Diseño Eficiente de Filtros Digitales para Sistemas CDMA IS-95

por

David Ernesto Troncoso Romero

Tesis sometida como requisito parcial para obtener el grado de

MAESTRO EN CIENCIAS EN LA ESPECIALIDAD DE ELECTRONICA

en el

Instituto Nacional de Astrofísica, Óptica y Electrónica

Septiembre 2008

Tonantzintla, Puebla

Supervisada por:

Dra. Gordana Jovanovic Dolecek, INAOE

Dr. Jorge Martínez Carballido, INAOE

Diseño eficiente de filtros digitales para sistemas CDMA IS-95

David Ernesto Troncoso Romero

Resumen

En esta tesis se presenta el diseño eficiente de un filtro digital de banda base, usado en sistemas de radio móvil celular bajo los estándares CDMA IS-95 y CDMA 2000. El filtro es requerido para la cancelación de interferencia inter-símbolo (ISI).

En principio se da una revisión general a los estándares CDMA IS-95 y CDMA 2000. Enseguida se explica la razón de utilizar el filtro formador de pulso en banda base para lograr la menor interferencia inter-símbolo. Además se estudian las propuestas existentes para realizar este filtro en forma eficiente.

Posteriormente se analizan algunas formas de diseñar filtros con Respuesta al Impulso Finita (FIR) de forma eficiente y se presentan las características de los filtros simples sin multiplicadores. También se muestran las técnicas para representar los coeficientes de filtros FIR como suma de potencias de dos. Con esto es posible evitar multiplicadores. Los métodos para ahorrar sumadores en un filtro FIR también son explicados.

Se propone un método sencillo y efectivo para diseñar un filtro FIR pasa bajas con las especificaciones de los estándares CDMA IS-95 y CDMA 2000. Además se realizan las comparaciones con los filtros diseñados previamente.

Al final de la tesis se incluye la descripción de los programas realizados en MATLAB y VHDL. También se presenta una lista con las publicaciones derivadas de este trabajo.

Abstract

This thesis presents an efficient design of a baseband digital filter used in IS-95 CDMA and CDMA 2000 systems. The proposed filter achieves zero Inter-Symbol Interference (ISI). Additionally, we compare the proposed method with existing designs in terms of number of adders.

We first present a general introduction of CDMA IS-95 and CDMA 2000 standards. Then it is explained the basic principles of the baseband filter for ISI cancelation. Moreover, current proposals for realizing the pulse-shaping filter are studied.

Further, we analyze some methods to efficiently design Finite Impulse Response (FIR) filters, and the characteristics of simple multiplierless filters are mentioned as well. We review novel techniques to represent each FIR filter coefficient as a sum of powers of two to avoid multipliers as well as recent methods to save adders in FIR filter structures.

Finally, an explanation of the MATLAB and VHDL files used in this work is included. The list of publications resulting from the research work is also given.

Agradecimientos

En primer lugar y con profunda veneración doy gracias a Dios, por cuidar mis riñones y contar mis lágrimas; por regalarme un poquito de ciencia y permitirme llegar hasta este momento y hasta este lugar. Doy gracias por permitir que todo cuanto he escrito delante y detrás de este párrafo, tenga motivo de existencia. A Dios canalizo en cada renglón escrito, mi insondable agradecimiento.

A la Dra. Gordana Jovanovic Dolecek, por la paciente dirección que me dio al realizar este trabajo. Por abrirme la puerta hacia el mundo de la investigación. Por sus amables consejos y observaciones precisas.

Al Dr. Jorge Martínez Carballido, por dirigir mi aprendizaje en el lenguaje VHDL.

A mis sinodales, por revisar este trabajo. Por sus valiosos comentarios y sugerencias para mejorarlo.

Al CONACyT, por el apoyo otorgado a través de la beca para estudios de maestría.

Al INAOE, por permitirme el uso de sus instalaciones. Por el incondicional apoyo otorgado en múltiples aspectos.

A mi papá, don Victoriano, por esa tremenda dosis de ternura y cariño amalgamada perfectamente con un tanto igual de disciplina. A mi mamá, doña Ricarda, por su ilimitado amor; por esa paciencia ligeramente confinada, pero adecuadamente transformada en carácter fuerte y en consistentes palabras de aliento. A mi hermana, Anita, por darme el ejemplo de cómo se debe luchar con ahínco para hacer lo que se desea. A mi amada esposa -mi bendición de Dios- Miriam, de corazón sincero, de mente ágil, de carácter alegre, de amor incondicional; por tolerar al David de las 24 horas (al que duerme, se enoja, llora, se desespera…). A mis amigos (hermanos) de

Agradecimientos

Acayucan, los bohemios, quienes no me han olvidado. A aquellos que dieron a mis días de vacaciones las horas más divertidas. A esos que llevaron serenata a mi mamá, mientras yo estaba lejos de ella. A mis compañeros del INAOE, que me enseñaron a sobrevivir lejos de casa. A quienes toleraron mi impuntualidad; a los que me explicaron con humildad; a los que detuvieron mi desesperado intento de huida; a aquellos que me han permitido ser su amigo.

David Ernesto Troncoso Romero.

Noviembre 2008.

A mi papá, don Victoriano; a mi mamá, doña Ricarda;
a Anita, mi hermana y a mi esposa Miriam

Contenido

Resumen..... **V**

Abstract..... **VII**

Agradecimientos..... **IX**

Contenido..... **XIII**

Prefacio..... **XVII**

CAPÍTULO 1: Introducción..... **1**

1.1 Introducción a CDMA IS-95 y CDMA 2000..... 1

1.1.1 CDMA IS-95..... 2

1.1.2 CDMA 2000..... 5

1.1.3 Ubicación del filtro de banda base..... 6

1.2 El problema de la ISI..... 8

1.3 Características del filtro para IS-95 y CDMA 2000..... 10

1.4 Diseños existentes..... 11

1.4.1 Diseño directo [11]..... 11

1.4.2 Diseño basado en prefiltro-ecualizador [12]..... 12

1.4.3 Diseño de Y. Lian y J. Yu [13]..... 14

1.4.4 Diseño mejorado de Y. Lian y J. Yu [14]..... 14

CAPÍTULO 2: Revisión de algunos métodos de diseño de filtros FIR y de filtros simples sin multiplicadores…… 17

2.1 Introducción…… 17

2.2 Optimización multiobjetivo en el diseño de filtros FIR……. 21

2.2.1 El método Goal Attainment…….... 27

2.3 Filtros IFIR……..... 38

2.4 Filtros simples sin multiplicadores……..... 43

2.4.1 Filtro de Hogenauer……..... 43

2.4.2 Filtro Coseno……..... 47

2.4.3 Filtro Coseno Modificado……. 50

CAPÍTULO 3: Síntesis de coeficientes como suma de potencias de dos…… 53

3.1 Obtención de coeficientes de precisión finita…… 53

3.1.1 Método de redondeo (Rounding)…… 54

3.1.2 Programación Lineal Entera Mixta (MILP)…… 58

3.2 Representación de los coeficientes…… 63

3.2.1 Algoritmo de Hörner para realizar multiplicaciones en base a sumas y corrimientos……. 63

3.2.2 Representación CSD…… 66

3.2.3 Representación MSD…… 70

3.3 Revisión de algunos métodos de eliminación de sub-expresiones comunes……. 73

3.3.1 Algoritmo sub-óptimo de dos etapas…….. 74

3.3.2 Optimización en el espacio de sub-expresiones usando MILP……. 83

CAPÍTULO 4: Filtro propuesto…….. 89
4.1 Introducción…….. 89
4.2 Procedimiento de diseño…… 90
4.2.1 Diseño del filtro interpolador…….. 92
4.2.2 Diseño del filtro modelo utilizando optimización multiobjetivo……. 95
4.3 Realización del filtro propuesto…… 101
4.4 Comparación con diseños previos……... 105
4.5 Simulación en VHDL…….. 107
4.5.1 Simulación del filtro modelo en forma directa transpuesta….. 110
4.5.2 Simulación del filtro interpolador…….. 115
4.5.3 Simulación del filtro total……... 119

CAPÍTULO 5: Conclusiones y trabajo futuro……... 123
5.1 Conclusiones…….. 123
5.2 Trabajo futuro…….. 124

APÉNDICE A: Funciones realizadas en MATLAB y programas de VHDL……... 127
A.1 Funciones realizadas en MATLAB…….. 127
A.2 Programas realizados en VHDL…….. 131

APÉNDICE B: Artículos publicados…… 133

Lista de figuras…… 135

Lista de tablas……. 139

Referencias…… 141

Prefacio

Los sistemas de radio móvil celular basados en los estándares CDMA IS-95 y CDMA 2000 requieren en banda base un filtro digital de fase lineal que permita cancelar la Interferencia Inter-Símbolo (ISI). Para este propósito, los filtros con Respuesta al Impulso Finita (FIR) son usualmente utilizados.

Los filtros FIR tienen muchas ventajas, como la propiedad de fase lineal, estabilidad garantizada, etc. Sin embargo, la complejidad de éstos es relativamente alta en términos de costo de implementación. Los principales recursos de hardware que deben evitarse en los filtros FIR son los multiplicadores, debido a que éstos consumen más potencia y requieren más espacio.

Desde las décadas pasadas se han realizado continuas investigaciones enfocadas en el diseño de filtros FIR de baja complejidad. Una de las estrategias más exitosas y útiles es representar a los coeficientes como suma de potencias de dos. Así las multiplicaciones por los coeficientes pueden ser llevadas a cabo solamente con sumas y corrimientos, dando como resultado un filtro sin multiplicadores. En un filtro sin multiplicadores el objetivo es entonces reducir la cantidad de sumadores requeridos. Además es posible desarrollar un diseño eficiente en base a estructuras multirazón. Estas estructuras permiten realizar el equivalente de un filtro relativamente complejo con la interconexión de filtros con pocos coeficientes muestreados a diferentes velocidades.

Por otra parte, la creciente necesidad de telefonía móvil en los últimos años ha provocado la realización de propuestas de diseño eficiente para el filtro que satisface las especificaciones de los estándares CDMA IS-95 y CDMA 2000.

El objetivo de esta tesis es hacer un diseño eficiente de un filtro de fase lineal sin multiplicadores aplicable a los estándares CDMA IS-95 y CDMA 2000, que contenga la menor cantidad posible de sumadores en comparación con diseños previos.

Para lograr el objetivo de la tesis, primero se investigaron las características del filtro y su ubicación en el sistema de comunicaciones basado en los estándares CDMA IS-95 y CDMA 2000. El capítulo 1 presenta estos temas. En el capítulo 2 se estudian los métodos de diseño eficiente de filtros FIR en base a filtros simples y optimización multiobjetivo. Posteriormente el capítulo 3 detalla los métodos para representar los coeficientes de un filtro FIR como suma de potencias de dos.

En base a los capítulos 2 y 3 se propone, en el capítulo 4, el diseño eficiente de un nuevo filtro que satisface las características requeridas en los estándares CDMA IS-95 y CDMA 2000. Las comparaciones con los diseños existentes son realizadas en este mismo capítulo.

Al final de la tesis se presentan las conclusiones y los trabajos que se pretenden realizar a futuro. Por último, en el apéndice A se incluyen las descripciones de las funciones realizadas en MATLAB y de las entidades de VHDL utilizadas. El apéndice B contiene una lista con las publicaciones logradas a partir de este trabajo.

Capítulo 1

Introducción

¡Oh profundidad de las riquezas de la sabiduría y de la ciencia de Dios! ¡Cuán incomprensibles son sus juicios, e inescrutables sus caminos! *Romanos* 11:33

En el presente capítulo se dará una revisión general a los estándares CDMA IS-95 y CDMA 2000 para sistemas de radio móvil celular, haciendo énfasis sobre los filtros formadores de pulso utilizados en banda base en estos sistemas. Se explicará la necesidad de utilizar estos filtros para satisfacer en la medida de lo posible el criterio de Nyquist y lograr la menor interferencia intersímbolo (ISI). Por último se estudiarán las propuestas existentes para realizar estos filtros en forma eficiente.

1.1 Introducción a CDMA IS-95 y CDMA 2000

Las comunicaciones inalámbricas actualmente abarcan muchas tecnologías, sistemas y servicios enfocados hacia diversas aplicaciones. Sin embargo, uno de los sistemas más ampliamente usados es el de radio móvil celular. Los sistemas celulares ofrecen libertad en cuanto a movilidad, cobertura amplia y comunicación de voz y datos. Estos sistemas se han desarrollado en el ámbito de tecnologías de radio digital, iniciando bajo los estándares GSM (*Global System for Mobile communication*) en Europa, JDC (*Japan Digital Communications*) en Japón e IS-136A e IS-95 en Estados Unidos [1].

El Acceso Múltiple por división de Código (CDMA) es uno de varios métodos para hacer multiplexaje de usuarios en un sistema inalámbrico, basado en comunicación de espectro expandido (*Spread Spectrum Communication*) [2]. En CDMA cada usuario es multiplexado por un código distinto. Así, todos pueden ocupar toda la banda de frecuencia para la transmisión al mismo tiempo.

1.1.1 CDMA IS-95

El estándar IS-95 (cuya abreviación IS significa *Interim Standard*) fue desarrollado por la Asociación de la Industria de Telecomunicaciones (*TIA*, *Telecommunications Industry Association*) luego de múltiples investigaciones por Qualcomm Inc. en el área de CDMA para sistemas celulares. Este estándar fue probado por primera vez en 1995 en Hong Kong y en 1997 recibió el nombre de CDMA-One [3].

El estándar CDMA-One utiliza división de frecuencia *duplex* (*FDD*, *Frequency Division Duplex*). Esto significa que la comunicación desde la estación base hasta la estación móvil y desde la estación móvil hasta la estación base ocurre en diferentes bandas de frecuencia [4]. El enlace de la estación base hasta la móvil se denomina enlace directo (*forward link*) o enlace de bajada (*down-link*). El enlace desde la estación móvil hasta la estación base es llamado enlace inverso (*reverse link*) o enlace de subida (*up-link*).

Para un enlace inverso la banda asignada va desde 824 MHz hasta 849 MHz y para un enlace directo va desde 869 MHz hasta 894 MHz. Un sistema CDMA-One es implementado usando *N* frecuencias portadoras, cada una capaz de soportar *M* usuarios, como se muestra en la figura 1.1 [4].

Cada canal es espaciado en 30 kHz. La tabla 1.1 presenta el esquema de numeración de los canales para los enlaces directo e inverso y las correspondientes frecuencias asignadas.

Dicho de forma general, el enlace de subida consta de dos canales físicos: canal de acceso (*access channel*) y canal de tráfico (*traffic channel*). En el canal de acceso una estación móvil puede enviar mensajes de señalización tales como una solicitud de llamada, una orden de mensaje o una solicitud de registro.

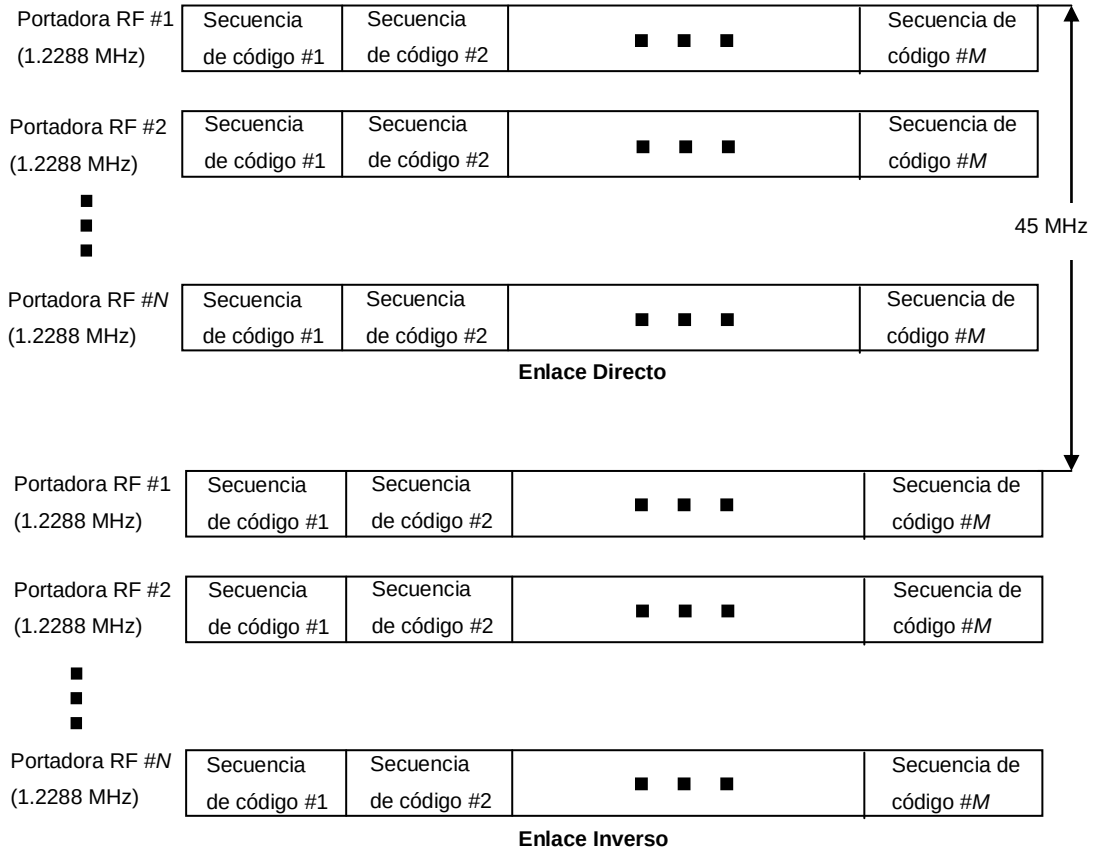


Figura 1.1. Sistema CDMA/FDD.

Un canal de tráfico lleva la información del usuario, como voz o datos, y puede ser usado también durante una llamada para emitir mensajes de señalización, como una realización de *handoff* (proceso de cambio entre una estación base y otra para comunicación con el móvil) o un reporte de medición de potencia a la estación base, por ejemplo. Por otra parte, de

Enlace	Frecuencia MHz	Números de Canal
Inverso	$f_u = 0.03N + 825$	$1 \leq N \leq 777$
	$f_u = 0.03(N\text{-}1023) + 825$	$1013 \leq N \leq 1023$
Directo	$f_d = 0.03N + 870$	$1 \leq N \leq 777$
	$f_d = 0.03(N\text{-}1023) + 870$	$1013 \leq N \leq 1023$

Tabla 1.1. Asignación de números de canal y bandas de frecuencia.

forma general el enlace de bajada consta de cuatro clases de canales físicos [5]:

- un canal piloto (*pilot channel*),
- un canal de sincronización (*sync channel*),
- hasta siete canales de paginación (*paging channels*) y
- hasta 55 canales de tráfico.

El canal piloto envía continuamente una portadora modulada por código Walsh y sirve para que la estación móvil pueda sincronizarse con la estación base. Además esta señal puede usarse como referencia en demodulación coherente en una estación móvil. El canal de sincronización es usado para proporcionar sincronización y configuraciones de información a la estación móvil. Los canales de paginación proporcionan parámetros de sistema, páginas de voz, servicios de mensajes cortos, etc. Los canales de tráfico llevan la información de voz o datos del usuario, así como mensajes de señalización durante una llamada.

El sistema CDMA IS-95 requiere que el espectro sea expandido por una secuencia PN (*Pseudonoise* o pseudo-aleatoria) cuya velocidad de chip es 1.2288 Mchips/s. Esto provoca que el ancho de banda resultante de las señales expandidas sea alrededor de 1.25MHz. En este sistema se utilizan dos códigos PN: el código corto y el código largo. El primero consiste de un par de secuencias PN utilizadas para la expansión y de-expansión de las

componentes de las señales en fase y en cuadratura en el enlace directo. El segundo es utilizado para la expansión en el canal inverso. Además de estos códigos, hay un conjunto de 64 códigos mutuamente ortogonales llamados *códigos Walsh*, que aseguran la ortogonalidad entre las señales recibidas por distintos usuarios desde la misma estación base. Así pues, un canal directo puede ser identificado por su frecuencia portadora de RF (Radio Frecuencia), su único código corto en fase y en cuadratura y su único código Walsh asignado. El canal inverso puede ser identificado por su frecuencia portadora de RF y su único código largo asignado.

1.1.2 CDMA 2000

El sector de estandarización de la *International Telecommunications Union-Radio Communication* (*ITU-R*) desarrolló las especificaciones *International Mobile Telecommunications*-2000 (*IMT*-2000) cerca del año 2000. Las especificaciones *IMT*-2000 están enfocadas a facilitar la introducción de nuevas capacidades en el servicio de telefonía móvil celular, llegando a lo que se conoce como Telefonía Móvil de Tercera Generación, 3G. Los sistemas 3G ofrecen una amplia gama de servicios de telecomunicaciones incluyendo voz, datos a alta velocidad, servicios multimedia y video.

CDMA 2000 es un estándar planeado para satisfacer las necesidades de los sistemas de comunicación inalámbrica 3G. Debido a que este estándar es una evolución de IS-95 hacia sistemas de Tercera Generación, entonces ambos son completamente compatibles [4].

Una diferencia de arquitectura entre los estándares IS-95 y CDMA 2000 es que este último requiere explícitamente las funciones de cuatro diferentes capas del modelo *OSI* (*Open System Interconection*). El modelo de Interconexión de Sistemas Abiertos *OSI* es un marco de referencia para la definición de arquitecturas de interconexión de sistemas de comunicaciones. Las capas requeridas son *Capa Física*, *Sub-capa de Control de Acceso al*

Medio, *Sub-capa de Control de Acceso al Enlace de Señalización* y *Capa Superior* [6].

- *Capa Física*. Ésta es responsable de la transmisión y recepción de bits sobre el medio físico.
- *Sub-capa de Control de Acceso al Medio (MAC, Medium Access Control)*. Controla el acceso desde las capas superiores al medio físico que está siendo usado.
- *Sub-capa de Acceso al Enlace de Señalización (LAC, Link Access Control)*. Esta subcapa es responsable de la fiabilidad de los mensajes de señalización que son enviados y recibidos.
- *Capa Superior*. Desempeña el control completo del sistema CDMA 2000. Es el punto donde se procesan todos los mensajes de señalización y se originan nuevos. Los mensajes con información de voz y datos también se controlan en esta capa.

El estándar CDMA 2000 presenta además las siguientes características:

- Tamaños de canal de 1, 3, 6, 9 y 12 x 1.25 MHz.
- Soporte para antenas de alta tecnología.
- Mayores velocidades de datos.
- Capacidad de operar en un amplio rango de entornos (interior/exterior, vehículos, etc.).

1.1.3 Ubicación del filtro de banda base

Tanto en los enlaces de subida y de bajada, después de las codificaciones y de la multiplicación por la secuencia PN, pero antes de la multiplicación por la señal portadora, están situados los filtros de banda base. Éstos pueden verse en las figuras 1.2 y 1.3.

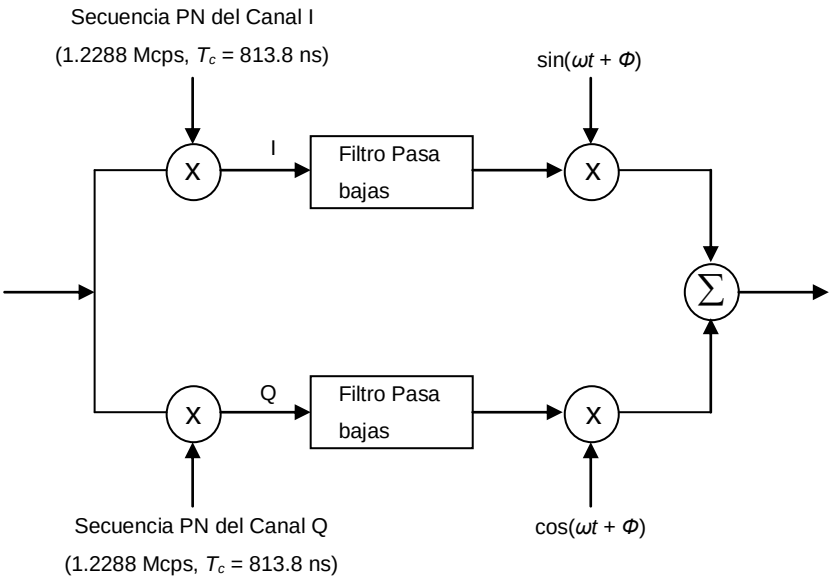


Figura 1.2. Ubicación de los filtros de banda base en el canal directo.

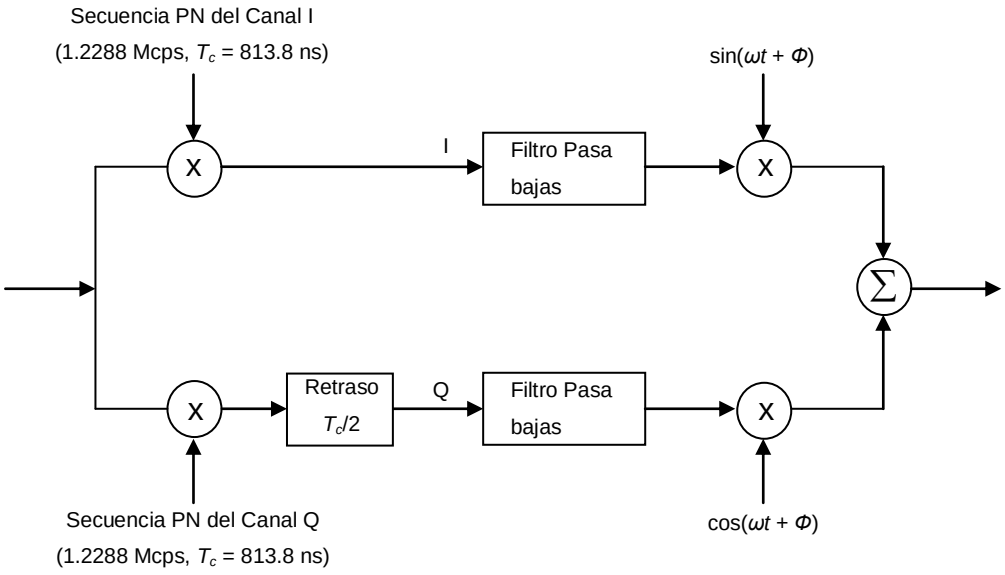


Figura 1.3. Ubicación de los filtros de banda base en el canal inverso.

La función de un filtro es formar adecuadamente los pulsos que llegan a él, para evitar en lo posible el efecto de interferencia inter-símbolo (ISI). Esto se mencionará en forma más detallada en la sección 1.2. En IS-95 y CDMA

2000 el filtro utilizado es normalmente un filtro digital FIR de 48 coeficientes, que no satisface exactamente el criterio de Nyquist para cero ISI pero proporciona resultados deseables. Sin embargo, existen nuevas propuestas para realizaciones más eficientes de este filtro. Éstas serán revisadas en la sección 1.4.

1.2 El problema de la ISI

Debe considerarse que un canal a través del cual se transmite información digital está limitado a algún ancho de banda específico, digamos W Hz. Entonces el canal puede ser modelado como un filtro pasa-bajas lineal cuya respuesta en frecuencia es cero para todas las frecuencias *f* mayores que W. Si este canal es ideal en su banda de paso, es posible transmitir a velocidades comparables a W. Sin embargo, en un canal real la transmisión de señales a una velocidad igual o mayor a W provoca la existencia de interferencia inter-símbolo (ISI) entre cierto número de símbolos adyacentes.

Considérese un sistema de banda base como el de la figura 1.4, con un modulador de amplitud de pulsos que modifica una secuencia de símbolos b_k de duración T_b y cuyos valores son 1 y 0, en símbolos a_k de amplitud +1 y -1. Dado que cada pulso cuadrado tiene un espectro en frecuencia en forma de función *sinc*, que se extiende desde un valor infinito negativo hasta un valor infinito positivo, es necesario limitar esta distribución espectral buscando la anchura de banda mínima que permita detectar al pulso con la tasa de error requerida. Entonces esta secuencia de pulsos es aplicada a un filtro transmisor con respuesta al impulso $g(t)$ y la salida de esto es transmitida a través del canal con respuesta al impulso $h(t)$. El canal añade ruido blanco aditivo gaussiano. La señal recibida pasa a través de un filtro receptor cuya respuesta al impulso es $c(t)$. La salida de esto es muestreada en un tiempo dado $t_i = iT_b$, para luego pasar a un sistema donde se obtiene finalmente cada símbolo en base a un criterio de decisión [7].

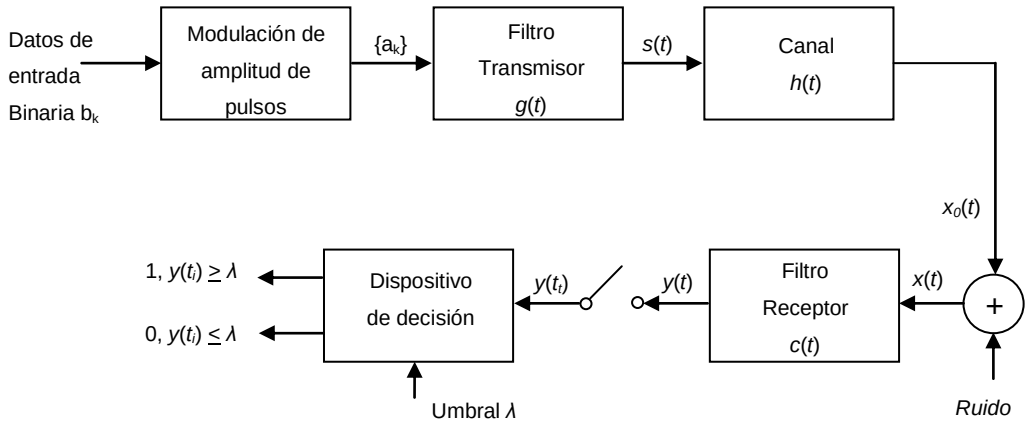


Figura 1.4. Sistema de transmisión de datos binarios en banda base.

La salida en el filtro receptor puede ser escrita como:

$$y(t) = \mu \sum_k a_k p(t - kT_b) + n(t) \qquad (1.1),$$

donde μ es un factor de escalamiento y $p(t)$ es la forma de pulso definida por una doble convolución entre los filtros y el canal, normalizada a $p(0) = 1$, lo que justifica al factor μ. La operación entre a_k y $p(t)$ es la convolución de la secuencia de pulsos original, con el pulso escalado $\mu p(t)$. El término $n(t)$ representa el ruido producido en la salida del filtro receptor debido al ruido añadido en el canal.

Dado que la salida en el filtro receptor es muestreada cada tiempo $t_i = iT_b$, entonces $y(t)$ ahora es expresada como:

$$\begin{aligned} y(t_i) &= \mu \sum_k a_k p[(i-k)T_b] + n(t_i) \\ &= \mu a_i + \mu \sum_{\substack{k=-\infty \\ k \neq i}}^{\infty} a_k p[(i-k)T_b] + n(t_i). \end{aligned} \qquad (1.2)$$

En este caso el término μa_i representa la contribución del i-ésimo bit transmitido, mientras que el segundo término representa la Interferencia

Inter-símbolo (*Intersymbol Interference, ISI*) y es el efecto residual de todos los otros bits transmitidos al momento de la codificación del *i*-ésimo bit. El tercer término representa la muestra de ruido en el tiempo t_i. Así pues, en el diseño de los filtros transmisor y receptor, el objetivo es minimizar los efectos de la ISI y del ruido. El problema consiste en determinar la forma de onda del pulso $p(t)$ con la cual la ISI sea completamente eliminada [8].

Comúnmente se encuentran ya definidas la respuesta en frecuencia del canal y la forma de onda del pulso enviado originalmente. El problema se enfoca en encontrar las respuestas en frecuencia del filtro transmisor y receptor. La forma de estas respuestas en frecuencia queda restringida por el *criterio de Nyquist para transmisión en banda base sin distorsión*. Éste indica que si el canal es de W Hz de ancho de banda, la velocidad de muestreo $(T_b)^{-1}$ debe ser el doble de W y el espectro en frecuencia de los filtros debe cortar su banda de paso exactamente en W Hz, de forma abrupta [7-8].

Dado que filtros con esta respuesta son prácticamente irrealizables, se permite cierta banda de transición entre las bandas de paso y rechazo. Además estas bandas consideran ciertas tolerancias en sus magnitudes. No obstante cuando la respuesta en amplitud en la banda de paso no es constante y la respuesta en fase respecto a la frecuencia no varía en forma lineal, una sucesión de pulsos transmitidos a través del canal a velocidades cercanas a W se distorsiona fácilmente [9].

1.3 Características del filtro para IS-95 y CDMA 2000

Los estándares CDMA IS-95 y CDMA 2000 recomiendan el uso de un filtro digital de banda base para lograr la cancelación del efecto de Interferencia Inter-Símbolo [10,11]. El filtro debe ser de Respuesta al Impulso Finita (FIR) para lograr fase lineal y es recomendado de orden 47, diseñado directamente con el método de Parks-McClellan. Este filtro es conocido como *filtro IS-95*.

Las características del filtro IS-95 deben ser:

- Velocidad de muestreo de 4.9152 MHz,
- Frecuencia límite de paso de 590 kHz,
- Frecuencia límite de rechazo de 740 kHz,
- Rizo en la banda de paso de 1.5 dB,
- Atenuación en la banda de rechazo de -40 dB.

Este filtro también puede ser usado en sistemas CDMA 2000 [10].

1.4 Diseños existentes

El filtro IS-95 ha sido rediseñado en varias ocasiones con la finalidad de satisfacer las características con menos recursos de hardware. Los diseños existentes fueron presentados en [11-14] y se resumen a continuación.

1.4.1 Diseño directo [11]

En [11] se presenta un filtro de longitud 47 utilizando la estructura directa para un filtro FIR de fase lineal explotando simetría.

Los coeficientes del filtro fueron representados con código *CSD* (*Canonical Signed Digit*), para evitar el uso de multiplicadores y ahorrar sumadores en cada coeficiente. Los coeficientes fueron redondeados usando la expresión siguiente:

$$\tilde{h}(k) = 2^{-10} \cdot \left\lfloor \frac{h(k)}{2^{-10}} \right\rfloor, \quad \text{con} \quad k = 0, 1, ..., 47. \tag{1.3}$$

En este caso $\tilde{h}(k)$ representa al k-ésimo coeficiente cuantizado, mientras $h(k)$ es el k-ésimo coeficiente en valor real obtenido directamente del proceso de diseño. El símbolo $\lfloor\ \rfloor$ representa la operación de redondeo hacia el

entero de menor valor más cercano (redondeo por defecto). Cada coeficiente fue representado en formato de 16 bits, con 10 bits para representar la parte fraccionaria. El resultado final con estructura directa fue obtenido con 66 sumadores.

Además del diseño con estructura directa, en [11] se presentan otras formas de implementación cuya finalidad es disminuir la complejidad en hardware. Estas formas están basadas en el enfoque de filtros multiplexados en el tiempo.

En un filtro multiplexado en el tiempo todos los coeficientes son almacenados en una memoria ROM y los datos de entrada son almacenados en una memoria RAM de entrada. El filtro multiplexado requiere un multiplicador de propósito general conectado a un acumulador. Esta unidad recibe el nombre de *MAC* (*Multiply-Accumulator*). El filtro debe ser sincronizado a una velocidad de muestreo más alta que la de una implementación basada en estructura directa. Esto debido a que el funcionamiento del filtro se desempeñará en forma secuencial, haciendo en cada ciclo la multiplicación entre un coeficiente y un dato de entrada. Así, si el filtro tiene una longitud L, la velocidad de muestreo del filtro debe ser L veces más grande que la velocidad de muestreo original. Además utiliza un sistema de control para elegir la dirección correcta en las memorias de coeficientes y de datos de entrada. Esta unidad de control también debe resetear a la unidad *MAC* cuando sea necesario.

