Bhavanam, S. N., P. Siddaiah, y P. R. Reddy. 2014. Zynq 7000 series FPGA based Efficient DTMF detection. In: 2014 IEEE International Conference on Computational Intelligence and Computing Research (ICCIC). p. 1–7.

Buccella, C., C. Cecati, y H. Latafat. 2012. Digital Control of Power Converters #x2014;A Survey. IEEE Transactions on Industrial Informatics. 8:437–447. doi:10.1109/TII.2012.2192280.

Costa, D., y C. S. Páez. 2015. A comparative analysis of hardware techniques for implementation of IIR digital filter on FPGA. In: 2015 XVI Workshop on Information Processing and Control (RPIC). p. 1–6.

Dagbagi, M., A. Hemdani, L. Idkhajine, M. W. Naouar, E. Monmasson, y I. Slama-Belkhodja. 2016. ADC-Based Embedded Real-Time Simulator of a Power Converter Implemented in a Low-Cost FPGA: Application to a Fault-Tolerant Control of a Grid-Connected Voltage-Source Rectifier. IEEE Transactions on Industrial Electronics. 63:1179–1190. doi:10.1109/TIE.2015.2491883.

Fysikopoulos, E., M. Georgiou, N. Efthimiou, S. David, G. Loudos, y G. Matsopoulos. 2014. Fully Digital FPGA-Based Data Acquisition System for Dual Head PET Detectors. IEEE Transactions on Nuclear Science. 61:2764–2770. doi:10.1109/TNS.2014.2354984.

Garufi, F., F. Acernese, A. Boiano, R. D. Rosa, R. Romano, y F. Barone. 2008. A Hybrid Modular Control and Acquisition System. IEEE Transactions on Nuclear Science. 55:295–301. doi:10.1109/TNS.2007.913940.

Idkhajine, L., E. Monmasson, M. W. Naouar, A. Prata, y K. Bouallaga. 2009. Fully Integrated FPGA-Based Controller for Synchronous Motor Drive. IEEE Transactions on Industrial Electronics. 56:4006–4017. doi:10.1109/TIE.2009.2021591.

Jayant, H. K., K. P. S. Rana, V. Kumar, S. S. Nair, y P. Mishra. 2015. Efficient IIR notch filter design using Minimax optimization for 50Hz noise suppression in ECG. In: 2015 International Conference on Signal Processing, Computing and Control (ISPCC). p. 290–295.

Madanayake, A., L. Bruton, y C. Comis. 2004. FPGA architectures for real-time 2D/3D FIR/IIR plane wave filters. In: 2004 IEEE International Symposium on Circuits and Systems. Vol. 3. p. 613–616.

Mukherjee, D., S. Mukhopadhyay, y G. P. Biswas. 2016. FPGA based parallel implementation of morphological filters. In: 2016 International Conference on Microelectronics, Computing and Communications (MicroCom). p. 1–6.

Paul, A., T. Z. Khan, P. Podder, M. M. Hasan, y T. Ahmed. 2015. Reconfigurable architecture design of FIR and IIR in FPGA. In: 2015 2nd International Conference on Signal Processing and Integrated Networks (SPIN). p. 958–963.

Sundaram, K., Marichamy, y Pradeepa. 2016. FPGA based filters for EEG pre-processing. In: 2016 Second International Conference on Science Technology Engineering and Management (ICONSTEM). p. 572–576.

Veiga, A., y C. Grunfeld. 2016. A Modular Pipelined Processor for High Resolution Gamma-Ray Spectroscopy. IEEE Transactions on Nuclear Science. 63:297–303. doi:10.1109/TNS.2016.2515851.

Wang, S., y T. Maruyama. 2016. An implementation method of the box filter on FPGA. In: 2016 26th International Conference on Field Programmable Logic and Applications (FPL). p. 1–8.

Winder, S. 2002. Chapter 16 - Digital fir filter design. In: Analog and Digital Filter Design (Second Edition). Newnes, Woburn. p. 377–394. Available from: https://www.sciencedirect.com/science/article/pii/B9780750675475500166

Zhao, C., y Z. Zhang. 2016. Digital filter design and performance analysis of dynamic temperature signal denoise based on FPGA. In: 2016 10th International Conference on Sensing Technology (ICST). p. 1–7.

