Decimadores CIC no Recursivos: Diseño e Implementación

por

Gerardo Molina Salgado

Tesis sometida como requisito parcial para obtener
el grado de

MAESTRO EN CIENCIAS EN LA ESPECIALIDAD DE ELECTRÓNICA

en el

Instituto Nacional de Astrofísica, Óptica y Electrónica

Julio 2011
Tonantzintla, Puebla
Supervisada por

Dra. Gordana Jovanovic Dolecek, INAOE
Dr. Alejandro Díaz Méndez, INAOE

Resumen

Esta tesis presenta dos propuestas para el diseño de decimadores CIC (Cascaded-Integrator-Comb) no recursivos, con bajo consumo de potencia y mejora de sus características en frecuencia.

Primero se presentan las bases de los sistemas *multi-razón* (múltiples frecuencias de muestreo), características de filtros CIC y revisión de algunos métodos para mejorar la atenuación en la banda de rechazo del filtro CIC.

Se propone una estructura que permite mejorar la atenuación en el primer doblez de la banda a costa de un ligero incremento en complejidad y consumo de potencia en comparación con el decimador CIC no recursivo. También, se propone una estructura para disminuir el consumo de potencia del decimador CIC no recursivo que mantiene un valor mínimo de atenuación en todos los dobleces de la banda.

Más adelante, se da una breve introducción a la implementación, la cual está enfocada en la implementación eficiente de los decimadores CIC no recursivos en forma directa y en componentes polifásicas. En base a esto se implementaron las estructuras propuestas y se hicieron simulaciones para verificar su funcionamiento lógico, consumo de potencia y área utilizada.

Finalmente, se hacen comparaciones entre las estructuras propuestas y los decimadores CIC no recursivos para verificar que satisfacen los criterios de diseño.

Abstract

This thesis presents design and implementation of non-recursive CIC decimation filters, with low power consumption and incremented attenuation in the first folding band, comparing with the CIC filter.

First, it is presented a general review of sampling rate conversion, the characteristics of CIC filter, and a brief review of methods for improving the attenuation in the stopband of CIC filter.

Two different designs have been proposed. The first design is a non-recursive CIC decimation filter structure that improves the alias rejection in the first folding band with a slight increase in the complexity and power consumption of the original non-recursive structure. The second design is a non-recursive CIC decimation filter structure that decreases the power consumption satisfying the minimum aliasing attenuation in all folding bands.

The brief introduction to the implementation, which is focused on the efficient implementation of non-recursive CIC decimation filter, in both direct and polyphase form is presented. The proposed decimation filters are implemented and simulated in order to verify their proper operations, the power consumption and the area.

Finally, the comparisons of CIC filters and the proposed structures are performed in order to present the benefits and the proper designs.

Agradecimientos

- En primer lugar doy gracias a Dios, por permitirme llegar hasta este momento de mi vida, por brindarme salud, fuerza y perseverancia para alcanzar mis metas.

- Al INAOE, por permitirme ingresar a sus instalaciones, por el apoyo y los servicios gratuitos que me proporcionó.

- Al Consejo Nacional de Ciencia y Tecnología (CONACYT), por haberme otorgado la beca de maestría.

- A mi asesora, la Dra. Gordana Jovanovic, por haber confiado en mí, por compartir su conocimiento conmigo y por propiciar mi superación académica.

- A mis sinodales, por sus favorables y valiosos comentarios, por sus sugerencias para mejorar mi trabajo de Tesis, por su disposición y ayuda proporcionada.

- A mi familia, mi mamá Isabel y mi hermano José Luis, por creer en mí, por apoyarme en todas mis decisiones y sobre todo por su invaluable amor y cariño.

- A mi novia, Ángeles, por su amor y apoyo incondicional que siempre me ha brindado.

Índice general

Prefacio

En los últimos años los decimadores CIC no recursivos han recibido una mayor atención en aplicaciones, para la reducción de la frecuencia de muestreo de una señal, donde se requiere de un bajo consumo de potencia y un diseño simple. Específicamente, en este trabajo se considerará la estructura donde el factor de sub-muestreo puede ser representado como $M = 2^P$, donde P es un número entero.