1.4.2 Diseño basado en prefiltro-ecualizador [12]

En [12] se realiza un nuevo filtro basado en la técnica de prefiltro-ecualizador. La estructura de prefiltro-ecualizador se presenta en la figura 1.5. El prefiltro está diseñado con la conexión en paralelo de un filtro de longitud par y uno de longitud impar. Éstos son $H_e(z^2)$ y $H_o(z^2)$ respectivamente. El filtro ecualizador antecede al prefiltro y está representado en la figura por $H_{eq}(z^2)$.

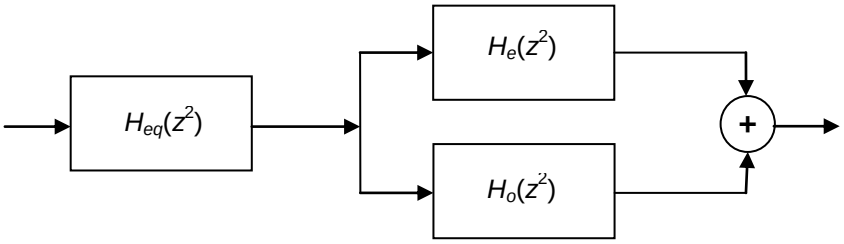


Figura 1.5. Estructura prefiltro-ecualizador.

Dentro del intervalo [0, π/2], el filtro de longitud par presenta valores positivos en su respuesta en frecuencia. Dentro del intervalo [π/2, π] los valores de amplitud de respuesta en frecuencia son negativos. El filtro de longitud impar presenta valores positivos de respuesta en frecuencia en ambos intervalos. La conexión en paralelo de ambos filtros provoca que las respuestas en frecuencia de éstos se sumen punto a punto. Así la respuesta en frecuencia resultante obtiene buena amplitud dentro del intervalo [0, π/2] y buena atenuación en el intervalo [π/2, π].

El filtro de longitud par es utilizado con dos coeficientes unitarios y utiliza dos retrasos entre cada coeficiente. El filtro de longitud impar es obtenido por el método de optimización *MILP* (*Mixed Integer Linear Programming*) partiendo de la respuesta en frecuencia del filtro de longitud par. El filtro ecualizador es obtenido después del filtro de longitud impar, utilizando el mismo método de optimización. Todos los filtros utilizan dos retrasos entre cada coeficiente.

El filtro resultante requiere 30 coeficientes en total. Esto representa un ahorro del 37.5 % en el número de coeficientes respecto al filtro presentado en [11], ya que aquél tiene 48 coeficientes.

La implementación del filtro presentado en [12] está basada en una arquitectura de filtro multiplexado en tiempo. Debido a la menor cantidad de coeficientes, el tamaño de la memoria ROM y de los registros internos es menor comparado con el del filtro presentado en [11]. Los coeficientes fueron representados con 8 bits.

1.4.3 Diseño de Y. Lian y J. Yu [13]

En [13] se presenta la misma estructura de diseño dado en [12]. El procedimiento para obtener el prefiltro y el ecualizador es el mismo. No obstante, los filtros son obtenidos por el método de optimización presentado en [15]. Éste es una modificación del método *MILP*. Luego de encontrar los valores de coeficientes óptimos para el filtro ecualizador, el filtro de longitud impar vuelve a optimizarse, para encontrar coeficientes que mejoren aún más la respuesta en frecuencia total.

El filtro resultante requiere 27 coeficientes en total con precisión de 6 bits para la parte fraccionaria. Esto representa un ahorro del 10% respecto al filtro presentado [12]. Los coeficientes obtenidos son representados en *CSD* para ahorrar sumadores. La implementación del filtro total en estructura directa requiere 38 sumadores y es libre de multiplicadores. Esto da un ahorro de 42.4% sobre el filtro presentado en [11].

1.4.4 Diseño mejorado de Y. Lian y J. Yu [14]

El filtro presentado en [14] está basado en la misma estructura de prefiltro-ecualizador, como los dos anteriores. Sin embargo, en esta propuesta el filtro ecualizador, junto con los dos filtros que conforman al prefiltro, son diseñados simultáneamente utilizando un método de optimización multiobjetivo.

El procedimiento de diseño comienza por realizar un filtro como en [12], utilizando programación lineal. La diferencia es que en este caso no hay necesidad de llegar a cuantizar los coeficientes. Los coeficientes obtenidos son tomados como valores iniciales. A continuación se realiza el proceso de optimización multiobjetivo simultáneamente sobre el prefiltro y el filtro ecualizador. Si el resultado obtenido no satisface las especificaciones, entonces se utiliza el algoritmo de [16] para modificar las tolerancias dadas en la optimización. El proceso de optimización multiobjetivo se repite las veces necesarias, hasta que las especificaciones logran ser satisfechas. Por último, los coeficientes son cuantizados utilizado el método *MILP*.

El filtro resultante tiene 29 coeficientes en total. La precisión de los coeficientes es de 5 bits para la parte fraccionaria y los coeficientes son representados en código *CSD* para ahorrar sumadores. La implementación es realizada en estructura directa y no requiere multiplicadores. En total, este filtro utiliza 34 sumadores. Esto representa un ahorro del 11% respecto a la estructura anterior.

Capítulo 2

Revisión de algunos métodos de diseño de filtros FIR y de filtros simples sin multiplicadores

Mejor es adquirir sabiduría que oro preciado; y adquirir inteligencia vale más que la plata. *Proverbios* 16:16

En este capítulo se estudiarán algunos métodos para diseñar filtros FIR de forma eficiente y se revisarán los filtros simples que serán utilizados en esta tesis. Se presentará la técnica de optimización multiobjetivo como herramienta para obtener los coeficientes de un filtro a partir de las especificaciones de éste en frecuencia. Además se explicará brevemente la técnica IFIR para realizar filtros con menos coeficientes que un diseño directo. Por último, algunos filtros simples sin multiplicadores serán explicados.

2.1 Introducción

Un filtro FIR con coeficientes constantes es un sistema Lineal Invariante en el Tiempo (LTI). Ante una secuencia de entrada muestreada $x(n)$, su salida está dada por la operación de convolución expresada como [17]

$$y(n) = x(n) * h(n) = \sum_{k=0}^{L-1} h(k)x(n-k)\,, \tag{2.1}$$

donde

- $x(n)$ es la secuencia de entrada y representa la muestra de entrada en el instante de tiempo n.
- $h(n)$ es la respuesta al impulso del filtro y representa el n-ésimo coeficiente del mismo.
- $y(n)$ es la secuencia de salida y representa la muestra de salida en el instante de tiempo n.
- L es el número de coeficientes del filtro y representa la *longitud* del mismo.

El filtro puede ser caracterizado por su función de sistema, dada por

$$H(z) = \sum_{k=0}^{L-1} h(k)z^{-k}\,. \tag{2.2}$$

Un filtro FIR tiene fase lineal si su respuesta al impulso tiene coeficientes reales y cumple la condición dada por

$$h(n) = \pm h(L-1-n) \qquad n = 0,1,...,L-1. \tag{2.3}$$

La condición dada en (2.3) recibe el nombre de *simetría* cuando el término del lado derecho recibe un signo positivo. Cuando recibe un signo negativo, se llama condición de *antisimetría*.

Al sustituir la ecuación (2.3) en (2.2) se obtiene

$$H(z)=\begin{cases} z^{-(L-1)/2}\left\{h\left(\dfrac{L-1}{2}\right)+\displaystyle\sum_{n=0}^{(L-3)/2}h(n)\left[z^{(L-1-2n)/2}\pm z^{-(L-1-2n)/2}\right]\right\} & L \text{ impar},\\ z^{-(L-1)/2}\displaystyle\sum_{n=0}^{(L/2)-1}h(n)\left[z^{(L-1-2n)/2}\pm z^{-(L-1-2n)/2}\right] & L \text{ par}. \end{cases} \tag{2.4}$$

La respuesta en frecuencia se obtiene al sustituir z por $e^{j\omega}$. Para un filtro simétrico con $h(n) = h(L–1–n)$ se tiene entonces

$$H(e^{j\omega})=\begin{cases} e^{-j\omega(L-1)/2}\left\{h\left(\dfrac{L-1}{2}\right)+2\displaystyle\sum_{n=0}^{(L-3)/2}h(n)\left[\cos\omega\left(\frac{L-1}{2}-n\right)\right]\right\} & L \text{ impar},\\ e^{-j\omega(L-1)/2}\cdot 2\displaystyle\sum_{n=0}^{(L/2)-1}h(n)\left[\cos\omega\left(\frac{L-1}{2}-n\right)\right] & L \text{ par}. \end{cases} \tag{2.5}$$

De igual manera, la respuesta en frecuencia para un filtro antisimétrico con $h(n) = –h(L–1–n)$ está dada por

$$H(e^{j\omega})=\begin{cases} e^{j[-\omega(L-1)/2+\pi/2]}\cdot 2\displaystyle\sum_{n=0}^{(L-3)/2}h(n)\left[\sin\omega\left(\frac{L-1}{2}-n\right)\right] & L \text{ impar},\\ e^{j[-\omega(L-1)/2+\pi/2]}\cdot 2\displaystyle\sum_{n=0}^{(L/2)-1}h(n)\left[\sin\omega\left(\frac{L-1}{2}-n\right)\right] & L \text{ par}. \end{cases} \tag{2.6}$$

Es posible notar, por el término exponencial, que en los cuatro casos representados en (2.5) y (2.6) la fase es una función lineal de la frecuencia. La tabla 2.1 presenta en forma resumida las expresiones correspondientes a la respuesta en frecuencia para cada uno de los 4 tipos de filtros FIR de fase lineal.

El problema de diseño de filtros FIR consiste entonces en determinar L coeficientes $h(0)$, $h(1)$, …, h(L – 1) a partir de una especificación de la respuesta en frecuencia deseada [18].

$H(e^{j\omega}) = e^{j\beta}e^{-j\frac{L-1}{2}\omega}H_r(\omega)$		
Tipo	β	$H_r(e^{j\omega})$
Tipo 1: *L* impar; *h*(*n*) simétrico	0	$\sum_{k=0}^{(L-1)/2} a(k)\cos\omega k$
Tipo 2: *L* par; *h*(*n*) simétrico	0	$\sum_{k=1}^{L/2} b(k)\cos\omega\left(k-\frac{1}{2}\right)$
Tipo 3: *L* impar; *h*(*n*) antisimétrico	$\frac{\pi}{2}$	$\sum_{k=1}^{(L-1)/2} c(k)\sin\omega k$
Tipo 4: *L* par; *h*(*n*) antisimétrico	$\frac{\pi}{2}$	$\sum_{k=1}^{L/2} d(k)\sin\omega\left(k-\frac{1}{2}\right)$

Tabla 2.1. Respuesta en frecuencia para filtros FIR de fase lineal.

La longitud *L* de un filtro FIR puede estimarse, partiendo de las especificaciones deseadas, como [19]

$$L = \frac{-20\log_{10}(\sqrt{\delta_p\delta_s}) - 13}{14.6\Delta f} + 1, \quad \text{con} \quad \Delta f = \frac{\omega_s - \omega_p}{2\pi}. \tag{2.7}$$

Obsérvese que en esta ecuación el término de ancho de banda de transición, Δf, tiene más impacto que las otras variables sobre la longitud del filtro. Además este término afecta de forma proporcional inversa. Nótese también que Δf aparece expresado en términos de frecuencias relativas ω_p y ω_s. Debido a esto, la frecuencia de muestreo juega un papel muy importante en el valor obtenido para Δf. Para ciertos valores dados de frecuencia de paso y de rechazo, las frecuencias relativas son más pequeñas cuanto más grande es la frecuencia de muestreo. Esto provoca un valor para Δf menor en tanto sea más alta sea la velocidad de muestreo, lo cual implica mayor longitud del filtro.

2.2 Optimización multiobjetivo en el diseño de filtros FIR

La optimización multiobjetivo permite hacer la minimización simultánea de un conjunto de funciones objetivo. Estas funciones pueden estar sujetas a cierto número de restricciones de igualdad o desigualdad.

Un problema de optimización multiobjetivo puede ser expresado como [20]

$$\text{Hallar} \quad \mathbf{X} = \{x_1, x_2, ..., x_n\}, \tag{2.8}$$

$$\text{que minimiza a} \quad \mathbf{F} = f_1(\mathbf{X}), f_2(\mathbf{X}), ..., f_k(\mathbf{X}) \tag{2.9}$$

$$\begin{aligned} \text{sujeto a} \quad & G_i(\mathbf{X}) = 0 \qquad i = 1, 2, ..., m_e, \\ & G_i(\mathbf{X}) \leq 0 \qquad i = m_e + 1, m_e + 2, ..., m, \\ & \mathbf{X}_L \leq \mathbf{X} \leq \mathbf{X}_U \ , \end{aligned} \tag{2.10}$$

donde k denota el número de funciones objetivo que serán minimizadas. Cualquiera de las funciones f_i (**X**) o G_i (**X**) (o incluso todas) pueden ser funciones no lineales.

Generalmente no existe un vector solución **X** que minimice a las k funciones objetivo simultáneamente. Esto significa que debe haber un balance que permita que las funciones objetivo puedan ser satisfechas de la mejor manera posible. Esta situación es enfrentada bajo un concepto llamado *solución óptima de Pareto* [20 - 22].

Un vector solución **X** es una solución óptima de Pareto si no existe otra solución factible **Y** tal que

$$\begin{aligned} & f_i(\mathbf{Y}) \leq f_i(\mathbf{X}) \qquad i = 1, 2, ..., m, \\ & f_j(\mathbf{Y}) < f_j(\mathbf{X}) \qquad \text{para al menos un valor } j. \end{aligned} \tag{2.11}$$

En otras palabras, una solución óptima de Pareto, también llamada conjunto de puntos no-inferiores, es aquella en la que ya no puede

optimizarse más una función objetivo a menos que se degraden otras. Es decir, no existe un conjunto **Y** que evaluado en todas las funciones objetivo, resulte en valores iguales o menores a los evaluados con **X** y a la vez pueda encontrar un valor más pequeño para una de las funciones objetivo. Si existiera, esa sería la solución óptima de Pareto.

Ejemplo 2.1 [21]

Considérense las funciones objetivo dadas por

$$f_1 = (x-3)^4, \tag{2.12}$$

$$f_2 = (x-6)^2. \tag{2.13}$$

Para estas funciones, todos los valores de x entre 3 y 6 (puntos A y B) forman parte de las soluciones óptimas de Pareto. La figura 2.1 muestra las gráficas correspondientes a las funciones f_1 y f_2.

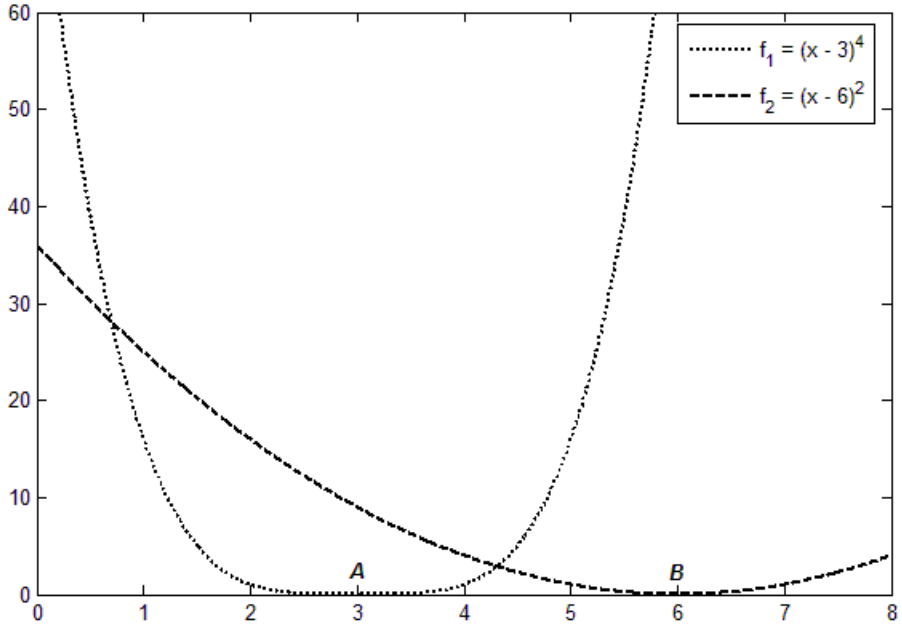


Figura 2.1. Soluciones óptimas de Pareto para f_1 y f_2.

Obsérvese que $x = 3$ minimiza a f_1, sin embargo no corresponde a un valor mínimo en f_2. De igual manera, $x = 6$ minimiza a f_2 pero no corresponde a un valor mínimo en f_1. No obstante, en este rango de valores es posible encontrar los valores mínimos factibles para ambas funciones. Esto puede ser visto con más claridad en la figura 2.2, donde se muestra la relación gráfica entre f_1 y f_2. Esta gráfica está dada en el *espacio de funciones objetivo*. Es fácil ver de la figura 2.2 que cuando se llega al mínimo en f_1 es necesario que f_2 crezca hasta 9. Igualmente, llegar al mínimo en f_2 implica que f_1 aumente hasta 81. Sin embargo, no hay otros valores que permitan este balance.

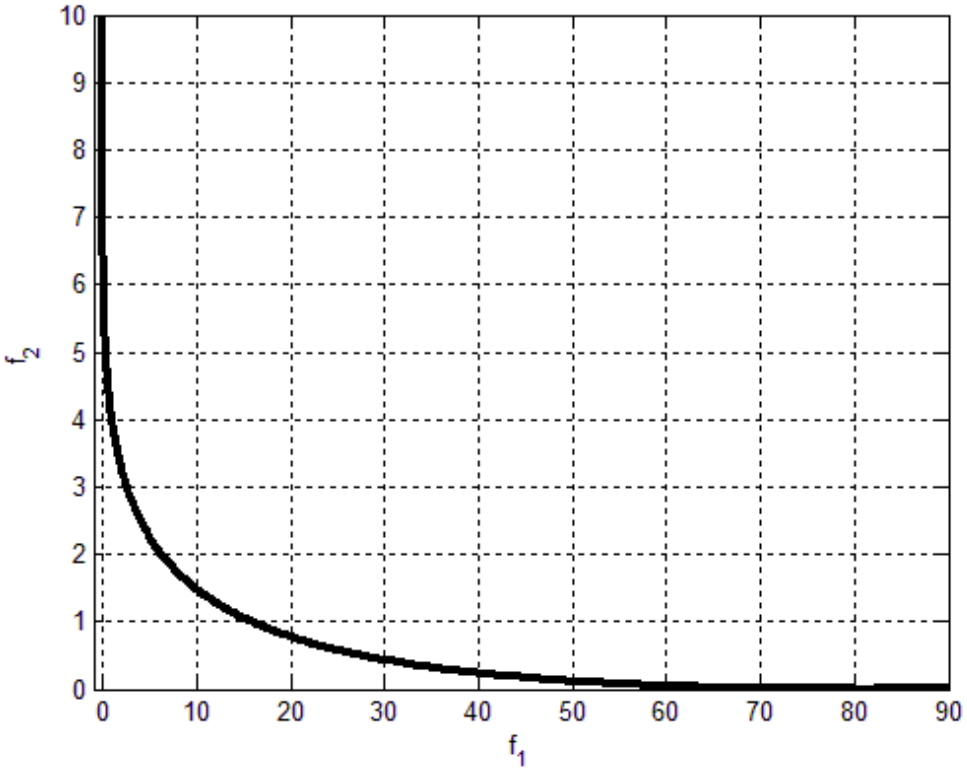


Figura 2.2. Espacio de funciones objetivo.

Para definir el concepto de solución óptima de Pareto en forma más general, considérese una región factible Ω, en el *espacio de parámetros* $\mathbf{X} \in \Re^n$ que satisface todas las restricciones dadas en (2.10). Entonces es posible

definir la correspondiente región factible para el espacio de funciones objetivo Λ como [23]

$$\Lambda = \left\{ \mathbf{Y} \in \Re^m \right\}, \quad \text{donde } y_i = f_i(\mathbf{X}) \in \mathbf{F} \text{ sujeto a } \mathbf{X} \in \Omega. \tag{2.14}$$

El vector de funciones objetivo **F** mapea el espacio de parámetros en el espacio de funciones objetivo para todos los valores factibles de Ω. Así se puede obtener el correspondiente espacio factible Λ. Esto es representado en la figura 2.3.

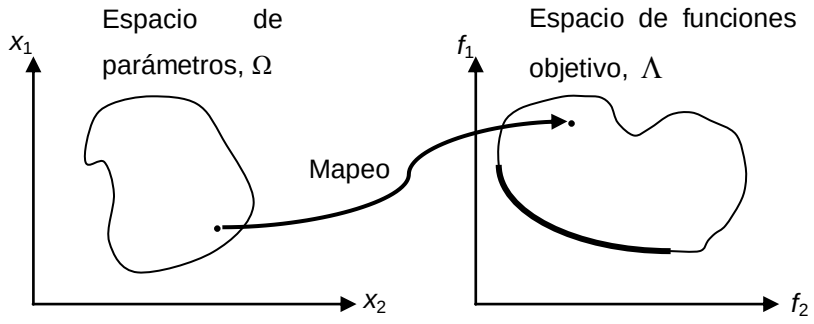


Figura 2.3. Mapeo del espacio de parámetros al espacio de funciones objetivo.

La figura 2.4 muestra el conjunto de soluciones no-inferiores entre los puntos *A* y *B*. Los puntos *C* y *D* representan puntos-solución específicos.

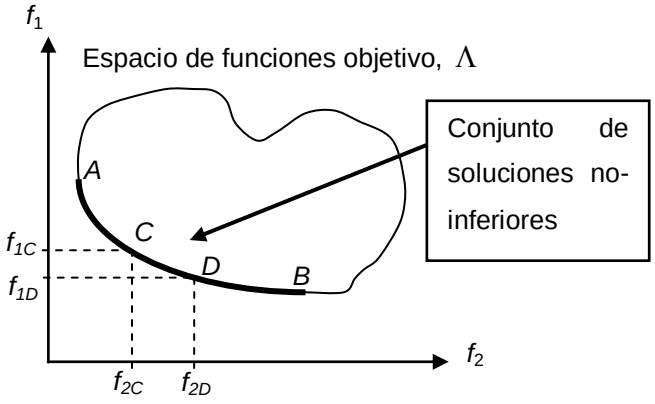


Figura 2.4. Conjunto de soluciones no-inferiores.

Nótese que los puntos *C* y *D* son soluciones no-inferiores debido a que una función evaluada en estos puntos mejora, pero la otra función empeora.

Entonces la optimización multiobjetivo consiste en la minimización de un vector de funciones objetivo a través de la generación y selección de puntos no-inferiores. Las funciones objetivo pueden estar sujetas a un cierto número de restricciones. Cada función depende de un único vector de parámetros independientes. Mapeando del espacio de parámetros al espacio de funciones objetivo se pueden encontrar los puntos no-inferiores, en los que la mejora de una función objetivo implica la degradación en otra.

Varios métodos han sido desarrollados para resolver el problema de optimización multiobjetivo. La mayoría de estos métodos genera un conjunto de soluciones óptimas de Pareto y utiliza algún criterio adicional para seleccionar una solución particular para el problema. A continuación se mencionan algunos de los métodos utilizados para hallar la solución a problemas de optimización multiobjetivo.

- *Método de función de utilidad*. En este método se define una función de utilidad para cada función objetivo, dependiendo de la importancia de cada una comparada con las otras. Entonces se da solución maximizando una función de utilidad total representada como la suma de los productos de cada función objetivo con su respectiva función de utilidad. Esta función total tiene las restricciones del problema de optimización multiobjetivo. Algunas veces es llamado *método de funciones ponderadas* [21,23].
- *Método de función de utilidad invertida.* Aquí el inverso del producto de cada función objetivo con su respectiva función de utilidad cuantifica que tan indeseable es esa función objetivo. La suma de todos estos términos constituye la función de no-deseabilidad total. El problema se aborda como la minimización de esta función bajo las restricciones del problema de optimización multiobjetivo [21].

- *Método de criterio global*. Consiste en encontrar la solución óptima minimizando un criterio global elegido previamente. Se expresa una función de desviación relativa (generalmente cuadrática) entre el valor deseado y el valor actual para cada función objetivo. El criterio global es formulado como la suma de las funciones de error dadas para cada función objetivo [20 - 22].
- *Método de función objetivo delimitada*. En este método se minimiza cada función objetivo en forma individual. Además de las restricciones propias del problema multiobjetivo se agregan las otras funciones objetivo como restricciones. Éstas están delimitadas entre el mínimo y máximo valor aceptable correspondiente a cada una de ellas [20 - 22].
- *Método Lexicográfico*. En el método lexicográfico las funciones objetivo se ordenan de acuerdo a su importancia. La primera función minimizada es la más importante, y tiene las mismas restricciones del problema multiobjetivo. Luego se minimiza la segunda función, incorporando a sus restricciones la función objetivo previamente optimizada. El procedimiento continúa hasta optimizar la última función objetivo, incluyendo como restricciones adicionales las funciones previamente minimizadas [22].
- *Método Goal Attainment*. Aquí se define un factor de relajación que permite estimar un error entre cada función objetivo y el valor deseado para ésta. El error es ponderado por un correspondiente valor para cada función objetivo. El problema se formula como la minimización del factor de relajación que permita obtener el menor error posible entre las funciones objetivo y su respectivo valor deseado [21, 23].

Utilizando el enfoque de optimización multiobjetivo se puede diseñar un filtro FIR especificando la respuesta en frecuencia deseada para cada valor de frecuencia. En el capítulo 4 se presentará la propuesta de un filtro

diseñado en base a optimización multiobjetivo, solucionando el problema planteado con el método *Goal Attainment*. Debido a esto, en la sección siguiente se presenta una explicación generalizada sobre este método. Además se muestra la forma en que se realiza el diseño de un filtro digital con esta técnica utilizando MATLAB.

2.2.1 El método Goal Attainment

En el método *Goal Attainment* se define un conjunto de valores deseados como [21]

$$\mathbf{F}^* = \left\{F_1^*, F_2^*, ..., F_m^*\right\}. \tag{2.15}$$

Éste está relacionado con un conjunto de funciones objetivo dado por

$$\mathbf{F}(\mathbf{X}) = \left\{f_1(\mathbf{X}), f_2(\mathbf{X}), ..., f_m(\mathbf{X})\right\}. \tag{2.16}$$

Así, cada función objetivo tiene su correspondiente valor deseado. Las funciones objetivo en el conjunto **F**(**X**) deben intentar alcanzar sus correspondientes valores deseados del conjunto $\mathbf{F}^*$. El grado de tolerancia para acercarse por defecto o por exceso al valor deseado está dado por un conjunto correspondiente de coeficientes de ponderación, expresado como

$$\mathbf{w} = \left\{w_1, w_2, ..., w_m\right\}. \tag{2.17}$$

El problema de optimización es formulado de la siguiente manera:

$$\begin{aligned} &\text{minimizar} \quad \gamma \\ &\gamma \in \Re, \mathbf{X} \in \Re^n \\ &\text{tal que} \quad f_i(\mathbf{X}) - w_i\gamma \le F_i^* \qquad i = 1,2,...,m\,. \end{aligned} \tag{2.18}$$

El término $w_i\gamma$ introduce un factor de relajación en el problema. Éste permite tener cierta tolerancia para alcanzar el valor deseado.

Es posible notar que la interpretación de un problema basado en optimización multiobjetivo es intuitiva y relativamente sencilla con el método *Goal Attainment*. Por tanto, diseñar un filtro FIR a partir de sus especificaciones en frecuencia no presenta demasiada dificultad utilizando este método.

En el paquete *Optimization Toolbox* de MATLAB se lleva a cabo la solución de un problema multiobjetivo con el método *Goal Attainment* a través de la función `fgoalattain` [23].

La función `fgoalattain` resuelve el problema de optimización multiobjetivo dado por (2.18), expresado en forma más general como

$$\begin{aligned} &\text{minimizar} \quad \gamma \\ &\gamma \in \Re, \mathbf{X} \in \Re^n \\ &\text{tal que} \quad \mathbf{F}(\mathbf{X}) - \mathbf{w}\gamma \le \mathbf{F}^*, \\ &\qquad\quad \mathbf{c}(\mathbf{X}) \le 0, \\ &\qquad\quad \mathbf{c}_{\text{eq}}(\mathbf{X}) = 0, \\ &\qquad\quad \mathbf{A} \cdot \mathbf{X} \le \mathbf{b}, \\ &\qquad\quad \mathbf{A}_{\text{eq}} \cdot \mathbf{X} = \mathbf{b}_{\text{eq}}, \\ &\qquad\quad \mathbf{lb} \le \mathbf{X} \le \mathbf{ub}, \end{aligned} \tag{2.19}$$

donde $\mathbf{X}$, $\mathbf{F}^*$, $\mathbf{w}$, $\mathbf{b}$, $\mathbf{b}_{\text{eq}}$, $\mathbf{lb}$ y $\mathbf{ub}$ son vectores. $\mathbf{A}$ y $\mathbf{A}_{\text{eq}}$ son matrices y $\mathbf{c}(\mathbf{X})$, $\mathbf{c}_{\text{eq}}(\mathbf{X})$ y $\mathbf{F}(\mathbf{X})$ son funciones parametrizadas que representan, cada una, un conjunto de funciones. Todas las funciones para $\mathbf{F}(\mathbf{X})$ pueden ser lineales o no lineales. Para $\mathbf{c}(\mathbf{X})$ y $\mathbf{c}_{\text{eq}}(\mathbf{X})$ las funciones deben ser solo funciones no lineales.

La expresión general para la función `fgoalattain` puede ser representada como [23]

[**X**, **Fval**, *gamma*] = `fgoalattain` (**F**(**X**), $\mathbf{X_0}$, $\mathbf{F}^*$, **w**, **A**, **b**, $\mathbf{A}_{eq}$, $\mathbf{b}_{eq}$,…
lb, **ub**, **c**(**X**), *opciones*, $\mathbf{P_1}$, $\mathbf{P_2}$, …), (2.20)

donde

- **X** es el vector que contiene las soluciones adecuadas para minimizar las funciones objetivo.
- **Fval** es el vector de valores obtenidos al evaluar las funciones objetivo en los puntos solución **X**.
- *gamma* es el factor que debe ser minimizado.
- **F**(**X**) es una función de **X** expresada en términos parametrizados. Es decir, **F**(**X**) representa, en función de los parámetros adecuados y de **X**, a todas las funciones objetivo (dependientes de **X**) que van a ser minimizadas. Por lo tanto, **F**(**X**) tiene un vector como valor de salida. Cada elemento de ese vector es el resultado de evaluar su correspondiente función objetivo en **X**.
- $\mathbf{X_0}$ es el vector de valores iniciales dados para comenzar el algoritmo de minimización.
- $\mathbf{F}^*$ es el vector de valores deseados correspondientes a cada función objetivo.
- **w** es el vector de ponderaciones dado a cada función objetivo.
- **A** es la matriz de coeficientes que representa un conjunto de restricciones de desigualdad lineales.
- **b** es el vector de valores restrictivos para las restricciones de desigualdad lineales.
- $\mathbf{A}_{eq}$ es la matriz de coeficientes que representa un conjunto de restricciones de igualdad lineales.
- $\mathbf{b}_{eq}$ es el vector de valores restrictivos para las restricciones de igualdad lineales.

- **lb** es un vector de valores restrictivos para **X**. Cada elemento en **lb** es un valor límite inferior para cada elemento en **X**. Las soluciones menores a estos valores no están disponibles.
- **ub** es un vector de valores restrictivos para **X**. Cada elemento en **ub** es un valor límite superior para cada elemento en **X**. Las soluciones mayores a estos valores no están disponibles.
- **c**(**X**) es también una función de **X** expresada en términos parametrizados. Representando a **c**(**X**) en función de los parámetros adecuados y de **X**, se tienen todas las funciones de restricción no lineales (dependientes de **X**, de igualdad o desigualdad). Por lo tanto, **c**(**X**) también tiene un vector como valor de salida.
- *opciones* permite utilizar algunos parámetros adicionales incluidos en el paquete *Optimization Toolbox* para la función `fgoalattain`. Estos parámetros pueden permitir algunas funciones extra, como controlar el número de iteraciones, etc. El elemento *opciones* es configurado a través de la función `optimset`, incluida en mismo paquete.
- $\mathbf{P_1}$, $\mathbf{P_2}$, …, representan los vectores de parámetros que están incluidos en las funciones **F**(**X**) y **c**(**X**).

La forma de usar la función `fgoalattain` puede ser vista con más claridad con el par de ejemplos que se presenta a continuación.

Ejemplo 2.2

Hallar la solución *x* para que las funciones dadas en (2.12) y (2.13) alcancen un valor de cero, con valores de tolerancia considerados como

a) Tolerancia cero para la función (2.12),

b) Tolerancia cero para la función (2.13),

c) Tolerancia uno para ambas funciones.

Este ejemplo está basado en las funciones mostradas en el ejemplo 2.1 y pretende dar una mejor explicación sobre la función `fgoalattain`.

Las funciones objetivo dadas en (2.12) y (2.13) son repetidas aquí por conveniencia: $f_1 = (x-3)^4$, $f_2 = (x-6)^2$.

El valor inicial para x es elegido como 0. Los valores que se desean alcanzar con las funciones f_1 y f_2 son ambos 0.

La función de MATLAB que define a ambas f_1 y f_2 puede ser representada como una función "en línea", utilizando el comando `inline`. Cada función del problema debe poder ser expresada en la misma función de MATLAB. Debido a esto se utilizan los parámetros P1 y P2.

Para el inciso a) la tolerancia es cero para f_1. En f_2 la tolerancia es tomada como 1. Esto significa que se permite el mismo porcentaje de tolerancia para acercarse al valor deseado por defecto o por exceso. El código correspondiente para resolver el problema se presenta enseguida.

```
x0 = 0;
val_deseados = [0 0];
P1 = [3 6];
P2 = [4 2];
f = inline('(x - (P1)).^P2');
w = [0 1]; %------------------>  no hay tolerancia para f1
[x,fval,gamma]=fgoalattain(f,x0,val_deseados,w,[],[],[],[],[],[],...
                           [],[],P2,P1)
```

Los corchetes [ ] representan campos vacíos. Esto significa que las condiciones que ocupan esa posición en la forma general dada en (2.20) no existen para este problema.

Al ejecutar el código anterior se obtienen los siguientes valores.

```
x  =
     3.0000
fval =
    0.0000    9.0000
gamma =
    9.0000
```

La solución obtenida es $x = 3$. Al ser evaluada en $x = 3$, la función f_1 vale 0, que es el valor deseado, sin embargo f_2 vale 9. Esto significa que se ha dado prioridad a f_1, exactamente como fue establecido.

Para el inciso b) la tolerancia es cero para f_2. En f_1 la tolerancia es tomada como 1 (se intercambian los valores de *w* en la línea 6 del código). Al ejecutar el programa se obtienen los siguientes valores.

```
x  =
     6.0000
fval =
    81.0000    0.0000
gamma =
    81.0000
```

Así, la solución obtenida es $x = 6$. Ahora la función f_2 vale 0, que es el valor deseado, pero f_1 vale 81. Ahora se ha dado prioridad a f_2 y a la vez se ha obtenido el valor posible más cercano a cero para f_1.

Para el inciso c) la tolerancia en ambas funciones es 1 (se modifican los valores de *w* en la línea 6 del código). Se obtienen los siguientes resultados.

```
x  =
     4.3028
fval =
    2.8806    2.8806
gamma =
    2.8806
```

La solución obtenida es $x = 4.3028$. En este caso ambas funciones han sido acercadas a 0 lo máximo posible. Obsérvese que ambas tienen el mismo valor al ser evaluadas en $x = 6$. Esto puede verse claramente en la figura 2.1. En la figura 2.2, el punto más cercano a cero es aquél dado por $(f_1,f_2) = (2.8806, 2.8806)$, sin embargo resulta un poco difícil apreciarlo.