Zhao, L., X. Hu, C. Feng, S. Tang, S. Liu, y Q. An. 2013. A 1.6-Gsps High-Resolution Waveform Digitizer Based on a Time-Interleaved Technique. IEEE Transactions on Nuclear Science. 60:2180–2187. doi:10.1109/TNS.2013.2257846.

APÉNDICE: PROGRAMAS EN MATLAB

Listado 1, Filtro Rechaza Banda

```
%close all; clear all; clc;
% General features
frc = 16; % fractional lenght of the ROM in bits
li  = 26; % total lenght of the ROM in bits
% Generate input signal
fsamp = 500; % (Hz)
dt = 1/fsamp; % (secs)
t = 0:dt:1;
fsine1 = 5; % (in passband: should remain in filter output)
fsine2 = 50; % (in stopband: should be removed by filter)
in = 10*sin(2*pi*fsine1*t) + 2*sin(2*pi*fsine2*t);

% Plot input signal
plot(t, in);
title('Input');
xlabel('Time (secs)');
ylabel('Amplitude');
grid on;

% Design filter
wp1 = 45; % Passband corner frequency 1
ws1 = 48; % Stopband corner frequency 1
wp2 = 55; % Passband corner frequency 2
ws2 = 52; % Stopband corner frequency 2
rp = 0.5; % Passband ripple (dB)
rs = 50;  % Stopband attenuation (dB)
w1 = 2*wp1/fsamp;
w2 = 2*ws1/fsamp;
w3 = 2*ws2/fsamp;
w4 = 2*wp2/fsamp;

       %--- Filtro Elíptico ---%
%[N,wn] = ellipord([w1,w4],[w2,w3],rp,rs);
%[b,a] = ellip(N,rp,rs,wn,'stop');
%file_1 = fopen('ROM_Q.txt','w');
       %---                  ---%

       %--- Filtro Butterworth ---%
%[N,wn] = buttord([w1,w4],[w2,w3],rp,rs);
%[b,a] = butter(N,wn,'stop');
%file_1 = fopen('ROM_W.txt','w');
       %---                     ---%
```

```
       %--- Filtro Chebyshev 1 ---%
%[N,wn] = cheb1ord([w1,w4],[w2,w3],rp,rs);
%[b,a] = cheby1(N,rp,wn,'stop');
%file_1 = fopen('ROM_C.txt','w');
       %---                    ---%

       %--- Filtro Chebyshev 1 ---%
[N,wn] = cheb2ord([w1,w4],[w2,w3],rp,rs);
[b,a] = cheby2(N,rs,wn,'stop');
file_1 = fopen('ROM_D.txt','w');
       %---                    ---%

flag = isstable(b,a);
if(flag == 1)
    fprintf('Stable filter\n');
else
    fprintf('Unstable filter\n');
end

fprintf('Order = %2u', N);

% Plot filter frequency response
[H,W] = freqz(b,a,fsamp);
%mag = 10*log(abs(H));
mag = 20*log(abs(H));
figure();
plot(fsamp*W/(2*pi), mag);
%plot(W/(pi), mag);
title('Frequency Response');
xlabel('Frequency (Hz)');
ylabel('(dB)');
%axis([0 0.4 -100 10])
axis tight;
grid on;

% Apply filter and plot output
out = filter(b, a, in);
figure();
plot(t, out);
xlabel('Time (secs)');
ylabel('Amplitude');
grid on;

%fixed dot (10.16)

PrintROM(li, frc, file_1, b, a);
```

