UNIVERSIDAD CENTRAL "MARTA ABREU" DE LAS VILLAS

VERITATE SOLA NOBIS IMPONETUR VIRILIS TOGA. 1948

**Facultad de Ingeniería Eléctrica**

**Centro de Estudios de Electrónica y Tecnologías de la Información**

# TRABAJO DE DIPLOMA

## Implementación de un Banco de Filtros Coseno Modulado en aritmética de punto fijo.

**Autor: Loraine Fernández Carrillo.**

**Tutor: DrC. Julián Cárdenas Barrera.**

**Santa Clara**

**2011**

**"Año 53 de la Revolución"**

# Universidad Central “Marta Abreu” de Las Villas

## Facultad de Ingeniería Eléctrica

### Departamento de Automática y Sistemas Computacionales



# TRABAJO DE DIPLOMA

## Implementación de un Banco de Filtros Coseno Modulado en aritmética de punto fijo.


**Autor: Loraine Fernández Carrillo.**

**e-mail:** carrillo@uclv.edu.cu

**Tutor: DrC. Julián Cárdenas Barrera.**

**e-mail:** julian@uclv.edu.cu



**Santa Clara**

**2011**

**"Año 53 de la Revolución"**

Hago constar que el presente trabajo de diploma fue realizado en la Universidad Central "Marta Abreu" de Las Villas como parte de la culminación de estudios de la especialidad de Ingeniería en Automática, autorizando a que el mismo sea utilizado por la Institución, para los fines que estime conveniente, tanto de forma parcial como total y que además no podrá ser presentado en eventos, ni publicados sin autorización de la Universidad.

Firma del Autor

Los abajo firmantes certificamos que el presente trabajo ha sido realizado según acuerdo de la dirección de nuestro centro y el mismo cumple con los requisitos que debe tener un trabajo de esta envergadura referido a la temática señalada.

Firma del Tutor

Firma del Jefe de Departamento
donde se defiende el trabajo

Firma del Responsable de
Información Científico-Técnica

# PENSAMIENTO

***"Solo hay un bien: el conocimiento.***

***Solo hay un mal: la ignorancia."***

***Descartes.***

# DEDICATORIA

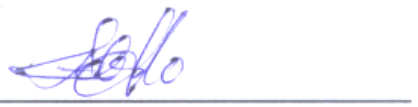
Dedicada a todos los que hicieron que pudiera cumplir en tiempo y forma esta importante tarea y de manera muy especial a mi mamá y toda mi familia.

# AGRADECIMIENTOS


- A toda mi familia en especial a mi mamá por el apoyo y la confianza que ha depositado en mí toda la vida.
- A mi novio Jandry por la ayuda, la entrega, la comprensión y el amor que me ha proporcionado todos los momentos difíciles.
- A mis suegros y muy especialmente a mi suegra por su constante apoyo, aliento y preocupación.
- A mi tutor ya que sin su ayuda y su paciencia no hubiera podido cumplir esta importante tarea.
- A mis abuelos maternos a los cuales debo haber llegado hasta donde estoy.

# TAREA TÉCNICA

- Estudio de bancos de filtro coseno modulado y sus implementaciones rápidas.
- Estudio de la representación en punto fijo de algoritmos y restricciones de dicha implementación con respecto a las implementaciones en punto flotante.
- Estudio del toolbox de Matlab de punto fijo y su funcionalidad para posterior implementación de algoritmos.
- Implementación de los procesos de filtrado en punto fijo.
- Comparación con las implementaciones punto flotante.
- Confección del informe final.



Firma del Autor

Firma del Tutor

# RESUMEN

En las investigaciones del CEETI se necesita el empleo de bancos de filtros con implementaciones eficientes para aplicaciones como implantes cocleares, prótesis auditivas, reconocimiento de locutor, etc. Por esta razón el uso de algoritmos eficientes de bancos de filtros conduce a mejores desempeños. El hecho de implementar en punto fijo contribuye a su realización en hardware de bajo costo computacional y bajo consumo de potencia que permita la implementación práctica de estos complejos procedimientos de tratamiento de señales.

Hasta el momento existen algoritmos de implementación de filtros que desde el punto de vista computacional son eficientes y a los cuales no se les ha realizado la implementación en punto fijo, ni el análisis de los errores que acarrea dicha implementación, lo cual es objeto de estudio en el mundo y constituye el punto de inicio de este proyecto.

El objetivo general del trabajo es realizar la implementación en punto fijo de un banco de filtros coseno modulado empleando el toolbox de punto fijo de Matlab. De manera específica se desea analizar el algoritmo rápido para la realización de un banco de filtros coseno modulado tipo FIR de reconstrucción aproximada, realizar la implementación del banco de filtros mencionado empleando aritmética de punto fijo y finalmente comparar el desempeño del algoritmo teniendo en cuenta las implementaciones en punto flotante y punto fijo.

La implementación del Banco de Filtros Coseno Modulado se llevará a cabo mediante un algoritmo rápido que reduce de manera considerable el costo computacional minimizando el número de operaciones requeridas. Este proceso es optimizado empleando aritmética de punto fijo, que además de simplificar considerablemente las operaciones reduce de manera favorable el consumo de potencia gracias a que requiere menor tiempo de procesamiento.

# TABLA DE CONTENIDOS

# INTRODUCCIÓN

El procesamiento digital de señales empleando bancos de filtros tiene diversas ventajas en aplicaciones como compresión de audio, imagen, video, sistemas de comunicación, etc. Para ello los Bancos de Filtros Coseno Modulados (CMFBs) son los más utilizados y muy especialmente los de Reconstrucción Aproximada (N-PR) con Respuesta al Impulso Finita (FIR). Estos sistemas, aunque no satisfacen la propiedad de Reconstrucción Perfecta (PR), tienen ventajas como:

1. Diseño de bancos de filtro con buena selectividad y discriminación.
2. El proceso de diseño se concentra en optimizar solo el filtro prototipo.
3. Existe la posibilidad – para algunos sistemas – de utilizar algoritmos rápidos para la implementación eficiente de las etapas de análisis y síntesis.

Hasta hace algún tiempo solo se habían mostrado resultados sobre la implementación de algoritmos rápidos para N-PR de bancos de filtro coseno modulado convencionales (C-CMFBs) obtenidos de un filtro prototipo de fase lineal con una longitud restringida lo cual acarreaba ciertas desventajas [20]. Ya en la actualidad se ha propuesto un algoritmo que permite la implementación de filtros FIR de orden elevado el cual se ha demostrado, contribuye a una importante reducción del costo computacional, pues reduce considerablemente el número de Multiplicaciones por Muestras de Entrada (MPIS) como de Adiciones por Muestras de Entrada (APIS) [20].

Para optimizar lo descrito anteriormente es factible utilizar aritmética de punto fijo debido a las ventajas que ella ofrece, entre las que se encuentran la disminución del costo computacional entre otras, lo que contribuye a la realización del proceso de tratamiento de señales en hardware de bajo costo y bajo consumo de potencia.

Por estas razones el trabajo tiene como objetivo general realizar la implementación en punto fijo de un banco de filtros coseno modulado empleando el toolbox de punto fijo de Matlab. De manera específica se desea analizar el algoritmo rápido para la realización de un banco de filtros coseno modulado tipo FIR de reconstrucción aproximada, realizar la implementación del banco de filtros coseno modulado empleando aritmética de punto fijo y finalmente comparar el desempeño del algoritmo teniendo en cuenta las implementaciones en punto flotante y punto fijo.

En el informe se hace referencia primeramente a aspectos generales de los bancos de filtros tales como teoría y aplicaciones. Después se procede a hacer una descripción del algoritmo rápido de implementación y de los bloques que componen el programa encargado de realizar el banco de filtros con las especificaciones deseadas empleando aritmética de punto fijo. Finalmente se hace un análisis del comportamiento del error en las implementaciones en punto flotante y punto fijo.

# CAPÍTULO 1. Generalidades de los Bancos de Filtros

## 1.1 Bancos de Filtros

Un banco de filtros es una colección de filtros que se dividen en dos grupos: uno que se encarga del análisis y otro que se encarga de la síntesis. La etapa de análisis divide la señal de entrada en subbandas que se pueden procesar por separado. En análisis también está incluido el proceso de submuestreo mientras que la etapa de síntesis se realiza un sobremuestreo [1], [2]. A lo largo de este capítulo se realizará un estudio referente sobre la teoría y aplicaciones de los bancos de filtros.

## 1.2 Tipos de bancos de filtros

Cuando se trata de bancos de filtros es necesario remitirse a mucho antes de los años 90.El tipo más antiguo de bancos de filtros es el codificador de voz [7]. Este contiene un banco de filtros coseno modulado de la forma

$$h_k[n] = h[n]cos\left[\frac{\pi}{M}(n+\alpha)(k+\beta)\right]. \tag{1.1}$$

Las constantes $\alpha$ y $\beta$ provocan variaciones al banco de filtros que pueden conducir a implementaciones eficientes. En el codificador de voz de canal, se calcula el valor absoluto de cada canal de muestreo, seguido por un filtrado paso bajo.

La Transformada de Fourier de tiempo Corto (STFT) puede ser considerada también como un tipo de banco de filtros de DFT uniforme (ver Fig. 1.1). Este fue popular en el área del procesamiento digital del habla en 1970 y cerca de los primero años de los 80.

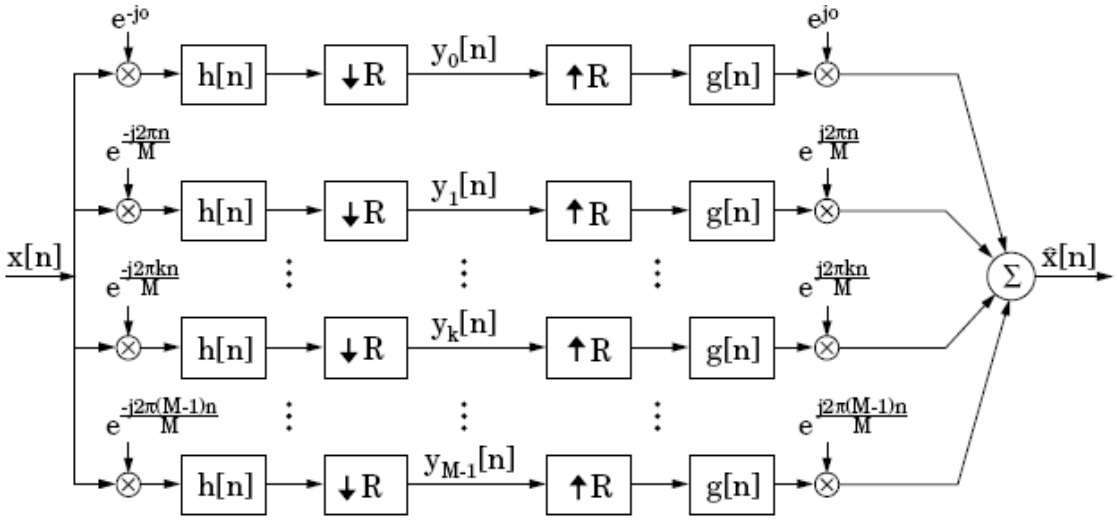



**Figura 1. 1 Sección de análisis-síntesis de un STFT discreto de M canales.**

La STFT usa moduladores exponenciales complejos para cambiar las regiones de alta frecuencia del espectro a una banda base, después de la cual se emplea un filtro pasobajo para aislar la banda cambiada. Debido a los moduladores parecidos al kernel de la DFT, las salidas son valores complejos. La formulación general tiene en cuenta la redundancia variable en la representación. Cuando los parámetros de decimación $R = 1$, la representación es completamente redundante, por otro lado está el caso cuando $R=M$ (donde M representa el número de bandas), en el cual hay una representación mínima del muestreo que puede ser implementada eficientemente utilizando una estructura polifásica concatenada con una FFT [7] .

### 1.2.1 Filtros espejo en cuadratura

En 1976 y 1977 se introducen los filtros espejo en cuadratura (QMFs) y el concepto de cancelación de aliasing. En esta aproximación, la descomposición está basada en un banco de filtros de dos bandas donde $M = 2$ y $R = 2$, como se muestra en la Figura 1.2.

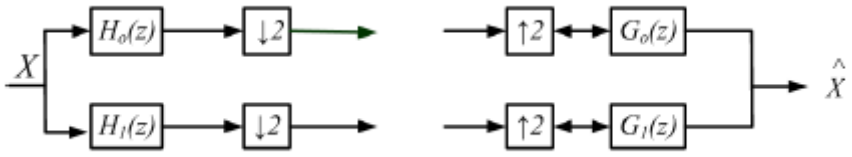



**Figura 1. 2 Análisis-síntesis de un banco de filtros de dos bandas.**

El objetivo del análisis del banco de filtros es descomponer la entrada en subbandas críticamente muestreadas de baja y alta frecuencia. En la reconstrucción las subbandas son interpoladas y unidas tal que la entrada es reconstruida con gran exactitud.

La ecuación de análisis-síntesis puede escribirse en el dominio de $Z$ conduciendo a la ecuación de reconstrucción

$$\hat{X} = \frac{1}{2}X(-z)[H_0(-z)G_0(-z) + H_1(-z)G_1(-z)] + \frac{1}{2}X(z)[H_0(z)G_0(z) + H_1(z)G_1(z)] \tag{1.2}$$

Donde $x[n]$ es la entrada y $\hat{x}\ [n]$ es la salida reconstruida. A esta ecuación están asociados dos componentes: un componente de aliasing

$$\frac{1}{2}[H_0(-z)G_0(z) + H_1(-z)G_1(z)]$$

y un término de la función de transferencia

$$\frac{1}{2}[H_0(z)G_0(z) + H_1(z)G_1(z)]$$

Idealmente el banco de filtros debe tener la propiedad de que los términos de aliasing se reduzcan a cero y los términos de la función de transferencia sean uno [8].

### 1.2.2 Filtros Conjugados en Cuadratura

Otro hito dentro de la historia de los bancos de filtros ocurre en 1984 con la introducción de los filtros conjugados en cuadratura (CQFs), con los cuales se puede obtener una reconstrucción perfecta. La solución de este tipo de banco de filtros está definida como

$$H_1(z) = H_0(-z^{-1}),$$

$$G_0(z) = H_0(z^{-1}),$$

y

$$G_1(z) = -H_0(-z)\,,$$

A diferencia de los QMFs, el análisis y la síntesis de los filtros CQF está desplazado en frecuencia y constituye una versión inversa de $H_0(z)$ [8].