Esta estructura con $M = 2^P$ está reportada como la de menor consumo de potencia. Sin embargo, presenta los mismos problemas de respuesta en magnitud que la estructura CIC recursiva, como son *caída en la banda de paso* (CBP) y poca *atenuación en la banda de rechazo* (ABR).

Se han propuesto diversos métodos para mejorar estas características. No obstante, el factor común entre las propuestas es el aumento en la complejidad de diseño e implementación, y que no se han reportado trabajos para mejorar la estructura CIC no recursiva.

El objetivo de esta tesis consiste en mejorar la atenuación en la banda de rechazo del decimador CIC no recursivo con un mínimo incremento en la complejidad de diseño e implementación de la estructura con $M = 2^P$.

No se consideró la compensación para la caída en la banda de paso, debido a que esta se realiza fácilmente de forma externa mediante un filtro compensador que no representa un incremento en la complejidad del decimador CIC no recursivo y trabaja en baja frecuencia de muestreo.

Se considerarán dos criterios de diseño:

El primero, va enfocado a mejorar la atenuación en el primer doblez de la banda a costa de un ligero incremento en la complejidad de diseño e implementación.

El segundo, va a tratar de mantener el consumo de potencia tan bajo como sea posible cumpliendo con una atenuación mínima en todos los dobleces de la banda.

Para lograr esto, primero se estudió la conversión de la frecuencia de muestreo, tanto reducción como incremento, lo cual es presentado en el Capítulo 1. Después se estudiaron las características en frecuencia de los filtros CIC, así como también su implementación no recursiva y en componentes polifásicas, lo cual se presenta en el Capítulo 2. En el Capítulo 3 se puede encontrar una breve descripción de los diferentes métodos propuestos para mejorar las características en frecuencia de los filtros CIC.

Con base en los primeros tres capítulos, en el Capítulo 4, se propuso la estructura que permitiera cumplir con el objetivo de este trabajo y de forma matemática se calcularon sus parámetros de diseño para satisfacer los dos criterios de diseño mencionados anteriormente.

En los Capítulos 5 y 6 se implementaron los métodos de diseño propuestos. En el Capítulo 7 se hizo la simulación y comparación de los métodos propuestos para verificar que se cumplían los objetivos de diseño.

Al final, se presentan las conclusiones generales y se plantea el trabajo a futuro.

Capítulo 1

Conversión de la frecuencia de muestreo

En este primer capítulo se explicará la decimación, a través de sus características en tiempo y frecuencia. A partir de esto se presentarán los elementos necesarios para su implementación. Después, se mostrará su representación en el dominio de la transformada z y sus identidades principales. Como punto final de decimación se tratará su implementación polifásica. Más adelante, de una forma más breve, se explicará la interpolación mediante sus características en tiempo y frecuencia. Finalmente, se mostrará la forma de realizar la conversión de frecuencia de muestreo para valores fraccionarios.

1.1 Introducción

Existen aplicaciones donde una señal digital con cierta frecuencia de muestreo debe convertirse en una señal equivalente pero con una frecuencia de muestreo diferente, más alta o más baja. Una forma de hacer esto consiste en convertir la señal digital en analógica mediante un convertidor digital a analógico (DAC por sus siglas en inglés) y después regresarla al dominio digital con un convertidor analógico a digital (ADC) a la frecuencia deseada. Sin embargo al hacer esto la señal se degrada debido, principalmente, a la distorsión introducida por el DAC en la reconstrucción de la señal y el ruido de cuantización introducido por el ADC [1]. Una alternativa a este procedimiento es convertir la frecuencia de muestreo directamente en el dominio digital mediante decimación e interpolación.

1.2 Decimación

La reducción de la frecuencia de muestreo es llamada decimación, porque el conjunto original de muestras es reducido (decimado) [2]. La decimación consiste de dos etapas: *filtrado* y *sub-muestreo*, ilustrado por la Fig. 1.1. Primero se considerará el sub-muestreo de la señal de entrada y a través de su descripción se ilustrará por qué es necesario aplicar un filtrado previo.