El ejemplo que se muestra enseguida trata en forma detallada el procedimiento para diseñar un filtro FIR utilizando el método *Goal Attainment*.

Ejemplo 2.3

Diseñar un filtro FIR pasabajas con las siguientes especificaciones:

R_p = 3 dB (δ_p = 0.171);

A_s = 40 dB (δ_s = 0.01);

ω_p = 0.5;

ω_s = 0.6.

Primero considérese el conjunto de valores deseados como

$$\mathbf{F}^* = \{H_{d1}, H_{d2}, ..., H_{dm}\}, \tag{2.21}$$

donde H_{d1} representa un valor deseado para la frecuencia ω_1, H_{d2} representa un valor deseado para la frecuencia ω_2 y así sucesivamente, para *m* puntos de frecuencia. Para cada valor deseado de respuesta en frecuencia, existe una correspondiente función objetivo que debe ser optimizada para alcanzar ese valor. Entonces el conjunto de funciones objetivo está dado por

$$\mathbf{F}(e^{j\omega}) = \left\{H_1(e^{j\omega_1}), H_2(e^{j\omega_2}), ..., H_m(e^{j\omega_m})\right\}. \tag{2.22}$$

Las funciones objetivo entonces son las expresiones de la respuesta en frecuencia de un filtro FIR, dadas en la tabla 2.1. Todas las funciones dentro del conjunto $\mathbf{F}(e^{j\omega})$ deben ser del mismo tipo (tipo 1 a tipo 4), dependiendo de qué filtro se pretende diseñar. De esta manera, es posible expresar cada elemento *i* del conjunto $\mathbf{F}(e^{j\omega})$ como

$$H_i(e^{j\omega_i}) = \sum_k \alpha(k) Trig(k, \omega_i) \tag{2.23}$$

donde el término *Trig* indica la función trigonométrica adecuada dependiendo del tipo de filtro deseado.

Para las especificaciones mencionadas en el ejemplo, el orden del filtro puede ser estimado de la ecuación (2.7). La longitud que se obtiene es L = 22. Ahora, de la tabla 2.1 debe elegirse la función adecuada para representar el conjunto de funciones objetivo. Debido a que el filtro que se pide es pasabajas, los tipos 3 y 4 de la tabla 2.1 no pueden ser utilizados. Dado que L es par, se elige el tipo 2. La función para un filtro FIR tipo 2 es repetida aquí por conveniencia.

$$H(e^{j\omega}) = \sum_{k=1}^{L/2} b(k)\cos\omega\left(k - \frac{1}{2}\right). \tag{2.24}$$

Obsérvese de (2.22) que se definirá una función objetivo por cada valor de frecuencia dado. Entonces el conjunto de valores de frecuencia será un vector de parámetros para la función que se realizará en MATLAB. Por otro lado, en la ecuación (2.24), k representa el índice de la sumatoria. Así, las variables independientes en esta función son solo el conjunto de coeficientes $b(k)$. De este modo, la función de MATLAB que representará a las funciones objetivo quedará expresada de la forma

$$\mathbf{y} = fun(\mathbf{coef}, \boldsymbol{\omega}), \tag{2.25}$$

donde

- **ω** es el vector de parámetros que contiene los m puntos de frecuencia elegidos.
- **coef** es el vector de coeficientes que se debe hallar para minimizar a la función.
- **y** es el vector de resultados de cada función. El primer elemento de **y** corresponde a la primera función objetivo evaluada en **coef** y ω_1, el

segundo corresponde a la segunda función objetivo evaluada en **coef** y ω_2, y así, hasta llegar a la *m*-ésima función objetivo.

Para realizar este ejemplo se elige *m* = 512, debido a que en el paquete *Signal Processing Toolbox* de MATLAB éste es el número de puntos que está definido por defecto para calcular la respuesta en frecuencia de un filtro con la función `freqz`.

El vector de frecuencias es formado por 512 valores de frecuencia igualmente espaciados, entre 0 y π. Para cada valor de frecuencia debe minimizarse una función objetivo y debe definirse un valor deseado correspondiente a esa función. Se eligen entonces 256 puntos de frecuencia para la banda de paso y 256 puntos para la banda de rechazo. En la banda de paso, los valores deseados son 1 para todas las funciones objetivo correspondientes. En la banda de rechazo los valores deseados son 0. El vector de valores deseados está dado entonces como

$$\mathbf{F}^* = [1,\ 1,\ \ldots,\ 1,\ 0,\ 0,\ \ldots,\ 0]. \tag{2.26}$$

Cada elemento del vector de valores deseados está relacionado además con un valor de tolerancia dado por el vector **w**. Para el diseño de filtros digitales, en [1] se recomienda utilizar $w_i = 1$ en la banda de paso y $w_i = \delta_s/\delta_p$ en la banda de rechazo. Entonces se tiene

$$\mathbf{w} = \left[1,\ 1,\ \ldots,\ 1,\ \frac{\delta_s}{\delta_p},\ \frac{\delta_s}{\delta_p},\ \ldots,\ \frac{\delta_s}{\delta_p}\right]. \tag{2.27}$$

Adicionalmente, se utiliza la función `optimset` para configurar la opción `'GoalsExactAchieve'`. Esta opción está disponible solo para la función `fgoalattain`. Con la opción `'GoalsExactAchieve'` se indica que la

función `fgoalattain` debe alcanzar exactamente los valores deseados en los puntos especificados por `optimset`. Por último, los valores iniciales para el vector **coef** serán incluidos en el vector **coef_ini**, y son elegidos como 1. El código correspondiente para la optimización de las funciones objetivo está dado como

```
opciones = optimset('GoalsExactAchieve',512);
[coef]=fgoalattain(fun,coef_ini,F_asterisco,w,[],[],[],[],[],[], ...
                   [], opciones, omega)
```

Nótese que *fun* es la función definida en (2.25). Ésta depende de los vectores **coef** y **ω**. El primero debe ser hallado para minimizar a *fun*. El segundo solo es vector de parámetros, por eso llega al final de la declaración, como es indicado en (2.20) para la expresión general de `fgoalattain`. Obsérvese además que algunos nombres se han modificado, como $\mathbf{F}^*$ por *f_asterisco* y **ω** por *omega*.

Debe tenerse en cuenta que los valores obtenidos en **coef** no corresponden a los coeficientes del filtro directamente, sino a aquellos representados por $b(k)$ en (2.24). La relación entre los valores obtenidos y los coeficientes del filtro está dada como función lineal [17, 18]. El valor de *gamma* resultante en el proceso de optimización es de 0.0614

Haciendo la relación correspondiente para representar los coeficientes del filtro, se obtiene el conjunto de coeficientes dado en la tabla 2.2. La respuesta en frecuencia obtenida de estos valores es graficada utilizando la función `freqz` del paquete *Signal Processing Toolbox* de MATLAB. Además puede comprobarse que los mismos valores de coeficientes se obtienen si el filtro se diseña utilizando la instrucción `firpm` mencionada antes. Sin embargo, la ventaja del método explicado en este ejemplo radica en que los valores deseados para el filtro pueden ser especificados en cada

correspondiente valor de frecuencia. La figura 2.5 muestra la respuesta en frecuencia obtenida.

$h_0 = h_{23} = -0.0283$	$h_6 = h_{17} = 0.0084$
$h_1 = h_{22} = -0.0504$	$h_7 = h_{16} = 0.0642$
$h_2 = h_{21} = -0.0139$	$h_8 = h_{15} = -0.0330$
$h_3 = h_{20} = 0.0334$	$h_9 = h_{14} = -0.1097$
$h_4 = h_{19} = 0.0040$	$h_{10} = h_{13} = 0.1217$
$h_5 = h_{18} = -0.0444$	$h_{11} = h_{12} = 0.4746$

Tabla 2.2. Coeficientes obtenidos para el ejemplo 2.3.

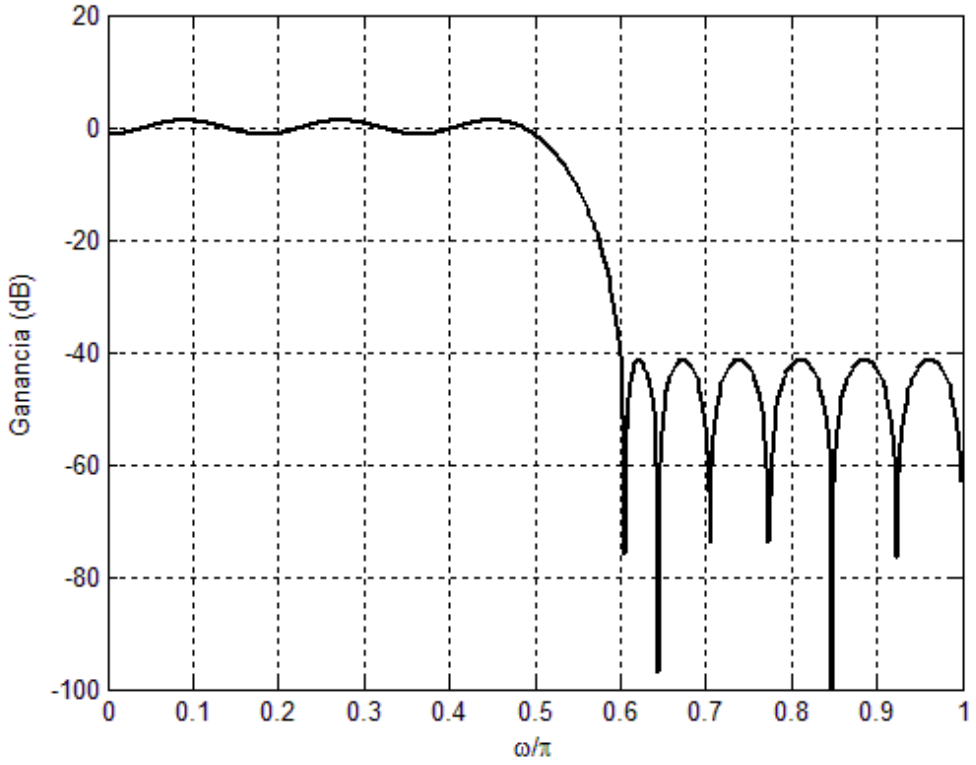


Figura 2.5. Respuesta en frecuencia del filtro diseñado en el ejemplo 2.3

2.3 Filtros IFIR

En un diseño eficiente de filtros FIR se pretende minimizar el número de coeficientes requeridos, la cantidad de operaciones aritméticas realizadas y la complejidad de las mismas. Esto se traduce en menos recursos de hardware para la implementación y menor consumo de potencia. La idea en los filtros IFIR (*Interpolated Finite Impulse Response*) es realizar un filtro eficiente, que resulte mejor que el obtenido de un diseño directo.

Los filtros IFIR permiten tomar ventaja de las características de los sistemas multirazón (*multirate systems*) para hacer un diseño eficiente. Dicho de forma general, un filtro IFIR implementa la función de un filtro con especificaciones dadas, utilizando la conexión en cascada de dos filtros relativamente simples. El filtro resultante exhibe las características requeridas, pero con menos coeficientes que los obtenidos de un diseño directo.

Considérese la figura 2.6. Ésta es la estructura para un filtro IFIR. El enfoque IFIR fue presentado por primera vez en [24]. Debe notarse que la estructura tiene una sola velocidad de muestreo. No obstante, el filtro expandido $G(z^M)$ resulta con pocos coeficientes, con la característica de utilizar M elementos de retraso en lugar de 1.

En la estructura de la figura 2.6, $G(z^M)$ es llamado *filtro modelo expandido en M*. El filtro $I(z)$ es llamado *filtro interpolador*.

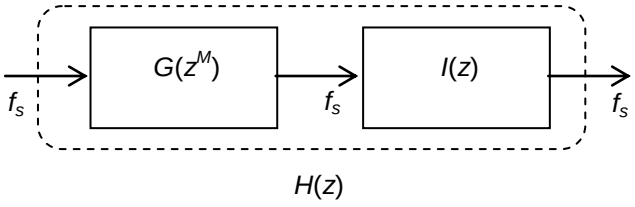


Figura 2.6. Estructura IFIR.

El filtro IFIR es diseñado en tres pasos:

1.- Diseñar el filtro modelo $G(z)$, con las especificaciones dadas por

$$\begin{aligned} \omega_{p,G} &= M \cdot \omega_p, \\ \omega_{s,G} &= M \cdot \omega_s, \\ R_{p,G} &= \frac{R_p}{2}, \\ A_{s,G} &= A_s, \end{aligned} \tag{2.28}$$

donde ω_p y ω_s son las respectivas frecuencias límite de paso y rechazo para el filtro deseado, y R_p y A_s son los rizos en las bandas de paso y rechazo en dB.

2.- Realizar el filtro modelo expandido en M, $G(z^M)$, sustituyendo cada elemento de retraso en $G(z)$ por M elementos de retraso. Esto significa que entre cada muestra de la respuesta al impulso del filtro modelo deben introducirse M-1 muestras de valor cero.

3.- Diseñar el filtro interpolador $I(z)$, con las especificaciones dadas por

$$\begin{aligned} \omega_{p,I} &= \omega_p, \\ \omega_{s,I} &= \frac{2\pi}{M} - \omega_s, \\ R_{p,I} &= \frac{R_p}{2}, \\ A_{s,I} &= A_s, \end{aligned} \tag{2.29}$$

donde ω_p, ω_s, R_p y A_s son los parámetros especificados para el filtro deseado.

Las respuestas en frecuencia para los filtros $G(z)$, $G(z^M)$, $I(z)$ y $H(z)$ son representadas en la figura 2.7. Nótese que la función del filtro interpolador es eliminar las imágenes introducidas por el filtro expandido $G(z^M)$. Es posible ver además que el enfoque IFIR puede aprovecharse para valores desde M = 2 en adelante. Debido a esto, el máximo valor relativo para ω_s es 0.5π; por lo tanto, se trata de un enfoque de diseño de filtros FIR de banda angosta.

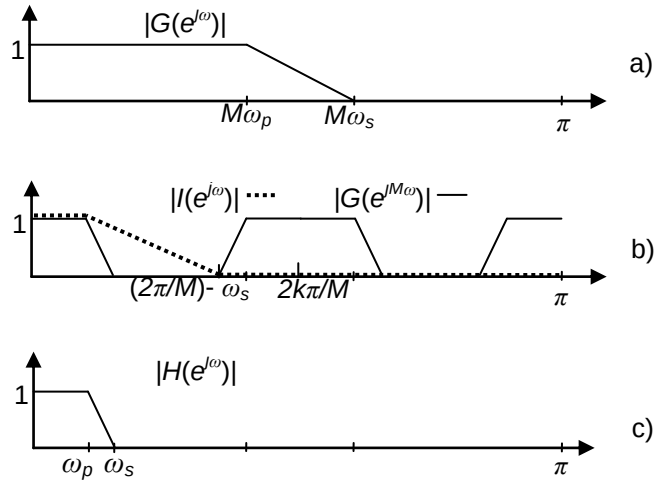


Figura 2.7. Respuestas en frecuencia de: a) Filtro Modelo, b) Filtro modelo expandido en M y filtro interpolador, c) Cascada del filtro modelo expandido con el interpolador.

Ejemplo 2.4

Diseñar un filtro FIR pasabajas con estructura IFIR, utilizando M = 5, con las siguientes especificaciones:

R_p = 0.1 dB;

A_s = 60 dB;

ω_p = 0.05;

ω_s = 0.1.

Comparar la complejidad del filtro resultante con la de un diseño directo.

A continuación se muestran los pasos a seguir en el diseño del filtro.

1. Se diseña el filtro modelo con las características dadas por (2.28), a saber:

   $R_{p,G}$ = (0.1) / 2 dB;

   $A_{s,G}$ = 60 dB;

   $\omega_{p,G}$ = 5*(0.05);

   $\omega_{s,G}$ = 5*(0.1).

La longitud del filtro modelo es L = 26. La figura 2.8a muestra la respuesta en frecuencia correspondiente.

2. Se realiza el filtro modelo expandido en 5 introduciendo 4 muestras de valor cero entre cada muestra del filtro obtenido en el paso 1. En la figura 2.8b se presenta la respuesta en frecuencia correspondiente.

3. Se diseña el filtro interpolador con las características dadas por (2.29). Éstas son:

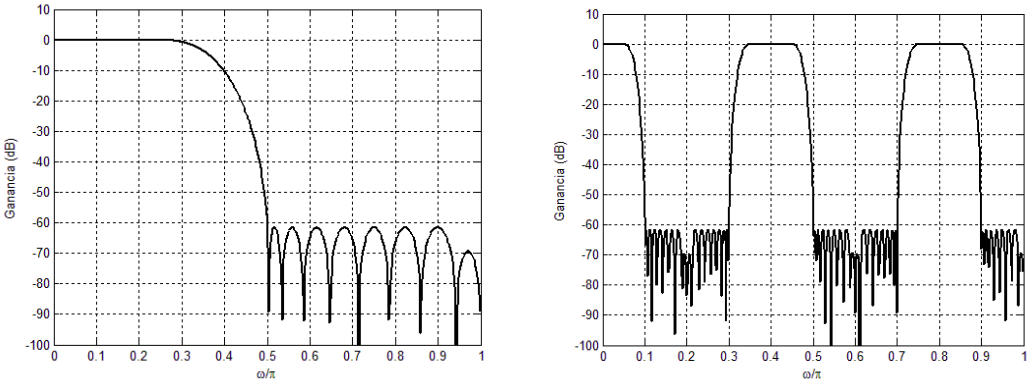
$R_{p,I}$ = (0.1) / 2 dB;

$A_{s,I}$ = 60 dB;

$\omega_{p,I}$ = 0.05;

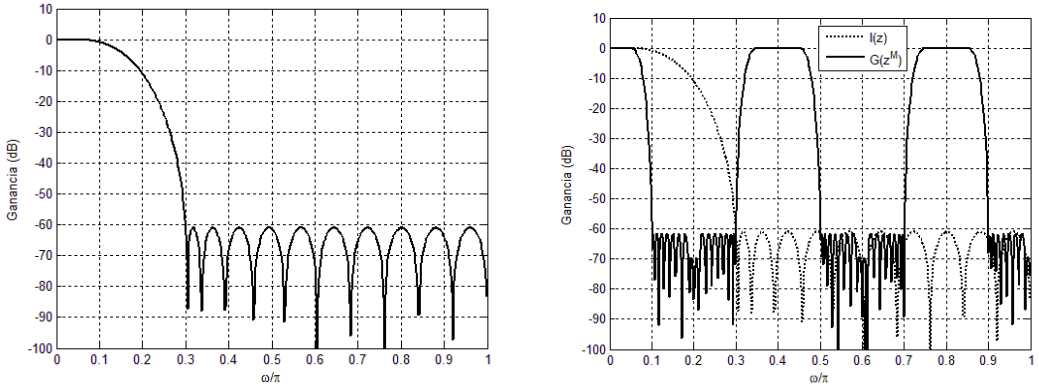
$\omega_{s,I}$ = (2/5) - (0.1).

La longitud del filtro interpolador es L = 26. La figura 2.9a muestra la respuesta en frecuencia correspondiente. La figura 2.9b muestra las respuestas en frecuencia de los filtros modelo expandido e interpolador. La figura 2.10 exhibe la respuesta en frecuencia de la conexión en cascada de ambos filtros. Obsérvese que se obtiene el resultado deseado.



a) b)

Figura 2.8. Respuesta en frecuencia de: a) filtro modelo $G(z)$, b) filtro modelo expandido $G(z^M)$, con M = 5.



a) b)

Figura 2.9. Respuesta en frecuencia de: a) filtro interpolador $I(z)$, b) filtro interpolador $I(z)$ y filtro modelo expandido $G(z^M)$, con $M = 5$.

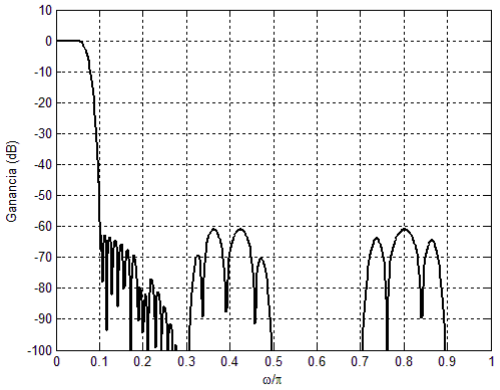


Figura 2.10. Respuesta en frecuencia del filtro $H(z)$, resultado de la conexión en cascada de $G(z^M)$ e $I(z)$.

Por otra parte, la longitud del filtro diseñado directamente es $L = 111$. Puede verse entonces que existe un ahorro de aproximadamente 50 % en coeficientes de filtro utilizando la estructura IFIR.

La tabla 2.3 muestra la comparación hecha entre el filtro obtenido de la estructura IFIR y aquél realizado por diseño directo. Es posible concluir que el enfoque IFIR permite realizar diseño de filtros de banda angosta en forma eficiente.

Enfoque de diseño	Longitud *L*	No. De coeficientes	Porcentaje de Ahorro
Diseño directo	111	56	-
Estructura IFIR	26 + 26 = 52	13 + 13 = 26	53.5 %

Tabla 2.3. Comparación de un diseño directo y un diseño con estructura IFIR.

2.4 Filtros simples sin multiplicadores

Se ha mencionado que la finalidad en un diseño eficiente de filtros digitales es minimizar los recursos de hardware requeridos. Además es sabido que los filtros con banda de transición angosta son más complejos de realizar. Así pues, los métodos para diseñar estos filtros de forma eficiente se basan en técnicas multirazón. Dicho de forma general, estas técnicas utilizan la combinación de filtros simples y bloques de cambio de velocidad de muestreo para llegar a un diseño que cumple con las características requeridas y a la vez demanda menos recursos que un diseño directo.

Existen filtros digitales muy simples que no requieren multiplicadores debido a que sus coeficientes son unitarios. Si bien su respuesta en frecuencia no es muy buena, en muchas ocasiones resulta adecuada para varios tipos de aplicaciones. Una explicación breve pero concisa respecto a estos filtros se presenta en las sub-secciones siguientes.

2.4.1 Filtro de Hogenauer

Se trata de un filtro de fase lineal propuesto por E. B. Hogenauer [25]. Este filtro consta de la conexión en cascada de un integrador y un diferenciador. A menudo también es llamado CIC, que significa *Cascade Integrator-Comb*.

La sección del integrador es una estructura recursiva de un solo polo, cuya función de transferencia está dada por

$$H_I(z) = \frac{1}{1 - z^{-1}}. \tag{2.30}$$

La sección del diferenciador es una estructura no recursiva, con su función de transferencia expresada como

$$H_D(z) = 1 - z^{-1}. \tag{2.31}$$

El filtro de Hogenauer resulta de la conexión en cascada de un integrador y un diferenciador, separados por un bloque reductor de velocidad de muestreo por un factor entero *M*. Debido al cambio en la velocidad de muestreo, debe introducirse un factor M^{-1} en el integrador, para expresar la ganancia en forma unitaria. La función de transferencia de este filtro está dada por

$$H_{CIC}(z) = \frac{1}{M} \cdot \frac{1 - z^{-M}}{1 - z^{-1}}. \tag{2.32}$$

La conexión en cascada de *K* filtros CIC tiene la función de transferencia expresada como

$$H^K{}_{CIC}(z) = \left(\frac{1}{M} \cdot \frac{1 - z^{-M}}{1 - z^{-1}} \right)^K. \tag{2.33}$$

La figura 2.11a muestra la estructura correspondiente a la ecuación 2.32. La figura 2.11b presenta la estructura para *K* filtros CIC conectados en cascada.

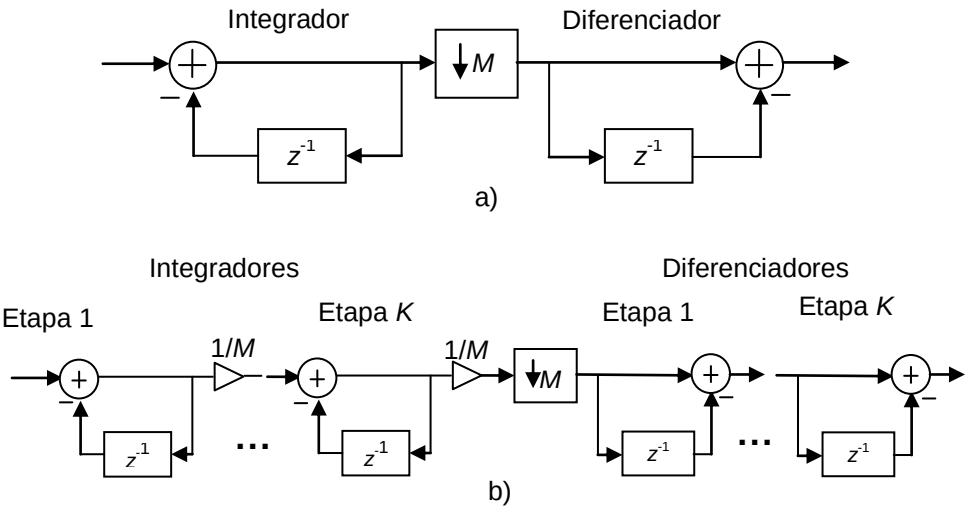


Figura 2.11. Estructuras de un filtro de Hogenauer: a) estructura simple, b) estructura con *K* etapas simples en cascada.

La respuesta en frecuencia de un filtro de Hogenauer está dada por

$$H_{CIC}(e^{j\omega}) = \left(e^{-\frac{j\omega(M-1)}{2}} \cdot \frac{\sin(\omega M/2)}{M \cdot \sin(\omega/2)} \right). \tag{2.34}$$

La figura 2.12 muestra las respuestas en frecuencia de dos filtros CIC, con diferentes valores de *M* y *K*. Nótese que la respuesta en frecuencia tiene un comportamiento pasa bajas. Además exhibe valores nulos en

$$\omega = \frac{2\pi k}{M}, \quad \text{con } k = 1,2,...,M-1. \tag{2.35}$$

Las ventajas de los filtros CIC son las siguientes:

- No utiliza multiplicadores.
- Utiliza pocos recursos de suma y almacenamiento.

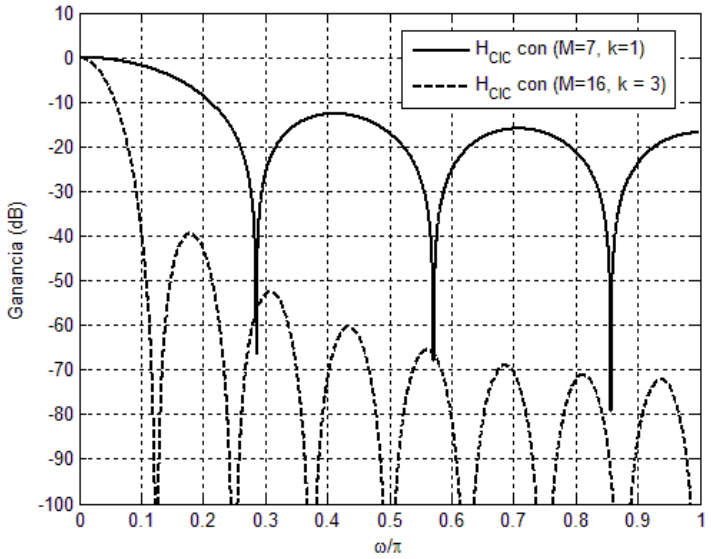


Figura 2.12. Respuesta en magnitud de dos filtros CIC con diferentes valores de *M* y *K*.

- Los valores nulos de respuesta en frecuencia lo hacen ideal para cancelar imágenes en filtros periódicos de banda angosta con el mismo factor de interpolación *M*.

Las desventajas de los filtros CIC se enlistan a continuación.

- Poca atenuación en la banda de rechazo.
- Caída en la banda de paso muy pronunciada.

La atenuación en los filtros CIC puede hacerse más alta conectando más filtros en cascada, pero la caída en banda de paso se intensifica.

Se han ideado varios métodos con la finalidad de disminuir la caída en la banda de paso. Algunos de ellos se enlistan a continuación.

- *Compensadores de segundo orden*. En estos métodos, se diseña un filtro compensador de segundo orden para mejorar la banda de paso del filtro CIC. El filtro compensador de segundo orden es de muy poca complejidad. Conectando en cascada el filtro

compensador con el filtro CIC, se logra disminuir la caída en banda de paso de este último notablemente. La desventaja es que en la banda de rechazo la atenuación disminuye un poco. En [26] y [27] se han realizado propuestas de filtros compensadores con muy poca complejidad, que no requieren multiplicadores para su implementación y mejoran en gran medida la banda de paso de los filtros CIC.

- *Sharpening*. Esta técnica, presentada en [28], es frecuentemente usada para mejorar simultáneamente la banda de paso y la banda de rechazo de los filtros CIC. Está basada en la interconexión del filtro CIC con otros iguales a él en diferentes formas, utilizando una función de cambio de amplitud. La función de cambio de amplitud indica cómo deben conectarse los filtros CIC para mejorar, en mayor o menor medida, las bandas de paso y de rechazo. La desventaja del método recae en la complejidad del filtro resultante, debido a que incluye la implementación de varios filtros CIC iguales en lugar de uno solo.

2.4.2 Filtro Coseno

Considérese la función de transferencia dada como

$$H_{\cos}(z) = \frac{1}{2}\left(1 + z^{-N}\right), \quad \text{con } N \text{ entero.} \tag{2.36}$$

Puede notarse que este filtro no tiene multiplicadores, debido a que sus coeficientes son unitarios. El factor (1/2) es introducido para expresar la respuesta en frecuencia del filtro en forma normalizada.

La respuesta en magnitud de este filtro está dada por

$$\left|H_{\cos}(e^{j\omega})\right| = \left|\cos\left(\frac{N\omega}{2}\right)\right|. \tag{2.37}$$

Debido a la ecuación anterior, este filtro es llamado filtro Coseno. La figura 2.13 muestra las respuestas en frecuencia de dos filtros coseno para N = 6 y N= 10.

Utilizando la conexión en cascada de filtros Coseno con diferentes valores de N, como es propuesto en [29], es posible obtener una respuesta en frecuencia total con comportamiento pasabajas. La expresión para la conexión cascada de filtros Coseno con diferentes valores enteros de N está dada por

$$\begin{aligned} &H_{C\cos}(i,N_1,N_2,...N_i,z) = H_{\cos}(z^{N_1})H_{\cos}(z^{N_2})...H_{\cos}(z^{N_i}), \\ &\text{con } N_1 > N_2 > ... > N_i. \end{aligned} \tag{2.38}$$

El valor principal N_1 es calculado de la siguiente manera,

$$N_1 \le \left\lfloor \frac{\pi}{\omega_s} + 1 \right\rfloor, \quad \text{con } \omega_p = 0.1\omega_s. \tag{2.39}$$

Ejemplo 2.5

Realizar un filtro pasa bajas con ω_p = 0.03 y ω_s = 0.3.

De la ecuación (2.39) se tiene que N_1 = 4. Utilizando 3 filtros adicionales con N_2 = 3, N_3 = 2 y N_4 = 1 se obtiene un filtro con característica pasa bajas en su respuesta en frecuencia. La figura 2.14 presenta la respuesta en frecuencia resultante.

El filtro obtenido tiene R_p = 0.3dB y A_s = 16 dB. Comparando con un filtro con estas características diseñado directamente, el filtro basado en Cosenos

requiere menos recursos. No obstante, la respuesta en frecuencia como filtro pasa bajas en general es deficiente y necesita muchas etapas para ser mejorada.

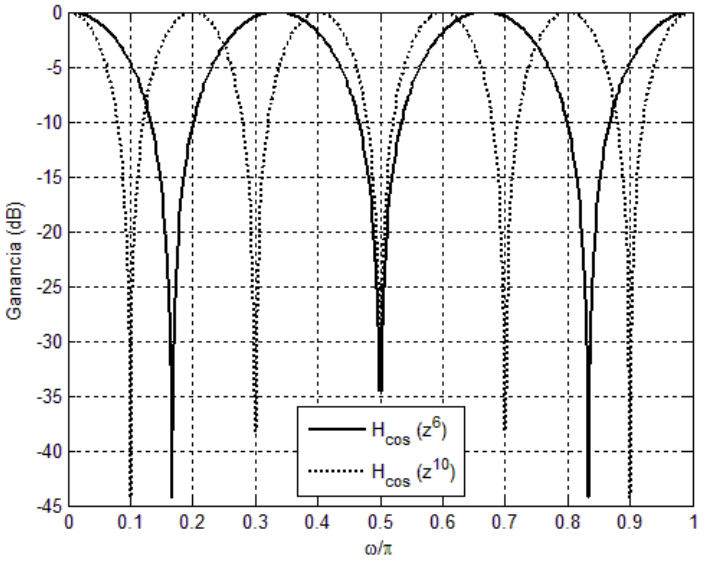


Figura 2.13. Respuesta en frecuencia de filtros Coseno con $N = 6$ y $N = 10$.

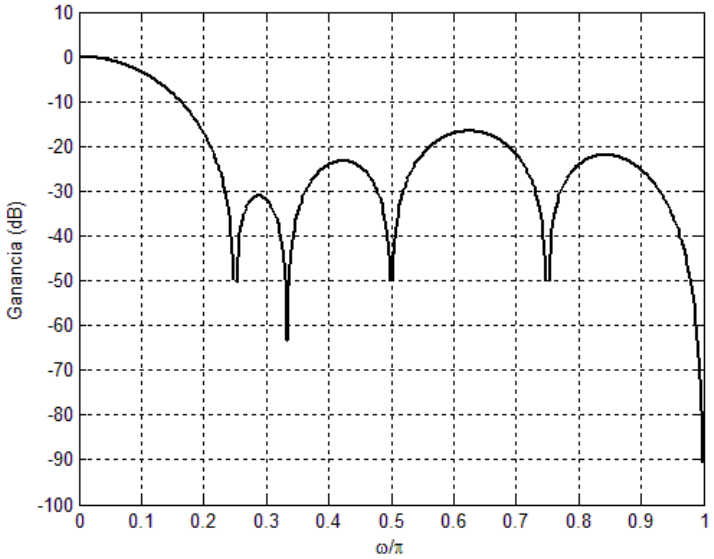


Figura 2.14. Respuesta en frecuencia de 4 filtros Coseno en cascada, con $N_1 = 4$, $N_2 = 3$, $N_3 = 2$ y $N_4 = 1$.

2.4.3 Filtro Coseno Modificado

El filtro propuesto en [30] está formado por la suma de un filtro Coseno con un filtro Coseno al cuadrado. Este último es obtenido de la conexión en cascada de dos filtros Coseno. Debido a que está basado en filtros Coseno, el filtro propuesto en [30] es llamado filtro Coseno Modificado.

De la ecuación (2.37) se tiene que la amplitud de la respuesta en frecuencia de un filtro Coseno con $N = 2$ exhibe valores positivos en frecuencias dentro del intervalo $[0, \pi/2]$. Dentro del intervalo $[\pi/2, \pi]$ sus valores de amplitud de respuesta en frecuencia son negativos. La idea en un filtro Coseno Modificado es utilizar un filtro cuya amplitud de respuesta en frecuencia exhiba valores positivos en el intervalo $[\pi/2, \pi]$. Conectando dicho filtro en paralelo con el filtro Coseno se pretende cancelar a los valores negativos de este último, formando una respuesta en frecuencia con característica pasa bajas.

El filtro que presenta valores positivos de amplitud de respuesta en frecuencia en ambos intervalos, $[0, \pi/2]$ y $[\pi/2, \pi]$, es el filtro con función coseno al cuadrado. Así pues, la función de transferencia de un filtro Coseno Modificado está representada como

$$H_{\cos\_M}(z) = \frac{1}{2}\left(H_{\cos}(z^N) + H_{\cos}^2(z^N)\right) = \frac{1}{2}\cdot H_{\cos}(z^N)\cdot\left(1 + H_{\cos}(z^N)\right). \qquad (2.40)$$

La estructura de un filtro Coseno Modificado puede verse en la figura 2.15. Obsérvese que, para igualar los retrasos en ambas líneas, es necesario introducir un retraso en la línea inferior. Nótese que N solo puede tomar valores pares positivos para evitar que el retraso sea fraccionario.