Es fácil de ver que los términos de la cancelación de aliasing se cancelan

$$[H_0(-z)H_0(z^{-1})\text{-}H_0(z^{-1})H_0(-z)] = 0$$

y que la función de transferencia puede ser diseñada tal que

$$[H_0(z)H_0(z^{-1}) - H_0(-z^{-1})H_0(-z)] = 2z^{-l}\,. \tag{1.3}$$

### 1.2.3 Bancos de filtros con estructura de árbol

Las descomposiciones en dos bandas son bastante poco interesantes desde una perspectiva de aplicación. Sin embargo, los bancos de filtros de dos bandas pueden ser colocados en cascada formando árboles que a su vez forman una infinita variedad de bancos de filtros multibanda con resolución en frecuencia variable. En la figura 1.3 se muestra la estructura de un filtro de ocho canales.

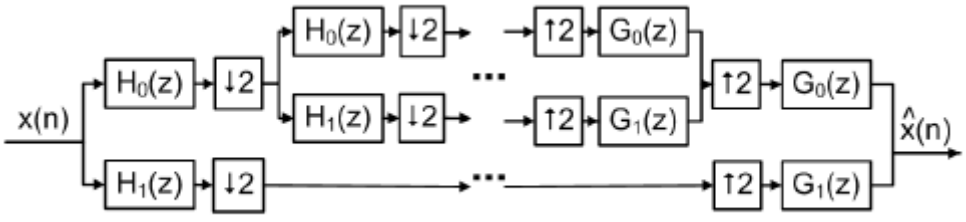



**Figura 1. 3 Estructura de árbol de ocho bandas (también conocida como wavelet diádica con estructura de árbol).**

En esta estructura de árbol simple, la señal original está dividida en dos subbandas. Cada nivel de descomposición está dividido en dos nuevas subbandas de la etapa anterior del

canal de baja frecuencia. El ancho de banda ideal para el análisis está ilustrado en la Figura 1.4 que muestra un gráfico de respuesta de frecuencia.


**Figura 1. 4 División de frecuencia de un filtro de ocho bandas.**

Este proceso de cascadeo de bancos de filtros de dos bandas puede ser repetido de forma apropiada para una aplicación particular. El proceso de reconstrucción incluye una estructura de árbol complementaria de un banco de filtros de síntesis de dos bandas como se muestra en la Figura 1.3 [8].

## 1.3 Implementación de un banco de filtros.

El proceso de implementación de un banco de filtros involucra dos etapas: el análisis de la señal que se desea procesar y la síntesis de la misma luego de haber realizado las modificaciones correspondientes.

Etapa de análisis: Comprende el análisis de M filtros $H_k$ (z), los cuales dividen la señal de entrada en x(n) en M subbandas de señales $x_k$(n). Primero x(n) es filtrada por

h0 [n]; h1 [n];: : : ; hM¡1[n];

lo cual representa las bandas de paso del filtro. En el dominio *Z*, estos filtros están dados por

H0 (z); H1 (z);: : : ; HM¡1(z).

La sección de análisis implementa una convolución descrita por las ecuaciones

$$v_k[n] = h_k[n] * x[n] = \sum_{m=0}^{L-1} x\,[n-m]h_k[m]\,, \qquad k = 0, 1, \cdots, M-1\,, \tag{1.4}$$

$$y_k[n] = v_k[R_n]\,.$$

Los parámetros *M* y los parámetros de decimación *R* ayudan a definir el tipo específico de banco de filtros. En dependencia de la aplicación, los bancos de filtros pueden tener como mínimo dos bandas ( *M* = 2) o un número elevado de bandas como 64, 128, o más. El factor de decimación *R* determina la redundancia inherente en la representación. Cuando *R* = 1, el banco de filtros incluye su grado más alto de sobremuestreo. En muchas aplicaciones, el factor de decimación se hace igual al número de bandas (es decir *R*= *M*), resultando en una representación en memoria críticamente muestreada de manera eficiente [1] [3].

Etapa de síntesis: Utiliza filtros $F_k$ (z) para crear una señal de salida reconstruida xh(n).La salida es recombinada para reconstruir la señal original. En este proceso, las subbandas son sobremuestreadas por un factor *N* y luego filtradas por el filtro de síntesis $F_k$ (z), después del cual los canales se suman juntos.

Dependiendo de la aplicación, algunos tipos de operaciones con señales se realizan a menudo en las subbandas antes de la reconstrucción. Para aplicaciones de compresión esto incluye cuantización y codificación de subbandas, por lo que el ruido de cuantización asociado con la cuantización está preferentemente distribuido en frecuencia, lo cual permite explotar el *roll off* espectral que típicamente acompaña la señal de entrada de interés, así como explotar las propiedades de la percepción humana.

El submuestreo por N se hace justo después del análisis del banco y el sobremuestreo por N se hace antes de la síntesis del banco. La aplicación del procesamiento subbanda se realiza usualmente en el dominio de la frecuencia a una razón más lenta.

Como muestra la Figura 1.5 un banco de filtros es una colección de filtros digitales con una entrada o una salida común [1] [3].

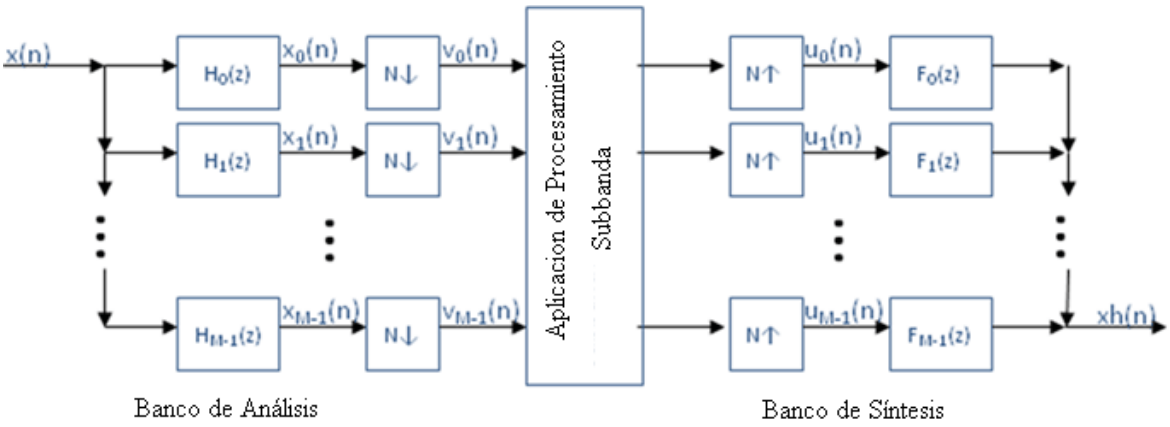



**Figura 1. 5 Estructura básica de un Banco de Filtros de M canales.**

El objetivo en la elección de varios parámetros del banco de filtros es minimizar el error en la señal reconstruida xh(n), mientras se optimiza la realización de la descomposición de la señal. Así el problema de diseño se convierte en cómo diseñar los coeficientes de $H_k$ (z),

$F_k$ (z) y en cómo sobre y submuestrear la razón (es) N.

Todo el proceso descrito anteriormente se puede llevar a cabo mediante estructuras de implementación eficiente como la polifásica [4], de rejilla [5] y de escalera [6].

**Estructura polifásica:** El banco de filtros polifásico es a menudo la forma más eficiente de implementar QMFs (los cuales que serán abordados en el próximo epígrafe) que utilizando una forma directa de implementación. La estructura polifásica es mostrada en la Figura 1.6. Para QMFs el número de multiplicaciones requeridas se reduce en un factor de dos.

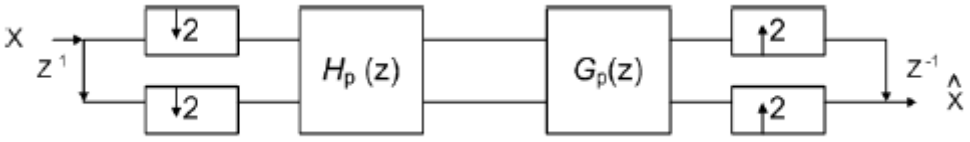



**Figura 1. 6 Estructura polifásica de un banco de filtros.**

**Estructura de rejilla:** Puede ser utilizada en la implementación de CQFs eficientemente. Está formada por las cascadas de una serie de rotaciones y elementos de retardo.

**Estructura de escalera:** Similar a la estructura de rejilla, esta estructura emplea mariposas pero con una sola ala. En la Figura 1.7 se muestra el esquema básico de una estructura de escalera dos bandas.

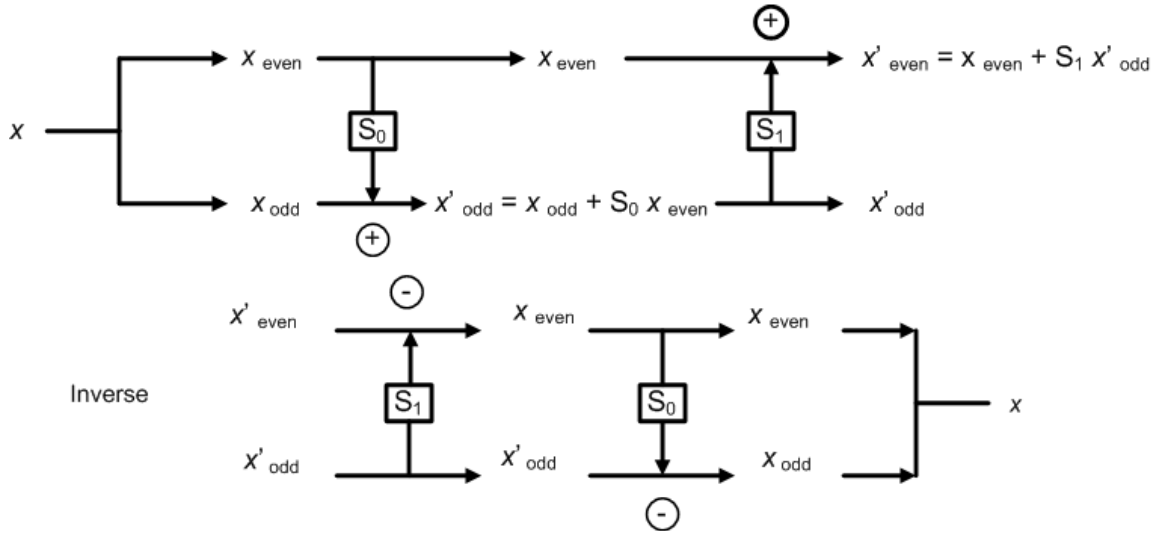



**Figura 1. 7 Estructura de escalera de un banco de filtros.**

## 1.4 Errores Generados en el Banco de Filtros

Partiendo de un entorno ideal en el que no existe la etapa intermedia de procesamiento, en la salida se pueden encontrar los siguientes errores:

- *Distorsión de Amplitud:* Está presente cuando el módulo de la función de transferencia de distorsión total no es constante, con el mismo valor para toda la banda de frecuencia.

- *Distorsión de fase:* Aparece cuando la función de fase de la respuesta de frecuencia de la función de distorsión total no es lineal.

- *Solapamiento:* Se debe al proceso de filtrado y diezmado producido en el banco de análisis en la salida de una determinada subbanda del banco de síntesis pueden aparecer componentes de la señal que pertenecen a otra banda distinta de la considerada.

## 1.5 Aplicaciones de los bancos de filtros.

Los bancos de filtros tienen una rica historia de investigación. Mucha de la atención ha sido motivada por las aplicaciones en la compresión del habla, la imagen, y de video. Sin

embargo los bancos de filtros subbanda han sido empleados también en otras áreas importantes como la telemedicina, detección de objetos y clasificación, eliminación de ruido, conversión del tamaño de la imagen y alteración de la razón de muestreo, y transmisión segura de la señal. En este contexto de aplicación los bancos de filtros se ven típicamente en la forma mostrada en la Figura 1.8, donde la parte izquierda comprende la etapa de análisis y la parte derecha la etapa de síntesis.

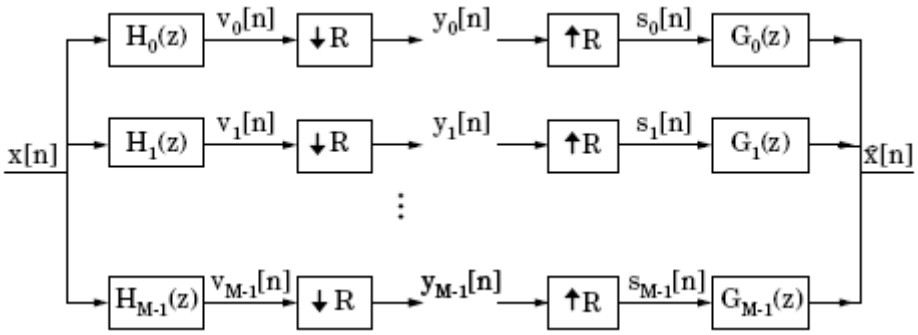



**Figura 1. 8 Diagrama en bloques de un Banco de Filtros 1-D.**

Por otra parte para muchas aplicaciones, la descomposición y el procesamiento en el dominio de la frecuencia pueden aportar beneficios significantes en varias esferas como:

- Selección del canal correcto en comunicaciones inalámbricas.
- Convergencia más rápida y menor complejidad en la ecualización adaptativa.
- Compresión flexible del habla y la música.
- Menor latencia y mayor compensación de frecuencia en enfermedades auditivas.
- Análisis espectral y síntesis más eficiente en corto tiempo.
- Compresión de la imagen con multiresolución y transformaciones Wavelets.
- Reconocimiento automático del habla con resultados confiables.
- Los bancos de filtros se emplean también en ecualizadores gráficos los cuales pueden atenuar las componentes diferentemente y recombinarlas en una versión modificada de la señal original.

Otra aplicación de los bancos de filtros es la compresión de señales, cuando algunas frecuencias son más importantes que otras. Después de la descomposición, las frecuencias importantes pueden ser codificadas con una buena resolución. Las diferencias pequeñas a estas frecuencias son significantes por lo que debe usarse un esquema de codificación que preserve estas diferencias. Por otra parte las frecuencias menos importantes no tienen que ser exactas [1].

También el codificador de voz emplea un banco de filtros para determinar la información de la amplitud de las subbandas de un modulador de señal (como la voz) y la utiliza para controlar la amplitud de las subbandas de la señal portadora (como la salida de una guitarra) así se impondrán las características dinámicas del modulador en la señal portadora.

## 1.6 Aplicaciones biomédicas

Los bancos de filtros como herramientas para el procesamiento digital de señales, son ampliamente utilizados en el campo de la medicina tanto para el diagnóstico de enfermedades como para investigaciones relacionadas con las mismas. Para cada una de estas aplicaciones es necesario tener en cuenta una serie de restricciones, que hacen de los resultados obtenidos luego del empleo del banco de filtros, una fuente confiable para el uso que se le quiera dar, dígase diagnóstico o evaluación del estado del paciente luego de un tratamiento etc.

También pueden encontrarse aplicaciones importantes de los bancos de filtros en las prótesis auditivas implantes cocleares, en el reconocimiento del locutor, así como en el análisis de señales de ECG, EEG, entre otras muy variadas aplicaciones que requieran de un procesamiento de la señal en una banda específica.