Listado 2, Prueba de filtro

```
% mostrar señal de entrada
x = load('female_1.mat');
x_c = x.cyl_ch1;
x = x_c(1,:);
num = 3000;          % valores de la base de datos size(x)
fs = 500;            % frecuencia de muestreo
ts = 1 / fs;         % periodo de la fs
tmax = (num - 1)*ts; % valor de tiempo máximo
t = 0 : ts : tmax;   % vector linea de tiempo

% Diseñar filtro
wp1 = 45; % Passband corner frequency 1
ws1 = 48; % Stopband corner frequency 1
wp2 = 55; % Passband corner frequency 2
ws2 = 52; % Stopband corner frequency 2
rp = 0.5; % Passband ripple (dB)
rs = 50;  % Stopband attenuation (dB)
w1 = 2*wp1/fs;
w2 = 2*ws1/fs;
w3 = 2*ws2/fs;
w4 = 2*wp2/fs;
[N,wn] = ellipord([w1,w4],[w2,w3],rp,rs);
[b,a] = ellip(N,rp,rs,wn,'stop');

%simular filtro
out = filter(b,a,x);

%mostrar graficas dominio del tiempo
entradat = subplot(2,1,1);
salidat  = subplot(2,1,2);

plot(entradat,t,x,'g');
title(entradat,'Input');
xlabel(entradat,'Time [s]');
ylabel(entradat,'Amplitude [mV]');
grid(entradat, 'on');

plot(salidat,t,out,'r');
title(salidat,'Output');
xlabel(salidat,'Time [s]');
ylabel(salidat,'Amplitude [mV]');
grid(salidat, 'on');

% Transformada de Fourier
ini = 1/fs;
esc = fs / num;
```

```
fin = fs / 2;
f = ini : esc : fin;

fx = fft(x);
fx = abs(2*fx(1:num/2)/num);

fo = fft(out);
fo = abs(2*fo(1:num/2)/num);

%mostrar graficas dominio de frecuencia
figure();
entradaf = subplot(2,1,1);
salidaf = subplot(2,1,2);

plot(entradaf,f,fx,'g');
title(entradaf,'Input');
xlabel(entradaf,'Frequency [Hz]');
ylabel(entradaf,'Amplitude [dB]');
plot(salidaf,f,fo,'r');
title(salidaf,'Output');
xlabel(salidaf,'Frequency [Hz]');
ylabel(salidaf,'Amplitude [dB]');
```