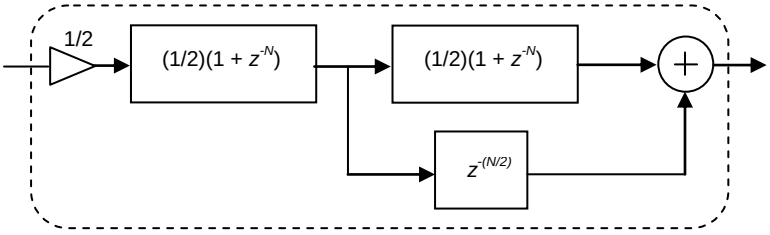


Figura 2.15. Estructura de un filtro Coseno Modificado.

Cuando se pretende mejorar la atenuación en la banda de rechazo, se conectan *K* filtros Coseno Modificado en cascada. La respuesta en frecuencia de estos filtros está dada por

$$H_{\cos\_M}(e^{j\omega}) = \left[\frac{1}{2}\left(\cos\left(\frac{N\omega}{2}\right) + \cos^2\left(\frac{N\omega}{2}\right)\right)\right]^K . \qquad (2.41)$$

Las respuestas en frecuencia para dos filtros Coseno Modificado con diferentes valores de *N* y *K* se presentan en la figura 2.16.

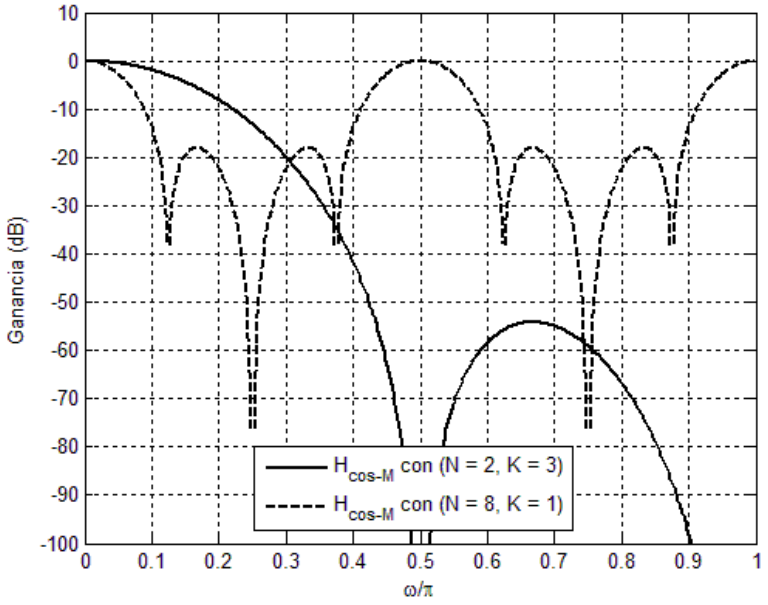


Figura 2.16. Respuesta en frecuencia de dos filtros Coseno Modificado con diferentes valores de *N* y *K*.

El filtro Coseno Modificado exhibe una respuesta en frecuencia con característica pasa bajas para $N = 2$. Además, la atenuación es mejorada en gran manera al aumentar el número K. Por otra parte, es posible notar que para $N > 2$, su respuesta en frecuencia tiene imágenes introducidas en el intervalo $[0, \pi]$.

Capítulo 3

Síntesis de coeficientes como suma de potencias de dos

Porque en la mucha sabiduría hay mucha molestia; y quien añade ciencia, añade dolor. *Ecclesiastés* 1:18

A través de este capítulo se presentarán algunas técnicas para obtener la representación de coeficientes de filtros como suma de potencias de dos. Además se explicarán las representaciones CSD (Canonical Signed Digit) y MSD (Minimal Signed Digit) y su aplicación en filtros digitales. Por último, se explicarán algunos métodos de eliminación de sub-expresiones comunes en la representación de coeficientes.

3.1 Obtención de coeficientes de precisión finita

En el diseño de filtros digitales, después de llevar a cabo el proceso de optimización, los coeficientes obtenidos resultan casi siempre de *precisión infinita*. Esto significa que se necesitaría una cantidad infinita de bits para representar esos valores de coeficientes en forma binaria. Debido a esto, los coeficientes obtenidos deben ser cuantizados para poder ser representados con palabra de longitud finita. Este proceso es llamado *síntesis de coeficientes*.

Los filtros digitales, junto con todos los sistemas digitales que desarrollan cálculos y operaciones aritméticas, deben ser clasificados entre sistemas de punto fijo o sistemas de punto flotante. La decisión entre aritmética de punto fijo o de punto flotante depende del tipo de sistema digital en cuestión. En general, se asume que los sistemas de punto fijo tienen más alta velocidad y requieren menos recursos de implementación. No obstante, los sistemas de punto flotante tienen más alto rango dinámico y pueden desempeñar algoritmos y operaciones más complicados, a expensas de consumir más recursos de hardware [31].

Los filtros digitales a menudo resultan adecuados para implementaciones en punto fijo, debido a que el proceso de convolución es únicamente desarrollado como una suma acumulativa de productos parciales. Debido a esto, en este capítulo solamente se abordarán temas relacionados con la generación y representación de los coeficientes en punto fijo.

Las sub-secciones siguientes presentarán los métodos más usuales para obtener coeficientes de precisión finita.

3.1.1 Método de redondeo (Rounding)

La técnica de redondeo presentada aquí está basada en la siguiente ecuación [32],

$$g(n) = \alpha \cdot g_i(n) = \alpha \cdot \left[h(n) / \alpha \right], \tag{3.1}$$

donde

- $h(n)$ es la respuesta al impulso con valores de precisión infinita de un filtro FIR que satisface ciertas especificaciones.
- α es un valor elegido de la forma 2^{-N}, que determina la aproximación de los coeficientes redondeados a los coeficientes de precisión infinita.
- $\left[\ \right]$ indica la operación de redondeo hacia el entero más cercano.

- $g_i(n)$ es la respuesta al impulso obtenida de la operación de redondeo, expresando los coeficientes en valores enteros.
- $g(n)$ es la respuesta al impulso obtenida de la operación de redondeo, escalada por α para expresar la ganancia en dB en la forma $(0 \pm R_p)$, con R_p como rizo en la banda de paso.

Debe notarse aquí que los coeficientes obtenidos en $g(n)$ son expresados en palabra de longitud finita, cuya precisión está determinada por α. Debido a que $g(n)$ representa coeficientes enteros escalados por una potencia de dos, los coeficientes obtenidos están expresados en representación de *Suma de Potencias de Dos*, *SOPOT* (*Sum-Of-Powers-Of-Two*). En la sección 3.2 se verá la importancia de tener representaciones *SOPOT* para los coeficientes de un filtro digital. Por el momento basta decir que esta representación permite implementaciones sin multiplicadores.

Ejemplo 3.1

Considérese un filtro FIR pasabajas con las siguientes especificaciones:

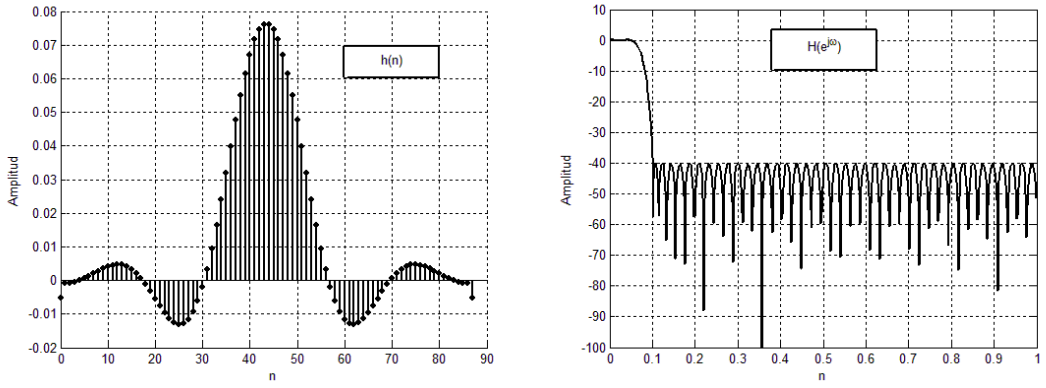
$R_p = 0.1$ dB;

$A_s = 40$ dB;

$\omega_p = 0.05$;

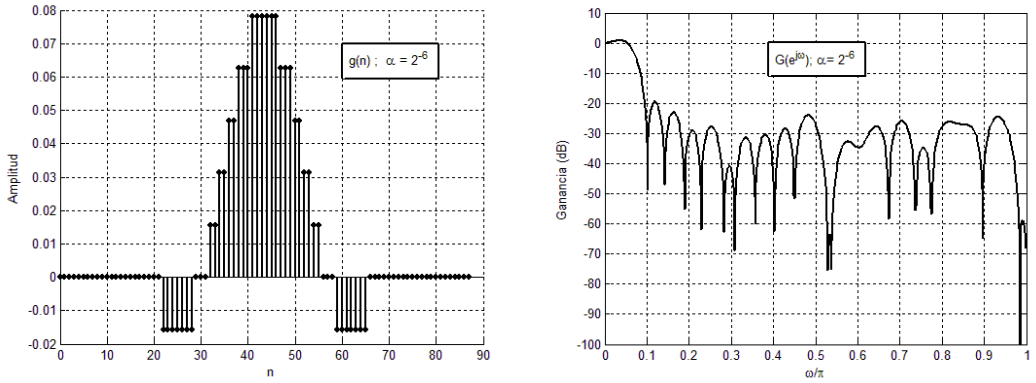
$\omega_s = 0.1$.

Para este filtro se calcula la respuesta al impulso $h(n)$ con el algoritmo Parks-McClellan, obteniéndose 88 coeficientes de precisión infinita. La figura 3.1a muestra la respuesta al impulso obtenida. La respuesta en frecuencia correspondiente $H(e^{j\omega})$ se exhibe en la figura 3.1b. La respuesta al impulso $g_1(n)$ obtenida con una constante de redondeo $\alpha = 2^{-6}$ se presenta en la figura 3.2a y la respuesta en frecuencia correspondiente $G_1(e^{j\omega})$ se muestra en la figura 3.2b. Las figuras 3.3a y 3.3b exhiben, respectivamente, la respuesta al impulso $g_2(n)$ y la respuesta en frecuencia $G_2(e^{j\omega})$ obtenidas con una constante de redondeo $\alpha = 2^{-10}$.



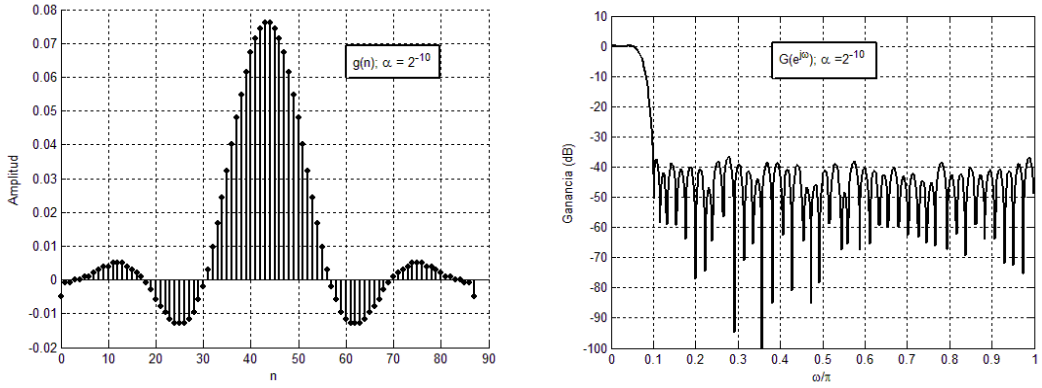
a) b)

Figura 3.1. a) Respuesta al impulso $h(n)$ con coeficientes de precisión infinita, b) Respuesta en frecuencia $H(e^{j\omega})$.



a) b)

Figura 3.2. a) Respuesta al impulso $g_1(n)$ con coeficientes de precisión $\alpha = 2^{-6}$, b) Respuesta en frecuencia $G_1(e^{j\omega})$.



a) b)

Figura 3.3. a) $g_2(n)$ con coeficientes de precisión $\alpha = 2^{-10}$, b) $G_2(e^{j\omega})$.

La técnica de redondeo implica cierto comportamiento en la respuesta al impulso de un filtro cuando se asignan diferentes valores de α. De igual manera, la respuesta en frecuencia exhibe cambios notables. Considérense los siguientes puntos:

- El proceso de redondeo introduce coeficientes de valor cero en la respuesta al impulso redondeada. Entonces el número de coeficientes con valor diferente de cero corresponde al número de sumadores utilizados en el filtro. Este número disminuye cuando el valor de α aumenta.
- Algunos de los valores de coeficientes que son distintos de cero tienen el mismo valor. Por lo tanto el número de coeficientes distintos en la respuesta al impulso redondeada disminuye, en comparación con la original. La cantidad de valores diferentes (sin contar el valor de α y su correspondiente negativo en $g(n)$, o bien, sin contar +1 y -1 en $g_i(n)$) corresponde al número de multiplicadores enteros necesarios en el filtro. Este número disminuye cuando el valor de α aumenta.
- Aplicar redondeo en la respuesta al impulso de un filtro implica que su respuesta en frecuencia experimentará cierto deterioro. Esta declinación en el comportamiento deseado se intensifica cuando aumenta el valor de α.

Obsérvese que cuando el valor de α aumenta, el número de coeficientes distintos utilizados en el filtro disminuye, así como el número de sumas requeridas. Sin embargo, esto provoca más deterioro en la respuesta en frecuencia correspondiente. Debido a esto, es necesario hacer una elección adecuada de la constante de redondeo α, tal que resulte en un balance apropiado entre el número de sumas y coeficientes necesarios para el filtro y, a la vez, una respuesta en frecuencia adecuada.

3.1.2 Programación Lineal Entera Mixta (MILP)

Se ha visto que el proceso de redondeo de los coeficientes de la respuesta al impulso de un filtro dado deteriora notablemente la respuesta en frecuencia del mismo. De hecho, se ha demostrado que la respuesta al impulso de filtros FIR con coeficientes de precisión finita obtenidos por redondeo lleva a un resultado distinto del óptimo [33, 34].

En [34] se presenta el diseño de filtros FIR planteado como un problema de *Programación Lineal Entera Mixta* (*MILP: Mixed Integer Linear Programming*). Básicamente, en este enfoque se encuentra la solución óptima cuando el problema tiene como restricción principal que algunas de sus variables independientes solo deben ser números enteros. Utilizando el método *MILP* en el diseño de filtros FIR, pueden encontrarse valores de coeficientes directamente factibles para representar con palabra de longitud finita, consiguiendo a la vez la mejor respuesta en frecuencia posible.

Considérese el problema de minimización dado como

$$\begin{aligned} &\text{minimizar } \delta \\ &\text{tal que } \quad H(\omega) - \delta w(\omega) \le H_d(\omega), \\ &\qquad\qquad H(\omega) + \delta w(\omega) \ge H_d(\omega), \\ &\text{con } w(\omega) = \begin{cases} 1 & \text{para } \omega \in [0, \omega_p], \\ \dfrac{\delta_s}{\delta_p} & \text{para } \omega \in [\omega_s, \pi], \end{cases} \end{aligned} \tag{3.2}$$

donde

- δ es la tolerancia del error entre la respuesta en frecuencia deseada y la respuesta en frecuencia actual.
- $H(\omega)$ es el conjunto de valores de respuesta en frecuencia que debe acercarse al valor deseado lo máximo posible al terminar el proceso de optimización.
- $H_d(\omega)$ es el conjunto de valores de respuesta en frecuencia deseados.

- $w(\omega)$ es el conjunto de ponderaciones de error para cada valor de frecuencia.
- ω_p y ω_s son las frecuencias límite de paso y de rechazo, respectivamente.
- δ_p y δ_s son los rizos en la banda de paso y de rechazo, respectivamente.

Tomando en cuenta que cada función objetivo puede ser evaluada en al menos 4*L* puntos de frecuencia, con *L* siendo la longitud del filtro, es posible plantear (3.2) como un problema de programación lineal. Para cada valor de frecuencia ω_i, se tiene una función de restricción que incluye a la función $H(\omega)$ dada por la ecuación (2.23). Cada función es una sumatoria que puede ser desglosada exhibiendo individualmente a todos los parámetros del filtro, $\alpha(k)$. Así pues, el problema de programación lineal quedaría expresado como

$$\begin{aligned}
&\text{minimizar } \delta \\
&\text{tal que } \sum_k \alpha(k)Trig(k,\omega_i) - \delta w(\omega_i) \le H_d(\omega_i), \\
&\qquad -\sum_k \alpha(k)Trig(k,\omega_i) - \delta w(\omega_i) \le -H_d(\omega_i); \quad i = 1,2,...,4L; \\
&\text{con } w(\omega) = \begin{cases} 1 & \text{para } \omega \in [0,\omega_p]; \\ \dfrac{\delta_s}{\delta_p} & \text{para } \omega \in [\omega_s,\pi]. \end{cases}
\end{aligned} \tag{3.3}$$

Recuérdese que *k* aumenta desde cero hasta un valor *K* que está en función de *L*, dependiendo del tipo de filtro (véase la tabla 2.1). Las restricciones del problema de optimización planteado en (3.3) pueden ser agrupadas en forma matricial, como **Ax** ≤ **b**. Desglosando las funciones para cada valor de ω_i y *k* se tiene:

$$\mathbf{x} = [\alpha(0),\alpha(1),...\alpha(K),-\delta]^T, \tag{3.4a}$$

$$\mathbf{b} = [H_d(\omega_1), H_d(\omega_2), \ldots H_d(\omega_{4L}), -H_d(\omega_1), -H_d(\omega_2), \ldots, -H_d(\omega_{4L})]^T, \qquad (3.4b)$$

$$\mathbf{A} = \begin{bmatrix} Trig(\omega_1,\alpha(0)) & Trig(\omega_1,\alpha(1)) & \ldots & Trig(\omega_1,\alpha(K)) & w(\omega_1) \\ Trig(\omega_2,\alpha(0)) & Trig(\omega_2,\alpha(1)) & & Trig(\omega_2,\alpha(K)) & w(\omega_2) \\ \vdots & & & & \\ Trig(\omega_{4L},\alpha(0)) & Trig(\omega_{4L},\alpha(1)) & & Trig(\omega_{4L},\alpha(K)) & w(\omega_{4L}) \\ -Trig(\omega_1,\alpha(0)) & -Trig(\omega_1,\alpha(1)) & & -Trig(\omega_1,\alpha(K)) & w(\omega_1) \\ -Trig(\omega_2,\alpha(0)) & -Trig(\omega_2,\alpha(1)) & & -Trig(\omega_2,\alpha(K)) & w(\omega_2) \\ \vdots & & & & \\ -Trig(\omega_{4L},\alpha(0)) & -Trig(\omega_{4L},\alpha(1)) & & -Trig(\omega_{4L},\alpha(K)) & w(\omega_{4L}) \end{bmatrix}_{(8L)\times(K+2)}, \qquad (3.4c)$$

$$K = \begin{cases} (L-1)/2 & \text{para filtros de longitud impar,} \\ (L)/2 & \text{para filtros de longitud par.} \end{cases} \qquad (3.4d)$$

Obsérvese en (3.4a) que, para poder obtener coeficientes de precisión finita en el resultado final, los valores $\alpha(1)$, $\alpha(2)$, …, $\alpha(K)$ deben ser enteros. Las restricciones deben ser escaladas adecuadamente por una potencia de dos si se desea obtener un resultado de valores con ganancia normalizada. Por otro lado, el valor δ solo proporciona una medida de tolerancia en el error, por lo tanto puede ser real. Así pues, se pide que en el vector **x** la mayoría de las variables independientes sean enteras, excepto δ. Entonces se trata de un problema de Programación Lineal Entera Mixta.

Para resolver problemas *MILP* uno de los métodos más efectivos es *Branch & Bound* (*B & B*) [21]. El procedimiento consiste en optimizar la función objetivo en un espacio continuo, como en el problema planteado en (3.3). Esto significa que no se consideran las restricciones de valor entero para los coeficientes. Si al resolver el problema inicial los valores obtenidos para los coeficientes son enteros, el problema se ha resuelto. Si se han alcanzado solo valores reales, entonces se elige una de las variables independientes y se redondea a los valores enteros más cercanos por defecto y por exceso. Cada nuevo valor de dicha variable independiente se

añade como restricción al problema original, dando lugar a un nuevo planteamiento que debe resolverse. Se sigue de la misma manera en cada nueva solución si vuelven a encontrarse algunos valores reales. Este procedimiento da lugar a dos problemas distintos por cada variable real que es redondeada y establecida como restricción adicional. Debido a lo anterior, este procedimiento es llamado *branching* (ramificación).

De todos los problemas generados, solo algunos deben resolverse. El criterio de decisión más adecuado es llamado *depth-first* (prioridad a la profundidad) [21]. En este enfoque, se resuelven los dos problemas derivados de la solución original. De ambos, el que obtuvo mejor resultado se elige para generar dos nuevos problemas, mientras que el otro se deja pendiente. Los dos nuevos problemas se resuelven y se comparan con la solución que fue aplazada. La peor solución de las tres es eliminada, la segunda se deja pendiente y la primera se continúa explorando. Este proceso continúa hasta que se obtengan las mejores soluciones enteras. Debido a que algunas ramas se eliminan, este procedimiento es llamado *bounding* (delimitación).

Obsérvese la figura 3.4. El problema P_0 es planteado originalmente como en (3.3). Suponiendo que las soluciones obtenidas $\alpha(0)$, $\alpha(1)$, …, $\alpha(K)$, δ fueron reales, entonces se elige alguna variable $\alpha(i)$ y se redondea como

$$\lfloor \alpha(i) \rfloor \leq \alpha(i) \leq \lceil \alpha(i) \rceil, \tag{3.5}$$

donde los símbolos $\lfloor \ \rfloor$ y $\lceil \ \rceil$ significan redondeo al entero más cercano por defecto y por exceso, respectivamente. Los problemas P_1 y P_2 son planteados como P_0, con las restricciones adicionales dadas por

$$\alpha(i) \leq \lfloor \alpha(i) \rfloor \quad \text{para } P_1, \tag{3.6a}$$

$$\alpha(i) \geq \lceil \alpha(i) \rceil \quad \text{para } P_2. \tag{3.6b}$$

De las soluciones obtenidas de P_1 y P_2 se elige la que haya proporcionado mejor resultado (más cercano al óptimo logrado en P_0). Enseguida se selecciona una nueva variable $\alpha(j)$ que no haya sido alcanzada con valor entero. Ésta se redondea como en (3.5) y se da lugar a dos nuevos problemas, P_3 y P_4, como en (3.6). El procedimiento continúa siguiendo el criterio *depth-first*, hasta que la solución óptima sea encontrada, cumpliendo las restricciones de valor entero para $\alpha(0)$, $\alpha(1)$, …, $\alpha(K)$.

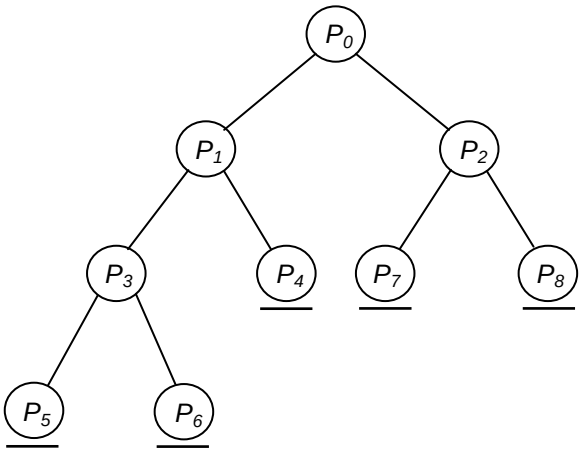


Figura 3.4. Ejemplo de un árbol formado por el método *Branch & Bound*.

El enfoque *MILP* para el diseño de filtros FIR proporciona mejores resultados que el proceso de redondeo directo. No obstante, con *MILP* el tiempo invertido en el diseño es relativamente alto, es decir, la carga computacional del algoritmo es muy costosa. Ésta es la principal desventaja del planteamiento de un diseño con *MILP*. Cuando se utiliza el criterio *depth-first*, es posible obtener soluciones sub-óptimas durante el proceso. Esto proporciona una ventaja cuando el tiempo de diseño requerido por el algoritmo debe restringirse.

3.2 Representación de coeficientes

La multiplicación es conocida como una operación conmutativa, donde no es necesario hacer diferencia entre los factores. No obstante, desde el punto de vista algorítmico, es necesario hacer la diferencia entre multiplicador y multiplicando. Los multiplicadores representan parámetros que cambiarán las características de los multiplicandos. Estos últimos representan datos que serán modificados por los multiplicadores. En los filtros digitales, cada coeficiente de un filtro digital es interpretado como un multiplicador. Los datos de entrada en el filtro son los multiplicandos.

La construcción de un multiplicador implica el uso de muchos recursos de hardware (mucho espacio, más consumo de potencia, etc.) [35]. Debido a esto, se prefieren filtros sin multiplicadores, es decir, filtros en donde las operaciones de multiplicación se realicen utilizando algún algoritmo que permita prescindir de los multiplicadores.

Existen varios algoritmos para realizar multiplicaciones en forma eficiente [36, 37], basados en la diferencia entre multiplicadores y multiplicandos. Representando adecuadamente a un multiplicador, la operación de multiplicación puede llevarse a cabo realizando operaciones simples de suma, resta y corrimiento únicamente sobre el multiplicando.

En las sub-secciones siguientes se revisará el método más comúnmente utilizado en filtros FIR para desempeñar operaciones de multiplicación en forma eficiente. Además, las representaciones *CSD* (*Canonical Signed Digit*) y *MSD* (*Minimal Signed Digit*) serán estudiadas.

3.2.1 Algoritmo de Hörner para realizar multiplicaciones en base a sumas y corrimientos

Considérense *X* como mutiplicador y *Y* como mutiplicando, dados por [37]

$$\begin{aligned} X &= x_{n-1}\cdot B^{n-1} + x_{n-2}\cdot B^{n-2} + \ldots + x_0 \cdot B^0, \\ Y &= y_{m-1}\cdot B^{m-1} + y_{m-2}\cdot B^{m-2} + \ldots + y_0 \cdot B^0, \quad x_i, y_i \in \{0,1,\ldots,B-1\}, \end{aligned} \tag{3.7}$$

donde *n* es el número de dígitos de *X*, *m* es el número de dígitos de *Y* y *B* es la base numérica de *X* y *Y*. La multiplicación entre ambos factores puede ser expresada como

$$\begin{aligned} &Z = X \cdot Y, \\ &\text{con} \\ &Z = z_{m+n-1} \cdot B^{n+m-1} + z_{m+n-2} \cdot B^{n+m-2} + ... + z_0 \cdot B^0. \end{aligned} \tag{3.8}$$

La expresión anterior puede reescribirse en la forma

$$Z = x_{n-1} \cdot Y \cdot B^{n-1} + x_{n-2} \cdot Y \cdot B^{n-2} + ... + x_0 \cdot Y \cdot B^0, \tag{3.9}$$

que puede ser expandida como

$$Z = ((...((0 \cdot B + x_{n-1} \cdot Y) \cdot B + (x_{n-2} \cdot Y) \cdot B + ... + x_2 \cdot Y) \cdot B + x_1 \cdot Y) \cdot B + x_0 \cdot Y \,. \tag{3.10}$$

La ecuación (3.10) es llamada *expansión de Hörner*. Obsérvese que, comenzando del paréntesis más anidado, se realiza un producto parcial entre el último elemento de *X* (representado como x_{n-1} en (3.7)) y *Y*. El resultado es entonces multiplicado por la base *B*, lo que significa correr hacia la izquierda el resultado obtenido de la multiplicación. Después, este resultado es sumado con el siguiente producto parcial, realizado como el producto entre el penúltimo elemento de *X* (llamado x_{n-2} en (3.7)) con *Y*. El procedimiento continúa hasta realizar el último producto parcial, entre x_0 y *Y*, y sumarlo al resultado acumulado de las operaciones anteriores. Este procedimiento es llamado *algoritmo de Hörner* [37] y es representado en la figura 3.5.

Puede notarse que para representaciones de multiplicadores en base *B* = 2, el procedimiento es muy simple. Esto se concluye a partir de que cada

elemento x_k del multiplicador X solo puede tomar valores 0 o 1. Así, los productos parciales generados solo indican la presencia o ausencia de Y.

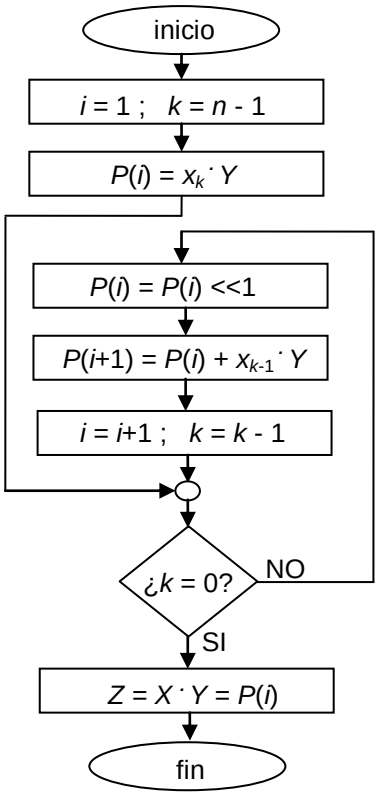


Figura 3.5. Algoritmo de Hörner.

Ejemplo 3.2

Considérense los valores dados como $X = 1101$ y $Y = 111$, con $B = 2$. Calcular $Z = X \cdot Y$, con X como multiplicador.

En base a la ecuación (3.7), se tiene $n = 4$; $x_3 = 1$, $x_2 = 1$, $x_1 = 0$, $x_0 = 1$.

La expansión de Hörner queda expresada, siguiendo la ecuación (3.10), como

$$Z = (((x_3 \cdot Y) \cdot B + x_2 \cdot Y) \cdot B + x_1 \cdot Y) \cdot B + x_0 \cdot Y \ ,$$
$$Z = (((1 \cdot 111) \cdot 10 + 1 \cdot 111) \cdot 10 + 0 \cdot 111) \cdot 10 + 1 \cdot 111 \ . \qquad (3.11)$$

El desarrollo del algoritmo de Hörner resulta en 4 pasos:

Paso 1. $P(1) = (1 \cdot 111) = 111$

Paso 2. $P(2) = (P(1) \cdot 10 + (1 \cdot 111)) = 1110 + 111 = 10101$

Paso 3. $P(3) = (P(2) \cdot 10 + (0 \cdot 111)) = 101010 + 000 = 101010$

Paso 4. $P(4) = (P(3) \cdot 10 + (1 \cdot 111)) = 1010100 + 111 = 1011011$

Se obtiene entonces Z = 1011011 en base 2, o bien 91 en decimal. Se comprueba la operación $Z = 1101 \cdot 111$ en base 2, o bien, $Z = 13 \cdot 7$ en decimal.

Es importante notar en este ejemplo que la operación de multiplicación puede llevarse a cabo utilizando solamente operaciones de suma y efectuando los corrimientos apropiados. Los bits del multiplicador X indican cuántas operaciones de suma serán necesarias. Si el multiplicador tiene k bits no-cero, entonces se requerirán $k - 1$ sumas. Nótese entonces que la representación de los multiplicadores determina cuántas operaciones de suma serán realizadas. Debido a esto, es conveniente revisar las representaciones eficientes para los multiplicadores.

3.2.2 Representación CSD

La representación *CSD* (*Canonical Signed Digit*) forma parte de una representación numérica general denominada *Signed Digit* (*SD*). Algunas veces la representación *SD* es llamada *SPT* (*Signed Powers-of-Two*). Las representaciones basadas en *SD* son diferentes de las representaciones binarias ordinarias debido a que están constituidas por un conjunto de tres dígitos distintos, {-1, 0, 1}. Algunas veces -1 puede ser denotado como $\bar{1}$.

Un número *N* codificado en *SD* puede ser expresado como [38]

$$N = \sum_{k=1}^{B} s_k 2^{p_k} , \tag{3.12}$$

donde

- $s_k \in \{-1, 0, 1\}$.
- $p_k \in \{-l_b, \ldots, -1, 0, 1, \ldots, u_b\}$, con l_b y u_b como enteros positivos que determinan el rango dinámico de *N*.
- *B* es el número de dígitos de *N*.

Las representaciones *SD* resultan mejores que las representaciones binarias tradicionales debido a que incluyen menos elementos no-cero en su código. Al representar un multiplicador con *SD*, es posible desempeñar una multiplicación con menos operaciones de suma o resta que las necesarias cuando el multiplicador es representado en forma tradicional.

La representación *CSD* para cierto número *N* es única, y tiene las siguientes características:

- El número de dígitos no-cero en el código es mínimo. En un número de *B* bits, solo puede haber un máximo de $\lfloor B/2 \rfloor$ bits no-cero.
- El producto entre dos dígitos adyacentes es 0. Esto significa que cada dígito en la representación *CSD* está separada al menos por un bit cero.

Existen varios procedimientos que permiten desarrollar la representación en *CSD* de un número dado *N*. A continuación se presenta uno muy simple tomado de [31].

Considérese un valor *N* entero positivo que se desea codificar en *CSD*.

1.- Representar el número *N* en forma binaria sin signo.

2.- Comenzando con el bit menos significativo, sustituir la primera secuencia de unos acomodados en forma consecutiva, como se muestra en la ecuación (3.13).

$$011...11 = 100...0\bar{1} \tag{3.13}$$

3.- Concatenar el resto de los bits situados a la izquierda, que no fueron sustituidos en el paso anterior, con la cadena de bits sustituida.

4.- Repetir desde el paso 2, comenzando ahora con el primer bit de la cadena de bits concatenada en el paso 3. Finalizar cuando toda la representación binaria inicial haya sido revisada.

Si el número *N* es negativo, entonces se sigue el procedimiento anterior utilizando la representación binaria para el valor positivo correspondiente. Al finalizar, se cambian los dígitos1 por $\bar{1}$ y viceversa. Además, si el número *N* es real, entonces debe aplicarse redondeo como en la ecuación (3.1). El procedimiento se efectúa sobre el valor entero obtenido de (3.1), al dividir toda la ecuación por la constante de redondeo.

Ejemplo 3.3

Hallar la representación *CSD* para los siguientes valores:

a) $N = 237$

b) $N = -45$

c) $N = 0.553$

a) Para el primer caso se tiene que *N* ya es un número entero positivo, entonces se lleva a cabo el procedimiento.

Paso 1. La representación del número $237\,|_{10}$ en forma binaria sin signo es $11101101\,|_2$.

Paso 2. Realizar la sustitución de la primera cadena de unos hallada y sustituir como en (3.13). Se encuentra 111**011**01 y se sustituye por 111**10$\bar{1}$**01.

Paso 3. Concatenar el resto de los bits situados a la izquierda. Se obtiene 11110$\bar{1}$01.

Paso 4. Repetir el paso 2. Ahora se encuentra **1111**0$\bar{1}$01 y se sustituye por **1000$\bar{1}$**0$\bar{1}$01. Se concluye debido a que se ha revisado toda la representación binaria inicial. El resultado es N = 1000$\bar{1}$0$\bar{1}$01 $|_{CSD}$.

b) Para el segundo caso se tiene que N = -45 es un valor entero negativo, entonces se desarrollará con el valor positivo correspondiente, N = 45.

Paso 1. La representación del número 15 $|_{10}$ en forma binaria sin signo es 101101 $|_2$.

Paso 2. Realizar la sustitución de la primera cadena de unos hallada y sustituir como en (3.13). Se encuentra **11** y se sustituye por **10$\bar{1}$**.