# CAPÍTULO 2. Implementación de un Banco de Filtros Coseno Modulado en aritmética de punto fijo.

## 2.1 Descripción de los Bancos de Filtros Coseno Modulados

Los bancos de filtros modulados presentan como ventaja frente al resto de los Bancos de Filtros la simplicidad en las fases de diseño y de realización del banco. Todos los filtros de análisis y de síntesis se obtienen a partir de uno o dos filtros prototipo, a los que se aplica una modulación para desplazar su respuesta en frecuencia y obtener los restantes filtros del banco.

En particular los bancos de filtros coseno modulado son conocidos por proporcionar una realización muy eficiente, pues entre otras ventajas, eliminan las componentes más significativas del solapamiento. Ellos consisten en dos etapas principales: una etapa de filtros polifase del prototipo y otra de transformada discreta de coseno (ver Fig. 2.1 ).

Normalmente los esquemas de modulación en coseno se basan en la transformada discreta del coseno. Se pueden distinguir cuatro tipos distintos de dicha transformada:

DCT I: $$\sqrt{\frac{2}{M}} \cdot k_m \cdot k_n \cdot \cos\left(\frac{m \cdot n \cdot \pi}{M}\right) \qquad m,n = 0,\cdots,M$$

DCT II: $$\sqrt{\frac{2}{M}} \cdot k_m \cdot \cos\left(\frac{m \cdot (n+1/2) \cdot \pi}{M}\right) \qquad m,n = 0,\cdots,M-1$$

DCT III: $$\sqrt{\frac{2}{M}} \cdot k_n \cdot \cos\left(\frac{n \cdot (m+1/2) \cdot \pi}{M}\right) \qquad m,n = 0,\cdots,M-1$$

DCT IV: $$\sqrt{\frac{2}{M}} \cdot \cos\left(\frac{(m+1/2) \cdot (n+1/2) \cdot \pi}{M}\right) \qquad m,n = 0,\cdots,M-1$$

donde:

$$k_j = \begin{cases} 1 & si \quad j \neq 0, j \neq M \\ 1/\sqrt{2} & si \quad j = 0, j = M \end{cases}$$

El índice *m* representa a la variable independiente de la señal transformada, mientras que *n* es la variable tiempo de la señal de entrada. En los bancos de filtros coseno modulado se suelen emplear los esquemas de modulación Tipo II y Tipo IV, debido a que los bancos de filtros que originan cubren todo el intervalo $0 \leq \omega \leq \pi$ y presentan el mismo ancho de banda. Sin embargo, los esquemas basados en la DCT I y DCT III no se utilizan normalmente para la construcción de bancos de filtros porque pueden originar sistemas en los que algunos filtros tienen una banda de paso que es la mitad del resto de las bandas de paso de los filtros.

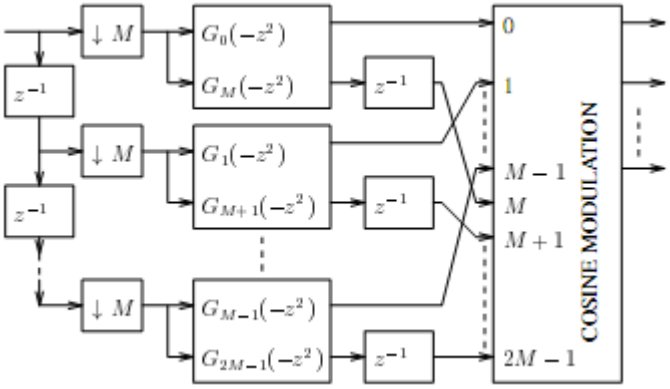



**Figura 2. 1 Etapa de análisis de un banco de filtros coseno modulado de M canales.**

En relación con los tres errores que se pueden presentar en un banco de filtros (Cap. 1), existen:

1 *Bancos de filtros de reconstrucción perfecta (PR)*. Se caracterizan por la ausencia total de distorsión y solapamiento.

2 *Bancos de filtros pseudo-QMF*. Surgen como alternativa a los sistemas de reconstrucción perfecta para soslayar las grandes dificultades que éstos presentan en la fase de diseño. Las dificultades se reducen considerablemente, si se eliminan totalmente alguno de los errores, pero se tolera un valor mínimo, aunque no nulo, en los restantes. Por ejemplo, los bancos de filtros coseno modulado pseudo-QMF

próximos a la reconstrucción perfecta ("Nearly Perfect Reconstruction") no presentan distorsión de amplitud ni de fase y el error de solapamiento es comparable con el valor de la ganancia en la banda atenuada del módulo de la respuesta en frecuencia del filtro prototipo [13].

### 2.1.1 Bancos de Filtros de Reconstrucción Perfecta

Sean $H_k(z)$ la columna k-ésima de la matriz de modulación del banco de análisis H(z), y $F_k(z)$ la columna k-ésima de la matriz de modulación del banco de síntesis F(z) tales que el solapamiento es completamente cancelado y la función de transferencia de distorsión global $T_0(z)$ que representa a un retardo puro en el dominio del tiempo, entonces el sistema es de reconstrucción perfecta (PR) y satisface que $\hat{x}[n] = c \cdot x[n - n_0]$, donde $\hat{x}[n]$ es la señal de salida y $x[n]$ es la señal de entrada . De manera equivalente, el banco es PR si satisface:

$$\text{f(z)} \cdot H(z) = [z^{-n_0} \quad 0 \ \ldots \ 0]^T.$$

Partiendo de la anterior expresión, se pueden derivar distintos procedimientos para encontrar $H_k(z)$ y $F_k(z)$. Las técnicas de diseño más habituales se basan en las matrices $\text{E}(z)$ y $\text{R}(z)$ que representan las componentes polifase de los bancos de análisis y de síntesis respectivamente.

Sea el sistema de fase cero de la Figura 2.2a. Si se cumple que:

$$\text{R}(z) \cdot \text{E}(z) = \text{I} \tag{2.1}$$

entonces la señal de salida $\hat{x}[n]$ coincide con la de entrada $x[n]$. Si se desplazan los sistemas $\text{E}(z)$ y $\text{R}(z)$ empleando las identidades Noble que resultan de gran aplicación en aquellos sistemas que lleven a cabo realizaciones eficientes de filtros y bancos de filtros, se

obtiene el sistema de la Figura 2.2b, que continúa teniendo la propiedad de reconstrucción perfecta [13].

De este modo, si se tiene un conjunto de filtros de análisis $H_k(z)$, con $0 \leq k \leq (M-1)$, la matriz E $(z)$ se encuentra determinada completamente. Asumiendo que E $(z)$ se puede invertir, entonces se puede obtener un sistema PR eligiendo R $(z)$ de modo que:

$$R(z)=E^{-1}(z)$$

Hay que tener en cuenta que la invertibilidad de E $(z)$ no es suficiente, ya que su inversa podría conducir a filtros inestables, e incluso si los filtros de análisis son FIR, se podrían obtener filtros de síntesis IIR [13].

A pesar de que aparentemente se van a encontrar las mismas dificultades (incluyendo la inestabilidad) que aparecieron en la inversión de la matriz H ($z$), se puede evitar invertir directamente E $(z)$ de muchas maneras. Una de ellas consiste en restringirla al caso de que sea paraunitaria [13].

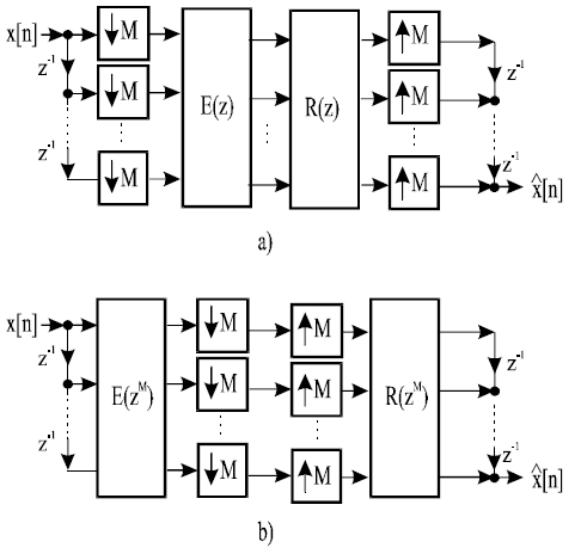



**Figura 2. 2 Banco de Filtros de diezmado máximo de M representado por sus matrices.**

La condición (2.1) es suficiente para reconstrucción perfecta si el sistema es FIR o IIR.

De forma más general se puede demostrar que el sistema tiene reconstrucción perfecta si el producto R $(z)$ por E $(z)$ es de la forma:

$$P(z) = R(z) \cdot E(z) = c \cdot z^{-m_0} \cdot \begin{bmatrix} 0 & I_{M-r} \\ z^{-1} I_r & 0 \end{bmatrix} \tag{2.2}$$

para cualquier entero $r$ con $0 \le r \le (M-1)$, cualquier entero $m_0$ y cualquier constante $c \neq 0$.

Un sistema PR es un sistema libre de solapamiento donde $T_0$(z) representa un retardo en el dominio del tiempo.

Algunos tipos de bancos de filtros de PR son:

1 Banco de filtros de fase lineal coseno-modulado paraunitarios.
2 Banco de filtros coseno-modulado FIR biortogonales.
3 Banco de filtros coseno-modulado IIR biortogonales.
4 Banco de filtros coseno-modulado PR IIR causales y estables.

En general y para una longitud dada de un filtro, el número de coeficientes que hay que encontrar para diseñar el banco de reconstrucción perfecta es directamente proporcional a su número de canales M [13].

### 2.1.2 Bancos de Filtros de Reconstrucción Aproximada

En los bancos de filtros de reconstrucción aproximada no se cumplen las relaciones que anteriormente se plantearon sobre los bancos de filtros de reconstrucción perfecta lo que provoca que exista cierto nivel de distorsión, pero a pesar de ello ofrecen ventajas como:

a) Diseño de bancos de filtro con mejor selectividad y discriminación.
b) El proceso de diseño se concentra solo en optimizar el o los filtros prototipo.
c) Posibilidad, para algunos sistemas, de emplear algoritmos rápidos para la implementación eficiente de las etapas de análisis y síntesis.

Los N-PR FBs se presentan como una alternativa a los de PR debido a que desaparece el proceso de optimización no lineal de diseño del banco, el cual puede resultar bien

complicado, además son sistemas casi libres de solapamiento, ya que se eliminan las componentes más significativas de dicho error y el valor de las restantes componentes disminuye a medida que se aumenta la atenuación en la banda eliminada del filtro prototipo. Además, la función de transferencia de distorsión global $T_0(z)$ tiene fase lineal y un módulo de la respuesta en frecuencia que aproximadamente es de valor constante.

En los N–PR CMFB, las respuestas impulsivas de los filtros de análisis $h_k[n]$ y de síntesis $f_k[n]$ se obtienen de la siguiente forma:

$$h_k[n] = p[n] \cdot c_{1,k}[n], \qquad f_k[n] = p[n] \cdot c_{2,k}[n], \tag{2.3}$$

donde p[n]es el filtro paso bajo prototipo y

$$c_{1,k}[n] = 2 \cdot \cos\left((2k+1)\frac{\pi}{2M}\left(n - \frac{N}{2}\right) + (-1)^k \frac{\pi}{4}\right), \tag{2.4}$$

$$c_{2,k}[n] = 2 \cdot \cos\left((2k+1)\frac{\pi}{2M}\left(n - \frac{N}{2}\right) - (-1)^k \frac{\pi}{4}\right),$$

Si el filtro prototipo de fase lineal se diseña adecuadamente de tal manera que satisfaga las condiciones de reconstrucción aproximada

$$\left|P\left(e^{j\omega}\right)\right| \approx 0 \text{ para } |w| > \frac{\pi}{M}, \tag{2.5}$$

y

$$\left|T_0\left(e^{jw}\right)\right| \approx 1, \qquad \forall\ w, \tag{2.6}$$

todas las componentes de solapamiento se cancelan. Además si el filtro prototipo es de fase lineal, el banco de filtros no introducirá distorsión de fase, siempre que los filtros de síntesis se elijan de acuerdo con

$$f_k[n] = h_k[N-n] \ , \quad \begin{cases} 0 \le n \le N \\ 0 \le k \le M-1 \end{cases} \tag{2.7}$$

Es por eso que, con los N–PR CMFB no se verifica la propiedad PR, pero asumiendo un margen razonable de error para la señal reconstruida, se consigue simplificar el gran esfuerzo de diseño del banco de filtros por el diseño de un prototipo paso bajo.

Para lograr una elevada calidad de la señal reconstruida, este prototipo de ser diseñado de manera que se aproxime todo lo posible a las siguientes condiciones:

$$|P(e^{jw})|^2 + |P(e^{j(w-\pi/M})|^2 = 1 \ , \quad 0 < w < \frac{\pi}{M} \ , \tag{2.8}$$

y

$$\left|P(e^{jw})\right| = 0, \ \ w > \frac{\pi}{M} \ , \tag{2.9}$$

Si se satisface (2.8), se elimina la distorsión de amplitud y si se cumple (2.9), no existirá solapamiento entre bandas adyacentes.

A partir de lo anterior se han propuesto varias alternativas para facilitar el diseño del filtro prototipo. Ejemplo de ello es el algoritmo Parks–McClellan el cual se basa en el diseño del prototipo cumpliendo con las condiciones (2.8) y (2.9). La longitud del filtro y la atenuación deseada en la banda a eliminar se fijan con anterioridad al proceso de optimización, mientras que la pulsación de corte de la banda de paso se ajusta para minimizar la función de coste siguiente:

$$\emptyset = máx_w \left\{ \left|P(e^{jw})\right|^2 + \left|P\left(e^{j(w-\frac{\pi}{M})}\right)\right|^2 - 1 \right\} , \qquad 0 < w < \frac{\pi}{M} \tag{2.10}$$

Recientemente se ha propuesto un método que simplifica la etapa de diseño del filtro prototipo mediante la minimización de la constante:

$$\emptyset = \left| \left| P(e^{j\pi/2M}) \right| - 1/\sqrt{2} \right| \tag{2.11}$$

Pare ello se diseña un filtro prototipo FIR mediante el método de la ventana o el algoritmo de Parks–McClellan, tomando como parámetro a modificar la pulsación de corte de 6 dB o la pulsación de corte de la banda de paso, respectivamente, dando como resultado un filtro prototipo cuya pulsación de corte de 3 dB se sitúa aproximadamente a π/2M. De esta forma se consigue reducir la distorsión de amplitud y el error de solapamiento en el banco de filtros [12].