Listado 3, Escribe los valores en la FPGA a través del puerto USB

```
% Transmitir valores de señal discreta a una BASYS-3 para efectuar filtro
% IIR
% Para usar el script es necesario que la señal a filtrar ya haya sido
% convertida a un objeto tipo "timeseries"
% Ejemplo :
% x = load('female_1.mat')
% x_c = x.cyl_ch1
% x = x_c(1,:)
% ins = timeseries(x,t,'Name','EMG_Female_1')
% ins.data(1:end) -- Valores de cada muestra
% ins.time(1:end) -- Valores de cada tiempo de muestreo
% ins.lenght      -- Total de valores o ancho

%Señal de ejemplo, cambiar el archivo por el adecuado para la prueba
% wss = load('Vin.mat');                % Wotkspace donde ya se encuentra un ejemplo
% width = length(wss.t);                % Muestras en total de la señal digitalizada

%wss = load('signal_1.mat');
%width = length(wss.t);

fs = 500;            % frecuencia de muestreo
ts = 1 / fs;         % periodo de la fs

%---------------------- Señal sin(t) contaminada -------------------------
fsine1 = 20; % (in passband: should remain in filter output)
fsine2 = 50; % (in stopband: should be removed by filter)
fsine3 = 70;
t = 0:ts:0.3;
v_in = 100*sin(2*pi*fsine1*t) + 15*sin(2*pi*fsine2*t) + 30*sin(2*pi*fsine3*t);
width = length(t);
%-------------------------------------------------------------------------

%------------------ Señal de la base de datos ----------------------------
% x = load('female_1.mat');
% x_c = x.cyl_ch1;
% v_in = x_c(1,:);
% width = length(v_in);% valores de la base de datos size(x)
% tmax = (width - 1)*ts; % valor de tiempo máximo
% t = 0 : ts : tmax;   % vector linea de tiempo
%-------------------------------------------------------------------------

v_out = 1:1:width;

BSS3 = serial('COM4');            % Seleccionar el puerto COMX donde se aloje la FPGA
set(BSS3, 'BaudRate', 9600);      % Definir velocidad de transmisión
set(BSS3, 'DataBits', 8);         % Número de bits
```

```
%BSS3.Terminator = 'CR';          % Comunicación constante sin caracter de terminación
set(BSS3, 'Parity', 'Odd');       % Paridad impar
set(BSS3, 'StopBits', 1);         % Bit de paro
set(BSS3, 'FlowControl', 'none'); % Sin control de hardware o software
fopen(BSS3);                      % Abre el puerto

fprintf(' --> Press Start <--\n');  % Inicio, solo para que sea más fancy 10416 61440
pause

for i = 1:1:width
    d = v_in(i);
    [byte1, byte2, byte3] = dissectBinWordtoBytes(d, 24, 16);
    pause(0.3);
    fwrite(BSS3, byte3, 'uint8');
    pause(0.3);
    fwrite(BSS3, byte2, 'uint8');
    pause(0.3);
    fwrite(BSS3, byte1, 'uint8');
    pause(0.3);

    v = fread(BSS3, BSS3.BytesAvailable, 'uint8');
    x3 = v(3,1);
    x2 = v(2,1);
    x1 = v(1,1);

    v_out(i) = assembleBytestoBinWord(x3, x2, x1);
end

fclose(BSS3);                     % Cierra el puerto
delete(BSS3);                     % Borra el puerto abierto
clear BSS3;                       % Limpia el puerto para un nuevo envío

%save('signal_1.mat');

%mostrar graficas dominio del tiempo
entradat = subplot(2,1,1);
salidat  = subplot(2,1,2);

plot(entradat,t,v_in,'g');
title(entradat,'Input');
xlabel(entradat,'Time [s]');
ylabel(entradat,'Amplitude [mV]');

plot(salidat,t,v_out,'r');
title(salidat,'Output');
xlabel(salidat,'Time [s]');
ylabel(salidat,'Amplitude [mV]');
```

```
grid on;

% Transformada de Fourier
ini = 1/fs;
esc = fs / width;
fin = (fs / 2); % ajuste necesario por valores
f = ini : esc : fin;

fx = fft(v_in);
fo = fft(v_out);

if(length(fo) ~= length(f))
    fx = abs(2*fx(1,1:round(width/2)+1)/width);
    fo = abs(2*fo(1,1:round(width/2)+1)/width);
else
    fx = abs(2*fx(1,1:round(width/2))/width);
    fo = abs(2*fo(1,1:round(width/2))/width);
end

%mostrar graficas dominio de frecuencia
figure();
entradaf = subplot(2,1,1);
salidaf = subplot(2,1,2);

plot(entradaf,f,fx,'g');
title(entradat,'Input');
xlabel(entradat,'Frequency [Hz]');
ylabel(entradat,'Amplitude [dB]');
plot(salidaf,f,fo,'r');
title(entradat,'Output');
xlabel(salidaf,'Frequency [Hz]');
ylabel(salidaf,'Amplitude [dB]');
```