Paso 3. Concatenar el resto de los bits situados a la izquierda. Se obtiene 110$\bar{1}$01.A la izquierda se encuentra un patrón similar y se sustituye de la misma forma.

Paso 4. Se concluye el procedimiento debido a que se ha revisado toda la representación binaria inicial. El resultado es N = 10$\bar{1}$0$\bar{1}$01 $|_{CSD}$.

Debido a que se inició con un valor negativo, el resultado obtenido en el procedimiento anterior se debe modificar, cambiando los valores 1 por $\bar{1}$ y viceversa. El resultado final es N = $\bar{1}$010101$\bar{1}$ $|_{CSD}$.

c) En el tercer caso se tiene que N = 0.553 es un valor positivo, pero no es entero. Además se observa que no puede ser representado como suma de potencias de 2. Entonces es necesario aplicar redondeo como en la ecuación (3.1). Eligiendo un valor $\alpha = 2^{-8}$ se obtiene un valor

aproximado $N = (2^{-8}) \cdot (142) = 0.5546875$. Se desarrollará el procedimiento con el valor entero correspondiente, $N = 142$.

Paso 1. La representación del número 142 $|_{10}$ en forma binaria sin signo es 10001110 $|_2$.

Paso 2. Realizar la sustitución de la primera cadena de unos hallada y sustituir como en (3.13). Se encuentra 100**0111**0 y se sustituye por 100**100$\bar{1}$**0.

Paso 3. Concatenar el resto de los bits situados a la izquierda. Se obtiene 100100$\bar{1}$0.

Paso 4. El procedimiento concluye debido a que ya no hay secuencias de unos consecutivos y se ha revisado toda la representación binaria inicial. El resultado es $N = 100100\bar{1}0\ |_{CSD}$.

Debido a que se inició con un valor fraccionario, el resultado obtenido en el procedimiento anterior se debe modificar, únicamente escalando el valor por $\alpha = 2^{-8}$.

Obsérvese que la representación *CSD* permite expresar un número con pocos dígitos no-cero. Al representar los coeficientes de un filtro digital en forma *CSD*, se pretende que la multiplicación entre los datos de entrada y cada coeficiente se lleve a cabo con el menor número posible de operaciones de suma o resta.

3.2.3 Representación MSD

La representación *MSD* (*Minimal Signed Digit*) también forma parte de las representaciones numéricas *SD*. En forma similar a *CSD*, en *MSD* la condición fundamental es tener el mínimo número de dígitos no-cero en la representación de cierto número *N.* No obstante, la representación *MSD* permite la posibilidad de que un valor dado *N* sea representado por varias

formas distintas. Esto hace la diferencia con *CSD*. Así, mientras en *CSD* la representación de un número es única, en *MSD* existen varias opciones [39].

Básicamente, las representaciones *MSD* para un valor dado *N* parten de la representación *CSD* correspondiente. La transformación de *CSD* a *MSD* se basa en la conversión de conjuntos de bits expresados como

$$0\,1\,1 = 1\,0\,\bar{1}, \tag{3.14a}$$

$$\bar{1}\,0\,1 = 0\,\bar{1}\,\bar{1}. \tag{3.14b}$$

Estas transformaciones hechas sobre grupos de tres bits son desarrolladas a lo largo de toda la palabra que representa al número *N* en *CSD.* Cada conversión realizada, basada en las expresiones de (3.14), dará lugar a una nueva representación *MSD*. Nótese que la representación *CSD* de un número dado forma parte del conjunto de representaciones *MSD* para ese número. Además, todos los valores del conjunto *MSD* tienen el mínimo número de dígitos no-cero.

En [40] fue propuesto un algoritmo para encontrar todas las representaciones *MSD* de un número dado, partiendo del código inicial en *CSD* de dicho número*.* Los pasos a seguir para hallar el conjunto *MSD* son los siguientes:

1.- Representar el número *N* en forma *CSD*. Formar un conjunto *S* que vaya almacenando a todas las representaciones *MSD* que se obtengan durante el procedimiento. En este paso, *S* solo debe contener a la representación *CSD.*

2.- Elegir un apuntador $s_p(i)$ para cada *i*-ésima representación. Iniciar con la primera representación (i = 1) incluida en el conjunto *S* y con $s_p(i)$ apuntando hacia el bit más significativo.

3.- Comenzando con el bit indicado por $s_p(i)$, sustituir el primer bloque de bits que pueda ser modificado por cualquiera de las dos ecuaciones (3.14). Incluir el resultado en el conjunto *S*. Para esa nueva representación

generada, llamada *j*-ésima representación, asignar al apuntador $s_p(j)$ el valor $s_p(i) - 2$.

4.- Disminuir en 2 el valor de $s_p(i)$ si se realizó una transformación en el paso anterior, o en 1 si no se efectuó. Repetir el paso 3 hasta que toda la cadena de bits de la representación haya sido revisada.

5.- Elegir la siguiente representación. Repetir los pasos 3 y 4 con el respectivo apuntador $s_p(j)$. Finalizar cuando ya no haya más representaciones para revisar.

Ejemplo 3.4

Hallar las representaciones *MSD* para los siguientes valores:

a) $N = 237$, b) $N = -45$, c) $N = 0.553$.

Siguiendo el procedimiento anterior, se obtienen los siguientes resultados.

a) $N = 237$,

$S = \{1000\bar{1}0\bar{1}01,\ 1000\bar{1}00\bar{1}\bar{1}\}$.

b) $N = -45$,

$S = \{\bar{1}010101\bar{1}, 0\bar{1}\bar{1}0101\bar{1}, \bar{1}010011,\ 0\bar{1}0\bar{1}\bar{1}0\bar{1},\ 0\bar{1}\bar{1}0011\}$.

c) $N = 0.553$,

$S = \{100100\bar{1}0\}$.

Nótese que en los conjuntos obtenidos, la primera representación es *CSD*. Además, los demás elementos del conjunto no contienen ninguna representación con más dígitos no-cero que los que contiene la representación *CSD*.

Se ha dicho que la representación *CSD* es adecuada para representar cada coeficiente en forma individual. Sin embargo, la representación *MSD* resulta atractiva debido a que existe más de una representación disponible

para cada coeficiente. Esto es interesante cuando el filtro FIR es representado en forma directa transpuesta, debido a que los coeficientes pueden ser combinados dentro de un bloque multiplicador [40, 41]. Cuando esto se lleva a cabo, la combinación adecuada entre coeficientes resulta más fácil de determinar si existen más representaciones disponibles para cada coeficiente. Una explicación más detallada de esto se verá en la siguiente sección.

En los últimos años se han desarrollado varios algoritmos para eliminar la mayor cantidad posible de sub-expresiones comunes. La finalidad es llegar a un bloque multiplicador que contenga a todos los coeficientes del filtro, representándolos con la menor cantidad posible de operaciones de suma/resta. No obstante, estos algoritmos deben tener cuidado en no ser demasiado complejos computacionalmente. La sección siguiente presenta los métodos más recientes para la búsqueda y eliminación de sub-expresiones comunes.

3.3 Revisión de algunos métodos de eliminación de sub-expresiones comunes

Para entender la idea implicada en los métodos de eliminación de sub-expresiones comunes con más claridad, considérese la figura 3.6. Debido a que los coeficientes interactúan con las señales de entrada simultáneamente, éstos pueden ser combinados e incluidos dentro de un conjunto general llamado bloque multiplicador, como se observa en la figura 3.6a. Esto significa que algunas sub-expresiones comunes entre dos o más representaciones de distintos coeficientes pueden ser factorizadas, es decir, implementadas solo una vez y eliminadas del resto de los coeficientes. Esto es más explícito en la figura 3.6b.

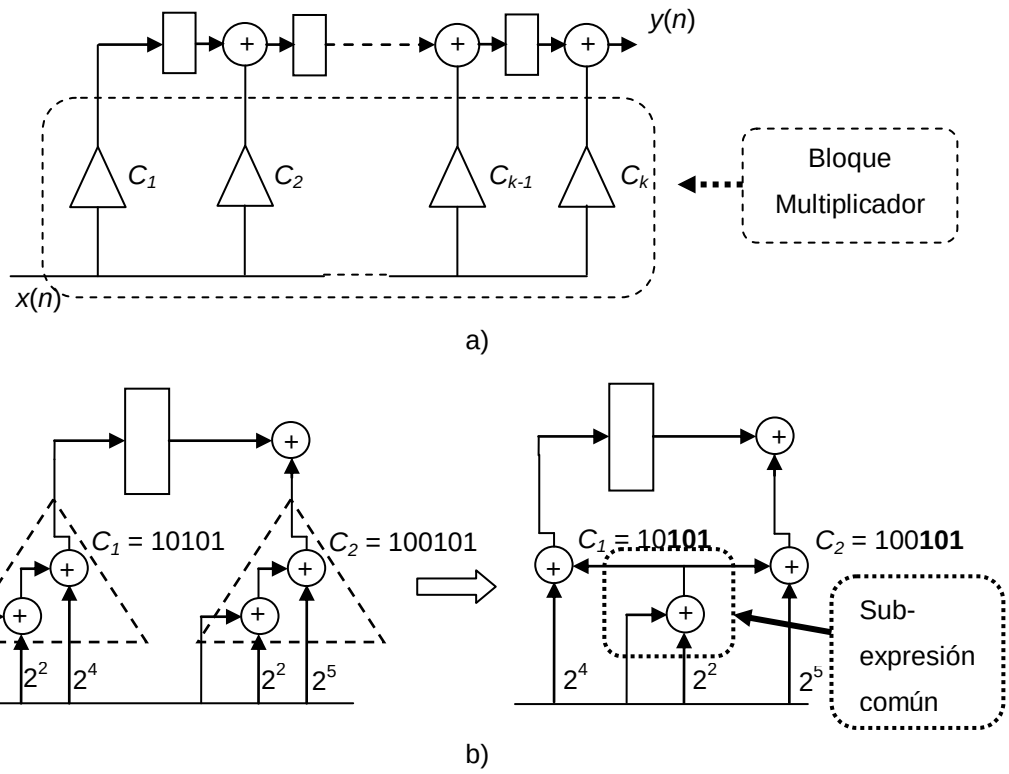


Figura 3.6. a) Estructura directa transpuesta de un filtro FIR, b) Ejemplo de eliminación de sub-expresiones comunes. Se elimina la sub-expresión "101" de cada coeficiente y se implementa una sola vez.

La eliminación de sub-expresiones comunes permite ahorrar más operaciones de suma/resta que las que se ahorran cuando los coeficientes se tratan en forma separada. Debido a esto, desde la última década aproximadamente, se han realizado investigaciones respecto a estos métodos. Dos de los métodos más recientes se explican en las sub-secciones siguientes.

3.3.1 Algoritmo sub-óptimo de dos etapas

En un problema de diseño de filtros FIR el principal requisito que se debe cumplir es que el filtro que va a ser diseñado satisfaga las especificaciones requeridas en la respuesta en frecuencia. A menudo estas especificaciones son dadas en forma relativa. Es decir, las frecuencias de paso y de rechazo,

ω_p y ω_s, son expresadas en relación a la frecuencia de muestreo del filtro. Además, los rizos en la banda de paso y de rechazo, δ_p y δ_s, son dados en relación a la ganancia en banda de paso del filtro. Por lo tanto, el parámetro que debe respetarse en cualquier momento durante el diseño del filtro es el Máximo Pico de Rizo Normalizado que es permitido en las especificaciones [15]. Esto se expresa como

$$NPRM = \max\left\{\frac{\delta_p\omega_p}{g}, \frac{\delta_s\omega_s}{g}\right\}, \tag{3.15}$$

donde *g* representa la ganancia del filtro, δ_p es el rizo en la banda de paso, δ_s es el rizo en la banda de rechazo, ω_p es la frecuencia de paso y ω_s es la frecuencia de rechazo.

El algoritmo sub-óptimo de dos etapas, presentado en [42], explica un procedimiento para minimizar el número de sumas utilizadas en una implementación como en la figura 3.6a, basado en las siguientes etapas generales:

- Partiendo de un diseño basado en el algoritmo Parks-McClellan, expresar los coeficientes en *SPT* (*Signed-Powers-of-Two*) y obtener el menor número de sumas requeridas. La expresión *SPT* de los coeficientes provocará que la condición del máximo *NPRM* (*Normalized Peak Ripple Magnitude*) (3.15) permitido deje de cumplirse.
- Corregir el error introducido en la primera etapa, asignando más términos no-cero a los coeficientes adecuados. Las especificaciones requeridas deben satisfacerse en esta etapa.

Considérese la expresión dada por

$$h_{spt}(i) = \sum_{k=1}^{L_i} s_{i,k} 2^{-p_i,k} , \tag{3.16}$$

donde

- $h_{spt}(i)$ es la representación *SPT* del *i*-ésimo coeficiente.
- $s_{i,k} \in \{-1,1\}$, es el *k*-ésimo término *SPT* (un dígito no-cero) dentro de la representación para el *i*-ésimo coeficiente.
- $p_{i,k} \in \{1,2,\ldots,B\}$, es la potencia de dos para el término $s_{i,k}$ y representa la posición del bit.
- L_i es la cantidad de términos *SPT* (dígitos no-cero) incluidos en la representación.

Se entiende entonces que el coeficiente $h_{spt}(i)$ tiene una longitud de palabra de *B* bits con L_i términos *SPT.*

El número total de términos *SPT* en un filtro con *N* coeficientes está expresado como

$$J = \sum_{i=1}^{N} \sum_{k=1}^{L_i} \left| s_{i,k} \right|, \tag{3.17}$$

con *N* siendo el número de coeficientes y L_i el número de términos *SPT* para el *i*-ésimo coeficiente. Nótese que la ecuación (3.17) es la suma total de términos *SPT* en los coeficientes del filtro, es decir, la cantidad total de dígitos no-cero. Esto significa que los dígitos están contados individualmente en cada coeficiente.

Considérense ahora las sub-expresiones comunes dadas por

$$x_{101} = 101, \tag{3.18a}$$

$$x_{10\bar{1}} = 10\bar{1}. \tag{3.18b}$$

Las sub-expresiones de la forma “101” o “$10\bar{1}$” y sus correspondientes formas negativas pueden ser incluidas en la expresión (3.16). Entonces se obtiene una expresión para cada coeficiente dada por

$$h_{cspt}(i) = \sum_{k=1}^{L_{101,i}} cs_{i,k}^{101}\left(2^{-p_{i,k}^{101}+2} + 2^{-p_{i,k}^{101}}\right) + \sum_{k=1}^{L_{10\bar{1},i}} cs_{i,k}^{10\bar{1}}\left(2^{-p_{i,k}^{10\bar{1}}+2} - 2^{-p_{i,k}^{10\bar{1}}}\right) + \sum_{k=1}^{L_i'} s_{i,k}' 2^{-p_{i,k}'}, \tag{3.19}$$

donde

- $h_{cspt}(i)$ es la representación *SPT* del *i*-ésimo coeficiente, que contiene términos de sub-expresiones comunes y términos no-cero aislados.
- $cs_{i,k}^{101}$, $cs_{i,k}^{10\bar{1}} \in \{-1,1\}$, son los signos de las sub-expresiones “101” y “$10\bar{1}$” respectivamente.
- $p_{i,k}^{101}$, $p_{i,k}^{10\bar{1}} \in \{1,2,\ldots,B\}$, son la potencia de dos para el bit *LSB* de las sub-expresiones “101” y “$10\bar{1}$” respectivamente, y representan la posición del bit *LSB* de la correspondiente sub-expresión.
- $L_{101,i}$ y $L_{10\bar{1},i}$ son la cantidad de sub-expresiones comunes de tipo “101” y “$10\bar{1}$” respectivamente.
- $s_{i,k}' \in \{-1,1\}$, representa al *k*-ésimo término no-cero aislado, que no pertenece a ninguna sub-expresión.
- $p_{i,k}' \in \{1,2,\ldots,B\}$, es la potencia de dos para el término aislado $s_{i,k}'$ y representa la posición del bit.
- L_i' es la cantidad de dígitos no-cero aislados incluidos en la representación.

La representación *SPT* dada en la expresión (3.19) incluye a los dígitos que se encuentran en alguna sub-expresión y a los que están aislados. Por lo tanto, $L_i^{'}$ puede ser expresado como

$$L_i^{'} = L_i - 2(L_{101,i} + L_{10\bar{1},i}) . \tag{3.20}$$

El objetivo del algoritmo dado en [42] es minimizar la cantidad total de términos *SPT* en todos los coeficientes, dada por

$$J^{'} = \sum_{i=1}^{N} L_i^{'} + L_{101,i} + L_{10\bar{1},i} . \tag{3.21}$$

La ventaja de tener los coeficientes del filtro representados con muchas sub-expresiones comunes compartidas es simple. Bajo cierto número dado de sumas, más términos pueden ser asignados a los coeficientes, lo cual permite mejorar su precisión.

A continuación obsérvese la *Pirámide de Ponderaciones de Hamming* (*HWP*: *Hamming Weight Pyramid*) [43], mostrada en la tabla 3.1. Ésta exhibe el número de términos *SPT* incluidos en la representación de cierto valor dado. Considérese el número que aparece en la posición (*r*, *c*), donde *r* representa un número de renglón y *c* representa un número de columna. Dicho número es la cantidad de términos *SPT* que están contenidos en un valor entero dado como $2^r + c$.

Ahora tómese en cuenta la *Pirámide de Ponderaciones de Hamming basada en Sub-expresiones Comunes* (*CS-HWP*: *Common Sub-expression HWP*) presentada en la tabla 3.2. La cantidad que aparece en la posición (*r*, *c*) representa ahora un término denominado *CSPT*, que significa *Común SPT*. Aquí, cada sub-expresión “101” o “$10\bar{1}$” es considerada como un solo término *CSPT*, debido a que es común con otros coeficientes. Entonces, si

un número dado como $2^r + c$ tiene una cantidad '*x*' de términos *SPT* en la pirámide *HWP*, puede tener menor o igual número de términos *CSPT* en la pirámide *CS-HWP*.

c / *r*	-9	-8	-7	-6	-5	-4	-3	-2	-1	0	1	2	3	4	5	6	7	8	9
0										**1**									
1										**1**									
2									**2**	**1**	**2**								
3								**2**	**2**	**1**	**2**	**2**							
4					**3**	**2**	**3**	**2**	**2**	**1**	**2**	**2**	**3**	**2**	**3**				
5	**3**	**2**	**3**	**3**	**3**	**2**	**3**	**2**	**2**	**1**	**2**	**2**	**3**	**2**	**3**	**3**	**3**	**2**	**3**

Tabla 3.1. Pirámide de ponderaciones de Hamming, *HWP*.

c / *r*	-9	-8	-7	-6	-5	-4	-3	-2	-1	0	1	2	3	4	5	6	7	8	9
0										**1**									
1										**1**									
2									**1**	**1**	**1**								
3								**2**	**2**	**1**	**2**	**2**							
4					**2**	**2**	**2**	**2**	**2**	**1**	**2**	**2**	**2**	**2**	**2**				
5	**3**	**2**	**3**	**3**	**2**	**2**	**2**	**2**	**2**	**1**	**2**	**2**	**2**	**2**	**2**	**3**	**3**	**2**	**3**

Tabla 3.2. Pirámide de ponderaciones de Hamming basada en Sub-expresiones comunes, *CS-HWP*.

Ejemplo 3.5

Hallar la cantidad de términos *CSPT* para los números: a) 19, b) 37. Comparar con los términos dados en la tabla *HWP*.

a) El número 19 aparece en las tablas como $2^4 + 3$. Entonces $r = 4$ y $c = 3$. De la tabla *HWP* se tiene que el número 19 tiene 3 términos *SPT.* Sin embargo, en la tabla *CS-SPT* se mencionan dos términos *CSPT*. Esto es porque la representación *SPT* de 19 es $10\,10\,\bar{1}$. Nótese que la sub-expresión $10\bar{1}$ es considerada como un solo término *CSPT* en la tabla *CS-HWP*.

b) El número 37 aparece en las tablas como $2^5 + 5$. Se tiene que $r = 5$ y $c = 5$. La tabla *HWP* señala 3 términos *SPT.* Sin embargo, en la tabla *CS-SPT* se mencionan 2 términos *CSPT*. La representación *SPT* de 37 es 100101. Nótese que la sub-expresión 101 es considerada como un solo término *CSPT* en la tabla *CS-HWP*.

Nótese que en ambos casos las sub-expresiones son consideradas como un término simple, debido a que estas pueden ser compartidas con otros coeficientes.

La pirámide *CS-HWP* resulta una herramienta auxiliar cuando se desarrolla la primera etapa del algoritmo. Luego de representar en *SPT* a los coeficientes (la representación recomendada es *CSD*), se buscan valores cercanos a cada valor de coeficiente en la pirámide. Los valores obtenidos de esta búsqueda deben cumplir con la restricción de máximo *NPRM* y a la vez contar con el mínimo número posible de términos *CSPT*.

En la segunda etapa, si la restricción de máximo *NPRM* dejó de satisfacerse, los coeficientes se modifican. Primero se define un rango de bits menos significativos para cada coeficiente (dependiendo de la longitud de palabra), con la finalidad de asignar términos *SPT* adicionales. Este rango está definido en la tabla 3.3. Luego se genera un conjunto de coeficientes nuevos que pueda sustituir a cada coeficiente formado en la primera etapa. Por último, se eligen los 5 mejores coeficientes nuevos (es decir, los que mejoren las características en el *NPRM*) y se elige el que tenga menor

aumento en la cantidad de términos *SPT*. Al finalizar la segunda etapa, las características requeridas deben ser alcanzadas.

El algoritmo es detallado a continuación.

Primera Etapa

1. Obtener el conjunto de coeficientes, {h}, para un filtro deseado.
2. Redondear {h} y representar en *CSD* (conjunto {h_{CSD}}).
3. Buscar un conjunto {h_{CSPT}}(i) para cada elemento $h_{CSD}(i)$ del conjunto {h_{CSD}} con igual o menor número de términos *CSPT.* Cada conjunto se busca en la pirámide CS-HWP.
4. Crear un conjunto de coeficientes posibles, {h_P}, con todos sus elementos igual a 0. Iniciar un contador k como $k = 1$.
5. Ordenar los elementos de cada conjunto {h_{CSPT}} en orden ascendente de términos *CSPT* (los elementos con un término *CSPT* van primero).
6. Permitiendo únicamente k términos *CSPT* en {h_P}, elegir de todos los conjuntos {h_{CSPT}} el elemento que minimice el *NPRM.* Sustituir el elemento $h_P(i)$ por el obtenido del conjunto {h_{CSPT}} (i).
7. Eliminar todos los elementos del conjunto {h_{CSPT}} (i) con k términos *CSPT .* Hacer $k = k + 1$.
8. Si aún hay elementos en los conjuntos {h_{CSPT}}, regresar al paso 6. De lo contrario, ir al paso 1 de la segunda etapa

Segunda Etapa

1. Definir un conjunto de valores-límite inferior, {h_{Low}}, con cada elemento $h_{Low}(i)$ dado por $h_P(i) - 2\Delta q_{max}$. Definir un conjunto de valores-límite superior, {h_{Up}}, con cada elemento $h_{Up}(i)$ dado por $h_P(i) + 2\Delta q_{max}$. (Δq_{max} depende de la longitud de palabra y está dado en la tabla 3.3).
2. Formar tantos conjuntos {h_{PMIN}} como bits permitidos puedan ser sumados o restados a cada elemento de {h_P}. El número de bits permitidos depende de Δq_{max} y también está dado por la tabla 3.3.

3. Elegir los 5 mejores conjuntos $\{h_{MIN}\}$ cuyos elementos dan los 5 mejores resultados para *NPRM.*
4. Obtener el número de términos *CSPT* de cada uno de los 5 conjuntos. Identificar cuántos conjuntos tienen el menor número posible de términos *CSPT* con respecto a los otros.
5. Si solo hay un conjunto con el menor número posible de términos *CSPT* con respecto a los otros, elegir dicho conjunto, llamarlo $\{h_{MIN\_BUENO}\}$ y pasar al paso 7. De lo contrario, parar al paso 6.
6. De los conjuntos con el menor número posible de términos *CSPT* con respecto a los otros, elegir el que tenga el mejor resultado para *NPRM* y llamarlo $\{h_{MIN\_BUENO}\}$.
7. Hacer $\{h_P\} = \{h_{MIN\_BUENO}\}$. Si el valor requerido del *NPRM* es alcanzado, terminar. De lo contrario, ir al paso 2.

Longitud de Palabra, *B*	**Términos discretos *SPT, Δq***	***Δq*$_{max}$**
8	$\pm 2^{-7}, \pm 2^{-8}$	2^{-7}
9	$\pm 2^{-8}, \pm 2^{-9}$	2^{-8}
10	$\pm 2^{-8}, \pm 2^{-9}, \pm 2^{-10}$	2^{-8}
11	$\pm 2^{-9}, \pm 2^{-10}, \pm 2^{-11}$	2^{-9}
12	$\pm 2^{-9}, \pm 2^{-10}, \pm 2^{-11}, \pm 2^{-12}$	2^{-9}
13	$\pm 2^{-10}, \pm 2^{-11}, \pm 2^{-12}, \pm 2^{-13}$	2^{-10}
14	$\pm 2^{-11}, \pm 2^{-12}, \pm 2^{-13}, \pm 2^{-14}$	2^{-11}

Tabla 3.3. Vecindad definida para compensación del error de cuantización

El algoritmo sub-óptimo de dos etapas presenta buenos resultados en cuanto a ahorro de sumas, aprovechando el uso de sub-expresiones comunes en la representación de los coeficientes. En la primera etapa, este algoritmo restringe la búsqueda de la solución óptima a un espacio limitado inicialmente por el redondeo de los coeficientes. Debido a eso es llamado

algoritmo sub-óptimo. No obstante, los resultados son mejorados en la segunda etapa. Además de esto, el algoritmo resulta relativamente sencillo y emplea poco tiempo para ser desarrollado.

3.3.2 Optimización en el espacio de sub-expresiones usando MILP

Se ha visto que la optimización en dos etapas desarrolla la búsqueda de sub-expresiones comunes en un espacio previamente limitado a los valores enteros obtenidos por redondeo. En el método dado en [42], luego de encontrar sub-expresiones comunes apropiadas, se mejora la respuesta en frecuencia introduciendo adecuadamente algunos términos *SPT*. Un enfoque distinto es desarrollar el proceso de optimización directamente en un espacio de valores de coeficientes representados con sub-expresiones comunes. El algoritmo propuesto en [44] lleva a cabo la búsqueda de los coeficientes de filtros FIR en un espacio de sub-expresiones previamente establecido. Para desarrollar la optimización, este método se basa en el enfoque *MILP*, revisado en la sub-sección 3.1.2.

Básicamente el método presentado en [44] consta de los siguientes pasos:

- Para una solución obtenida con coeficientes de precisión infinita, establecer la longitud de palabra deseada y la cantidad de sub-expresiones permitidas por coeficiente. Determinar el conjunto de sub-expresiones comunes que podrán ser utilizadas en la representación de los coeficientes con palabra de longitud finita.
- Realizar el proceso de optimización en el espacio de sub-expresiones. Dicho espacio contiene únicamente a los valores de coeficientes que pueden ser expresados con las condiciones establecidas en el paso anterior.

Sabido es que un valor de coeficiente representado en palabra de longitud finita puede ser convertido a entero multiplicándolo por la potencia de dos adecuada. Debido a esto, en lo sucesivo será suficiente suponer solamente coeficientes con valores enteros.

Considérese la ecuación dada por

$$h_L(i) = \sum_{k=1}^{L_i} s_{i,k} 2^{p_i,k}, \quad s_{i,k} \in S, \tag{3.22}$$

donde

- $h(i)$ es la representación en sub-expresiones comunes del i-ésimo coeficiente.
- $s_{i,k}$ es la k-ésima sub-expresión dentro de la representación para el i-ésimo coeficiente.
- $p_{i,k} \in \{0,1,2,\ldots,B\}$, es la potencia de dos para el término $s_{i,k}$ y representa la posición del bit *LSB* de la sub-expresión.
- L_i es la cantidad de sub-expresiones incluidas en la representación del i-ésimo coeficiente.
- S es el conjunto de sub-expresiones permisibles en la representación de todos los coeficientes.

Puede verse que la ecuación (3.22) es una forma general que incluye a las expresiones (3.16) y (3.19), tomando en cuenta que (3.22) está expresada para valores enteros. La ecuación (3.16) formula la representación de un coeficiente únicamente considerando los términos *SPT* 1 y -1. En (3.19) solamente son incluidas las sub-expresiones $10\bar{1}$, 101 y sus correspondientes negativas. Sin embargo, la ecuación (3.22) permite expresar a un coeficiente con L sub-expresiones tomadas de un conjunto S.

Las sub-expresiones permitidas en el conjunto *S* dependen de las especificaciones de longitud de palabra dadas.

Antes de revisar el criterio para elegir las sub-expresiones que deben ser incluidas en el conjunto *S*, es conveniente considerar algunas características básicas de *S*. Primero, debe tenerse en cuenta que una sub-expresión es una cadena de bits que puede o no incluir los tres dígitos permitidos en representación *SD*, es decir, 1, 0 y $\bar{1}$. Por otro lado, un conjunto S_1 = {0, $\pm$1, $\pm 10\bar{1}$, $\pm$101, $\pm 100\bar{1}$, $\pm$1001} puede ser expresado en forma equivalente como S_1 = {0, $\pm$1, $\pm$3, $\pm$5, $\pm$7, $\pm$9}. En cada conjunto *S*, las sub-expresiones incluidas siempre están acompañadas por su correspondiente negativa, debido a que estas no añaden complejidad. Considérense ahora las siguientes características:

- *Contigüidad*.- Un conjunto *S* es contiguo si los valores de las sub-expresiones incluidas en él son valores impares consecutivos. Por ejemplo, el conjunto S_1 = {0, $\pm$1, $\pm$3, $\pm$5, $\pm$7, $\pm$9} es contiguo, pero el conjunto S_2 = {0, $\pm$1, $\pm$5, $\pm$7, $\pm$9} no lo es.
- *Orden*.- El orden de un conjunto *S* está dado por el número de sumas que se necesitan para representar a todas las sub-expresiones incluidas en *S*, sin contar las negativas. El conjunto S_1 es de 4º orden, porque se requiere una suma para representar 3, una para 5, una para 7 y una para 9. Las correspondientes sub-expresiones negativas no añaden complejidad y no son tomadas en cuenta para determinar el orden del conjunto *S*.

En [44] se recomienda utilizar un conjunto contiguo para la búsqueda de coeficientes óptimos representados con sub-expresiones. Además, se aconseja que cada coeficiente deba incluir como máximo 2 sub-expresiones. Debido a esta restricción, es necesario aumentar el tamaño del conjunto *S*, de manera que exista más variedad de sub-expresiones. El contenido del conjunto *S* aumenta en función del tamaño de palabra especificado para los coeficientes, de acuerdo a la siguiente expresión

$$Orden = \begin{cases} 2, & \text{para } EWL \leq 9, \\ 2 + (EWL - 9) \times 3, & \text{para } EWL > 9. \end{cases} \quad (3.23)$$

En esta expresión *EWL* representa la *Longitud Efectiva de Palabra* (*Effective Word-Length*), es decir, la longitud de palabra designada para representar los valores de coeficientes sin contar el bit de signo.

Una vez que se conoce el conjunto *S* con las sub-expresiones adecuadas, se debe realizar el proceso de optimización utilizando el enfoque *MILP*. Recuérdese de la sub-sección 3.1.2 que el procedimiento en *MILP* parte de la previa obtención de coeficientes con precisión infinita, continuando con el método *Branch & Bound* para encontrar los coeficientes óptimos enteros. Durante el procedimiento seguido en *B & B*, se van generando nuevos problemas con restricciones añadidas a partir del redondeo de alguna variable independiente.

Debe notarse que, en este caso, el método *B & B* no puede ser utilizado directamente sobre el resultado de precisión infinita obtenido previamente. Esto se debe a que los valores redondeados de las soluciones de precisión infinita no siempre se pueden representar con las sub-expresiones incluidas en el conjunto *S*. Debido a lo anterior, primero es necesario encontrar el entero más cercano a cierto coeficiente de precisión infinita, que pueda ser expresado con las sub-expresiones del conjunto *S*. Enseguida deben elegirse los enteros más cercanos a dicho coeficiente por defecto y por exceso, que puedan ser representados con las sub-expresiones comunes de *S*. Habiendo logrado esto, el método *B & B* puede ser empleado con la seguridad de que los valores óptimos resultantes serán coeficientes factibles de representar con las sub-expresiones de *S*.

Para hallar el entero más cercano a algún valor de coeficiente de precisión infinita, que pueda ser representado con las sub-expresiones comunes de *S*, se propone en [44] el procedimiento siguiente.

Algoritmo 1

1. Inicializar $m = 1$ y $z_0 = h(i)$, con $h(i)$ siendo el valor de precisión infinita del coeficiente i-ésimo.
2. En base a la ecuación (3.22), buscar en el conjunto S la subexpresión $s_{i,m}2^{p_i,m}$ que minimice el error dado como $\left|z_{m-1} - s_{i,m}2^{p_i,m}\right|$.
3. Si $s_{i,m} = 0$ o $m = L_i$, ir al paso 6. De otro modo, ir al paso 4.
4. Actualizar $z_m = z_{m-1} - s_{i,m}2^{p_i,m}$.
5. Incrementar m. Ir al paso 2.
6. Hacer $h_L(i) = \sum_{k=1}^{L_i} s_{i,k}2^{p_i,k}$. Terminar.

Luego de encontrar el valor entero más cercano a cierto coeficiente de precisión infinita dado, es necesario encontrar el otro valor entero más cercano. Esto significa que si en el algoritmo 1 se encontró el entero más cercano por defecto, en el algoritmo 2 debe hallarse el correspondiente por exceso y viceversa. El algoritmo 2 se desarrolla como sigue.

Algoritmo 2

1. Hallar $h_L(i)$ utilizando el algoritmo 1.
2. Si $h_L(i) = h(i)$ hacer $\lceil h(i) \rceil = \lfloor h(i) \rfloor = h_L(i)$; ir al paso 7.
3. Si $h_L(i) > h(i)$ hacer $\lceil h(i) \rceil = h_L(i)$ e $i = -1$; de lo contrario, hacer $\lfloor h(i) \rfloor = h_L(i)$ e $i = 1$.
4. Hacer $h'(i) = h(i)$.
5. Actualizar $h'(i) = h'(i) + i$. Si $h'_L(i) = h_L(i)$, repetir el paso 5. De lo contrario ir al paso 6.
6. Si $i > 0$, hacer $\lceil h(i) \rceil = h'_L(i)$, de otro modo, hacer $\lfloor h(i) \rfloor = h'_L(i)$.
7. Terminar.

Nótese que el método presentado en [44] lleva a cabo la optimización directamente en un espacio definido por valores de coeficientes factibles de representar con las sub-expresiones del conjunto *S.* Esta es la diferencia con el enfoque *MILP* tradicional. Los coeficientes resultantes garantizan tener sub-expresiones comunes entre ellos, permitiendo ahorrar sumas en la implementación.

El método presentado en [44] presenta como ventaja el hecho de hallar los valores óptimos de los coeficientes logrando buena respuesta en frecuencia, usando cierta cantidad de sub-expresiones comunes permitidas. La desventaja es que el tiempo requerido para el diseño de filtros es relativamente alto, sobre todo en filtros de gran longitud.

Capítulo 4

Filtro propuesto

Más maravillosa es la ciencia que mi capacidad; alta es, no puedo comprenderla. *Salmos* 139:6

En el presente capítulo se revisará la propuesta de un filtro FIR eficiente aplicable a los estándares CDMA IS-95 y CDMA 2000. Se ilustrará el procedimiento utilizado para abordar el problema de diseño de este filtro. Además se presentarán las comparaciones entre el filtro propuesto y aquellos encontrados hechos previamente. Por último se mostrará la simulación en VHDL de la respuesta en el tiempo del filtro propuesto.