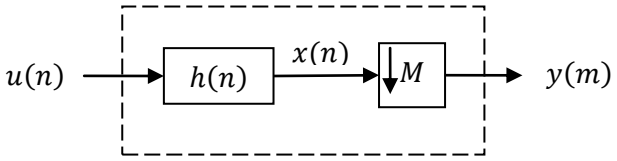


Fig. 1.1 Decimación.

1.2.1 Sub-muestreo

El *sub-muestreo* reduce la frecuencia de muestreo de entrada f_s por un valor entero M, el cual es conocido como *factor de sub-muestreo*. El bloque digital que realiza esta operación es llamado *Downsampler* (DS) y es presentado en la Fig. 1.2. La señal de salida $y(m)$ se obtiene a partir de solo aquellas muestras $x(n)$ que tienen un índice que es múltiplo entero de M y descartando todas las demás.

$$y(m) = x(mM). \tag{1.1}$$

Esta operación no es reversible debido a que las muestras que se establecen a cero no pueden ser recuperadas con su valor exacto, únicamente con un valor aproximado.

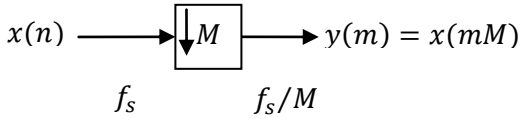


Fig. 1.2 Downsampler (DS).

Ejemplo 1.1: Considérese una señal $x(n)$ compuesta por tres tonos senoidales de frecuencia 1KHz, 2KHz y 3KHz, cuya frecuencia de muestreo es $f_s = 128KHz$, mostrada en (1.2) y la Fig. 1.3 (a).

$$x(n) = \sin\left(2\pi\frac{n}{128}\right) + 0.2 \times \sin\left(4\pi\frac{n}{128}\right) + \sin\left(6\pi\frac{n}{128}\right). \qquad (1.2)$$

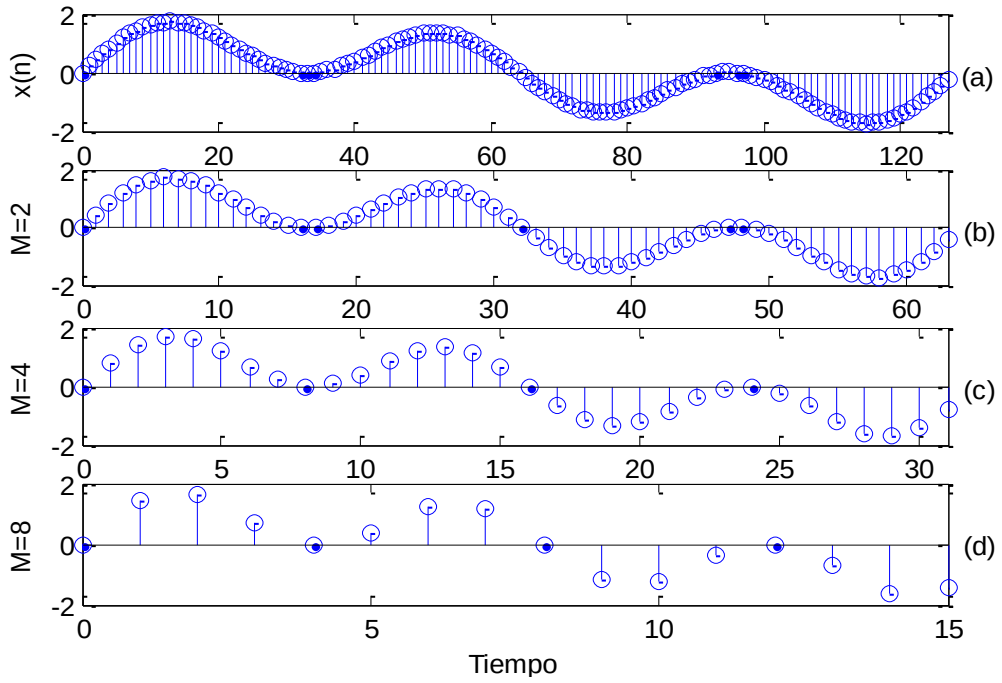


Fig. 1.3 (a) Señal original de (1.1), (b) señal sub-muestreada por $M = 2$, (c) $M = 4$, (d) $M = 8$.