Listado 4, genera la base de coeficientes para el componente ROM_Q

Nota: El archivo que genera es un tipo “txt”, por lo que para usarlo directamente en el diseño de VHDL basta con cambiar el formato a “vhd”.

```
function [ ] = PrintROM( l, f, fileID, a, b )
%l = ancho de bits, f = parte fraccionaria, fileID = fopen(...) a y b son
%los coeficientes a convertir

ra = length(a);
rb = length(b);
xa = 0:1:(ra + rb - 1);
ha = log2(ra + rb);
Ha = round(ha);
if(ha > Ha) %0.000xxx
    bina = Ha + 1;
else
    bina = Ha;
end

%imprimir para ROM_Q
fprintf(fileID,'Library    IEEE;\r\nUse    IEEE.std_logic_1164.all;\r\n\r\nEntity    ROM_Q
is\r\n\tport(\r\n\t');
fprintf(fileID,'SEL   :   in      std_logic_vector(%2u   downto   0);\r\n\tQX      :   out
std_logic_vector(%2u downto 0));\r\nend ROM_Q;\r\n',bina - 1, l - 1);
fprintf(fileID, '\r\nArchitecture DataFlow of ROM_Q is\r\nbegin\r\n\tWith(SEL) select QX
<=');
formatSpec = '\r\n\t"%*s" when "%*s", -- %20.16f';
for i = 1:ra
    sel = fi(xa(i),0,bina,0);
    bsel = bin(sel);
    ax = fi(a(i),1,l,f);
    bax = bin(ax);
    fprintf(fileID, formatSpec, l, bax, bina, bsel, a(i));
end
fprintf(fileID,'\r\n');
%imprimir coeficientes B, recordar que todos lo valores de B deben ser negados
aux = ra + 1;
for i = 2:rb
    sel = fi(xa(aux),0,bina,0);
    bsel = bin(sel);
    bx = fi(-b(i),1,l,f);
    bbx = bin(bx);
    fprintf(fileID, formatSpec, l, bbx, bina, bsel, -b(i));
    aux = aux + 1;
end
dflt = fi(0,0,l,f);
bd = bin(dflt);
```

```
fprintf(fileID, '\r\n\t"%*s" when others;\r\nend DataFlow;', 1, bd);
fclose(fileID);
end
```

Listado 5, Diseccionar una palabra en Bytes

```
function [ b1, b2, b3 ] = dissectBinWordtoBytes( sv, wl, fl )
%Método para diseccionar la palabra de 24 bits de la señal que será enviada
%a la FPGA, por medio de comunicación UART, para ser filtrada.
% sv - valor (double) directo de la base de datos
% wl - tamaño de la palabra en bits
% fl - tamaño de la parte fraccionaria en bits

fi_sv = fi(sv,1,wl,fl);     % Objeto tipo fi del valor sv con signo (1) de tamaño
                            % (wl) en bits con parte fraccionaria (fl) en bits
bi_sv = bin2dec(fi_sv.bin); % Convertir a decimal sin signo la palabra fv_sv
ubi_sv = de2bi(bi_sv,wl);   % Convertir a vector de de (wl) columnas el valor
                            % decimal bi_sv, representando la palabra en
                            % binario
ubyte3 = ubi_sv(1,17:24);   % Toma la fracción del byte más significativo de la
ubyte2 = ubi_sv(1,9:16);    % palabra completa del valor ubi_sv
ubyte1 = ubi_sv(1,1:8);

b1 = bi2de(ubyte1);         % valor en decimal sin signo de 8 bits (int8)
b2 = bi2de(ubyte2);
b3 = bi2de(ubyte3);

end
```

Listado 6, Ensamblar Bytes en una sola palabra

```
function [ v ] = assembleBytestoBinWord( ib3, ib2, ib1 )
%Metodo para conjuntar los bytes recibidos en una sola palabra binaria de tamaño
% wl con parte fraccionaria fl.
% v - valor real flotante obtenido
% ib1, ib2, ib3 son los bytes recibidos como un entero de 8 bits

vib3 = de2bi(ib3,8); % Convierte el entero recibido en un vector que corresponde
vib2 = de2bi(ib2,8); % a su representación en binario de 8 bits
vib1 = de2bi(ib1,8);

binx = 1:1:24;       % Se genera el vector de alojamiento para la palabra completa
binx(1,1:8) = vib1;  % Se ordena del byte 1 en el lsb al MSB
binx(1,9:16) = vib2;
binx(1,17:24) = vib3;

bini = 1:1:8;                % Vector donde se colocara la parte entera de la palabra binx
binf = 1:1:16;               % parte fraccionaria
bini(1,1:8)  =  binx(1,17:24); % asignar a vectores diferentes la parte entera y la
fraccionaria
binf(1,1:16) = binx(1,1:16);

pot = 0;
frc = 0;
for i = 16:-1:1
    pot = pot + 1;
    if binf(1,i) == 1
        frc = frc + (1/(power(2,pot)));
    end
end

f = frc;
deci = bi2de(bini);           % convierte la parte entera a un entero sin signo

if deci > 127
    v = deci - 256 + f;
else
    v = deci + f;
end

end
```