## 2.2 Algoritmos rápidos de implementación

Como ya se había mencionado anteriormente, la razón por la cual los N-PR CMFB son ampliamente utilizados es la ventaja que ofrecen por poder obtener realizaciones eficientes. El filtro prototipo puede expresarse como:

$$P(z) = \sum_{l=0}^{2M-1} z^{-l} \cdot G_l(z^{2M}) = \sum_{l=0}^{2M-1} z^{-l} \cdot K_l(z^{2M}) \tag{2.12}$$

Donde $G_l(z)$ y $K_l(z)$ son respectivamente las componentes polifase de tipo 1 y 2 de orden $2M$ del filtro $P(z)$ [12].

Empleando las Ecuaciones 2.3 y 2.4 el filtro de análisis que da de la siguiente manera

$$h^T(z) = \begin{bmatrix} H_0(z) \\ H_1(z) \\ \vdots \\ H_{M-1(z)} \end{bmatrix} = \hat{C}_A \cdot \begin{bmatrix} g_0(-z^{2M}) \\ z^{-M} g_1(-z^{2M}) \end{bmatrix} \cdot e(z) \tag{2.13}$$

donde:

$\hat{C}_A$ es la matriz de modulación coseno.

$g_0(z)$ y $g_1(z)$ son las matrices formadas por las componentes polifase.

$e_0(z)$ es el vector de desplazamientos.

El valor de estas matrices es:

$$\left[\hat{C}_A\right]_{k,l} = c_{1,k}[l] \qquad 0 \leq k \leq (2M-1)\ , \tag{2.14}$$

$$g_0(z) = diag[G_0(z)G_1(z)\cdots G_{M-1}(z)]\ , \tag{2.15}$$

$$g_1(z) = diag\,[G_M(z)G_{M+1}(z)\cdots G_{2M-1}(z)]\ , \tag{2.16}$$

y

$$e(z) = diag\left[1 \quad z^{-1} \quad \cdots \quad z^{-(M-1)}\right]^T\ , \tag{2.17}$$

El vector equivalente de los filtros de síntesis se puede expresar:

$$f(z) = [F_0(z)F_1(z)\ \cdots\ F_{M-1}(z)] = z^{-(M-1)}\cdot e^T(z^{-1})\cdot[z^{-M}k_1(-z^{-2M})k_0(-z^{2M})]\cdot \hat{C}_B{}^T \tag{2.18}$$

donde

$$k_0(z) = diag\,[K_{M-1}(z)K_{M-2}(z)\ \cdots\ K_0(z)]\,, \tag{2.19}$$

$$k_0(z) = diag\,[K_{2M-1}(z)K_{2M-2}(z)\ \cdots\ K_M(z)]\,, \tag{2.20}$$

y

$$[\hat{C}_B]_{k,l} = c_{2,k}[2M-1-l] \quad 0 \leq k \leq (M-1)\ ,\ 0 \leq l \leq (2M-1)\,. \tag{2.21}$$

La realización polifase del banco de filtros se muestra en la Fig. 2.3.

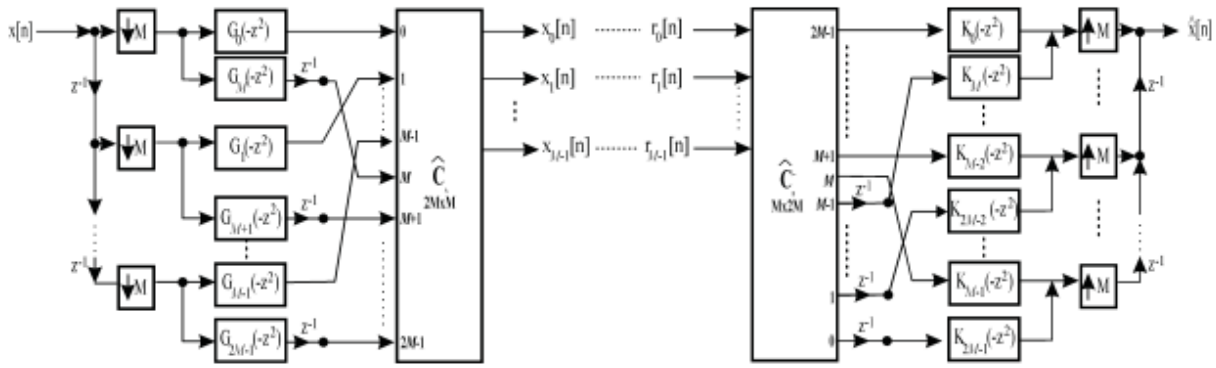


**Figura 2. 3 Implementación polifase del N-PR CMFB**

### 2.2.1 Realizaciones eficientes cuando N + 1 = (2m+ 1) ·M

Con el objetivo de simplificar la Ec. (2.13) se parte de la siguiente expresión para la matriz de modulación coseno:

$$\hat{C}_A = \sqrt{2M} \cdot [\Lambda_C \cdot C + \Lambda_S \cdot \Gamma \cdot C \cdot J \quad \Gamma \cdot \Lambda_S \cdot C - \Lambda_C \cdot C \cdot J] \tag{2.22}$$

Donde $\Lambda_C$ y $\Lambda_S$ son matrices diagonales de dimensión M ×M cuyos elementos son:

$$[\Lambda_C]_{k,k} = \cos\left((0.5+k)\cdot\frac{\pi}{M}\cdot\frac{(N+1)}{2} - (-1)^k\cdot\frac{\pi}{4}\right) \tag{2.23}$$

y

$$[\Lambda_S]_{k,k} = \sin\left((0.5+k)\cdot\frac{\pi}{M}\cdot\frac{(N+1)}{2} - (-1)^k\cdot\frac{\pi}{4}\right) \tag{2.24}$$

y C es la matriz DCT–IV de dimensión M×M definida como:

$$[C]_{k,n} = \sqrt{\frac{2}{M}}\cos\left(\frac{\pi}{M}\cdot(0{,}5+k)\cdot(0{,}5+n)\right). \tag{2.25}$$

Dependiendo de la longitud del filtro prototipo, los valores de $\Lambda_C$ y $\Lambda_S$ se simplifican de la manera que se muestra en la Tabla 2.1.

Cuando $m \in Z+$ es impar, la matriz de modulación coseno correspondiente a la Ec. 2.22 queda

$$\hat{C}_A = \sqrt{2M} \cdot [\Lambda_S \cdot \Gamma \cdot C \cdot J \quad \Gamma \cdot \Lambda_S \cdot C\,]. \tag{2.25}$$

**Tabla 2. 1 Valores de $\Lambda_C$ y $\Lambda_S$ para casos especiales.**

| *Longitud del filtro* | $m \in \mathbb{Z}^+$ | $\mathbf{\Lambda}_C$ | $\mathbf{\Lambda}_S$ |
|---|---|---|---|
| $N+1=(2m+1)\cdot M$ | Impar | $0$ | $\pm 1 \cdot \boldsymbol{I}$ |
| $N+1=(2m+1)\cdot M$ | Par | $\pm 1 \cdot \boldsymbol{I}$ | $0$ |

Como $\Lambda_S$ y $\Gamma$ son ambas matrices diagonal, se puede aplicar la propiedad conmutativa siendo $\Lambda_S \cdot \Gamma = \Gamma \cdot \Lambda_S$ simplificándose así $\hat{C}_A$ :

$$\hat{C}_A = \sqrt{2M} \cdot \Gamma \cdot \Lambda_S \cdot C \cdot [\,J \;\; I\,]. \tag{2.26}$$

Trasladando el valor simplificado de la matriz de modulación coseno a la Ec. 2.13, se llega a la ecuación del banco de análisis:

$$h^T(z) = \sqrt{2M} \cdot \Gamma \cdot \Lambda_S \cdot C \cdot [\,J \;\; I\,] \cdot \begin{matrix} g_0(-z^{2M}) \\ z^{-M} g_1(-z^{2M}) \end{matrix} \cdot e(z) \tag{2.27}$$

de la que se obtiene la correspondiente realización eficiente [12].

El otro caso que resta por contemplar es cuando $m \in Z+$ es par, quedando la matriz de modulación de esta manera:

$$\hat{C}_A = \sqrt{2M} \cdot [\Lambda_C \cdot C \quad -\Lambda_C \cdot C \cdot J] \tag{2.28}$$

En este caso, (2.13) se simplifica como:

$$h^T(Z) = \sqrt{2M} \cdot \Lambda_C \cdot C \cdot [I \quad -J] \cdot \begin{bmatrix} g_0(-z^{2M}) \\ z^{-M} g_1(-z^{2M}) \end{bmatrix} \cdot e(z) \tag{2.29}$$

### 2.2.2 Realización eficiente cuando N + 1 = 2mM

En el caso en que *m* sea un número par se llega a una modulación coseno del banco de análisis de la estructura de la Fig. 2.3 que se puede expresar como:

$$\hat{C}_A = \sqrt{M} \cdot \Lambda_C \cdot C \cdot [\ (I-J) \quad -(I+J)\ ] \tag{2.30}$$

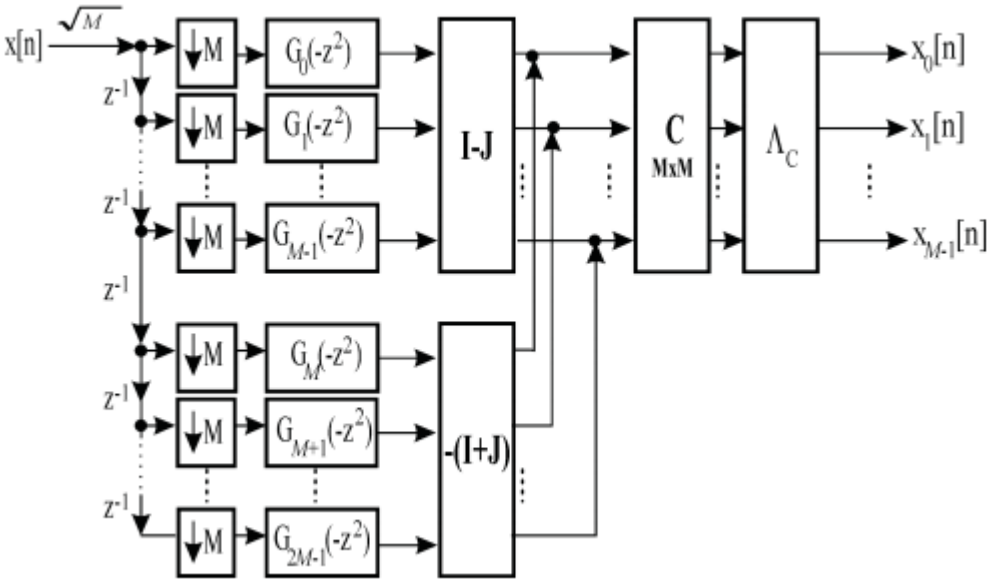



**Figura 2. 4 Realización polifase eficiente del banco de análisis para el caso *N+1=2mM* con *m* par.**

Incluyendo la Ecuación 2.30 en 2.13 se obtiene:

$$h^T(Z) = \sqrt{M} \cdot \Lambda_C \cdot C \cdot [\ (I-J) \quad -(I+J)\ ] \cdot \begin{bmatrix} g_0(-z^{2M}) \\ z^{-M} g_1(-z^{2M}) \end{bmatrix} \cdot e(z) \tag{2.31}$$

y la estructura correspondiente al banco de análisis se puede dibujar como en la Fig. 2.4 a partir de la cual se realiza el cálculo del coste computacional resultante [12].

**Tabla 2. 2 Coste computacional por cada M muestras de señal de entrada para N-PR CMFB (la longitud del prototipo cuando N + 1 = 2mM con *m* par).**

| Valor de M | Multiplicaciones | Sumas |
|---|---|---|
| ***Par*** | $(M/2)\log_2 M + 2M + (N+1)$ | $(3M/2)\log_2 M + N + 3M^2$ |
| ***Impar*** | $(M/2)\log_2 M + 3M + (N+1)$ | $(3M/2)\log_2 M + N + 3M^2 - 2M$ |

## 2.3 Costo computacional de implementaciones eficientes

La implementación polifase directa del banco de análisis requiere N multiplicaciones y N sumas por muestra. Por otro lado, el coste computacional para implementar el banco de análisis del N–PR CMFB de M canales usando la estructura de la Fig. 2.4 se obtiene teniendo en cuenta lo siguiente:

1. (N+1)/M multiplicaciones y N/M sumas por cada muestra de entrada para la etapa de los filtros polifase.
2. ((M/2) log2M +M) multiplicaciones y (3M/2) log2M sumas para calcular los M puntos de la DCT-IV.
3. Como *m* es un número par, $\Lambda_C$ sólo cambia los signos de las señales subbanda en determinados casos. No se consideran estas operaciones como multiplicaciones.
4. La matriz (I − J) requiere M sumas por muestra de entrada si M es un número par, y (M −1) sumas por muestra de entrada si M es un número impar.
5. La matriz − (I + J) requiere M sumas por muestra de entrada si M es un número par, y (M − 1) sumas y 1 multiplicación por muestra de entrada si M es un número par.
6. Una multiplicación y M sumas por muestra de entrada para el resto de las operaciones.

Teniendo en cuenta esto, en la tabla 2.2 se muestra el coste computacional calculado para valores de M pares e impares. Los cálculos se han realizado para un total de M muestras de entrada al banco de filtros [12].

## 2.4 Implementación en aritmética de punto fijo del NPR-CMFB

Como se pudo apreciar en el epíg. 2.1 las transformadas discretas de coseno son herramientas esenciales en el análisis numérico y en el procesamiento digital de señales, donde a menudo los procesadores emplean aritmética de punto fijo a pesar de que la aritmética de punto flotante es la más conocida. Aunque la aritmética de punto flotante ofrece una mayor precisión y rango dinámico, los diseñadores que operan en el procesamiento digital de señales eligen la aritmética de punto fijo con el objetivo de lograr mayor velocidad y menor costo de hardware en sus diseños.

### 2.4.1 Descripción de la estructura del Banco de Filtros a implementar.

El banco de filtros que se desea implementar en este trabajo es un banco de filtros coseno modulado que siga un algoritmo eficiente con el objetivo de optimizar el desempeño del núcleo de procesamiento de mayor costo computacional. Los filtros utilizados para conformar el banco de filtros se diseñaron según requerimientos para aplicaciones de codificación de señales biomédicas como el EEG y el ECG. En trabajos previos [25] se consideró suficiente emplear filtros de 16 y 32 canales con un ancho de banda de 100 Hz.

En el diagrama en bloques mostrado en la figura 2.4 se muestra la estructura para la realización eficiente de un banco de filtros de análisis coseno modulado. Tanto en los filtros de análisis como de síntesis las operaciones más relevantes las constituyen los filtros polifase y la DCT tipo IV, además de las multiplicaciones matriciales que le preceden a esta última.