4.1 Introducción

El filtro que se desea realizar es utilizado en sistemas de radio móvil celular basados en los estándares CDMA IS-95 y CDMA 2000. La función de este filtro es eliminar la interferencia inter-símbolo (ISI) en banda base. Las especificaciones del filtro están dadas como

$$\begin{aligned} f_p &= 590 \text{ kHz}, \\ f_s &= 740 \text{ kHz}, \\ R_p &= 1.5 \text{ dB}, \\ A_s &= -40 \text{ dB}, \end{aligned} \tag{4.1}$$

donde f_p representa la frecuencia de paso y f_s la frecuencia de rechazo en valores de frecuencia lineal. R_p simboliza el rizo en la banda de paso y A_s la atenuación en la banda de rechazo. La frecuencia de muestreo es de 4.9152 MHz. En frecuencia normalizada, los valores de frecuencia de paso y de rechazo están dados como

$$\begin{aligned} \omega_p &= 0.2400\pi, \\ \omega_s &= 0.3012\pi. \end{aligned} \tag{4.2}$$

Dentro de los filtros digitales que no necesitan multiplicadores, los sumadores son los elementos que más recursos utilizan [40]. Así, la principal mejora en un diseño de filtros digitales que cumplen la misma especificación radica en el ahorro de sumadores. La finalidad del filtro propuesto es lograr un diseño con menos de 34 sumadores, debido a que ésta es la cantidad mínima obtenida en diseños previos. El filtro está basado en una estructura IFIR, utilizando como interpolador la conexión en cascada de un filtro Coseno Modificado y un filtro CIC de dos etapas. La ventaja de usar estos filtros en el interpolador es que no tienen multiplicadores. Esto permite llegar a un diseño de baja complejidad.

4.2 Procedimiento de diseño

La desventaja inherente a los filtros simples es la gran caída en la banda de paso y la pobre atenuación en la banda de rechazo. Cuando los filtros deseados son de banda muy angosta, la respuesta en frecuencia de los filtros simples que conforman al interpolador puede ser corregida. Esto significa que es posible aplicar compensadores o la técnica sharpening, como en [26 - 28]. No obstante, las especificaciones de los filtros IS-95 no pueden ser completamente satisfechas utilizando estos métodos, porque la banda de paso de los filtros IS-95 no es muy angosta.

Para vencer el problema anterior, aquí se propone diseñar primero el filtro interpolador y después el filtro modelo. Este último debe compensar la caída en banda de paso introducida por el filtro interpolador y además debe cumplir con las especificaciones requeridas.

El diagrama de flujo correspondiente al procedimiento de diseño se presenta en la figura 4.1.

Los pasos a seguir para el diseño del filtro propuesto se enlistan enseguida. Los pasos 1 y 2 son detallados en las sub-secciones siguientes, debido a que incluyen el procedimiento de diseño del filtro interpolador y del filtro modelo, respectivamente.

Paso 1. Diseñar un filtro interpolador (filtro anti-imágenes en una estructura IFIR) basado en la conexión en cascada de *K* filtros simples mostrados en la sección 2.4. La atenuación en la banda de rechazo de este filtro interpolador debe ser igual o más alta que la deseada. Más detalles en la sub-sección 4.2.1.

Paso 2. Diseñar el filtro modelo que cumpla las características del filtro deseado y compense la caída en la banda de paso del filtro interpolador. Esto se explica en la sub-sección 4.2.2.

Paso 3. Realizar la expansión por $M = 3$ del filtro modelo y conectarlo en cascada con el filtro interpolador.

Paso 4. Si las especificaciones en la banda de rechazo no se cumplen, regresar al paso 1 y aumentar el número de etapas *K*.

Paso 5. Si las especificaciones en la banda de paso no se cumplen, regresar al paso 2 y restringir la especificación en banda de paso, a la vez que se relaja la especificación en banda de rechazo. Aumentar el orden del filtro modelo de ser necesario.

Paso 6. Redondear los coeficientes del filtro modelo partiendo de un factor $\alpha = 2^{-5}$. Si las especificaciones no se cumplen, disminuir α.

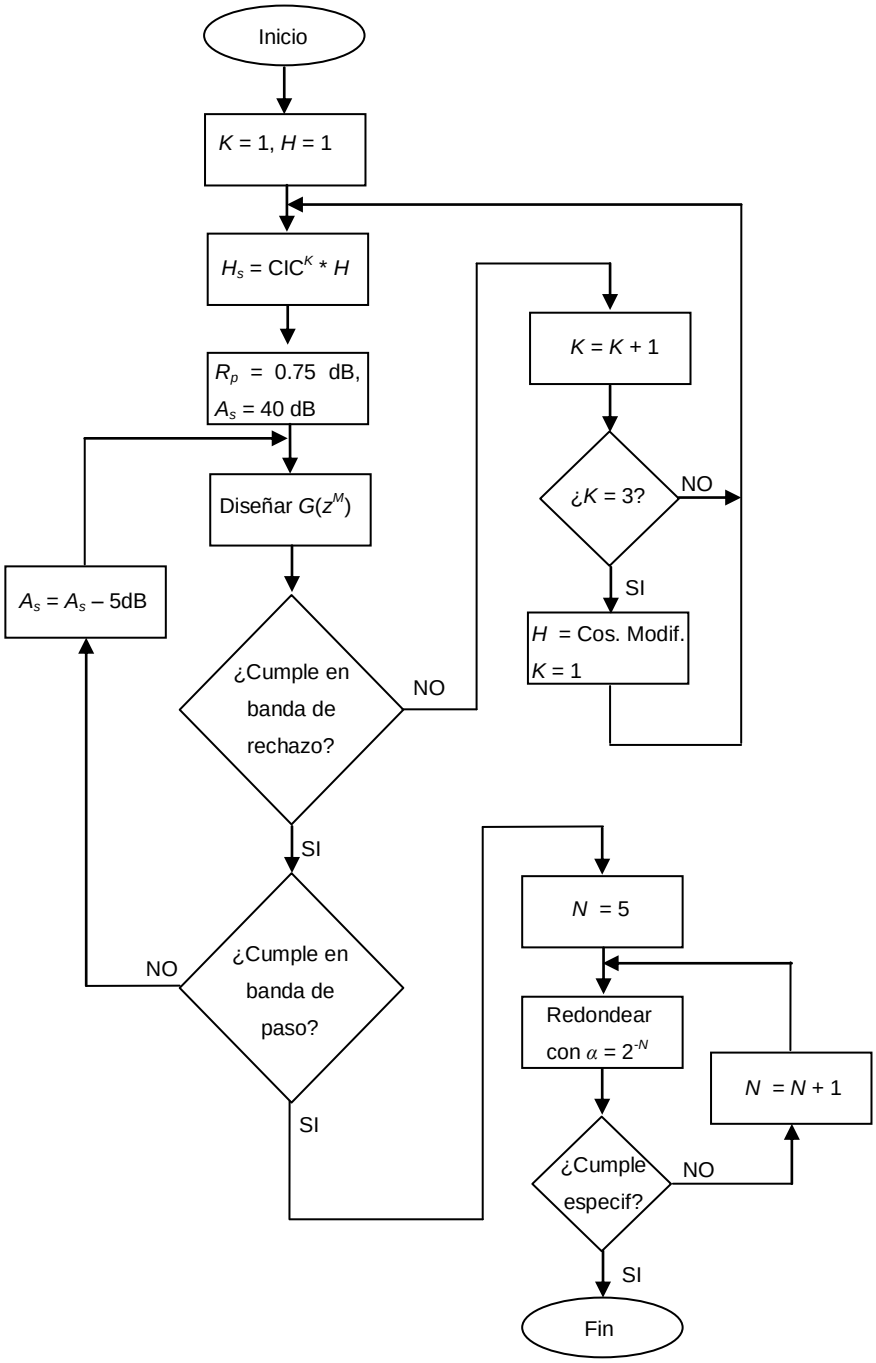


Figura 4.1. Diagrama de flujo correspondiente al procedimiento de diseño.

4.2.1 Diseño del filtro interpolador

Los filtros CIC presentan nulos en frecuencias con valor múltiplo de ($2k\pi/M$). Las imágenes de respuesta en frecuencia de un filtro modelo expandido por el mismo valor M, incluidas en el intervalo [0, π], pueden ser

atenuadas por los nulos del filtro CIC. No obstante, el número *K* necesario para atenuar adecuadamente las imágenes de respuesta en frecuencia del filtro modelo alcanza valores considerablemente altos cuando estas imágenes no son muy angostas. Este es el caso de los filtros IS-95. Por esta razón, no es posible considerar esta alternativa como viable en el diseño del filtro interpolador.

Debido a lo anterior se propone partir de un filtro CIC diseñado con $M = 5$, cuyo primer nulo aparece centrado en $\omega = 0.4\pi$. Con esto es posible lograr atenuación suficiente en la banda de rechazo sin intensificar demasiado la caída en banda de paso. Los pasos de diseño del filtro interpolador son presentados a continuación.

Paso 1. Elegir un filtro CIC utilizando un factor de decimación $M = 5$. Realizar el procedimiento para el diseño del filtro deseado.

Paso 2. Si el filtro anterior no se ajusta a las características deseadas, aplicar más etapas en cascada. Si se requieren más de 3 etapas y el filtro aún no se ajusta a las características deseadas, ir al paso 3.

Paso 3. Aplicar un filtro Coseno-Modificado con $N = 2$ en cascada y repetir el proceso de diseño, reduciendo la cantidad de etapas en cascada del filtro CIC.

Se ha establecido en el procedimiento un número máximo de etapas de filtro CIC en cascada igual a 3, con la finalidad de limitar el número de sumadores requeridos. El problema inherente es que un pequeño número de etapas tiene como consecuencia la deficiencia en la respuesta en frecuencia tanto en la banda de paso como en la de rechazo. No obstante, la banda de rechazo puede verse mejorada con la conexión en cascada de un filtro Coseno Modificado, de acuerdo al paso 3 de diseño del filtro interpolador.

La respuesta en frecuencia $H_s(e^{j\omega})$ del filtro interpolador obtenido al seguir los pasos de diseño se presenta en la figura 4.2. El filtro resultante está formado por la conexión en cascada de un filtro CIC de dos etapas y un filtro

Coseno Modificado. Cada filtro CIC requiere 2 sumadores, y el filtro Coseno Modificado utiliza 3 sumadores. Así, el filtro interpolador es logrado utilizando solamente 7 sumadores. La figura 4.3 muestra que en la frecuencia de paso deseada el filtro interpolador tiene una caída de 15.5 dB. Además en la frecuencia de rechazo deseada la atenuación es de -27 dB. Debido a que no se satisfacen las especificaciones deseadas, el filtro modelo debe compensar estos errores.

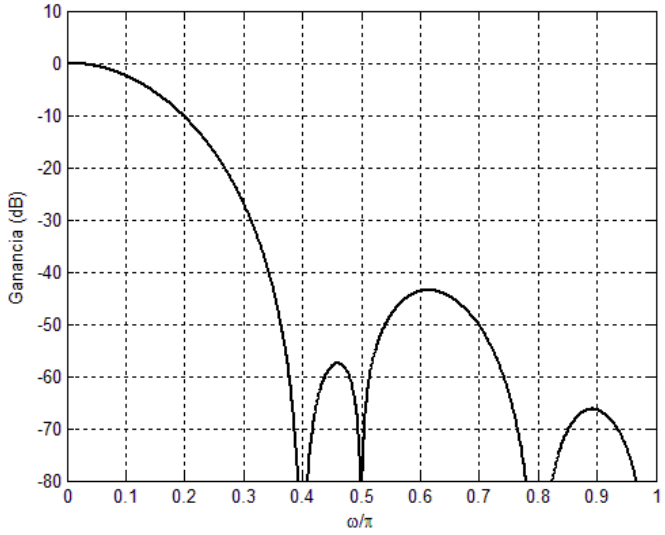


Figura 4.2. Respuesta en frecuencia del filtro interpolador, $H_s(e^{j\omega})$.

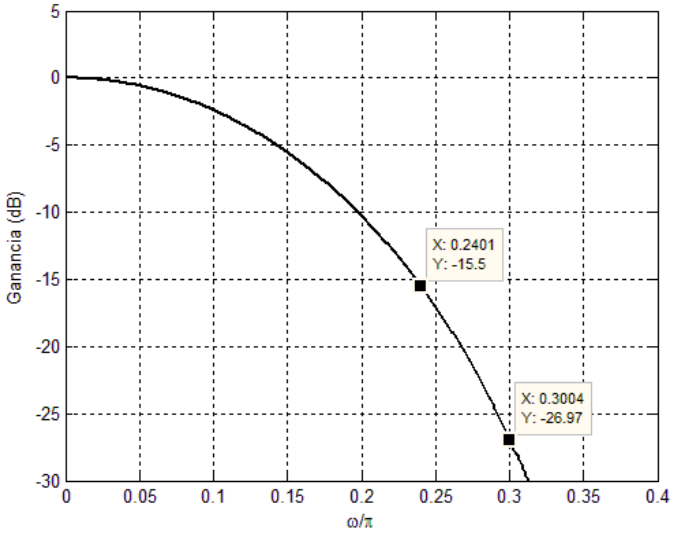


Figura 4.3. Ganancia del filtro interpolador en las frecuencias de paso y de rechazo.

4.2.2 Diseño del filtro modelo utilizando optimización multiobjetivo

El filtro modelo debe diseñarse para llevar a cabo la compensación en la banda de paso del filtro interpolador. Además el filtro modelo debe alcanzar la atenuación requerida en la frecuencia de rechazo. Para este fin el diseño es planteado como un problema de optimización multiobjetivo.

Debido a que el filtro interpolador no depende del factor de interpolación *M*, es posible aumentar este valor y llegar a un filtro modelo más simple. Para el caso especial de diseño de filtros IS-95, los únicos valores permitidos para hacer la expansión del filtro modelo son $M = 2$ y $M = 3$. Valores más altos exceden el límite dado por la velocidad de muestreo. La tabla 4.1 muestra la comparación entre las longitudes estimadas de los filtros modelo e interpolador para las dos estructuras IFIR tradicionales permitidas.

Factor de interpolación *M*	**Longitud del filtro modelo**	**Longitud del filtro interpolador**
2	26	8
3	18	30

Tabla 4.1. Comparación de las longitudes de filtros modelo e interpolador para una estructura IFIR clásica, con $M = 2$ y $M = 3$.

Obsérvese que el filtro modelo requiere menos coeficientes cuando se utiliza $M = 3$. Así, en el diseño propuesto el factor de interpolación utilizado para el filtro modelo será $M = 3$.

La respuesta en frecuencia deseada en la banda de paso es inversa a la respuesta en frecuencia del filtro interpolador utilizado. Esto se ilustra en la figura 4.4. Por otra parte, debido a que el filtro interpolador tiene -27 dB en la frecuencia de rechazo, la respuesta deseada en la banda de rechazo es de -20 dB. Al realizar la conexión en cascada entre el filtro modelo expandido y

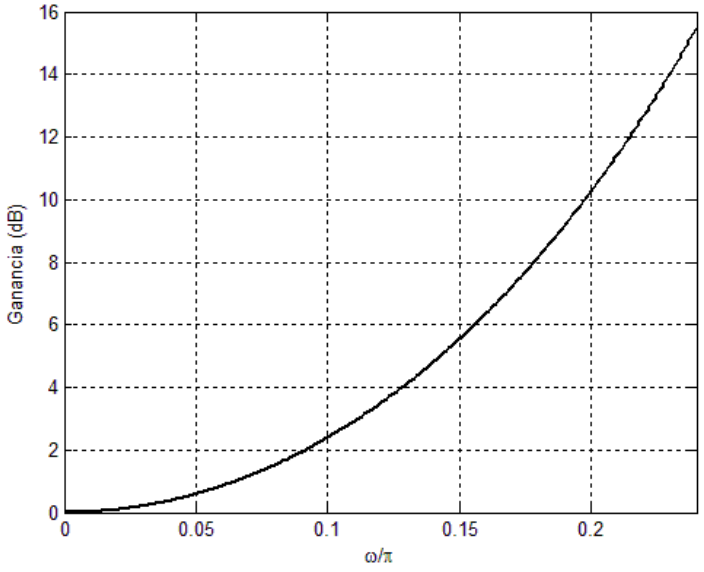


Figura 4.4. Ejemplo de la respuesta en frecuencia deseada en la banda de paso del filtro modelo.

el interpolador, el filtro resultante tendrá una respuesta plana en la banda de paso y, a la vez, la atenuación requerida en la banda de rechazo. Las especificaciones para el filtro modelo están dadas como

$$\begin{aligned} \omega_{p,G} &= M*(\omega_p), \\ \omega_{s,G} &= M*(\omega_s), \\ R_{p,G} &= 1.5 \text{ dB}, \\ A_{s,G} &= -20 \text{ dB}, \end{aligned} \tag{4.3}$$

con ω_p y ω_s dados en la expresión (4.2) y con factor $M = 3$.

Para llevar a cabo la optimización, los valores deseados de respuesta en frecuencia deben ser incluidos en un conjunto $\mathbf{F}^*$ dado como

$$\mathbf{F}^* = \left[\frac{1}{H_s(\omega_1)}, \frac{1}{H_s(\omega_2)}, \ldots, \frac{1}{H_s(\omega_p)}, 0, 0, \ldots, 0 \right]. \tag{4.4}$$

El conjunto $\mathbf{F}^*$ contiene *m* valores deseados correspondientes a *m* puntos de frecuencia. Siguiendo el ejemplo 2.3 se utiliza *m* = 512, asignando 256 a la banda de paso y 256 a la banda de rechazo.

La expresión (4.4) demanda conocer 256 valores que contengan de forma justa a la porción de respuesta en frecuencia del filtro interpolador que se encuentra dentro de la banda de paso del filtro deseado. Una forma sencilla de obtener estos valores es calcular la respuesta en frecuencia del filtro interpolador en *x* cantidad de puntos, tal que los primeros 256 contengan la parte que se busca. La cantidad *x* puede estimarse como

$$x = \frac{256}{\omega_p}, \tag{4.5}$$

donde *x* se usa en la función `freqz` del paquete *Signal Processing Toolbox* de MATLAB para indicar cuántos puntos de frecuencia se requieren. De todos estos puntos, solo los primeros 256 serán utilizados para expresar los valores deseados en banda de paso en la expresión (4.4).

Cada elemento del conjunto $\mathbf{F}^*$ tiene una correspondiente función objetivo. El conjunto de funciones objetivo está dado por

$$\mathbf{H}(e^{j\omega}) = \left\{H_1(e^{j\omega_1}), H_2(e^{j\omega_2}), ..., H_m(e^{j\omega_m})\right\}. \tag{4.6}$$

Cada función objetivo es representada como

$$H_i(e^{j\omega_i}) = \sum_{k=1}^{12} b(k) \cos \omega_i \left(k - \frac{1}{2}\right). \tag{4.7}$$

La ecuación dada en (4.7) indica que el filtro modelo resultante será pasa bajas de longitud par.

El proceso de optimización consiste en encontrar los valores *b*(*k*) que provoquen que las funciones dadas en (4.6) se aproximen lo máximo posible a los valores deseados expresados en (4.4).

El grado relativo de tolerancia para alcanzar los valores deseados dados en (4.4) es controlado con el vector de ponderaciones, expresado como

$$\mathbf{w} = \left[1,\ 1,\ \ldots,\ 1,\ \frac{\delta_s}{\delta_p},\ \frac{\delta_s}{\delta_p},\ \ldots,\ \frac{\delta_s}{\delta_p}\right]. \tag{4.8}$$

El vector de ponderaciones es igual a 1 en la banda de paso y δ_s/δ_p en la banda de rechazo. Aquí δ_p representa el rizo en banda de paso y δ_s representa el rizo en banda de rechazo en valor normalizado. A partir de las correspondientes cantidades expresadas en dB en la ecuación (4.3), los valores de δ_p y δ_s son calculados como

$$\begin{aligned} \delta_p &= \left(\frac{(10^{(R_p/20)}-1)/(10^{(R_p/20)}+1)}{2}\right), \\ \delta_s &= 10^{(-A_s/20)}. \end{aligned} \tag{4.9}$$

El filtro modelo se obtiene al aplicar el método de optimización multiobjetivo explicado en el capítulo 2, utilizando la función `fgoalattain`. Dicha función es utilizada como

[**coef**] = `fgoalattain` (*fun*, $\mathbf{coef_0}$, $\mathbf{F}^*$, **w**, [ ], [ ], [ ], [ ], [ ], [ ], [ ], **ω**), (4.10)

donde $\mathbf{coef_0}$ es el vector de valores iniciales de coeficientes, $\mathbf{F}^*$ está dado en la expresión (4.4) y **w** está representado por la expresión (4.8). La función *fun* es una función de MATLAB que debe ser creada para desarrollar el proceso de optimización. Ésta está dada por

$$\mathbf{y} = fun(\mathbf{coef}, \boldsymbol{\omega}), \tag{4.11}$$

donde

- **ω** es el vector de parámetros que contiene los *m* puntos de frecuencia elegidos.
- **coef** es el vector de coeficientes que se debe hallar para minimizar a la función.
- **y** es el vector de resultados de cada función. El primer elemento de **y** corresponde a la primera función objetivo evaluada en **coef** y ω_1, el segundo corresponde a la segunda función objetivo evaluada en **coef** y ω_2, y así, hasta llegar a la *m*-ésima función objetivo.

La figura 4.5 muestra la respuesta en frecuencia del filtro modelo resultante.

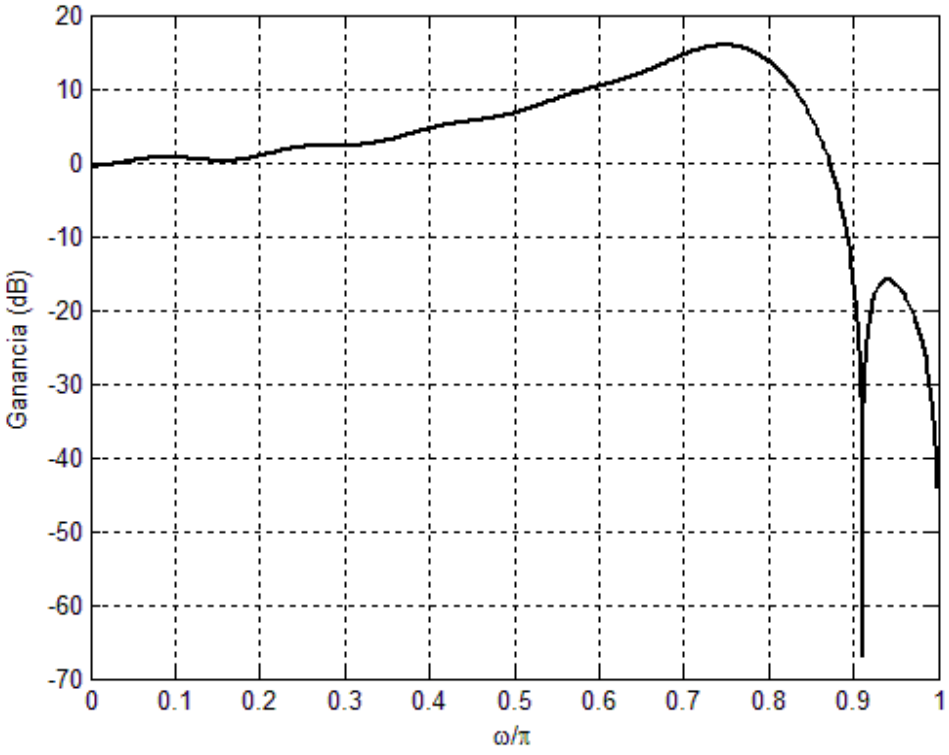


Figura 4.5. Respuesta en frecuencia del filtro modelo.

La respuesta en frecuencia del filtro modelo expandido es presentada en la figura 4.6. Obsérvese en esta figura que esta respuesta tiene una máxima subida de 15.4 dB en la frecuencia ω_p. Esto compensa la caída producida por el filtro interpolador. Además en la frecuencia ω_s el filtro modelo expandido tiene -20 dB, como fue calculado. Nótese que los rizos de banda de rechazo están ligeramente arriba de -20 dB. Sin embargo en esa frecuencia la atenuación del filtro interpolador es suficiente para lograr un resultado final con atenuación por debajo de -40 dB. La tabla 4.2 presenta los valores de coeficientes del filtro modelo obtenidos al aplicar el método de optimización.

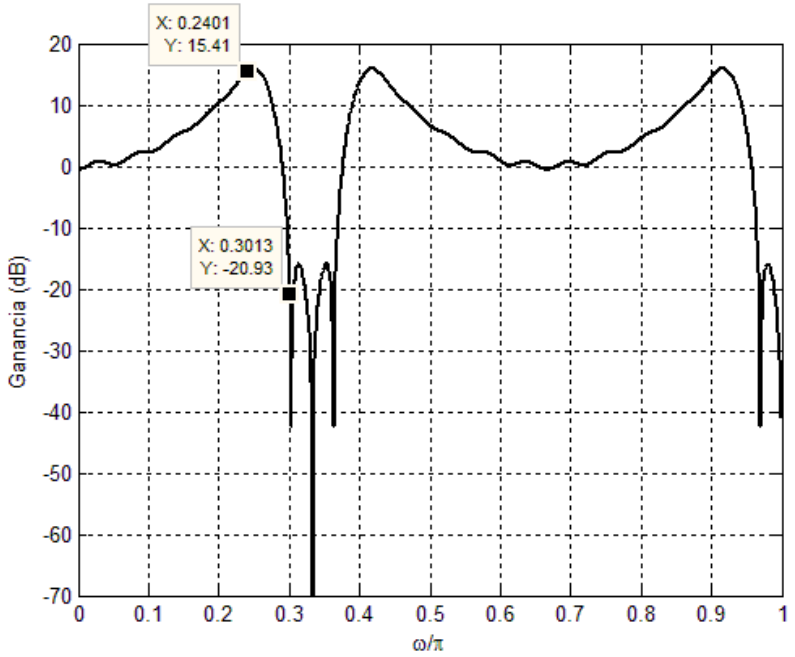


Figura 4.6. Respuesta en frecuencia del filtro modelo expandido en *M* = 3.

$h_0 = h_{23} = -0.0645$	$h_4 = h_{19} = 0.1398$	$h_8 = h_{15} = -0.0078$
$h_1 = h_{22} = 0.0750$	$h_5 = h_{18} = -0.2942$	$h_9 = h_{14} = 0.5860$
$h_2 = h_{21} = -0.0640$	$h_6 = h_{17} = 0.3885$	$h_{10} = h_{13} = -1.2208$
$h_3 = h_{20} = -0.0073$	$h_7 = h_{16} = -0.3168$	$h_{11} = h_{12} = 1.2555$

Tabla 4.2. Coeficientes del filtro modelo, con precisión infinita.

4.3 Realización del filtro propuesto

El filtro propuesto resulta de conectar en cascada el filtro modelo expandido $G(z^M)$ con el filtro interpolador $H_s(z)$. A su vez, el filtro interpolador está formado por la conexión en cascada de un filtro CIC de dos etapas y un filtro Coseno Modificado. La figura 4.7 muestra el diagrama a bloques del filtro propuesto.

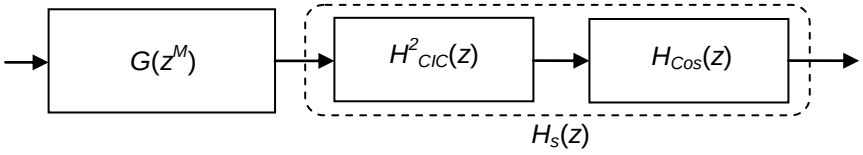


Figura 4.7. Estructura del filtro propuesto.

Las respuestas en frecuencia del filtro modelo expandido, del filtro interpolador y del filtro completo se muestran en la figura 4.8.

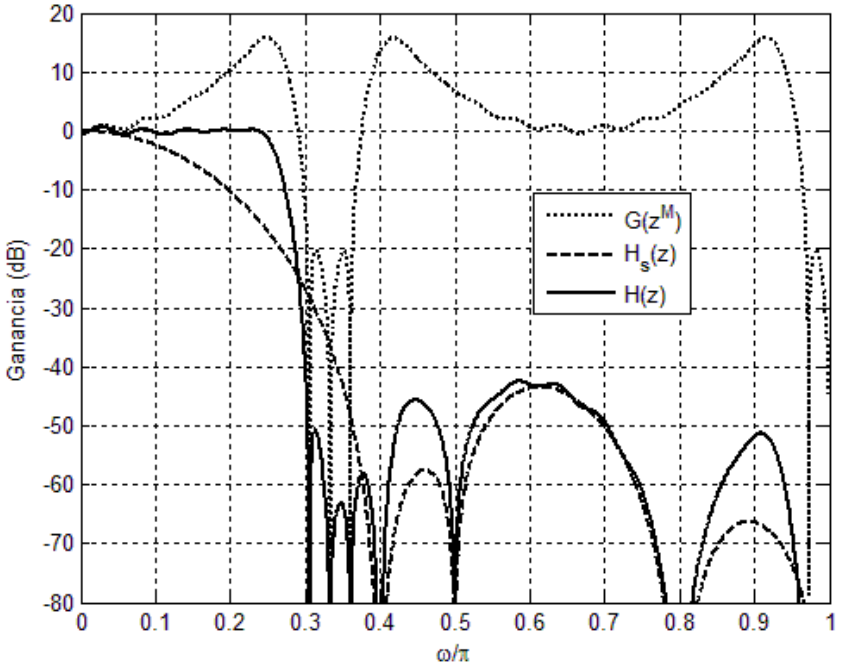


Figura 4.8. Respuestas en frecuencia del filtro modelo expandido, filtro interpolador y filtro completo.

La representación en dB de las respuestas en frecuencia permite que, para una conexión en cascada, éstas se puedan sumar. Así, obsérvese que en la banda de paso el resultado final es plano, debido a que el filtro modelo expandido ha compensado la caída del filtro interpolador. En la banda de rechazo la respuesta del filtro propuesto siempre se encuentra por debajo de -40 dB. Nótese que en la frecuencia $\omega = 0.42\pi$ el filtro modelo expandido presenta una subida pronunciada, sin embargo ésta es eliminada por el nulo del filtro CIC, centrado en $\omega = 0.4\pi$.

Los coeficientes del filtro modelo son redondeados con la ecuación (3.1), utilizando una constante de redondeo $\alpha = 2^{-5}$. Este factor permite la mínima longitud de palabra en los coeficientes y a la vez consigue que la respuesta en frecuencia del filtro modelo aún cumpla con las especificaciones. La figura 4.9 muestra la respuesta en frecuencia del filtro final propuesto. Además, una ampliación en la banda de paso es presentada en la figura 4.10.

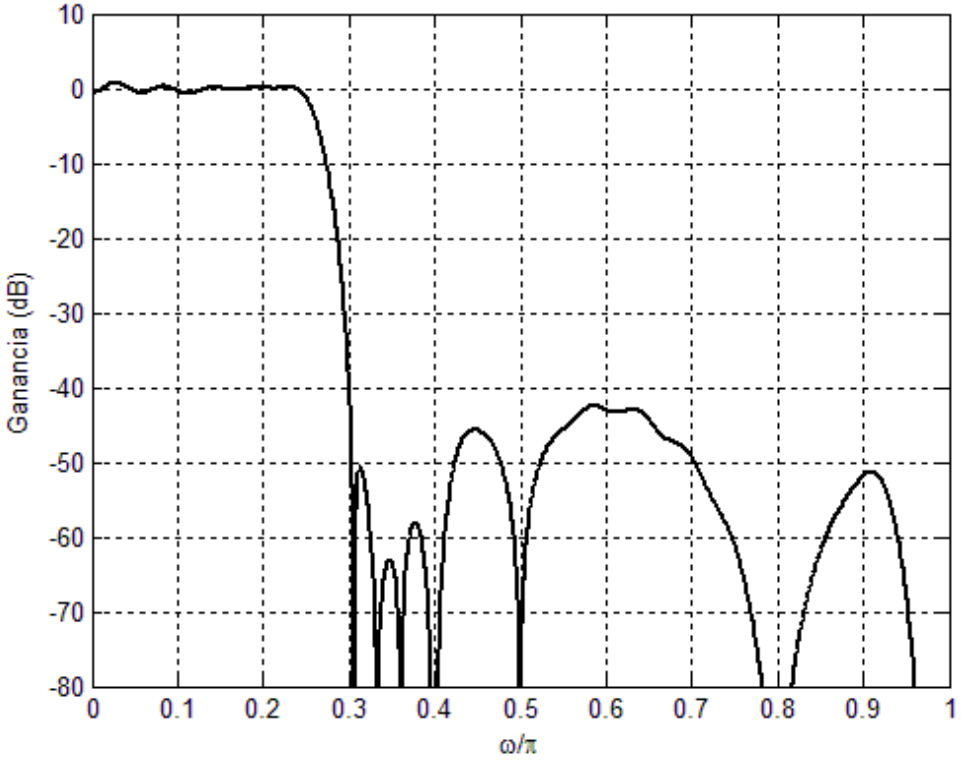


Figura 4.9. Respuesta en frecuencia del filtro propuesto.

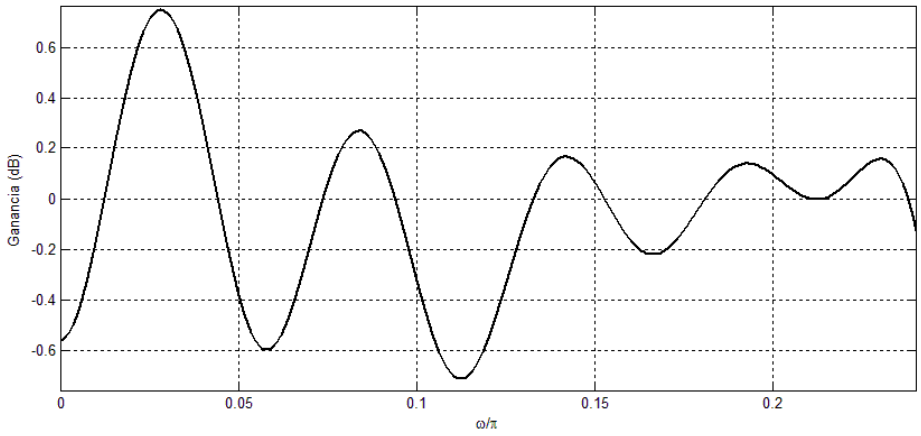


Figura 4.10. Ampliación en la banda de paso.

El filtro propuesto satisface las especificaciones del estándar IS-95 dadas en (4.1).

Los coeficientes del filtro modelo se representan en código *CSD* para ahorrar sumadores en su representación. Además se utiliza la forma directa transpuesta para el filtro modelo, agrupando todos los coeficientes en un bloque multiplicador. La tabla 4.3 muestra los coeficientes del filtro modelo representados en *CSD*. Debido a la estructura directa transpuesta, algunos coeficientes pueden ser expresados como la suma de otros dos. Esto puede apreciarse en la tabla 4.4. Recuérdese que la multiplicación por una potencia de dos no añade complejidad a los coeficientes.

$h_0 = h_{23} = -2^{-4}$	$h_6 = h_{17} = 2^{-1} - 2^{-3}$
$h_1 = h_{22} = 2^{-4}$	$h_7 = h_{16} = -2^{-2} - 2^{-4}$
$h_2 = h_{21} = -2^{-4}$	$h_8 = h_{15} = 0$
$h_3 = h_{20} = 0$	$h_9 = h_{14} = 2^{-1} + 2^{-3} - 2^{-5}$
$h_4 = h_{19} = 2^{-3}$	$h_{10} = h_{13} = -2^{0} - 2^{-2} + 2^{-5}$
$h_5 = h_{18} = -2^{-2} - 2^{-5}$	$h_{11} = h_{12} = +2^{0} + 2^{-2}$

Tabla 4.3. Coeficientes del filtro modelo para el filtro propuesto.