Si a esta señal se hace pasar por el bloque de la Fig. 1.2 con $M = 2, 4, 8$ ¿Cómo serán las señales de salida $y(m)$?

Para $M = 2$ y siguiendo (1.1), $y_1(m) = x(2m)$, es decir se conservan solo aquellas muestras $x(n)$ que tienen un índice que es múltiplo de 2, (0 2, 4, 6…).

En la Fig. 1.3 (b) se presenta esta señal $y_1(m)$ la cual tiene una forma parecida a la señal original $x(n)$, pero con la mitad del número de muestras. Esto equivale a un periodo de muestreo dos veces mayor o una frecuencia de muestreo que es la mitad de la señal original (64KHz).

Para $M = 4$, $y_2(m) = x(4m)$, se conservan solo aquellas muestras $x(n)$ que tienen un índice que es múltiplo de 4, (0 4, 8, 12…).

En la Fig. 1.3 (c) se presenta $y_2(m)$, donde el periodo de muestreo aumenta por cuatro y la frecuencia se reduce en este mismo factor (32KHz).

Para $M = 8$, $y_3(m) = x(8m)$, se conservan solo aquellas muestras $x(n)$ que tienen un índice que es múltiplo de 8, (0 8, 16, 24…).

En la Fig. 1.3 (d) se presenta esta señal $y_3(m)$, donde la frecuencia de muestreo se reduce a 16KHz.

En este ejemplo la forma de las ondas de salida son muy parecidas a la señal original. Sin embargo, se puede ver que debe existir un límite en cuanto al factor de sub-muestreo M, ya que si un número elevado de muestras es descartado la señal después del sub-muestreo puede perder la forma de la señal original.

1.2.2 Descripción en frecuencia

En el dominio de la frecuencia es posible ver con mayor claridad los efectos del sub-muestreo de una señal y determinar el factor máximo de sub-muestreo antes que se pierda información de la señal original. Para esto el sub-muestreo es considerado en dos pasos. La salida del *primer paso* es la señal $x'(n)$, la cual se obtiene estableciendo todas las muestras con índices que no son múltiplos enteros de M a cero. En el *segundo paso* todos los ceros que fueron introducidos en el paso anterior son descartados.

Paso 1: En este paso se conservan todas las muestras que sean múltiplos enteros de M y se vuelven cero las demás. Otra forma de ver esto es como la multiplicación, en tiempo discreto, de la señal original de la entrada por una secuencia periódica $C_m(n)$

$$x'(n) = x(n) * C_M(n), \tag{1.3}$$

donde

$$C_M(n) = \begin{cases} 1 & n = mM; \\ 0 & otros\ casos \end{cases} \quad ; m = entero. \tag{1.4}$$

Este paso no cambia la frecuencia de muestreo porque las muestras que son establecidas a cero no son eliminadas en la representación de $x'(n)$.

La señal $C_m(n)$ tiene un periodo igual a M y puede ser representada mediante su serie de Fourier la cual consta de M componentes exponenciales complejas [3]

$$C_M(n) = \frac{1}{M}\sum_{k=0}^{M-1} C(k)e^{\frac{j2\pi kn}{M}}, \tag{1.5}$$

donde los coeficientes complejos $C(k)$ están definidos por

$$C(k) = \frac{1}{M}\sum_{k=0}^{M-1} C_m(n)e^{\frac{-j2\pi kn}{M}}. \tag{1.6}$$

Como $C_m(n) = 1$ para múltiplos de M, $C(k) = 1$ para todos los valores de k, por lo tanto

$$C_m(n) = \frac{1}{M}\sum_{k=0}^{M-1} e^{\frac{j2\pi kn}{M}}. \tag{1.7}$$

Sustituyendo (1.7) en (1.3) se obtiene:

$$x'(n) = x(n) * \frac{1}{M}\sum_{k=0}^{M-1} e^{\frac{j2\pi kn}{M}}. \tag{1.8}$$

Para obtener la representación en frecuencia del paso 1, se calcula la transformada de Fourier en tiempo discreto (DTFT por sus siglas en inglés) de (1.8) como:

$$X'\left(e^{j\omega}\right) = \sum_{n=-\infty}^{\infty} x'(n) * e^{-j\omega n} = \sum_{n=-\infty}^{\infty} x(n) * \frac{1}{M}\sum_{k=0}^{M-1} e^{\frac{j2\pi kn}{M}} * e^{-j\omega n}. \tag{1.9}$$