Con vistas a realizar experimentos que evidencien el desempeño de la implementación en punto fijo del banco de filtros, se han elegido señales de entradas aleatorias, deterministas y reales. En todos los casos, estas señales se someten a escalado al rango de [-1,1) para lograr así la máxima precisión en su representación ya que se va a emplear aritmética de punto fijo. Dado que la mayoría de los procesadores de punto fijo actuales tienen una precisión de 16 bit y registros internos (acumulador y producto) de 32 bits, la representación del rango mencionado puede realizarse empleando 15 bits para fracción y 1 bit para signo. Al emplear las herramientas del *toolbox* de punto fijo del Matlab®, la representación numérica en punto fijo se realiza mediante objetos *fi*. Estos objetos permiten definir varias

propiedades entre las que se encuentran *InputWordLength* e *InputFracLength*, que se inicializan apropiadamente con los valores previamente mencionados.

La etapa de filtros polifase se realiza empleando filtros FIR [20].Los filtros FIR son buenos candidatos a la hora de trabajar con máxima precisión lo que hace que el proceso de convertir los filtros de punto flotante a punto fijo sea mucho más sencillo; sin embargo es muy importante tomar en consideración el rango de entrada para el análisis del rango estático ya que en esto se basa la implementación en máxima precisión. Es de destacar que el proceso de conversión de un filtro de punto flotante a punto fijo requiere de dos etapas principales, una en la que se efectúa la cuantización de los coeficientes y otra en la que se realiza el análisis del rango dinámico de la señal. En el caso de la cuantización primero es preciso determinar la cantidad de bits necesarios para representar los coeficientes,

La DCT puede realizarse mediante el uso de la FFT [26], es por ello que primeramente se estudia la realización de esta transformación en aritmética de punto fijo de manera que permita identificar el efecto de los errores numéricos y llegar a conclusiones referentes a arquitecturas hardware apropiadas para implementarla.

Con vistas a extraer conclusiones sobre el desempeño de los algoritmos de la FFT y por tanto de la DCT IV, se realizan experimentos que comparan resultados numéricos y desempeños en tiempos de ejecución de implementaciones en punto flotante y punto fijo. Como algoritmo de referencia de la FFT se toma la realización de la FFT interna del Matlab®. La FFT interna del Matlab® implementa un algoritmo DIT. Se elige una implementación similar para la FFT en punto fijo, tomada también del *toolbox Signal Processing.*

La implementación del algoritmo de la FFT en Matlab® supone que la señal de entrada se encuentre normalizada al rango (-1,1]. Esta restricción permite hacer un uso adecuado de la aritmética de punto fijo y minimizar y controlar los errores que su empleo supone. El escalado de la señal de entrada a este rango de valores garantiza que no se provoquen pérdidas de precisión producto de las operaciones de multiplicación y se minimicen los efectos dañinos del desbordamiento (*overflow).* La operación de la FFT puede generar overflow producto de la suma que realiza (Ecuación 2.32).

$$X_k = \sum_{n=0}^{N-1} x_n w_N^{nk} \tag{2.32}$$

En la ecuación anterior *x* es la señal de entrada, *N* es el número de puntos y *w* es un factor complejo.

Además entonces de asegurar que la señal de entrada se encuentre normalizada, se debe realizar otro escalamiento que garantice que la suma de *n* elementos sea menor que la unidad. Empleando un escalado completo por *1/n* se puede prevenir el desbordamiento. Este escalado puede resultar en una significativa reducción en la precisión de los datos de entrada. Si se escala solo la entrada de la primera etapa de longitud-n de la FFT por *1/n*, se obtiene una relación señal a ruido inversamente proporcional a $n^2$ . Sin embargo escalando la entrada en cada una de las log2(n) etapas de la FFT por ½, se obtiene un escalado completo de *1/n* con una relación señal a ruido inversamente proporcional a *n*.

La DCT-IV no viene implementada en el Matlab, por lo que se programó dicha función con el objetivo de utilizarla en el algoritmo rápido del Banco de Filtros Coseno Modulado tipo FIR de reconstrucción aproximada descrito en el epígrafe 2.1.2.

La etapa de síntesis del banco de filtros realiza el mismo proceso que la de análisis pero de manera inversa para obtener nuevamente la señal original con una reconstrucción aproximada.

Siguiendo lo planteado en los epígrafes anteriores, se implementó un programa encargado de llevar a cabo dicho algoritmo empleando aritmética de punto fijo y punto flotante con el objetivo de hacer una comparación de su desempeño empleando estos dos tipos de aritmética. En la programación se encontraron algunos errores producto de la propia aritmética. Ejemplo de ello son problemas debido al escalamiento, los cuales fueron resueltos realizando un estudio referente a las características y propiedades de la aritmética de punto fijo. Los resultados obtenidos de la implementación en cuestión se discutirán en el próximo capítulo.

# CAPÍTULO 3. RESULTADOS Y DISCUSIÓN

En este capítulo se muestra un análisis del desempeño de la implementación del algoritmo rápido del banco de filtros coseno modulado con el objetivo de evaluar el comportamiento del error que se comete empleando aritmética de punto fijo y punto flotante. Para validar el funcionamiento del algoritmo no solo se emplea una señal aleatoria, también se procesan una señal determinística y una real.

## 3.1 Implementación del algoritmo en punto flotante.

Con el objetivo de disponer de una referencia de desempeño del banco de filtros coseno modulado, se realizó una implementación en punto flotante utilizando Matlab. Teóricamente la reconstrucción no es perfecta, dado que no es una especificación en el diseño del banco de filtros. La señal de error calculada como la sustracción de la señal de entrada al banco de filtros y su salida no es nula y supondremos que esta condición está dada por las especificaciones de diseño del banco y no por su implementación con la aritmética de punto flotante de 64 bits que emplea el Matlab®. La figura 3.1 presenta las señales de entrada, salida del banco de filtro así como la señal de error. Las métricas de desempeño muestras valores de Relación Señal a Ruido de 55 dB, Relación Señal a Ruido Pico de 60 dB, Error Medio Cuadrático de 1.0045e-006 y de Error Máximo de 0.0051. Estos valores se tomarán como referencia para realizar las comparaciones con las implementaciones en punto fijo.

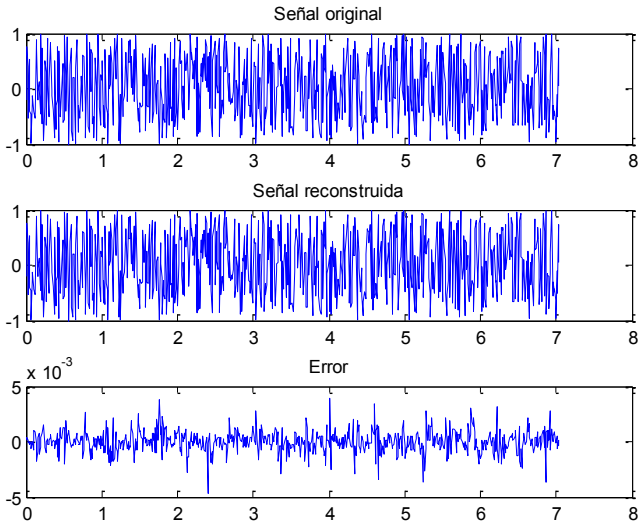



**Figura 3. 1 Señales de entrada, salida y error del banco de filtros.**

## 3.2 Implementación del algoritmo en punto fijo.

En el epígrafe anterior quedaron definidos los parámetros que se emplearán para implementar el algoritmo en punto fijo, pero antes de comenzar a analizar los resultados de la implementación resulta conveniente analizar el desempeño de los filtros polifase y de la DCT-IV ya que ambos juegan un papel muy importante dentro del algoritmo.

### 3.2.1 Implementación de los filtros polifase.

La estructura polifase de los filtros se emplea para el procesamiento subbanda ya que se divide la señal de entrada en varios componentes. En nuestro caso, siguiendo con el diagrama para la etapa de análisis del banco de filtros (ver Fig.2.4), empleamos para la construcción de los filtros polifase la función *dfilt* de Matlab que se encarga de construir un filtro de tiempo discreto con una estructura determinada. Para nuestra implementación utilizamos la estructura *dffirt* que realiza un filtro FIR de forma inversa.

La Tabla 3.1 lista los parámetros asociados a la implementación en punto fijo de cada uno de los filtros polifase. La primera especificación de diseño que se siguió fue instruir al objeto *dfilt* del Matlab que la entrada al filtro tiene una precisión de Wx bits y Wx-1 bits para la fracción. De esta manera se indica que el rango de valores de la señal de entrada a los filtros se encuentra entre los valores [-1,1). Como se planteó en el capítulo 2, Wx se refiere a la palabra de datos del procesador y en el ejemplo que se muestra en la tabla su

valor corresponde a Wx=16 bits. Otro elemento importante en el diseño es permitir que el objeto de filtrado realice el autoescalado de los coeficientes del filtro, de manera que se logre la mayor precisión en los resultados de la operación de convolución que realiza el filtro. Con esta decisión se observa que el número de bits de fracción efectivo es de 21 bits, amén de que la palabra de datos es de 16 bits. Otra especificación de diseño instruye emplear aritmética "*wrap*" en vez de aritmética de saturación. Esta decisión permite minimizar y en nuestro caso eliminar, los efectos asociados al desbordamiento durante operaciones de suma intermedias que sobrepasen el margen dinámico que ofrece la representación con Wx bits.

**Tabla 3. 1 Parámetros obtenidos de la implementación de los filtros polifase.**

| | |
|---|---|
| FilterStructure | 'Direct-Form FIR Transposed' |
| States | [8x1 embedded.fi] |
| Numerator | [1x9 double] |
| Arithmetic | 'fixed' |
| CoeffWordLength | 16 |
| CoeffAutoScale | 1 |
| Signed | 1 |
| RoundMode | 'convergent' |
| OverflowMode | 'wrap' |
| InputWordLength | 16 |
| InputFracLength | 15 |
| NumFracLength | 21 |
| FilterInternals | 'FullPrecision' |

Con estas especificaciones de diseño los filtros muestran un excelente desempeño y de esto da fe la Figura 3.2 en las que se observa que el error de redondeo resultante, al emplear aritmética de punto fijo, tiene una potencia de alrededor de -150 dB.

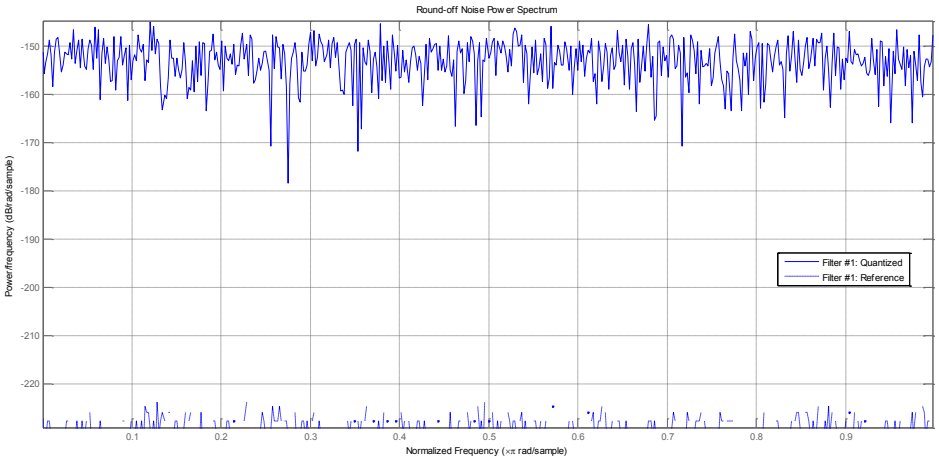



**Figura 3. 2 Espectro de potencia del ruido en los filtros polifase.**

### 3.2.2 Implementación de la DCT-IV.

El cálculo de la DCT-IV forma una parte importante del algoritmo para la implementación rápida de un banco de filtros coseno modulado. De ahí la importancia de analizar su comportamiento cuando se emplean aritmética de punto flotante y aritmética de punto fijo. Este análisis se lleva a cabo empleando una señal determinística y sus resultados se reflejan en las figuras 3.3 y 3.4 respectivamente.

Para cuantificar el ruido introducido por la DCT IV, se realizó un experimento en el que una señal de entrada se le aplica la transformada DCT IV directa e inversa. La señal de error se calcula como la diferencia entre las señales de entrada y reconstruidas. La Figura 3.3 presenta los oscilogramas de la señal de entrada, la señal reconstruida, los valores de la transformada y el oscilograma de la señal de error. La señal de entrada se corresponde con un ruido blanco gaussiano con media cero y desviación estándar 1. Como puede observarse, los límites del error tienen un orden inferior a $10^{-15}$, correspondiente a una relación señal a ruido superior a 300dB.

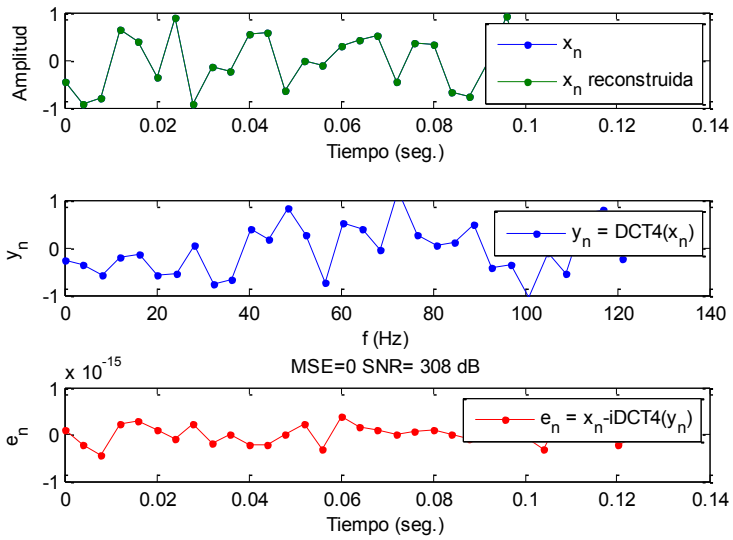



**Figura 3. 3 Transformada DCT-IV en punto flotante.**

Al emplear aritmética de punto fijo con palabras de datos de Wx=16 bits, la relación señal a ruido se deprime, sin embargo su valor se encuentra alrededor de 66dB que representa una excelente figura de mérito.