$h_0 = h_{23} = -2^{-4}$	$h_6 = h_{17} = 2^{-1} - 2^{-3}$
$h_1 = h_{22} = 2^{-4}$	$h_7 = h_{16} = -2^{-2} - 2^{-4}$
$h_2 = h_{21} = -2^{-4}$	$h_8 = h_{15} = 0$
$h_3 = h_{20} = 0$	$h_9 = h_{14} = -h_5 - h_7$
$h_4 = h_{19} = 2^{-3}$	$h_{10} = h_{13} = 2^2 (h_7) + 2^{-5}$
$h_5 = h_{18} = -2^{-2} - 2^{-5}$	$h_{11} = h_{12} = -2^2 (h_7)$

Tabla 4.4. Representación de los coeficientes en un bloque multiplicador.

El filtro modelo presenta un total de 25 sumadores: únicamente 5 sumadores en los coeficientes y 20 en la estructura.

La tabla 4.5 presenta las características del filtro propuesto. Obsérvese que en total este filtro requiere únicamente 32 sumadores. Esto es mejor que cualquiera de los filtros que se han encontrado hechos previamente.

Longitud del filtro modelo $G(z^M)$	24
Número de sumadores del filtro modelo $G(z^M)$	25
Número de sumadores del filtro interpolador $H_s(z)$	7
Número total de sumadores	32
Constante de redondeo (coeficientes del filtro modelo)	2^{-5}

Tabla 4.5. Características del filtro propuesto.

4.4 Comparación con diseños previos

Se ha mencionado que la finalidad en un diseño eficiente de filtros digitales es minimizar los recursos de hardware requeridos. Dentro de los filtros digitales que no necesitan multiplicadores, los sumadores son los elementos que más recursos utilizan [40]. Así, la principal mejora en un diseño de filtros digitales que cumplen la misma especificación radica en el ahorro de sumadores.

Como ya se ha visto en el capítulo 1, existen realizaciones previas presentadas en [11 - 14]. El filtro propuesto en [11] está basado en un diseño directo obtenido con el algoritmo Parks-McClellan, mientras que los filtros propuestos en [12 - 14] fueron realizados utilizando una estructura de prefiltro- ecualizador.

En [11] se presentan distintas estructuras para implementar el filtro IS-95, además de la estructura directa. Esto es porque que el filtro basado en la estructura directa transpuesta, como la de la figura 3.6, consumiría muchos recursos debido a la gran cantidad de sumadores requeridos. De la misma manera, el filtro presentado en [3] muestra una forma de implementar el filtro en FPGA, en forma multiplexada en el tiempo.

La comparación realizada aquí será en base a estructura directa, es decir, a aquellos filtros que no sean multiplexados en el tiempo, como el presentado en [12]. Considérese la tabla 4.6. En ella se muestran los resultados reportados en [11], [13] y [14] sobre los diseños realizados para filtros IS-95. Además esta tabla muestra los resultados correspondientes al filtro propuesto desarrollado aquí.

Obsérvese que el filtro propuesto requiere una precisión de 5 bits para representar la parte fraccionaria de los coeficientes. De los filtros realizados antes, solamente el presentado en [14] requiere la misma precisión.

Respecto al número de sumadores utilizados, el filtro propuesto en [14] presenta los mejores resultados entre todos aquellos tomados de las

referencias. No obstante, el filtro propuesto supera la cantidad límite establecida en [5], con únicamente 32 sumadores utilizados.

Ref.	***L* de $G(z^M)$**	***M* de $G(z^M)$**	**No. de sumas $G(z^M)$**	**No. de sumas $H_s(z)$**	**No. Total de sumas**	α
[11]	48	_	66	_	66	2^{-10}
[13]	22	2	30	8	38	2^{-6}
[14]	22	2	No especificado	No especificado	34	2^{-5}
Propuesto	24	3	25	7	32	2^{-5}

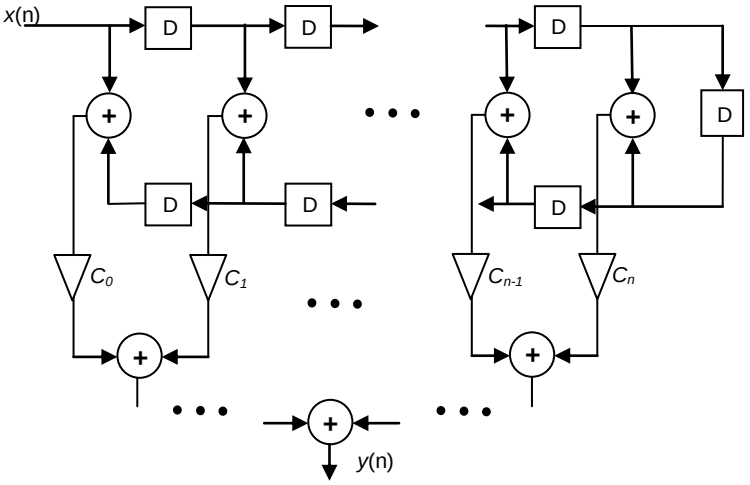
Tabla 4.6. Comparación entre el filtro propuesto y los filtros existentes.

Por otra parte, debe verse que el filtro propuesto en [11] se realizó con una técnica simple, aunque se trata de un diseño menos eficiente. Los filtros presentados en [13] y [14] fueron logrados utilizando el enfoque *MILP*, explicado en la sección 3.2. Además, el filtro en [14] fue diseñado en base a dos procedimientos de optimización distintos, uno para el cálculo de coeficientes y el otro para controlar el factor de relajación. Nótese entonces que el procedimiento de diseño utilizado en el filtro propuesto es más sencillo. Esto es porque el proceso de síntesis basado en *MILP* requiere mucho tiempo en llevarse a cabo.

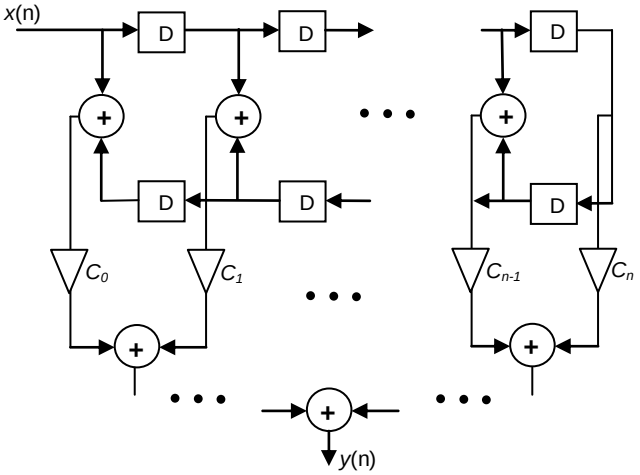
Así, el filtro propuesto requiere una menor cantidad de sumadores que cualquiera de los diseños existentes, por lo tanto puede ser considerado un diseño más eficiente. Nótese además que el procedimiento propuesto para el diseño de filtros IS-95 puede ser considerado útil y práctico.

4.5 Simulación en VHDL

Existen varias estructuras para la realización de filtros FIR. No obstante, la mayoría de sistemas en los que se prefiere alto rendimiento se realizan con implementaciones en forma directa [31]. Por otra parte, se ha mencionado que en los filtros IS-95 la fase lineal es un requisito indispensable, por lo tanto estos filtros deben ser simétricos. La figura 4.11a muestra la realización en forma directa para un filtro de fase lineal de longitud par. La figura 4.11b muestra el diagrama correspondiente a un filtro simétrico de longitud impar.



a) filtro con longitud par.



b) filtro con longitud impar

Figura 4.11. Filtros de fase lineal con estructura directa.

La implementación presentada en la figura 4.11 lleva a cabo el proceso de convolución expresado en la ecuación (2.1) de forma directa; de ahí el nombre de *estructura directa*. Nótese además que las estructuras de la figura 4.11 explotan las condiciones de simetría. De la figura anterior es posible reconocer tres elementos básicos incluidos en la implementación de un filtro FIR. Éstos se enlistan a continuación [17].

- *Sumador*. Es una unidad de dos entradas y una salida que realiza la suma de dos cantidades. La suma de más de dos números se obtiene implementando sucesivamente sumadores de dos entradas.
- *Multiplicador*. Es un elemento de una sola entrada y una sola salida. Cada multiplicador representa cierto valor. La salida de un multiplicador es el producto entre una cantidad de entrada y el valor del multiplicador.
- *Elemento de retraso*. Éste da un retraso de un periodo de muestreo a la señal que pasa a través de él. Se trata entonces de un registro paralelo de *n* bits, con *n* siendo el ancho de bus de la entrada del filtro.

En la entrada, la muestra que aparece sobre el índice 0 indica que ese valor es el que existe en el instante 0, y se menciona como $x(0)$. A su vez, la muestra que existe en el índice 1 es la que llega en el siguiente periodo de muestreo, denotada como $x(1)$. De esta manera se describe completamente la secuencia de entrada. Antes del tiempo cero las muestras que existen tienen valor 0, lo cual implica que la secuencia de entrada está en tiempo real (no puede existir algo que aún no ha sucedido) y por lo tanto es causal.

Obsérvese que cuando llega a la estructura la primera muestra de la señal de entrada, ésta se multiplica en ese instante con el primer coeficiente del filtro. El resultado se suma a otros productos que son obtenidos al multiplicar las muestras que llegaron en periodos anteriores con sus respectivos coeficientes de filtro (ocurren tantos productos como coeficientes tenga el filtro). Como se trata del primer instante, no ha llegado nada antes y por lo

tanto se interpreta que las muestras anteriores son cero. Así los otros coeficientes del filtro se multiplican por cero.

En el periodo siguiente, la muestra que llega se multiplica por el primer coeficiente y la del instante anterior por el segundo coeficiente, debido al retraso *D*. De nuevo los demás productos son cero (porque solo existe la muestra actual y la del ciclo anterior) y la salida resultante para ese instante es la suma de todos los productos. Esto se repite en los periodos de operación siguientes.

Obsérvese la figura 4.12. Ésta representa la forma en que opera un filtro en VHDL, desde un enfoque de comportamiento. En el código de VHDL se especifica primero un tipo de datos *coef_array* como un arreglo de *n*/2 elementos de tipo con signo. En este caso *n* representa la longitud del filtro. Enseguida se declara una constante llamada *coef* cuyo tipo es *coef_array* y contiene todos los coeficientes del filtro. Además se define el tipo de datos *ac_array* como un arreglo de *n* elementos de tipo con signo. La variable llamada *ac* es declarada con tipo *ac_array*. Ésta guardará los valores de entrada e inicialmente contiene ceros en cada elemento del arreglo.

La arquitectura está compuesta por un proceso secuencial, sensible a los cambios en el reloj. Antes de que ocurra un cambio en el reloj, se realiza un corrimiento del contenido de cada elemento del arreglo *ac* hacia el elemento superior. Es decir, el contenido del elemento 0 se transfiere al 1, el del elemento 1 al elemento 2, y así sucesivamente. Enseguida se espera hasta que ocurra una transición positiva en el reloj. Una vez ocurrido el flanco de subida, se asigna al primer elemento del arreglo *ac* el valor de entrada y se calcula la suma de todos los productos entre los arreglos *ac* y *coef*. Nótese que debido a la simetría del filtro, cada coeficiente de *coef* es multiplicado por la suma de dos elementos de *ac*. El resultado se asigna a la variable *sal* (que es del mismo tipo que el puerto *salida*) y la variable *sal* asigna su valor al puerto *salida*. Este proceso se repite a cada flanco de subida de reloj,

provocando que en cada nuevo ciclo se efectúe un corrimiento en los valores de *ac*, lo cual implica la realización de los retrasos.

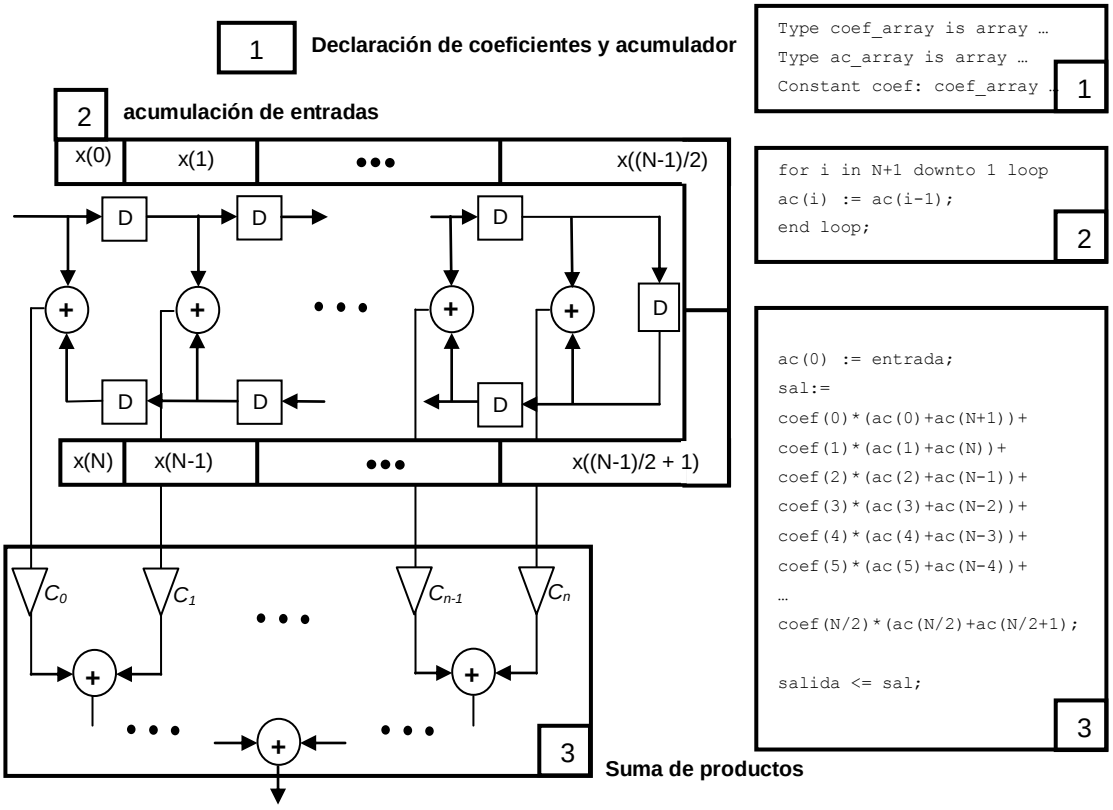


Figura 4.12. Relación entre un filtro de fase lineal con estructura directa y su código VHDL.

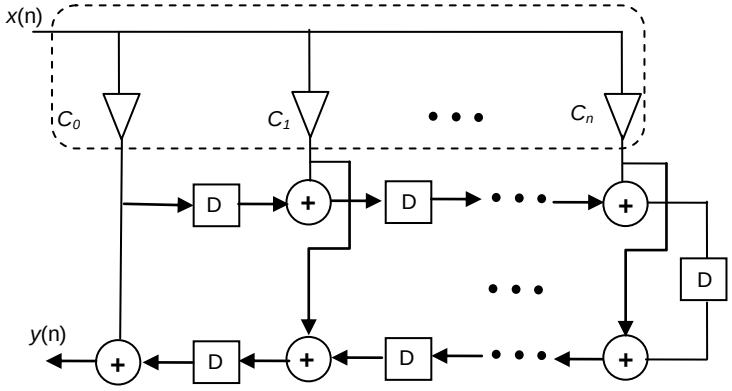
4.5.1 Simulación del filtro modelo en forma directa transpuesta

Si bien la estructura presentada en la figura 4.11 es adecuada para filtros FIR de fase lineal, existen formas distintas que presentan ciertas ventajas. La estructura utilizada en esta propuesta es llamada *estructura directa transpuesta*.

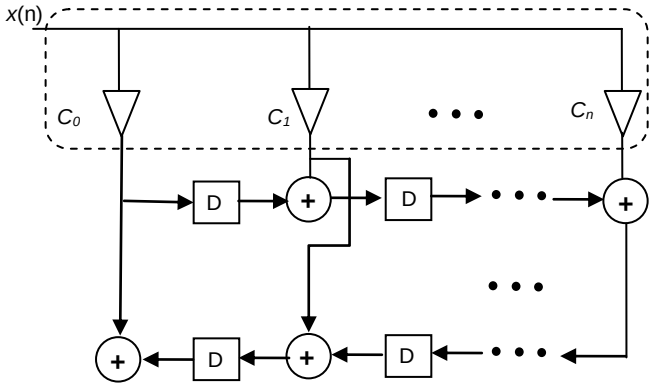
La implementación de fase lineal en forma directa traspuesta es una variación a la realización de un filtro FIR de fase lineal en forma directa. La implementación en forma transpuesta está basada en una estructura directa, bajo las modificaciones que se enlistan a continuación [31]:

- Intercambiar la entrada y la salida.
- Invertir la dirección del flujo de señal.
- Sustituir cada sumador por un punto de bifurcación, y viceversa.

La figura 4.13a muestra la estructura directa transpuesta para un filtro FIR de fase lineal con longitud par. La figura 4.13b presenta la estructura transpuesta correspondiente a un filtro con longitud impar. Obsérvese que en ambos casos los coeficientes pueden ser considerados como un bloque de multiplicadores, denotado por las líneas punteadas.



a) filtro con longitud par



b) filtro con longitud impar

Figura 4.13. Filtros de fase lineal con estructura directa transpuesta.

En el capítulo 3 se estudió que la operación de multiplicación puede ser sustituida por sumas y corrimientos, logrando así menor complejidad. Además se revisaron en la sección 3.3 algunos métodos eficientes para la eliminación de sub-expresiones comunes contenidas entre coeficientes dentro de un bloque de multiplicadores. Así, la principal ventaja de una implementación basada en una estructura directa transpuesta es el hecho de poder reunir todos los coeficientes en un solo bloque. Con esto es posible reducir en gran manera la cantidad de sumadores empleados en el bloque multiplicador. Los resultados exhibidos en [40 - 44] demuestran que una implementación basada en estructura directa transpuesta logra más ahorros en sumadores que una implementación basada en estructura directa.

La realización de un filtro FIR con la estructura directa transpuesta presentada en la figura 4.13 es relativamente sencilla siguiendo un enfoque estructural. Esto se debe a que cada elemento puede ser sustituido directamente por la unidad digital correspondiente dentro de una sola entidad. Así, el filtro modelo propuesto es simulado utilizando instanciación de componentes, donde cada elemento de retraso es un registro paralelo y cada punto de suma es un sumador de dos entradas. De acuerdo a la tabla 4.4, el bloque de multiplicadores puede ser sustituido por 5 sumas únicamente. El diagrama correspondiente es presentado en la figura 4.14. Nótese que, dado que se trata de un filtro modelo expandido en *M*, cada retraso debe ser sustituido por *M* retrasos, con *M* = 3.

Una consideración de gran importancia en la implementación de un filtro digital es el cuidado de que no ocurra desbordamiento en la salida. Esto significa que la salida de un filtro debe tener un ancho de bus suficiente para representar en forma adecuada las muestras de salida [11]. Considérese la expresión dada por

$$n = n_i + n_f + n_e + 1. \tag{4.12}$$

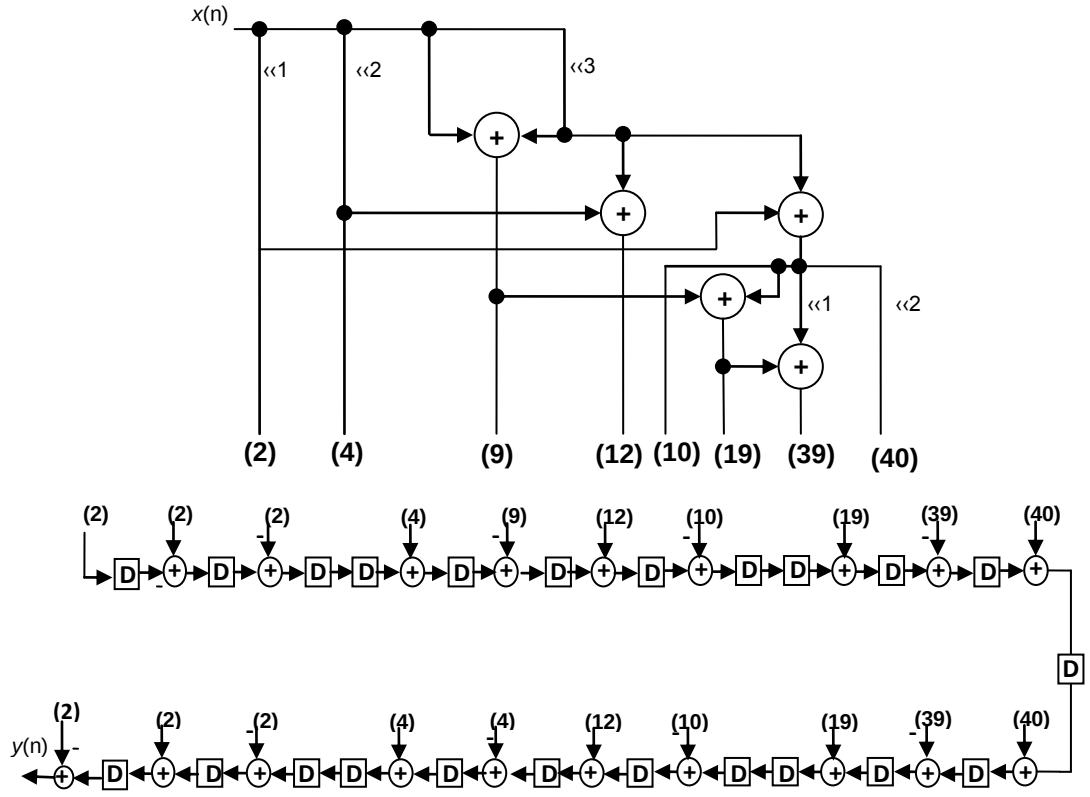


Figura 4.14. Estructura directa transpuesta para el filtro modelo propuesto.

Aquí n representa el número de bits de salida de un filtro FIR, n_i es el número de bits de entrada, n_f es el número de bits utilizados en la parte fraccionaria de los coeficientes y n_e simboliza el número de bits extras incluidos debido al proceso acumulativo de convolución. El número n_e se calcula a partir de la ecuación expresada como

$$n_e = \left\lfloor \log_2 \left(\sum_{k=0}^{K} |h_k| \right) \right\rfloor, \tag{4.13}$$

donde K es el orden del filtro y h_k representa al k-ésimo coeficiente.

Respecto al filtro modelo propuesto, considérense los valores de coeficientes presentados en la tabla 4.4. Para un ancho de palabra de entrada de 8 bits, la cantidad de bits extras necesarios se calcula utilizando la

expresión (4.13). Se obtiene n_e = 3. El ancho de palabra de salida puede ser calculado con la expresión (4.12). Considerando que se han utilizado 5 bits para representar la parte fraccionaria de los coeficientes, el ancho de palabra de salida para el filtro modelo es obtenido como n = 17.

Por otra parte, debe tenerse en cuenta que la ganancia del filtro modelo puede ser controlada directamente por un número representado en potencias de dos [44]. Debido a que los valores de coeficientes de longitud finita pueden ser convertidos a enteros multiplicándolos por $(\alpha)^{-1} = 2^5$, es suficiente considerar solamente coeficientes con valores enteros.

La respuesta al impulso del filtro modelo propuesto obtenida en VHDL es presentada en la figura 4.15.

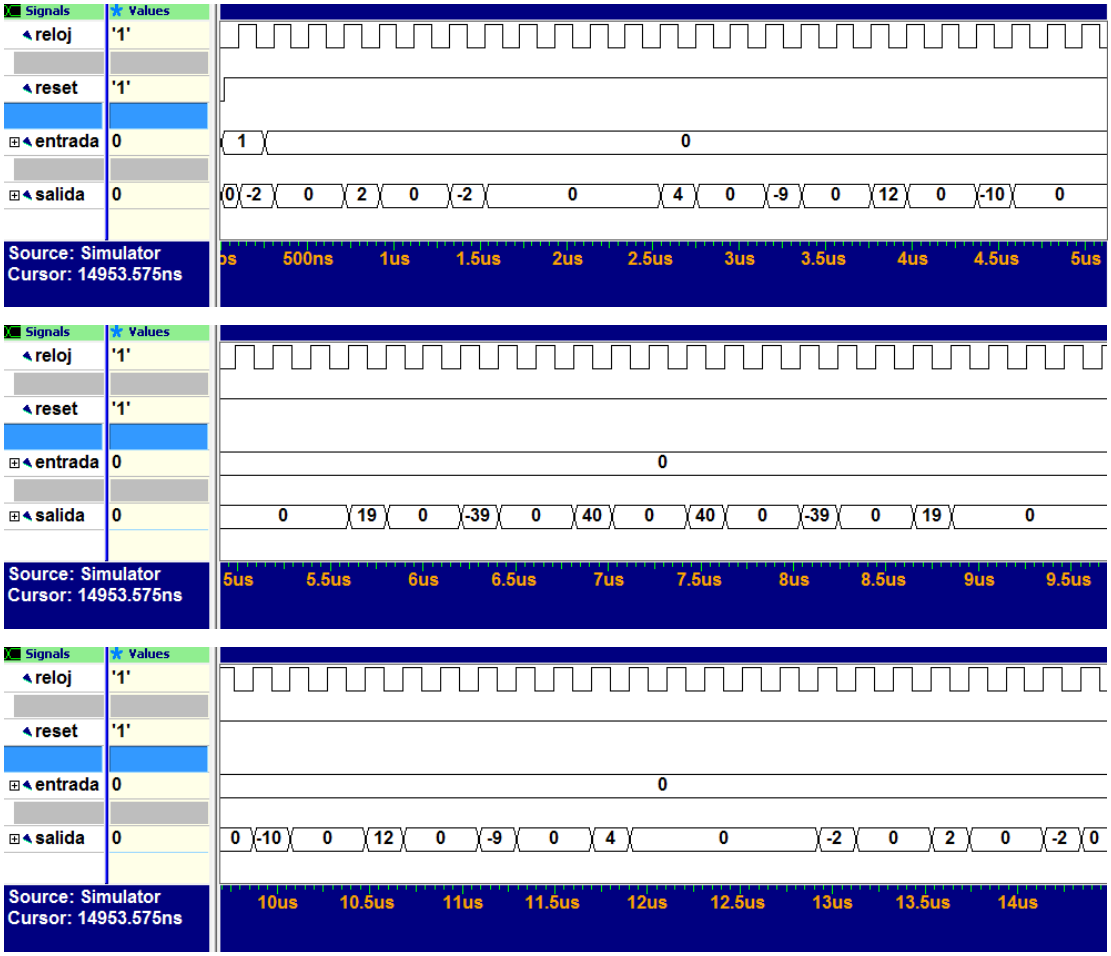


Figura 4.15. Simulación en VHDL de la respuesta al impulso del filtro modelo expandido.

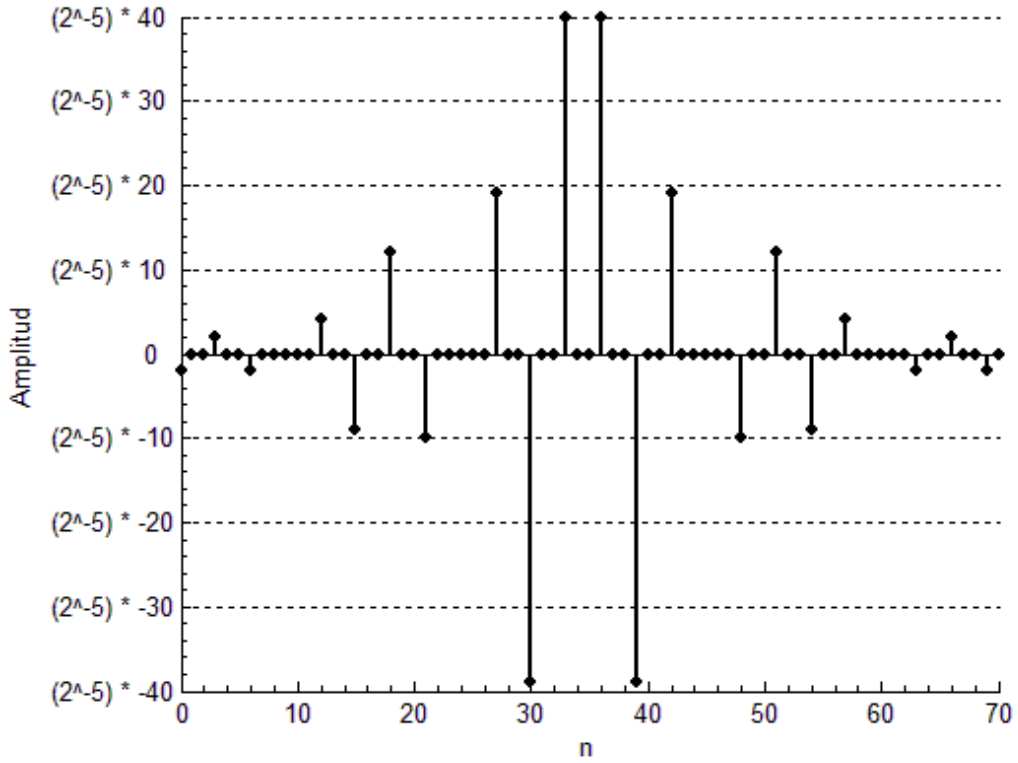


Figura 4.16. Respuesta al impulso del filtro modelo expandido en *M* = 3.

La respuesta al impulso del filtro modelo obtenida en MATLAB se muestra en la figura 4.16. Nótese que la salida simulada en diagrama de tiempo corresponde a la respuesta al impulso calculada en MATLAB. Además, debido a que los valores han sido representados como enteros, la figura 4.16 presenta las muestras de salida como enteros multiplicados por el factor de escalamiento utilizado, $\alpha = 2^{-5}$.

4.5.2 Simulación del filtro interpolador

El filtro interpolador está compuesto por la conexión en cascada de 2 filtros CIC y un filtro Coseno Modificado.

El ancho de palabra requerido en el filtro CIC puede calcularse como [31]

$$b = \log_2(B),$$
$$B = (M)^K, \tag{4.14}$$

donde b representa el número de bits necesarios en los registros del filtro CIC, n_i es el tamaño de palabra de entrada, M es el factor de decimación y K el número de etapas.

Debe distinguirse aquí que el factor M del filtro CIC utilizado en esta propuesta no representa la reducción en la velocidad de muestreo del filtro total. Más bien se trata de M retrasos incluidos en la etapa *Comb*. El diagrama a bloques del filtro CIC propuesto puede observarse en la figura 4.17.

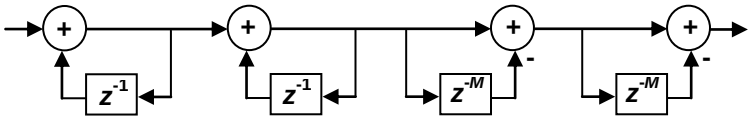


Figura 4.17. Estructura del filtro CIC de dos etapas utilizado en el interpolador.

El filtro CIC es simulado utilizando una entidad basada en un enfoque estructural, de la misma forma en que se realizó la simulación del filtro modelo. Los componentes que forman parte de la entidad son solo registros y sumadores de dos entradas, como puede verse en la figura 4.17. El resultado de la simulación en VHDL del filtro CIC se muestra en la figura 4.18.

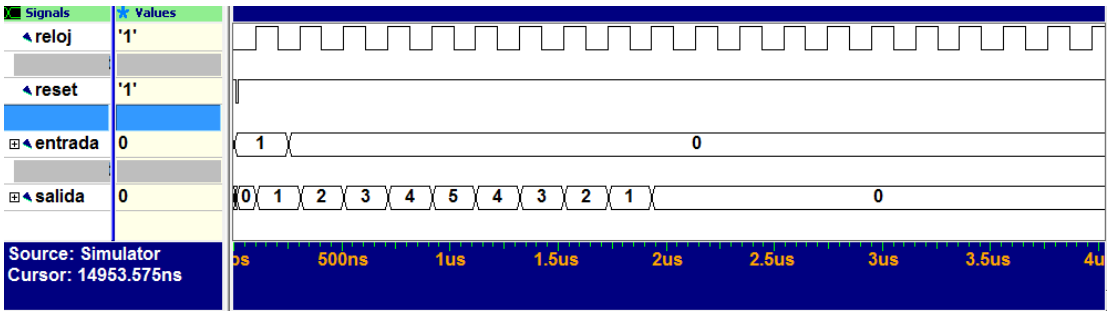


Figura 4.18. Simulación en VHDL de la respuesta al impulso del filtro CIC de dos etapas.

La respuesta al impulso de cada filtro CIC es equivalente a la de un filtro con *M* coeficientes unitarios, escalados por un factor $(M)^{-1}$. En el filtro CIC de dos etapas propuesto aquí la respuesta al impulso correspondiente puede obtenerse en MATLAB como la convolución entre dos filtros con *M* coeficientes unitarios cada uno, siendo *M* = 5. La figura 4.19 presenta la respuesta al impulso del filtro CIC de dos etapas obtenida en MATLAB. Obsérvese que el resultado de la entidad simulada en VHDL corresponde exactamente con la respuesta calculada en MATLAB. Debido a que el factor de escalamiento puede ser controlado en forma externa, como en el filtro modelo explicado antes, las muestras de salida se presentan solo con valores enteros.

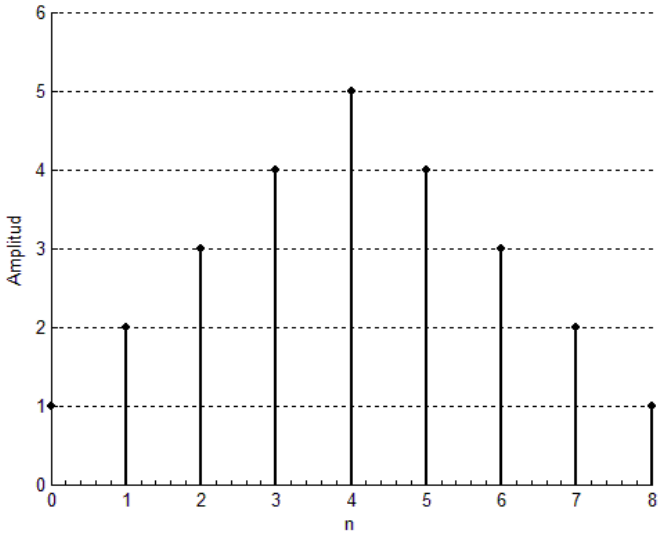


Figura 4.19. Respuesta al impulso del filtro CIC de dos etapas con *M* = 5.

Por otra parte, el filtro Coseno Modificado utiliza tres sumadores, como se muestra en la figura 4.20. Cada sumador requiere un bit adicional al número de bits más alto entre sus dos operandos. Esto significa que la salida del filtro Coseno Modificado debe tener 3 bits más que la entrada. El filtro utilizado en esta propuesta tiene valor *N* = 2 y es creado como los anteriores, siguiendo el enfoque estructural.

La figura 4.21 muestra la respuesta al impulso obtenida de la simulación usando VHDL.

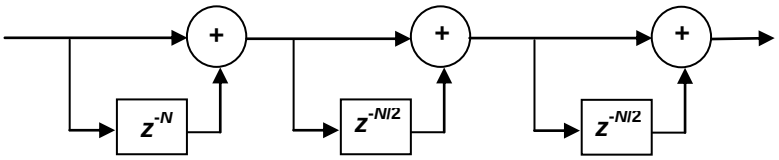


Figura 4.20. Estructura del filtro Coseno Modificado.