Rescribiendo (1.9) se obtiene:

$$X'\left(e^{j\omega}\right) = \frac{1}{M}\sum_{k=0}^{M-1}\sum_{n=-\infty}^{\infty} x(n) * e^{-jn\left(\omega-\frac{2\pi k}{M}\right)}. \tag{1.10}$$

$$X'\left(e^{j\omega}\right) = \frac{1}{M}\sum_{k=0}^{M-1} X\left(e^{-jn\left(\omega-\frac{2\pi k}{M}\right)}\right). \tag{1.11}$$

De (1.11) puede verse que el espectro de la señal que se obtiene en el paso 1 es el espectro de la señal $x(n)$ repetido cada $2\pi/M$, un total de $M-1$ veces en el intervalo de $[0, 2\pi]$, y que la magnitud se escala por el factor $1/M$.

Ejemplo 1.2: Considérese una señal $x(n)$ arbitraria, cuyo espectro es mostrado en la Fig. 1.4 (a). Si a esta señal arbitraria se le aplica un sub-muestreo por $M = 2, 4\ y\ 8$. ¿Cómo serán los espectros $X'(e^{j\omega})$ al aplicar el paso 1?

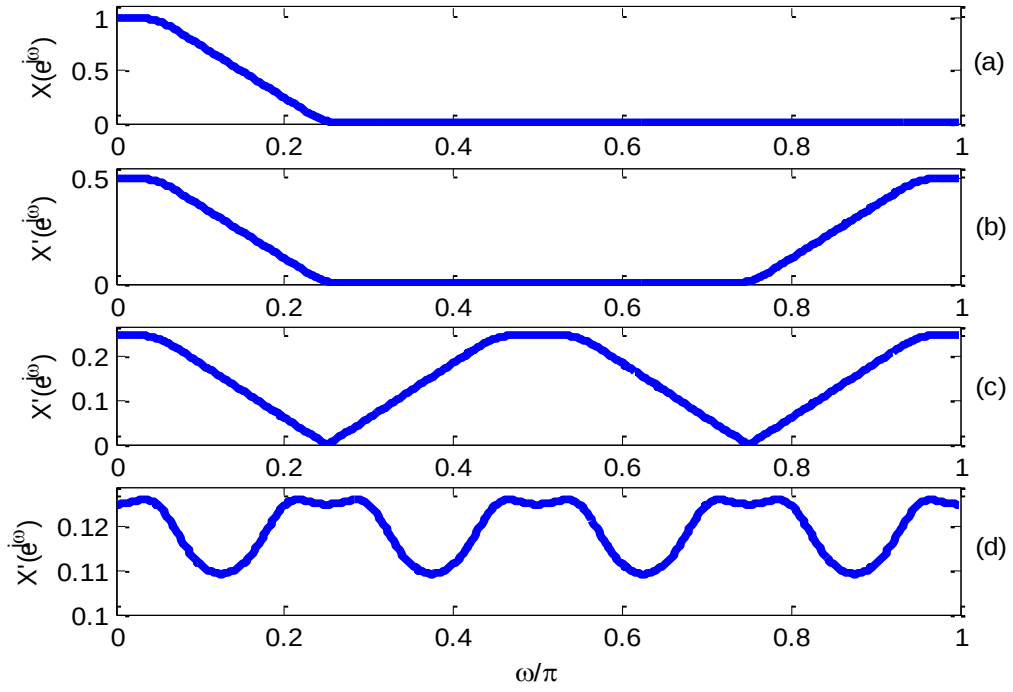


Fig. 1.4 Espectro del Ejemplo 1.2 (a) original, (b) después del paso 1 con $M = 2$, (c) $M = 4$, (d) $M = 8$.