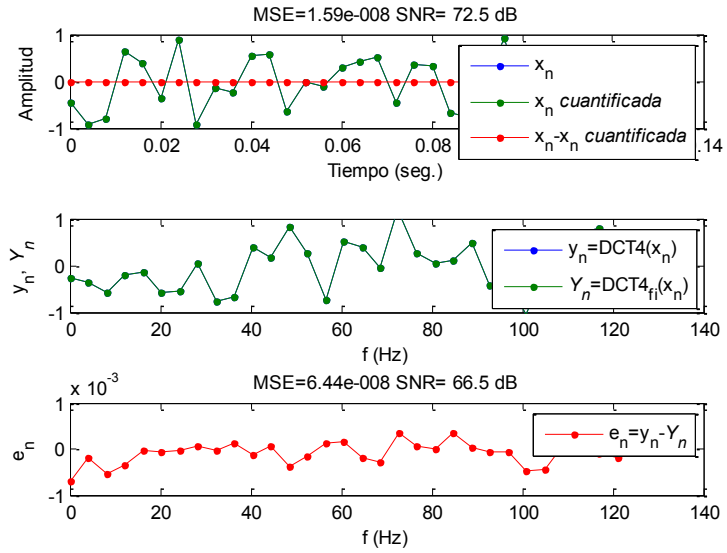



**Figura 3. 4 Transformada DCT-IV en punto fijo.**

### 3.2.3 Desempeño del banco de filtros.

En los análisis previos, se ha empleado, como se especificó anteriormente, señales aleatorias correspondientes a ruidos blancos gaussianos que satisfacen una distribución

probabilística normal. Con vistas a extender este análisis, se decidió emplear otro tipo de señales que ayudasen a caracterizar el comportamiento del sistema en situaciones más cercanas a su funcionamiento en las aplicaciones en que el banco de filtros se empleará. En el CEETI, este banco de filtros se ha empleado fundamentalmente en la compresión de señales EEG. Por esta razón, se eligió un segmento de señal de EEG obtenida de la base de datos CHB-MIT Scalp EEG Database. En las Figuras 3.5 y 3.6 se muestran las señales original, reconstruida y de error para la señal aleatoria y la señal real respectivamente, que se aplican y obtienen del banco de filtros implementado con aritmética de punto fijo, simulando su operación en procesadores de 16 bits.

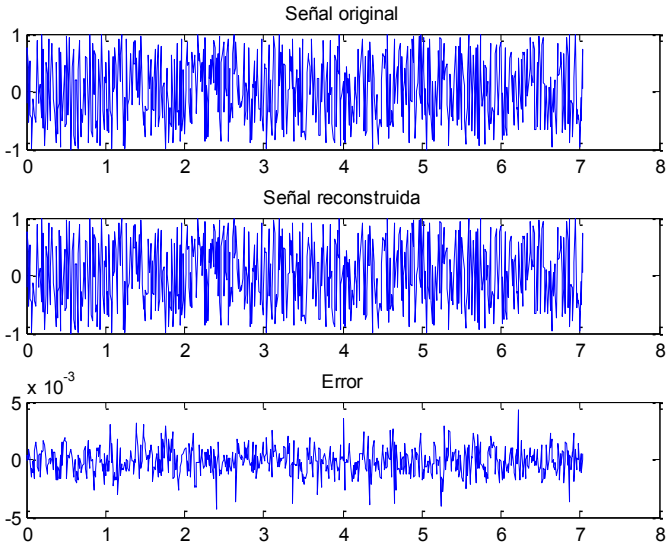



**Figura 3. 5 Señales de entrada, salida y error del banco de filtros empleando una señal aleatoria.**

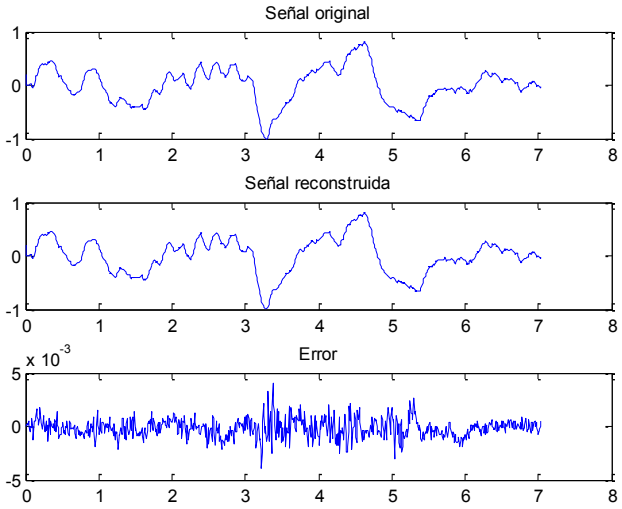



**Figura 3. 6 Señales de entrada, salida y error del banco de filtros empleando una señal real.**

Como se puede observar el las figuras si analizamos el orden del error en ambos casos podemos observar que el comportamiento es muy similar y se corresponde con el error obtenido durante la operación de la DCT. La tabla 3.2 nos muestra el resultado de las métricas calculadas par ambas señales y al igual que en el caso de la representación gráfica, en este caso se puede apreciar que la variación en los resultados es mínima.

**Tabla 3. 2 Métricas calculadas a la señal aleatoria y la señal real.**

| Métricas | Señal aleatoria | Señal Real |
|---|---|---|
| SNR (dB) | 53.8343 | 51.6452 |
| PSNR | 58.7579 | 60.9746 |
| MSE | 1.3311e-006 | 7.9899e-007 |
| ME | 0.0043 | 0.0041 |

## 3.3 Análisis de los errores de cuantificación en la implementación del banco de filtros.

Empleando una señal aleatoria con distribución normal cuando la implementación se realiza con aritmética de punto flotante se puede apreciar la presencia de un error que tiene lugar producto de la utilización de un algoritmo de filtrado con características de

reconstrucción aproximada. Considerando el hecho de que la señal de entrada está en el rango [-1,1) el error, que tiene un valor de $3{,}1 \cdot 10^{-3}$ con una relación señal a ruido de 55 dB, puede considerarse bastante pequeño (ver fig.3.7).

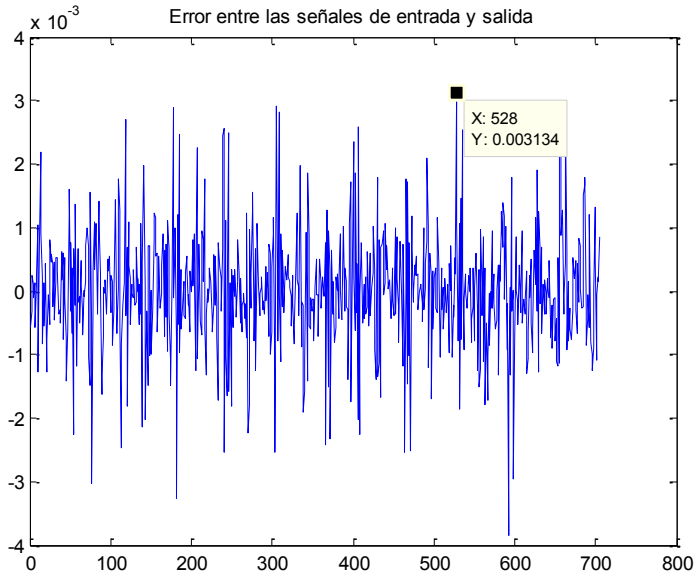



**Figura 3. 7 Error entre la señal de entrada y la señal reconstruida empleando punto flotante.**

Cuando el algoritmo se implementa con aritmética de punto fijo también existe un error entre la entre la entrada y la salida del filtro que se encuentra aproximadamente por el orden de $3{,}9 \cdot 10^{-3}$.En la aritmética de punto fijo además de la característica de reconstrucción aproximada influye también el error de precisión que introduce la propia aritmética (ver fig. 3.8).

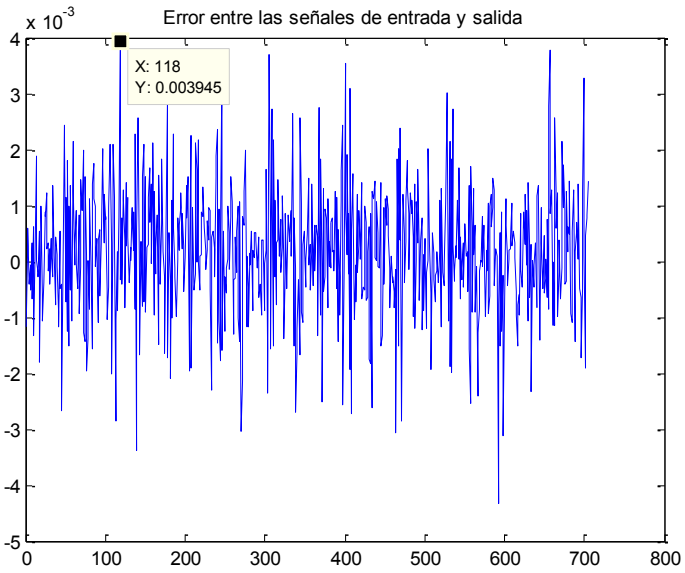



**Figura 3. 8 Error entre la señal de entrada y la señal reconstruida empleando punto fijo.**

De forma general cuando hacemos una comparación entre el error que se comete con aritmética de punto flotante respecto a la aritmética de punto fijo vemos que las diferencias no son significativas por lo que la precisión con que se efectuaron las operaciones fue manejada de manera tal que los resultados no se afectaron considerablemente (ver fig.3.9).

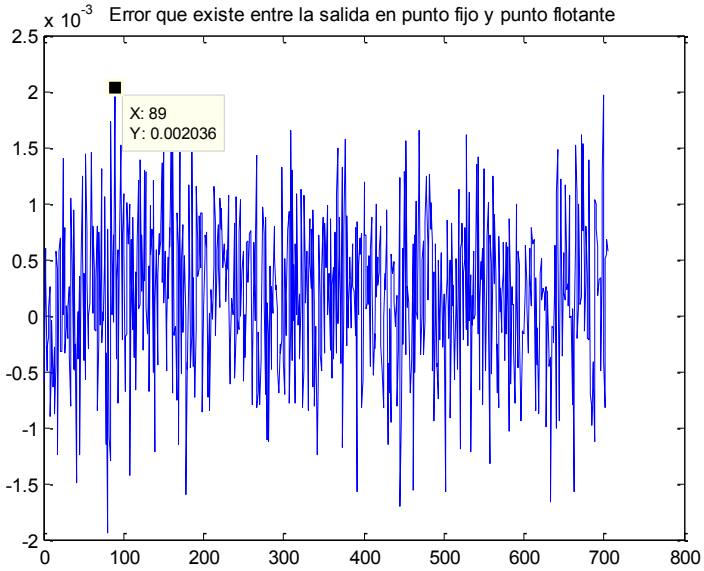



**Figura 3. 9 Error que existe entre las salida del banco de filtros en punto flotante y punto fijo.**

La Tabla 3.3 respalda los resultados obtenidos gráficamente y el análisis realizado al error que se comete producto de la cuantificación empleando aritmética de punto flotante y de

punto fijo. El incremento del error entre la implementación en punto flotante y la implementación en punto fijo es de aproximadamente 2dB.

**Tabla 3. 3 Métricas calculadas a las implementaciones en punto flotante y punto fijo.**

| Implementaciones | SNR(dB) | MSE | PSNR | ME |
|---|---|---|---|---|
| Punto Flotante | 55.7653 | 8.5331e-007 | 60.6889 | 0.0039 |
| Punto Fijo | 53.8343 | 1.3311e-006 | 58.7579 | 0.0043 |

Aquí vemos como las medidas de calidad adoptan valores similares y que se consideran dentro del rango de valores deseables para esta aplicación.

Es importante observar que la implementación de las partes fundamentales (filtrado polifase y DCT IV) en punto flotante conducen a errores mucho más pequeños que los mostrados por el banco de filtros. Esta situación nos dice que los errores fundamentales se deben a que el banco de filtros no ha sido diseñado para lograr reconstrucción perfecta. El elemento que más incide en el error de reconstrucción es por tanto esta especificación de diseño.

No obstante, la implementación en punto fijo añade otro error de reconstrucción que se atribuye, fundamentalmente a la implementación de la DCT IV, según pudimos observar del epígrafe 3.2.3.

### 3.3.1 Implementación con un procesador de menor longitud de palabra. Efectos para el error de cuantificación.

Como habíamos visto en epígrafes anteriores al trabajar con aritmética de puno fijo se configuran una serie de propiedades como la longitud de palabra a la entrada y la salida del procesador, la longitud de palabra deseada para el resultado del producto y la longitud de palabra deseada para el acumulador. Cuando se convierte a punto fijo también se especifican parámetros como el modo de precisión con que se desea que se efectúen la suma y el producto, el modo de manejo de overflow entre otros. En nuestro caso se emplearon unas longitudes de palabra de entrada y salida de 16 bits así como longitudes de palabra del acumulador y del producto de 32 bits. Para estos valores se realizó la

configuración de los demás parámetros con el objetivo obtener los resultados deseados entre los que se encuentra lograr un mínimo error en el proceso de filtrado. Esto no se cumple cuando se emplea un procesador de 12 bits, pues como se puede observar en la figura 3.10 aumenta el error producto de la cuantificación de los coeficientes.

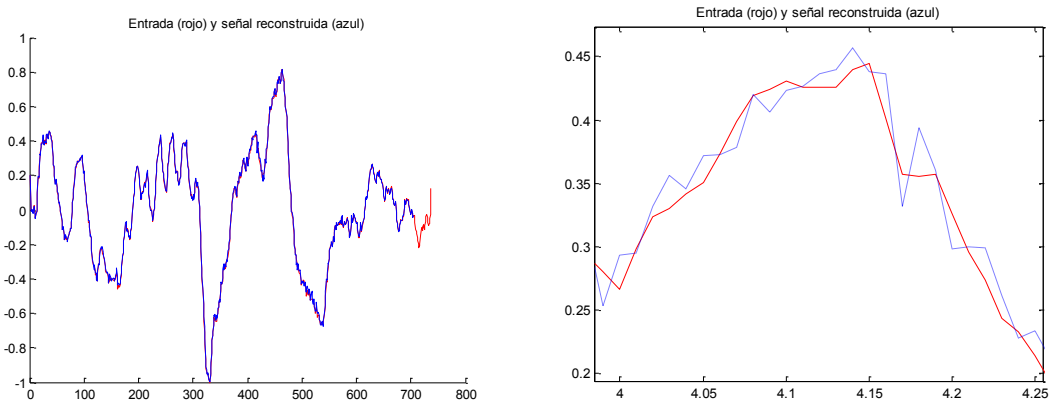



**Figura 3. 10 Resultado del filtrado con un procesador de 12 bits.**

Este error también se ve reflejado en la Relación Señal a Ruido (29 dB), la Relación Señal a Ruido Pico (38dB), el Error Medio Cuadrático (1.3840e-004) y el Error Máximo (0.0457).

En el peor de los casos cuando se emplea un procesador de 8 bits el error aumenta en una mayor medida como se puede apreciar en la figura 3.11 y en las métricas calculadas que se muestran en la tabla 3.3.