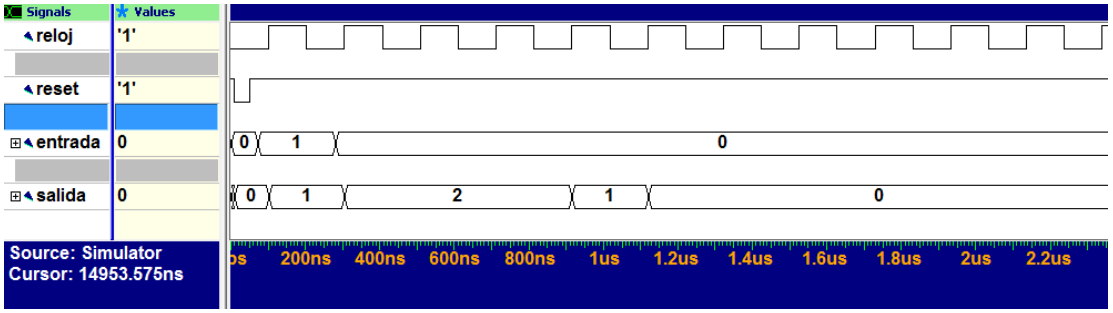


Figura 4.21. Simulación en VHDL de la respuesta al impulso del filtro Coseno Modificado.

La función de transferencia del filtro Coseno Modificado puede ser expresada como

$$H_{Cos\_M}(z) = (1+z^{-N})(1+z^{-N/2})(1+z^{-N/2}) = 1+2z^{-N/2}+2z^{-N}+2z^{-3N/2}+z^{-2N}. \quad (4.15)$$

De la ecuación (4.15) es posible ver que, para N = 2, la respuesta al impulso es equivalente a un filtro con coeficientes [1 2 2 2 1]. Esto comprueba que el resultado obtenido en la simulación de la figura 4.21 es correcto.

El filtro interpolador resulta de la conexión en cascada del filtro CIC de dos etapas con el filtro Coseno Modificado. Así, la entidad correspondiente al filtro interpolador incluye como componentes tanto al filtro CIC como al Coseno Modificado. Aplicando la ecuación (4.14) y añadiendo los 3 bits del filtro Coseno Modificado, se obtienen 6 bits adicionales en el puerto de salida.

La respuesta al impulso del filtro interpolador obtenida en VHDL es presentada en la figura 4.22. De nueva cuenta los filtros son considerados con valores enteros.

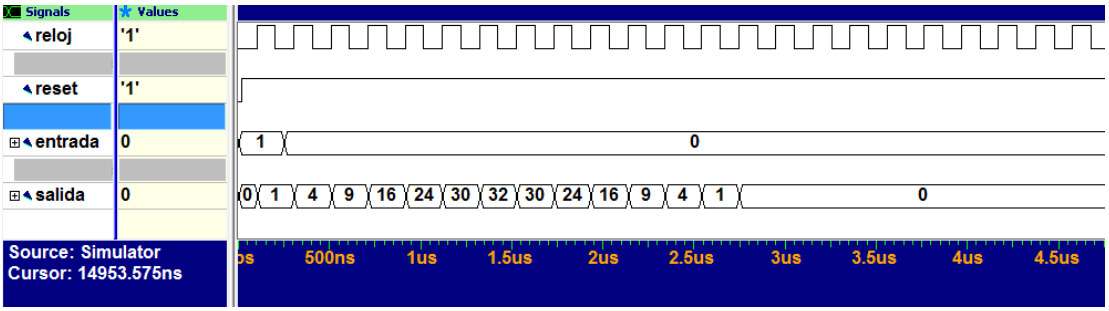


Figura 4.22. Simulación en VHDL de la respuesta al impulso del filtro interpolador.

4.5.3 Simulación del filtro total

El filtro final propuesto resulta de la conexión en cascada del filtro modelo expandido y el filtro interpolador. El enfoque estructural ha sido utilizado una vez más para llevar a cabo la simulación en VHDL. La entidad contiene como componentes al filtro modelo y al filtro interpolador. La entrada del filtro final es de 8 bits y la salida tiene un ancho de bus de 23 bits, considerando 17 bits en la entrada del filtro interpolador y con 6 bits añadidos a la salida de éste. La figura 4.23 presenta el resultado de la simulación en VHDL de la respuesta al impulso del filtro propuesto.

En MATLAB el resultado anterior puede comprobarse llevando a cabo la convolución entre la respuesta al impulso del filtro modelo expandido y la del filtro interpolador. Esto es presentado en la figura 4.24. Por último, la figura 4.25 muestra la respuesta en frecuencia correspondiente al filtro completo. Obsérvese que ésta es exactamente igual a la que fue presentada en la figura 4.9, con la única diferencia en el valor de ganancia obtenido al utilizar coeficientes con valores enteros en la simulación.

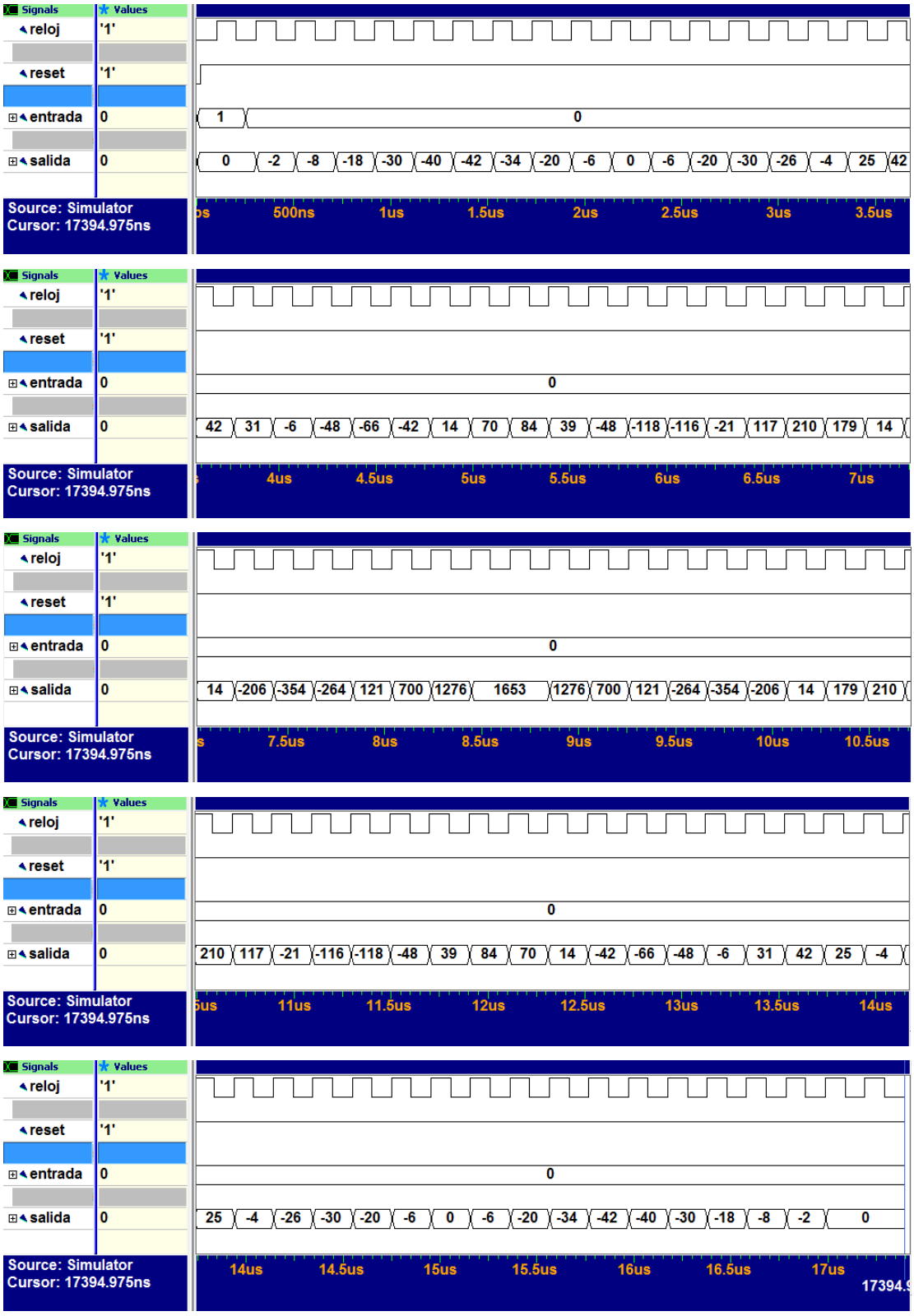


Figura 4.23. Simulación en VHDL de la respuesta al impulso del filtro propuesto.

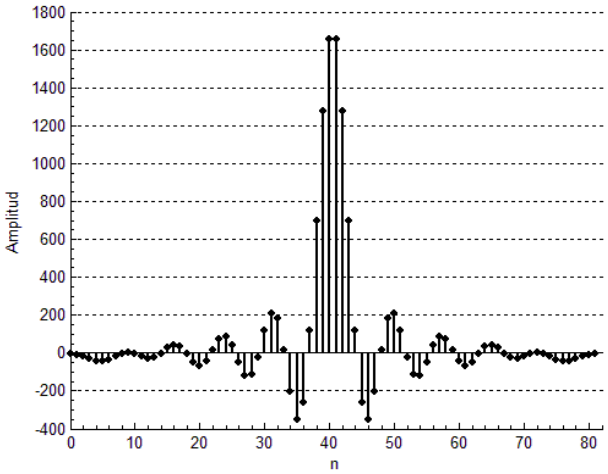


Figura 4.24. Respuesta al impulso del filtro propuesto.

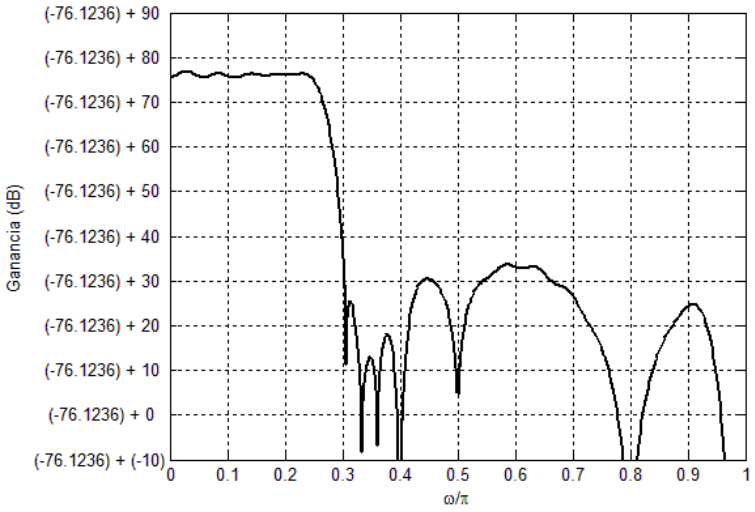


Figura 4.25. Respuesta en frecuencia del filtro propuesto, con ganancia de valores enteros.

El resultado obtenido de la simulación en VHDL es el mismo que aquél calculado en MATLAB y presentado en la figura 4.24. Obsérvese que en esta figura las muestras de salida se presentan con valores enteros en amplitud. La ganancia requerida para normalizar la amplitud de las muestras de salida entre 0 y 1 (o la respuesta en frecuencia a 0 dB) debe ser de 0.15625×10^{-4}; esto equivale a -76.1236 dB. En base a la figura 4.25 es posible comprobar que el filtro realizado en VHDL cumple con las características en frecuencia exigidas en los estándares IS-95 y CDMA 2000.

Capítulo 5

Conclusiones y trabajo futuro

Y di mi corazón a conocer la sabiduría, y también a entender las locuras y los desvaríos: conocí que aun esto era aflicción de espíritu. *Ecclesiastés* 1:17

5.1 Conclusiones

Una parte necesaria dentro de los sistemas de radio móvil celular es la cancelación de la Interferencia Inter-Símbolo (ISI) en banda base. En los estándares CDMA IS-95 y CDMA 2000 el problema de la ISI se resuelve con un filtro digital pasa bajas de fase lineal. El requisito de fase lineal solo puede satisfacerse con filtros FIR. Este filtro tiene banda de paso entre 0 y 590 kHz y banda de rechazo desde 740 kHz en adelante, con 4.9152 MHz de velocidad de muestreo.

La complejidad de un filtro FIR puede estimarse en términos de la cantidad de coeficientes que utiliza. En una implementación en estructura directa, el número de sumadores utilizados en la línea de retrasos es directamente proporcional al número de coeficientes. En una implementación multiplexada en el tiempo el tamaño de la memoria ROM utilizada y la velocidad interna con la que son sincronizados los registros son directamente proporcionales a la cantidad de coeficientes. De cualquier forma, una mayor cantidad de coeficientes implica un aumento en el uso de recursos de hardware y, como consecuencia, mayor consumo de potencia.

Pese a que cada coeficiente representa un multiplicador, muchas investigaciones se han realizado para evitar el uso de multiplicadores y representar a los coeficientes únicamente con sumadores. Además se han desarrollado métodos eficientes para reducir la cantidad de sumadores requerida originalmente por los coeficientes. Así, la finalidad en un diseño eficiente es realizar filtros que satisfagan las especificaciones dadas utilizando la menor cantidad de sumadores.

En esta tesis se ha propuesto un filtro que satisface las especificaciones para sistemas CDMA IS-95 y CDMA 2000 utilizando 32 sumadores. EL filtro propuesto fue diseñado en base a la estructura IFIR. El filtro interpolador se realizó con la conexión en cascada de filtros simples que requieren solo 7 sumadores en total. Esto proporcionó la ventaja de tener menor complejidad y la factibilidad de usar un filtro modelo con factor de interpolación más alto. No obstante, la caída en banda de paso fue la principal desventaja del filtro interpolador. Así, para compensar esta caída, el diseño del filtro modelo se planteó como la solución a un problema de optimización multiobjetivo. Los coeficientes del filtro modelo fueron redondeados y ajustados con un algoritmo sub-óptimo para ahorrar sumadores.

En conclusión, el filtro propuesto en esta tesis no requiere multiplicadores y tiene como principal ventaja el uso de menos sumadores que aquellos realizados previamente. Además, el método para realizar este filtro es sencillo y a la vez efectivo.

5.2 Trabajo futuro

Como trabajo futuro se pretenden realizar los siguientes puntos:

- Llevar a cabo la implementación en FPGA del filtro propuesto con la técnica de multiplexaje en el tiempo, utilizando bloques reconfigurables.

- Desarrollar un nuevo método basado en optimización multiobjetivo para hallar directamente los coeficientes del filtro expresados como suma de potencias de dos. Este método podría disminuir la cantidad de sumadores requeridos.

Apéndice A

Funciones realizadas en MATLAB y programas de VHDL

A.1 Funciones realizadas en MATLAB

[CIC w] = **filtrocic**(*M*, *K*, *P*)

Esta función obtiene *P* puntos de la respuesta en frecuencia de un filtro CIC con factor de decimación *M* y *K* etapas en cascada.

Entradas

- *M*.- Factor de decimación.
- *K*.- Número de etapas.
- *P*.- Número de puntos de frecuencia calculados.

Salidas

- CIC.- Respuesta en frecuencia del filtro CIC.
- w.- Valores de frecuencia entre 0 y π.

[COS w] = **filtrocos**(*N*, *K*, *P*)

Esta función obtiene *P* puntos de la respuesta en frecuencia de un filtro Coseno con factor de interpolación *N* y *K* etapas en cascada.

Entradas

- *N*.- Factor de interpolación.
- *K*.- Número de etapas.
- *P*.- Número de puntos de frecuencia calculados.

Salidas

- COS.- Respuesta en frecuencia del filtro Coseno Modificado.
- w.- Valores de frecuencia entre 0 y π.

[COSMOD w] = **filtrocosmod**(*N*, *K*, *P*)

Esta función obtiene *P* puntos de la respuesta en frecuencia de un filtro Coseno Modificado con factor de interpolación *N* y *K* etapas en cascada.

Entradas

- *N*.- Factor de interpolación.
- *K*.- Número de etapas.
- *P*.- Número de puntos de frecuencia calculados.

Salidas

- COSMOD.- Respuesta en frecuencia del filtro Coseno Modificado.
- w.- Valores de frecuencia entre 0 y π.

[H] = **firtipo2**(*h*, *w*)

Esta función obtiene la amplitud de la respuesta en frecuencia de un filtro FIR tipo 2 (simétrico de longitud par), a partir de sus coeficientes de Fourier.

Entradas

- *h*.- Vector que contiene los coeficientes de Fourier del filtro.

- *w*.- Vector que contiene los valores de frecuencia normalizados a π, entre 0 y 1.

Salidas

- H.- Amplitud de la respuesta en frecuencia del filtro, obtenida en valores normalizados, entre 0 y 1.

[cf H gamma] = **firgoal**(*frel*, *d*, *goal_p*, *goal_s*, *n*, *A*, *b*)

Ésta es una función para diseñar un filtro pasa bajas, simétrico de longitud par, proporcionando los puntos de frecuencia necesarios en la respuesta en frecuencia.

Entradas

- *frel*.- Es un vector que contiene dos elementos. El primero es la frecuencia relativa en la banda de paso, y el segundo es la frecuencia relativa en banda de rechazo.
- *d*.- Es un vector que contiene dos elementos. El primero es el rizo relativo (no en dB) de banda de paso y el segundo es el rizo relativo en banda de rechazo.
- *goal_p*.- Es el vector de valores deseados de respuesta en frecuencia (magnitud) del filtro en banda de paso.
- *goal_s*.- Es el vector de valores deseados de respuesta en frecuencia (magnitud) del filtro en banda de rechazo.
- *n*.- Es el número estimado de orden del filtro.
- *A*.- Es una matriz que contiene los coeficientes para restricciones lineales de desigualdad. Estas restricciones son de la forma $\boldsymbol{Ax} \leq \boldsymbol{b}$.
- *b*.- Es un vector que contiene los resultados para las restricciones lineales de desigualdad.

Salidas

- cf.- Es el vector fila que contiene los coeficientes de Fourier, relacionados en forma lineal con los coeficientes del filtro.
- H.- Es el vector fila que contiene los coeficientes del filtro.
- gamma.- Es el número que se minimizó para realizar la aproximación entre los valores de frecuencia deseados y el resultado final.

[escala pot_CSD bin CSD MSD] = **csd_msd**(*N*, *num*)

Esta función obtiene la representación *CSD* y el conjunto de representaciones *MSD* de cada coeficiente de un filtro.

Entradas

- *N*.- Número que se va a convertir a *CSD*.
- *num*.- Número de bits deseados para la parte fraccionaria.

Salidas

- escala.- Es un vector de dos elementos que contiene la escala en potencia de dos por la que debe ser multiplicada el número resultante. El primer elemento es 2, el segundo es la potencia a la que debe ser elevado el 2.
- pot_CSD.- Es un vector que contiene las potencias de dos correspondientes a cada término *CSD* del resultado.
- bin.- Es un vector que contiene la representación binaria del número deseado.
- CSD.- Es un vector que contiene la representación en *CSD* del número deseado.
- MSD.- Es una matriz que contiene todas las representaciones *MSD* del número deseado.

A.2 Programas realizados en VHDL

modelo1.vhd

Esta entidad representa al filtro Modelo de la estructura propuesta.

Entradas

- *reloj*.- Señal de reloj a 4.9152 MHz.
- *reset*.- Señal de reset activada en bajo.
- *entrada*.- Palabra de entrada de 8 bits.

Salidas

- *salida*.- Palabra de salida de 17 bits.

interpolador1.vhd

Esta entidad representa al filtro Interpolador de la estructura propuesta.

Entradas

- *reloj*.- Señal de reloj a 4.9152 MHz.
- *reset*.- Señal de reset activada en bajo.
- *entrada*.- Palabra de entrada de 17 bits.

Salidas

- *salida*.- Palabra de salida de 23 bits.

filtroprop1.vhd

Esta entidad representa al filtro propuesto. La arquitectura está hecha como la conexión en cascada del filtro modelo y el filtro interpolador.

Entradas

- *reloj*.- Señal de reloj a 4.9152 MHz.
- *reset*.- Señal de reset activada en bajo.
- *entrada*.- Palabra de entrada de 8 bits.

Salidas

- *salida*.- Palabra de salida de 23 bits.

Apéndice B

Artículos publicados

- David Ernesto Troncoso Romero, Gordana Jovanovic Dolecek, “Diseño de filtros digitales de banda base en sistemas CDMA IS-95,” *Memorias del Octavo Encuentro de Investigación*, INAOE, Puebla, México, Noviembre 2007, pp. 113 – 116.

- David Ernesto Troncoso Romero, Gordana Jovanovic Dolecek, “On a multiplierless FIR filter design for IS-95 CDMA systems,” *Mosharaka International Conference on Communications, Propagation and Electronics, MIC-CPE 2008*, Amman, Jordan, March 2008, pp. 1 – 6.

- David Ernesto Troncoso Romero, Gordana Jovanovic Dolecek, “Novel multiplierless FPGA implementation of CDMA 2000 baseband filter,” *IEEE International Symposium on Communication and Information Technologies, ISCIT 2008*, Vientiane, Lao PDR, October 2008, pp. 289 – 294.

- David Ernesto Troncoso Romero, Gordana Jovanovic Dolecek, “Diseño de un filtro de banda base sin multiplicadores para sistemas CDMA IS-95 y CDMA 2000,” *Memorias del Noveno Encuentro de Investigación*, INAOE, Puebla, México, Noviembre 2008, pp. 51 – 54.

Lista de figuras

Capítulo 1

1.1 Sistema CDMA/FDD…….. 3
1.2 Ubicación de los filtros de banda base en el canal directo……..... 7
1.3 Ubicación de los filtros de banda base en el canal inverso…… 7
1.4 Sistema de transmisión de datos binarios en banda base…….. 9
1.5 Estructura prefiltro-ecualizador…… 13

Capítulo 2

2.1 Soluciones óptimas de Pareto para f_1 y f_2…….. 22
2.2 Espacio de funciones objetivo…….. 23
2.3 Mapeo del espacio de parámetros al espacio de funciones objetivo… 24
2.4 Conjunto de soluciones no-inferiores…….. 24
2.5 Respuesta en frecuencia del filtro diseñado en el ejemplo 2.3…….. 37
2.6 Estructura IFIR……..... 38
2.7 Respuestas en frecuencia de: a) Filtro Modelo, b) Filtro Modelo expandido en M y filtro interpolador, c) Cascada del filtro modelo con el interpolador …….. 40
2.8 Respuesta en frecuencia de: a) filtro modelo $G(z)$, b) filtro modelo expandido $G(z^M)$, con $M = 5$…… 41
2.9 Respuesta en frecuencia de: a) filtro interpolador $I(z)$, b) filtro interpolador $I(z)$ y filtro modelo expandido $G(z^M)$, con $M = 5$……. 42
2.10 Respuesta en frecuencia de $H(z)$, resultado de la conexión en cascada de $G(z^M)$ e $I(z)$…….. 42
2.11 Estructuras de un filtro de Hogenauer: a) estructura simple, b) estructura con K etapas simples en cascada……. 45

2.12 Respuesta en magnitud de dos filtros CIC con diferentes valores de M y K……. 46
2.13 Respuesta en frecuencia de filtros Coseno con N = 6 y N = 10…….. 49
2.14 Respuesta en frecuencia de 4 filtros Coseno en cascada, con N_1 = 4, N_1 = 3, N_3 = 2, N_4 =1……. 49
2.15 Estructura de un filtro Coseno Modificado……... 51
2.16 Respuesta en frecuencia de dos filtros Coseno Modificado con diferentes valores de N y K…… 51

Capítulo 3

3.1 a) Respuesta al impulso de $h(n)$ con coeficientes de precisión infinita, b) Respuesta en frecuencia de $H(e^{j\omega})$…….. 56
3.2 a) Respuesta al impulso $g_1(n)$ con coeficientes de precisión $\alpha = 2^{-6}$, b) Respuesta en frecuencia $G_1(e^{j\omega})$…….. 56
3.3 a) $g_2(n)$ con coeficientes de precisión $\alpha = 2^{-10}$, b) $G_2(e^{j\omega})$……..... 56
3.4 Ejemplo de un árbol formado por el método *Branch & Bound*……..... 62
3.5 Algoritmo de Hörner……... 65
3.6 a) Estructura directa transpuesta de un filtro FIR, b) Ejemplo de eliminación de sub-expresiones comunes. Se elimina la sub-expresión “101” de cada coeficiente y se implementa una sola vez 74

Capítulo 4

4.1 Diagrama de flujo correspondiente al procedimiento de diseño…… 92
4.2 Respuesta en frecuencia del filtro interpolador $H_s(e^{j\omega})$…… 94
4.3 Ganancia del filtro interpolador en las frecuencias de paso y de rechazo…… 94
4.4 Ejemplo de la respuesta en frecuencia deseada en la banda de paso del filtro modelo…….. 96
4.5 Respuesta en frecuencia del filtro modelo……. 99

4.6 Respuesta en frecuencia del filtro modelo expandido en M = 3…….. 100
4.7 Estructura del filtro propuesto…… 101
4.8 Respuestas en frecuencia del filtro modelo expandido, filtro interpolador y filtro completo……..... 101
4.9 Respuesta en frecuencia del filtro propuesto…….. 102
4.10 Ampliación en la banda de paso……..... 103
4.11 Filtros de fase lineal con estructura directa…….. 107
4.12 Relación entre un filtro de fase lineal con estructura directa y su código VHDL…….. 110
4.13 Filtros de fase lineal con estructura directa transpuesta……... 111
4.14 Estructura directa transpuesta para el filtro modelo propuesto……. 113
4.15 Simulación en VHDL de la respuesta al impulso del filtro modelo expandido…….. 114
4.16 Respuesta al impulso del filtro modelo expandido en M = 3…….. 115
4.17 Estructura del filtro CIC de dos etapas utilizado en el interpolador 116
4.18 Simulación en VHDL de la respuesta al impulso del filtro CIC de dos etapas……..... 116
4.19 Respuesta al impulso del filtro CIC de dos etapas con M = 5…… 117
4.20 Estructura del filtro Coseno Modificado…… 118
4.21 Simulación en VHDL de la respuesta al impulso del filtro Coseno Modificado…….. 118
4.22 Simulación en VHDL de la respuesta al impulso del filtro interpolador…… 119
4.23 Simulación en VHDL de la respuesta al impulso del filtro propuesto…… 120
4.24 Respuesta al impulso del filtro propuesto…….. 121
4.25 Respuesta en frecuencia del filtro propuesto, con ganancia de valores enteros…….. 121

Lista de tablas

Capítulo 1

1.1 Asignación de números de canal y bandas de frecuencia…….. 4

Capítulo 2

2.1 Respuesta en frecuencia para filtros FIR de fase lineal…….. 20

2.2 Coeficientes obtenidos para el ejemplo 2.3……... 37

2.3 Comparación de un diseño directo y un diseño con estructura IFIR 43

Capítulo 3

3.1 Pirámide de ponderaciones de Hamming, *HWP*…….. 79

3.2 Pirámide de ponderaciones de Hamming basada en Sub-expresiones comunes, *CS-HWP*…….. 79

3.3 Vecindad definida para compensación de error de cuantización….. 82

Capítulo 4

4.1 Comparación de las longitudes de filtros modelo e interpolador para una estructura IFIR clásica, con $M = 2$ y $M = 3$……... 95

4.2 Coeficientes del filtro modelo, con precisión infinita……. 100

4.3 Coeficientes del filtro modelo para el filtro propuesto…….. 103

4.4 Representación de los coeficientes en un bloque multiplicador……. 104

4.5 Características del filtro propuesto…….. 104

4.6 Comparación entre el filtro propuesto y los filtros existentes……. 106

Referencias

[1] V. K. Garg and J. E. Wilkes, *Wireless and Personal Communication Systems*, Prentice Hall, USA, 1996.

[2] L. W. Couch, *Modern Communication Systems: Principles and Applications*, Prentice Hall, Englewood Cliffs, New Jersey 07632, 1995.

[3] R. Steele, C. C. Lee and P. Gould, *GSM, CdmaOne and 3G Systems*, John Wiley & Sons, England, 2001.

[4] V. K. Garg, *IS-95 CDMA and CDMA 2000: Cellular/PCS Systems Implementation*, Prentice Hall, USA, 2000.

[5] M. R. Karim and M. Sarraf, *W-CDMA and CDMA 2000 for 3G Mobile Networks*, McGraw Hill, USA, 2002.

[6] S. C. Yang, *3G CDMA 2000 Wireless System Engineering*, Artech House, USA, 2004.

[7] S. Haykin, *Communication Systems*, John Wiley & Sons, 4th Edition, USA, 2001.

[8] J. G. Proakis, *Digital Communications*, McGraw Hill, 3rd Edition, Singapore, 1995.

[9] F. J. Harris, *Multirate signal processing for communication systems*, Prentice Hall, First Edition, USA, 2004.

[10] F. Wang, J. J. Chen and G. Agami, “Optimal receiver filter design with applications in IS-2000 CDMA systems,” *IEEE Wireless Communications and Networking*, vol. 1, March 2003, pp. 496 – 501.

[11] G. L. Do and K. Feher, “Efficient filter design for IS-95 CDMA systems,” *IEEE Trans. on Consumer Electronics*, vol. 42, No. 4, Nov. 1996, pp. 1011 – 1020.

[12] Y. Lian and M. Poh, “FPGA implementation of IS-95 CDMA baseband filter,” *Proc. of the 4th International Conference on ASIC*, Oct. 2001, pp. 411 – 415.

[13] Y. Lian and J. Yu, “VLSI implementation of multiplier-free low power baseband filter for CDMA systems,” *2003 IEEE Workshop on Signal Processing Systems*, Aug. 2003, pp. 111 – 115.

[14] Y. Lian and J. Yu, “The synthesis of low power baseband filters for CDMA,” *IEEE 6th CAS Symposium on Emerging Technologies: Mobile Wireless Communications*, Shanghai, China, May 31 – June 2, 2004, pp. 599 – 602.

[15] Y. C. Lim, “Design of discrete-coefficient-value linear phase FIR filters with optimum normalized peak ripple magnitude,” *IEEE Trans. on Circuits and Systems- I, Reg. papers*, vol. 37, No. 12, Dec. 1990, pp.1480 – 1486.

[16] Y.C.Lim, J. H. Lee, C. K. Chen and R. H. Yang “A weighted least squares algorithm for quasi-equiripple FIR and IIR digital filter design,” *IEEE Trans. on Signal Processing*, vol. 40, March 1992, pp. 551 – 558.

[17] V. K. Ingle and J. G. Proakis, *Digital Signal Processing Using MATLAB*, Brooks/Cole, USA, 2000.

[18] J. G. Proakis and D. G. Manolakis, *Digital Signal Processing: Principles, Algorithms and Applications*, Pearson Prentice Hall, USA, 2005.

[19] J. F. Kaiser, “Nonrecursive digital filter design using the I_0-sinh window function,” *IEEE Proceedings of the International Symposium on Circuits and Systems 1974*, 1974, pp. 20 – 23.

[20] S. K. Mishra and G. Giorgi, *Invexity and Optimization*, Springer, USA, 2008.

[21] S. S. Rao, *Engineering Optimization: Theory and Practice*, John Wiley & Sons, USA, 1996.

[22] A. Abraham, L. C. Jain and R. Goldberg, *Evolutionary Multiobjective Optimization: Theoretical Advances and Applications*, Springer, 2005.

[23] *Optimization Toolbox 3*, Natick, MA, The MathWorks Inc., 2007.

[24] Y. Neuvo, Y. C. Dong and S. K. Mitra, “Interpolated finite impulse response filters,” *IEEE Transactions on Acoustic Speech and Signal Processing*, vol. ASSP-32, June 1984, pp. 563 – 570.

[25] E. B. Hogenauer, “An economical class of digital filters for decimation and interpolation,” *IEEE Transactions on Acoustic Speech and Signal Processing*, vol. 29, No. 2, April 1981, pp. 155 – 162.

[26] S. C. Chan and K. S. Yeung, “On the design and multiplier-less realization of digital IF for software radio receivers with prescribed output accuracy,” *14th International Conference on Digital Signal Processing*, 2002, Hellas, Greece, vol. 1, July 2002, pp. 277 – 280.

[27] F. J. T. Torres and G. J. Dolecek, “Compensated CIC-cosine decimation filter,” *International Symposium on Communication and Information Technologies,* 2007, Sydney, NSW, Oct. 2007, pp. 256 – 259.

[28] J. F. Kaiser and R. W. Hamming, “Sharpening the response of a symmetric nonrecursive filter,” *IEEE Trans. on Acoustics, Speech and Signal Processing*, vol. ASSP-25, no. 5, Oct. 1977, pp. 415 – 422.

[29] Y. L. Tai and T. P. Lin, “Design of multiplierless FIR filter by multiple use of the same filter,” *Electronics Letters*, vol. 28, January 1992, pp. 122 – 123.

[30] Y. Lian and C. Yang, “A new structure for design narrowband lowpass FIR filters,” *Proceedings of IEEE Region 10 International Conference on Electrical and Electronic Technology, 2001*. *TENCON*, vol. 1, August 2001, pp. 274 – 277.

[31] U. M. Baese, *Digital Signal Processing with Field Programmable Gate Arrays*, Springer, Germany, 2004.

[32] A. Bartolo, B. D. Clymer, R. C. Burgess and J. P. Turnbull, “An efficient method of FIR filtering based on impulse response rounding,” *IEEE Transactions on Signal Processing*, vol. 46, No. 8, August 1998, pp. 2243 – 2248.

[33] Y. C. Lim, S. R. Parker and A. G. Constantinides, “Finite wordlength FIR filter design using integer programming over a discrete coefficient space,” *IEEE Trans. Acoust., Speech, Signal Processing,* vol. ASSP-30, Aug. 1982, pp. 661 – 664.

[34] Y. C. Lim and S. R. Parker, “FIR filter design over a discrete powers of two coefficient space,” *IEEE Trans. Acoust., Speech, Signal Processing,* vol. ASSP-31, No. 3, Jun. 1983, pp. 583 – 591.

[35] S. Brown and Z. Vranesic, *Fundamentals of Digital Logic with VHDL design*, Mc Graw Hill, USA, 2005.

[36] K. Chang, *Digital Systems Design with VHDL and Synthesis: An Integrated Approach.* USA, IEEE Computer Society, 1999.

[37] J. P. Deschamps, G. J. A. Bioul and G. D. Sutter, *Synthesis of Arithmetic Circuits: FPGA, ASIC and Embedded Systems*, John Wiley & Sons, USA, 2006.

[38] W. J. Oh and Y. H. Lee, “Implementation of programmable multiplierless FIR filters with powers-of-two coefficients,” *IEEE Transactions on Circuits and Systems –II: Analog and Digital Signal Processing*, vol. 42, issue 8, Aug. 1995, pp. 553 – 556.

[39] I. C. Park and H. J. Kang, “Digital filter synthesis based on minimal signed digit representation,” *Design Automation Conference, 2001.Proceedings*, 2001, pp. 468 – 473.

[40] I. C. Park and H. J. Kang, “Digital filter synthesis based on an algorithm to generate all minimal signed digit representations,” *IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems*, vol. 21, No. 12, Dec. 2002, pp. 1525 – 1529.

[41] M. D. Macleod and A. G. Dempster, “Multiplierless FIR filter design algorithms,” *IEEE Signal Processing Letters*, vol. 12, No. 3, March 2005, pp. 186 – 189.

[42] F. Xu, C. H. Chang and C. C. Jong, “Design of low-complexity FIR filters based on signed-powers-of-two coefficients with reusable common

subexpressions," *IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems*, vol. 26, No. 10, Oct. 2007, pp. 1898 – 1907.

[43] F. Xu, C. H. Chang and C. C. Jong, "Hamming weight pyramid –A new insight into canonical signal digit representation and its applications," *J. Comput. Elect. Eng.*, vol. 33, No. 3, May 2007, pp. 195 – 207.

[44] Y. J. Yu, Y. C. Lim, "Design of linear phase FIR filters in subexpression space using Mixed Integer Linear Programming," *IEEE Trans. on Circuits and Systems-I, Reg. Papers*, vol. 54, No. 10, Oct. 2007, pp. 2330 – 2338.