Con el factor de sub-muestreo $M = 2$, en el espectro original de la Fig. 1.4 (a) se introduce una réplica de $X(e^{j\omega})$ en $2\pi/2$ y la amplitud se escala por 0.5, como es mostrado en la Fig. 1.4 (b).

Para $M = 4$, se introduce una réplica de $X(e^{j\omega})$ cada $2\pi/4$, es decir en 0.5π , π y 1.5π, y además la amplitud se escala por 0.25 (Fig. 1.4 (c)).

Finalmente, para $M = 8$, se introduce una réplica de $X(e^{j\omega})$ cada $2\pi/8$,es decir en 0.25π, 0.5π, 0.75π, 1π, 1.25π, 1.5π y 1.75π, y la amplitud se escala por 0.125, (Fig. 1.4 (d)).

Paso 2: Este paso consiste en eliminar las muestras de la señal $x'(n)$ que son iguales a cero para obtener la señal $y(m)$. Esta operación no cambia el contenido de $x'(n)$ y simplemente introduce escalamiento en el tiempo por un factor de $1/M$. Las señales sub-

muestreadas se representan con un menor número de muestras, por lo que sufren una compresión en el tiempo, lo que equivale a una expansión en frecuencia que provoca que el espectro $X'\left(e^{j\omega/M}\right)$ se convierta en $Y\left(e^{j\omega}\right)$. Esto es mostrado a continuación.

$$X'\left(e^{j\omega/M}\right) = \sum_{n=-\infty}^{\infty} x'(n) * e^{-j\omega n/M}. \tag{1.12}$$

Dado que la secuencia $x'(n)$ solo tiene valores diferentes de cero en $n = mM$, se puede rescribir (1.12) como:

$$X'\left(e^{j\omega/M}\right) = \sum_{m=-\infty}^{\infty} x'(mM) * e^{-j\omega mM/M} = \sum_{m=-\infty}^{\infty} x'(mM) * e^{-j\omega m}. \tag{1.13}$$

Substituyendo (1.1) en (1.13) y usando la DTFT para $y(m)$ se obtiene

$$X'\left(e^{j\omega/M}\right) = \sum_{m=-\infty}^{\infty} y(m) * e^{-j\omega m} = Y\left(e^{j\omega}\right). \tag{1.14}$$

Usando (1.11) se puede rescribir (1.14) como

$$Y\left(e^{j\omega M}\right) = X'\left(e^{j\omega}\right) = \frac{1}{M}\sum_{k=0}^{M-1} X\left(e^{-jn\left(\omega-\frac{2\pi k}{M}\right)}\right). \tag{1.15}$$

De (1.15) se puede ver que el espectro que se obtiene en el paso 2 es el espectro del paso 1 ampliado por el factor M.

Ejemplo 1.3: Aquí se determina el espectro del Ejemplo 1.2 al aplicar el paso 2.

Con el factor de sub-muestreo $M = 2$, el espectro de la Fig. 1.4 (b) se expande por un factor de 2. Por ejemplo los puntos 0.1 y 0.25 se convierten en 0.2 y 0.5 respectivamente, tal y como lo muestra la Fig. 1.5 (b), esto sucede debido a que el espectro del paso 1 de la Fig. 1.4 (b) sufre una expansión por 2.

Con el factor de sub-muestreo $M = 4$, el espectro de la Fig. 1.4 (c) se expande por un factor de 4 y los puntos 0.1 y 0.25 se convierten en 0.4 y 1 respectivamente, según la Fig. 1.5 (c).

Así mismo, con $M = 8$, el espectro de la Fig. 1.4 (d) se expande por un factor de 8 y se obtiene el de la Fig. 1.5 (d), donde los puntos 0.1 y 0.25 se convierten en 0.8 y 2 respectivamente.