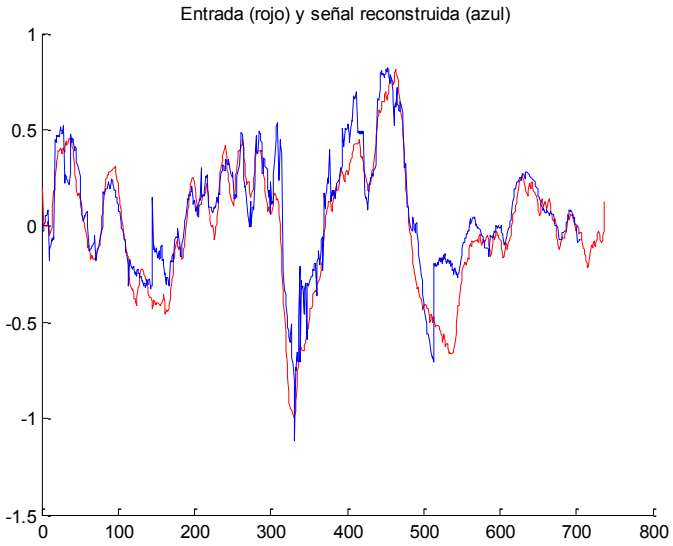



**Figura 3. 11 Diferencia entre las señales de entrada y salida utilizando un procesador de 12 bits.**

| SNR(dB) | PSNR | MSE | ME |
|---|---|---|---|
| 6.9403 | 16.2714 | 0.0236 | 0.5926 |

Es posible concluir entonces que a implementación en punto fijo debe realizarse, al menos en un procesador de 16 bits.

# CONCLUSIONES Y RECOMENDACIONES

Los bancos de Filtros Coseno Modulados proporcionan una realización eficiente y en la actualidad tienen diversas aplicaciones en casi todas las aristas del procesamiento digital de señales por lo que constituyen una opción atractiva a los ojos de diseñadores e investigadores de estas esferas.

## Conclusiones

1 Conclusión 1: El empleo del toolbox de punto fijo de Matlab es un mecanismo adecuado para las simulaciones de este tipo de implementaciones en ausencia de hardware.

2 Conclusión 2: Se logró llevar a cabo la implementación del algoritmo empleando punto flotante y punto fijo.

3 Conclusión 3: Se realizaron una serie de experimentos que demuestran la factibilidad de la implementación en punto fijo del banco de filtros sin que conduzca a errores apreciables producto de la aritmética.

4 Conclusión 4: La mayor potencia de error se debe a las especificaciones de reconstrucción casi perfecta y no a la propia aritmética.

5 Conclusión 5: Una arquitectura de procesador con16 bits de datos y 32 de acumulador y producto representa tan solo un incremento poco mayor a 2 dB de la potencia de ruido y debiera ser suficiente para un gran número de aplicaciones que requieran de este procesamiento.

**Recomendaciones**

Recomendación 1: Realizar el algoritmo desarrollado en Matlab en sistemas de desarrollo para C y C++ de manera tal que los programas puedan ser compilados para su ejecución en procesadores de punto fijo.

Recomendación 2: Realizar la implementación en hardware FPGA y específicamente en el dispositivo NEXYS2 que se encuentra disponible en el CEETI.

# REFERENCIAS BIBLIOGRÁFICAS

[1] Bhate,K.R (2004). IMPLEMENTATION AND COMPARISON OF COSINE MODULATED FILTER BANKS ON A FIXED POINT DIGITAL SIGNAL PROCESSOR. Texas, Texas Tech University.

[2] filter banks, part1: Principles and design techniques DSP DesignLine, January15, 2009.

[3] FIXED-ANALYSIS ADAPTIVE-SYNTHESIS FILTER BANKS A Thesis Presented to The Academic Faculty by Clyde Alphonso Lettsome. Georgia Institute of Technology May 2009.

[4] Bellanger, M., Bonnerot, G., and Coudreuse, M. (1976),Digital Filtering by polyphase network: Application to sampling rate alteration and Filter banks,"*IEEE Trans. on Acoustics, Speech, and Signal Processing*, pp. ASSP-24:109- 114, April.

[5] Vaidyanathan, P. and Hoang, P., Lattice structures for optimal design and robust implementaion of two-channel perfect reconstruction qmf banks," *IEEE Trans. on Acoustics, Speech, and Signal Processing*, pp. ASSP-36:97-112, January 1988.

[6] Mitra, S. K. and Sherwood, R. J., Canonic realizations of digital bersusing the continued fraction expansion," *IEEE Trans. on Audio and Electroacoustics*, pp. 185-194, August 1972.

[7] Dudley, H., (1939),The vocoder," *Bell Labs Rec.*, vol. 177, pp. 122-126.

[8] Croisier, A., Estaban, D., and Galand, C. Perfect channel splitting by use of interpolation/decimation/tree decomposition techniques," in *In Proc. Conf on Inf. Sciences and Systems*, pp. 191-195, May 1976.

[9] T. Q. Nguyen, "Near-perfect-reconstruction pseudo-QMF banks," IEEE Transactions

a. on Signal Processing, vol. 42, no. 1, pp. 65-76, Jan. 1994.

[10] Application Note 33, Fixed Point Arithmetic on the ARM, Copyright Advanced RISC Machines Ltd (ARM),( 1996), Document Number: ARM DAI 0033A, September.

[11] Lettsome, C.A. (2009). Fixed-Analysis Adaptative-Syntesis Filter Banks. Facultad Académica. Georgia, Instituto de Tecnología.

[12] Velazco, M.B. (2004). Compresión de Electrocardiogramas Mediante Bancos de Filtros Coseno Modulado y Análisis Multiresolución. Depto. De Teoría de la Señal y Telecomunicaciones Madrid, Alcalá: 62-70.

[13] Roldan, F.C. (2009).Sistemas de Tasa Múltiple y Bancos de Filtros Fernando Cruz Roldán.

[14] Manuel Blanco-Velazco, F.C-Roldan, Eduardo Moreno-Martínez, Juan Ignacio Godino-Llorenteb, Kenneth E. Barrer (2008).”Embedded filter bank-based algorithm for ECG compresion. Signal Processing 88: 1402-1412.

[15] E.Cooper and Ph.D., Minimizing Quantization Effects Using TMS320.Digital Signal Processor Family. (1994) Application Report.

[16] M. Peirone, Herramientas de Diseño y Aritmética de Punto Fijo. Sept / 3/2009.

[17] E. Usevitch and C. L. Betancourt (1999), Fixed-Point Error of Two-Channel Perfect Reconstruction Filter Banks with PERFECT Alias Cancellation, *IEEE Trans. Circuits Syst. II.* Vol. 46, no. 11. pp. 1437-1440.

[18] Tomado de: http://en.wikipedia.org/wiki/Discrete_cosine_transform

[19] M. Tsai, IIR filter Design on the TMS320C54X.( 1996) Application Report.

[20] Cruz-Roldán and M. Monteagudo-Prim, Efficient Implementation of Nearly Perfect Reconstruction FIR Cosine-Modulated Filterbanks, *IEEE Trans. on Signal Processing.* Vol. 52. No. 9. pp. 2661-2664. Sept. 2004.

[21] Michael Cerna and Audrey F. Harvey, The Fundamentals of FFT-Based Signal Analysis and Measurement, *Application Note 041*

[22] R.W.Bäuml and W. Sörgel, Uniform Polyphase Filter Banks for Use in Hearing Aids: Design and Constraints, Conf. EUSPICO 2008.

[23] Shojan, Numerical Performance of Fixed-Point DSP Data Path, or, What are Guard Bits and Why Should I Care? , DSPx Exhibition and Symp. San José, California, March 11-14. pp. 1-7. 1996.

[24] F. Cruz-Roldán, P. Amo-López, S. Maldonado-Bascón and S. S. Lawson, An efficient and simple method for designing prototype filters for cosine- modulated pseudo-QMF banks, IEEE Signal Processing Lett. , vol. 9, pp. 29-31, Jan. 2002.

[25] Bazán-Prieto, C, Cárdenas-Barrera, J, Blanco-Velasco, M, Cruz-Roldán, F, "Análisis de esquemas de compresión de EEG basados en Bancos de Filtros Modulados y Transformadas Wavelet", Work Shop 2010, Santiago de Cuba, Marzo 2010.

[26] Xuancheng, S, Steven G. Johnson, Type-IV DCT, DST, and MDCT algorithms

[27] with reduced numbers of arithmetic operations, Department of Mathematics, Massachusetts Institute of Technology, Cambridge.

# ANEXOS

## Anexo I Función DCT-IV

```
function b = fidct4(a,n,Tdct,F)
%FIDCT4  Discrete cosine transform (type IV).
%
%   This is an implementation of  DCT IV using floating or fixed point
%   arithmetic. It uses, as a template, the matlab's
DCT(II)implementation.
%   Following is the DCT help text from matlab:
%%%%%%%%%%%%%%%%%
%       Y = DCT(X) returns the discrete cosine transform of X.
%       The vector Y is the same size as X and contains the
%       discrete cosine transform coefficients.
%
%       Y = DCT(X,N) pads or truncates the vector X to length N
%       before transforming.
%
%       If X is a matrix, the DCT operation is applied to each
%       column.  This transform can be inverted using IDCT.
%
%       See also FFT, IFFT, IDCT.
%%%%%%%%%%%%%%%%%
%   The new DCT IV function uses floating point arithmetic if vector X is
%   not in fixed point (fi object) format and less that 3 arguments are
%   passed to the function. Here are the available options of passing
%   parameters:
%    1. Invoking FIDCT4 with 1 input argument
%
%       Y = FIDCT4(X) returns the discrete cosine transform (type IV)of
X.
%       The vector Y is the same size as X and contains the
%       discrete cosine transform coefficients.
%       - If X is a floating point vector, then execution will be carried
%         out with matlab's floating point arithmetic
%       - If X is a fixed point (fi object) vector, then fixed point
%         implementation will be used and fixed point internal
specifications
%         will be derived from X's numeric type and X's arithmetic
%
%    2. Invoking FIDCT4 with 2 parameters
%
%       Y = FIDCT4(X,N) pads or truncates the vector X to length N
```

```
%       before transforming.
%       Previous behaviour holds for this syntax
%
%    3. Y = FIDCT4(X,N,T) pads or truncates the vector X to length N
%       before transforming and uses T numeric
%

%   Author(s): C. Thompson, 2-12-93
%              S. Eddins, 10-26-94, revised
%   Copyright 1988-2004 The MathWorks, Inc.
%   $Revision: 1.7.4.2 $  $Date: 2004/12/26 22:15:37 $

%   References:
%   1) A. K. Jain, "Fundamentals of Digital Image
%      Processing", pp. 150-153.
%   2) Wallace, "The JPEG Still Picture Compression Standard",
%      Communications of the ACM, April 1991.

%   Modified by Julian Cárdenas Barrera. December 2009

%% Initialization
do_norm = true;

if nargin == 0,
    error('Not enough input arguments.');
end

if isempty(a)
   b = [];
   return
end

if ((nargin < 3) && isfi(a))
    Tdct = numerictype;
    F = a.fimath;
end

if (nargin>2)
    % fixed point implementation will be used
    if (~isfi(a))
        a = fi(a,Tdct);
    end
    if (nargin==3)
        F = fimath;
    end
    a.fimath = F;
end

if isfi(a)
    defaultTCexp = numerictype;
    defaultTCexp.DataType =         'Fixed';
    defaultTCexp.Scaling =          'BinaryPoint';
    defaultTCexp.Signed =           true;
    defaultTCexp.WordLength =       16;
    defaultTCexp.FractionLength =   15;
```

```
    if (nargin < 3)
        TCexp = defaultTCexp;
    else
        TCexp = Tdct;
    end
end

% If input is a vector, make it a column:
do_trans = (size(a,1) == 1);
if do_trans, a = a(:); end

if nargin==1,
  n = size(a,1);
end
m = size(a,2);

% Pad or truncate input if necessary
if size(a,1)<n,
  aa = zeros(n,m);
  aa(1:size(a,1),:) = a;
else
  aa = a(1:n,:);
end

%% Compute weights to multiply DFT coefficients

% Compute pre-processing weights
pp = (exp(-1i*(0.5+(0:n-1))*pi/(2*n))).';

% distribute weights columnwise if X is a matrix (each column of input
% matrix will be a vector to transform
P = pp(:,ones(1,m));

% Compute post-processing weights
ww = (exp(-1i*(0:n-1)*pi/(2*n))).';

% distribute weights columnwise if X is a matrix (each column will be a
% vector to transform
W = ww(:,ones(1,m));

% if fixed point is to be used, then apply proper arithmetic to the
% weights. Both P and W have maximum complex modulus of one (1). So, full
% precision can be used for fractions (plus sign and a bit for
representing
% positive one)
if (isfi(a))
    W = fi(W,Tdct, F);%, TCexp, F);
    P = fi(P,Tdct, F);% TCexp, F);
end

%% Compute transform

% generate fft twiddle factors
w0 = fi_radix2twiddles(2*n);
if (isfi(a))
```

```
    w0 = fi(w0,Tdct, F);% TCexp, F);
end

% Apply preprocessing weights
% Complex-complex multiplication involves an addition as well as
% multiplication, so the word length of the full-precision result has
% one more bit than the sum of the word lengths of the multiplicands
z = P.* aa;

if (isfi(a))
    % check output's best precision
    intlen = z.wordlength-z.fractionlength - 1;
%     if (fix(intlen/(2*n))>0)
        y = fi(zeros(2*n,m),1,TCexp.wordlength,TCexp.wordlength-intlen);
%      else
%         y = fi(zeros(2*n,m),1,TCexp);
%     end
    y(1:n,:) = z;
    y.fimath = F;
else
    y = zeros(2*n,m);
    y(1:n,:) = z;
end

y = fi_radix2fft_withscaling(y, w0); % y number of samples must be a
power of 2
y = y(1:n,:);

% Multiply FFT by post-processing weights:
if isfi(a)
    intlen = ceil(log2(n))+1;
    b = fi(zeros(size(y)),1,y.Wordlength, y.Wordlength-intlen-1);
    b.fimath = F;
end

b = 2* n * W .* y;

%% Return proper values
if isreal(a), b = real(b); end
if do_trans, b = b.'; end
if do_norm b = b * sqrt(2/n); end
```

## Anexo II Programa para la implementación de un Banco de filtros Coseno Modulado en aritmética de punto flotante y punto fijo.

```
%% Señal de entrada al filtro (Ojo!, en estos momentos potencia de 2)
% Crear un ruido uniforme con margen dinámico entre -1:1
N = 1024;

% N tiene que ser por ahora una potencia de dos (por la implementación de
% la FFT usada)
```

```
N = 2^nextpow2(N);
Nsymbols = 256;
alphabet = -1:1/(fix(Nsymbols/2) -1):1-eps;
probs = ones(size(alphabet))./numel(alphabet);
in = randsrc(1,N,[alphabet;probs]);

%% Carga del filtro a emplear
filterpath = 'J:\tesis LORA\filter_bank fixed point';
filtername = 'bf_32chan_spl_3db';
load ([filterpath filesep filtername])

proto = filter3b;

%% invocar a la función de filtrado con empleo de punto flotante
M = 32;
%
[outsr] = fi_fastcm_caso1(in,proto,M);

%% invocar la función de filtrado con empleo de punto fijo

% Especificación de las características del procesador
Wx    = 16; % Effective word length of the data, x
Wy    = 16; % Effective word length of the output data, y
Wprod = 32; % Effective word length of the product
Wacc  = 32; % Effective word length of the sum


F = fimath('ProductMode','KeepLSB', 'ProductWordLength',Wprod,...
           'SumMode','KeepLSB', 'SumWordLength',Wacc,...
           'OverFlowMode','wrap');

% trataremos de poner la entrada normalizada entre -1:1 en punto fijo
fi_in = fi(in, true, Wx, Wx-1);

fi_in.fimath = F;

[fi_outsr] = fi_fastcm_caso1(fi_in,proto,M,Wy);
```

## Anexo III Función para la realización del filtrado de la señal.

```
Ejemplo de filtrado de fichero
% Entradas:
%   'in'    ->  Señal de entrada a filtrar
%   'proto' ->  Filtro prototipo coseno modulado
%   'M'     ->  Número de subbandas
% Salidas:
%   'outsr'   ->  Señal filtrada

function [outsr]=fi_fastcm_caso1(in,proto,M,Wx,Wy);
%% debugging info
debug = false;
dbg_dir = './debug';
```

```
fl_dbg_file = fullfile(dbg_dir,'fl_data.txt');
fi_dbg_file = fullfile(dbg_dir,'fi_data.txt');
if debug
    fipref('LoggingMode', 'on');
else
    fipref('LoggingMode', 'off');
end

%%
Wx=16;% Effective word length of the data, x
Wy=16;% Effective word length of the output data, y
longitud=length(proto);
k=length(proto)/(2*M);

%%%%%%%%%%%%%%%%%%%%%%
%       Inicializaciones
%%%%%%%%%%%%%%%%%%%%%%

% Longitud de los filtros menos 1
N=length(proto)-1;

% Longitud de la señal de entrada
L=length(in);

% Inicializaciones de análisis o recepción
[estatana,bufferana,fpolifasana,deltacana,N,deltapana,deltasana]=csiniana
lisis(proto,M);
iteraciones=fix(L/M);

% Inicializaciones de síntesis o transmisión
[estat2sint,buffer2sint,fpolifas2sint,deltacsint,N,deltapsint,deltassint]
=csinisintesis(proto,M);

%% Convertir los filtros a objetos dfilt (Digital Filter implementations)
%% que permiten usarlos en punto flotante y punto fijo
% Modificaremos la estructura de los filtros polifase. Sería bueno
hacerlo
% cuando se crean; pero para modificar lo menos posible el programa lo
% haremos acá
[nfilters, filterlen] = size(fpolifasana);
for i=1:nfilters
    ahk(i)= dfilt.dffirt(fpolifasana(i,:));
    afk(i)= dfilt.dffirt(fpolifas2sint(i,:));
    ahk(i).PersistentMemory = true;
    afk(i).PersistentMemory = true;
    if isfi(in)
        set(ahk(i),'Arithmetic','fixed', 'InputWordlength',
in.WordLength, 'InputFracLength', in.FractionLength  );
        set(afk(i),'Arithmetic','fixed', 'InputWordlength',
in.WordLength, 'InputFracLength', in.FractionLength  );
    end
end


%% Continúa según algoritmo original de Fernando
```

```
if rem(k-0.5,2)==0
    disp('Comienza análisis rápido para N=(2K+1)M, K par');
    muestraact=1;

    % Señales de análisis
    out=zeros([M iteraciones]);
    k=length(proto)/(2*M);

    % Bucle de análisis o recepción
    for mue=1:iteraciones,
      %disp('Comienza análisis rápido para N=(2K+1)M, K par');

[subbandasar(:,mue),estatana,bufferana]=fi_ana2k1Mparcm_caso3(in(muestraa
ct:muestraact+M-1),estatana,bufferana,ahk,deltacana,M,N,Wx);
      % Actualizar el número de muestra
      muestraact=muestraact+M;
    end;

    if debug
        if ~exist(dbg_dir, 'dir')
            mkdir(dbg_dir)
        end
        if ~isfi(in)
            dlmwrite(fl_dbg_file, subbandasar', 'delimiter', '\t', ...
                'precision', 6)
        else
            dlmwrite(fi_dbg_file, double(subbandasar)', 'delimiter',
'\t', ...
                'precision', 6)
        end
    end

    disp('Comienza síntesis rápida para N=(2K+1)M, K par');
    % Longitud de la señal de entrada
    L2=length(subbandasar);
    %i=1;
    iteraciones2=L2;
    muestraact2=1;

    % Señales de análisis
    outsr=zeros([1 L2*M]);

    % Bucle de análisis - síntesis
    for mue=1:iteraciones2,
        % Síntesis rápida
        %disp('Comienza síntesis rápido para N=(2K+1)M, K par');
        [outsr(muestraact2:muestraact2+M-
1),estat2sint,buffer2sint]=fi_sint2k1Mparcm_caso3(subbandasar(:,mue),esta
t2sint,buffer2sint,afk,deltacsint,M,N,16);
        % Actualizar el número de muestra
        muestraact2=muestraact2+M;
    end;
```

```
    if debug
        if ~exist(dbg_dir, 'dir')
            mkdir(dbg_dir)
        end
        if ~isfi(in)
            filename = fl_dbg_file;
        else
            filename = fi_dbg_file;
        end
        dlmwrite(filename, outsr(:), 'delimiter', '\t', ...
            'precision', 6)
    end


    %Ajuste de desfase
    outsr(1:(longitud-M))=[];
    in((length(outsr)-M+1):length(in)-1)=[];

else
    error('la longitud del prototipo debe ser (2K+1)M, con K par')
end

% Síntesis
entrada=length(in);
outsr(1:2*M-1)=[];
salida=length(outsr);
plot(in,'r');grid;
title('entrada (rojo)');
figure;
hold on;

plot(outsr,'b');grid;
title('salidab (azul)');
figure;
hold on;%plot(outst,'r');
plot(in,'r');
plot(outsr,'b')
title('Entrada (rojo) y señal reconstruida (azul)');% y rojo)');zoom
on;%pause;close;
%title('Entrada (rojo)');
in=double(in(1:end-M));

outsr=double(outsr);

disp('Cálculo de valores para comparar la señal de entrada con la de
salida ');
snrdb=10*log10(sum(in.^2)/sum((in-outsr).^2));
disp('S/N');
disp(snrdb);

valormse=mse(in,outsr);
disp('mean square error');
disp(valormse);
```

```
valorp=psnr(in,outsr);
disp('peak signal noise ratio');
disp(valorp);

valorme=maxerror(in,outsr);
disp('maximum error');
disp(valorme);

figure;
t=(0:length(in)-1)/100;
plot(t,in,'r');grid;
hold on;
plot(t,outsr,'b:');grid;
text(0.085,0.55,'···xre[n]');
text(0.085,0.75,' - xin[n]');
title('Entrada (rojo) y señal reconstruida (azul)');
hold off;
```

## Anexo IV Función que implementa el banco de filtro de análisis 'rápido' para M muestras.

```
% Función que implementa el banco de filtro de análisis N=(2K+1)M, K even
'rápido' para M muestras
% Entradas:
% - 'in': Señal de entrada a filtrar, M muestras.
% - 'estat': Variables de estado del banco de filtros.
% - 'buffer': Buffer de señal de entrada de 2M muestras
% - 'fpolifas': Filtros de análisis G2M(-z^2)
% - 'deltac': Matriz parte de T
% - 'M': Número de subbandas
% - 'N': Longitud del filtro prototipo -1 --> N=(2K+1)M, K even
% Salidas:
% - 'out': Señal de salida subbandas, 1 muestra por subbanda.
% - 'estat': Variables de estado del banco de filtros.
% - 'buffer': Buffer de señal de entrada de 2M muestras
%
% Esta función permite ahora la ejecución con aritmética de punto fijo
% del algoritmo implementado y descrito en :
%
%   F. Cruz-Roldán, M. Monteagudo, "Efficient implementation of
%   nearly-perfect reconstruction cosine-modulated filterbanks", IEEE
%   Transactions on Signal Procesing, vol. 52, no. 9, pp. 2661-2664,
%   September 2004.
%
% Para la ejecución en punto fijo se debe garantizar que la señal de
% entrada (in) esté expresada en punto fijo, de lo contrario el algoritmo
% se ejecutará en punto flotante. Obsérvese el siguiente ejemplo:
%

% Modificada por Julián Cárdenas. Diciembre 2009 hjgfdaf akjhk asdkfjhk
% asd
```

```
function
[out,estat,buffer]=fi_ana2k1Mparcm_caso3(in,estat,buffer,fpolifas,deltac,
M,N,Wx);

% Diezmado y Filtrado
entrada=[in(M:-1:1) buffer];

inputa=entrada(M:(3*M-1));
% INPUTA=fft(inputa,2*M)./HK;
% %INPUTA=fft(inputa,2*M);
% input=ifft(INPUTA,2*M);



input=inputa;


for f=(1:2*M)
   fpolifas(f).States = estat(:,f);
   [out(f)]=filter(fpolifas(f),input(f));
   estat(:,f) = fpolifas(f).States;

end;

% for (f=1:2*M),
%    [out(f),estat(:,f)]=filter(fpolifas(f,:),1,entrada(M-
1+f),estat(:,f));
% end;
buffer=entrada(1:2*M);
%%%%%%%%%%%%%%%%%%%%%%
% matrices
%%%%%%%%%%%%%%%%%%%%%%
% I
I = diag(ones(1,M));
 if isfi(input)
    I = fi(I);
 end
 outa=transpose(I*transpose(out(1:M)));


% -J
J = fliplr(I);
outb=transpose((-J)*transpose(out((M+1):(2*M))));

out=outa+outb;

%%%%%%%%%%
%  DCT4
%%%%%%%%%%
%out=dctiv(out.');
if isfi(in)
    nT= get(in,'numerictype');
    F= get(in,'fimath');
    outdouble=double(outa+outb);
```

```
    out1=fi(outdouble,1,Wx);
    l=length(out1);
    out1=fidct4(out1.',l,nT);
    %Reescalar (Mirar c'omo garantizarlo dentro de la dct
    out1 = fi (out1,'numerictype', nT);
    out = fi(out1.*(deltac.'),'numerictype', nT); % deltac
else
  out=fidct4(out.');
    out=out.*(deltac.'); % deltac
end
if isfi(input)
    out=fi(out*sqrt(2*M),'numerictype', nT);

 else
    out=out*sqrt(2*M);
 end



% % Entradas:
% - 'in': Señal de entrada a filtrar, M muestras.
% - 'estat2': Variables de estado del banco de filtros.
% - 'buffer2': Buffer de señal de salida de M muestras
% - 'fpolifas2': Filtros de análisis G2M(-z^(-2))
% - 'deltac': Matriz parte de T
% - 'M': Número de subbandas
% - 'N': Longitud del filtro prototipo -1 --> N=(2KM+1), K even.
% Salidas:
% - 'out': Señal de salida subbandas, 1 muestra por subbanda.
% - 'estat2': Variables de estado del banco de filtros.
% - 'buffer2': Buffer de señal de salida de M muestras
function
[out,estat2,buffer2]=fi_sint2k1Mparcm_caso3(in,estat2,buffer2,fpolifas2,d
eltac,M,N,Wy);
multival=sqrt(2*M)*M;
out=(in.');%out=(in.')*multival;
out=out.*deltac; % deltac
% DCT4,
if isfi(in)
    nT=get(in,'numerictype');
    F= get(in,'fimath');
    outdouble= double(out);
    out1= fi(outdouble,1,Wy);
    n=length(out1);
    out=fidct4(out1.',n,nT);
    %reescalado
    out = fi (out,'WordLength', 16);
   else
out=fidct4(out.');
end
%out=dctiv(transpose(out));

% I
% outa=out;
outa=diag(ones(1,M))*out(1:M);
```

```
% -J
menosjota=-fliplr(diag(ones(1,M)));
outb=menosjota*out;

out=[outa outb];

% Filtrado
for f=1:2*M,
    fpolifas2(f).States = estat2(:,f);
   [out(f)]=filter((fpolifas2(f)),out(f));
     estat2(:,f) = fpolifas2(f).States;
end;
% %
% % Los dos siguientes pasos no es necesario hacerlo
% %
% %
% OUTA=fft(out(1:2*M),2*M);
% %
% %
% % A partir de aquí el estándar para 2M
% %
%  out=ifft(OUTA,2*M);



% Interpolación y Retardos
out=out(2*M:-1:1)*multival;%out=out(2*M:-1:1)
temp=out(M+1:M+M);
out=out(1:M)+buffer2;
buffer2=temp;
```

## Anexo V Función que implementa el banco de filtro de síntesis 'rápido' para M muestras.

```
% Función que implementa el bando de filtro de síntesis 'rápido' para M
muestras
% Entradas:
% - 'in': Señal de entrada a filtrar, M muestras.
% - 'estat2': Variables de estado del banco de filtros.
% - 'buffer2': Buffer de señal de salida de M muestras
% - 'fpolifas2': Filtros de análisis G2M(-z^(-2))
% - 'deltac': Matriz parte de T
% - 'M': Número de subbandas
% - 'N': Longitud del filtro prototipo -1 --> N=(2KM+1), K even.
% Salidas:
% - 'out': Señal de salida subbandas, 1 muestra por subbanda.
% - 'estat2': Variables de estado del banco de filtros.
% - 'buffer2': Buffer de señal de salida de M muestras
function [out,
estat2,buffer2]=fi_sint2k1Mparcm_caso3(in,estat2,buffer2,fpolifas2,deltac
,M,N,Wy);
multival=sqrt(2*M)*M;
out=(in.');%out=(in.')*multival;
```

```
out=out.*deltac; % deltac
% DCT4,
if isfi(in)
    nT=get(in,'numerictype');
    F= get(in,'fimath');
    outdouble= double(out);
    out1= fi(outdouble,1,Wy);
    n=length(out1);
    out=fidct4(out1.',n,nT);
    %reescalado
    out = fi (out,'WordLength', 16);
   else
out=fidct4(out.');
end
%out=dctiv(transpose(out));

% I
% outa=out;
outa=diag(ones(1,M))*out(1:M);

% -J
menosjota=-fliplr(diag(ones(1,M)));
outb=menosjota*out;

out=[outa outb];

% Filtrado
for f=1:2*M,
    fpolifas2(f).States = estat2(:,f);
   [out(f)]=filter((fpolifas2(f)),out(f));
     estat2(:,f) = fpolifas2(f).States;
end;
% %
% % Los dos siguientes pasos no es necesario hacerlo
% %
% %
% OUTA=fft(out(1:2*M),2*M);
% %
% %
% % A partir de aquí el estándar para 2M
% %
%  out=ifft(OUTA,2*M);



% Interpolación y Retardos
out=out(2*M:-1:1)*multival;%out=out(2*M:-1:1)
temp=out(M+1:M+M);
out=out(1:M)+buffer2;
buffer2=temp;
```