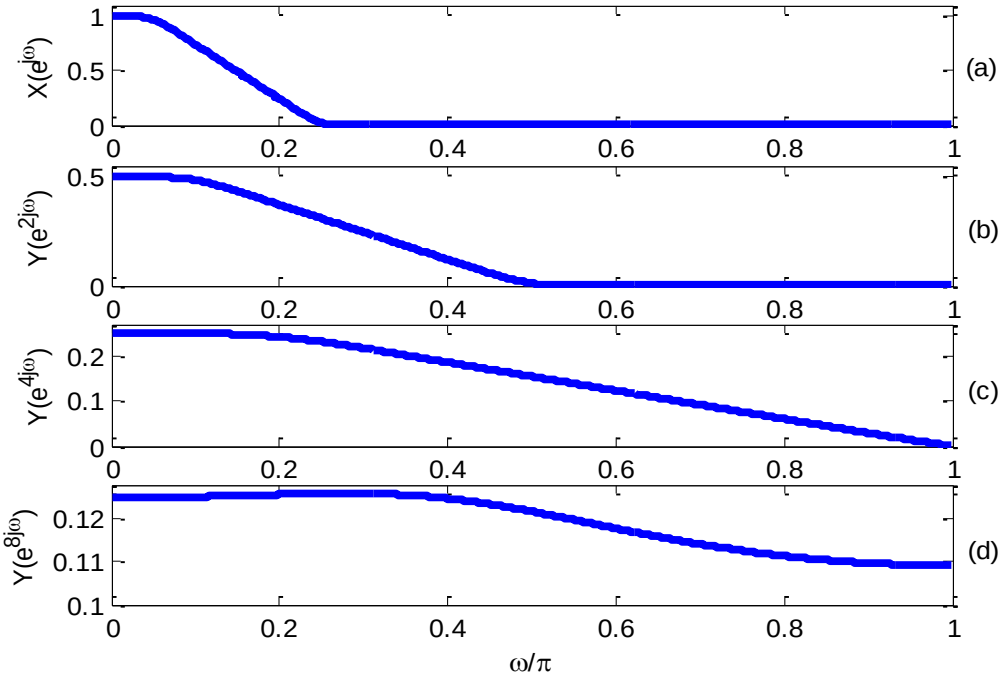


Fig. 1.5 Espectro del Ejemplo 1.3 (a) original, (b) después del paso 2 con $M = 2$, (c) $M = 4$, (d) $M = 8$.

1.2.3 Aliasing

En el Ejemplo 1.2 con el último factor de sub-muestreo ($M = 8$, Fig. 1.4 (d)), se puede ver que el espectro original sufre una distorsión debido a la réplica que se introduce en 0.25π. Este efecto indeseable se conoce como distorsión por *aliasing*. Para evitarlo es necesario el uso de un filtro digital pasa-bajas antes de que la señal sea pasada por el DS (Fig. 1.1). Este se conoce como *filtro antialiasing* y debe cumplir con las siguientes características [1];

$$\left|H\left(e^{j\omega}\right)\right| = \begin{cases} 1 & |\omega| \leq \omega_c/M \\ 0 & \pi/M \leq |\omega| \leq \pi \end{cases}, \tag{1.16}$$

donde ω_c es la máxima frecuencia en $x(n)$ que se necesita preservar. Para evitar por completo la distorsión por aliasing es necesario que la banda de rechazo inicie exactamente en π/M. Sin embargo, para ω_c cercanas a π/M la banda de transición del filtro será muy angosta, elevando su orden. Dependiendo de la aplicación algunas veces es tolerada cierta

cantidad de aliasing por lo que la banda de rechazo puede ser algo mayor que π/M y con esto se permiten relajar las especificaciones del filtro antialiasing.

Para el Ejemplo 1.2, con $M = 8$, si se quiere evitar por completo la distorsión por aliasing se debe filtrar la señal original $x(n)$, antes que ingrese al DS, mediante un filtro antialiasing con:

$$\omega_p \leq \omega_c/8 \tag{1.17}$$

$$\pi/8 \leq \omega_s \leq \pi. \tag{1.18}$$

Por tal motivo es necesario usar un filtro digital previo al sub-muestreo que permita limitar la banda de la señal de entrada para que la señal no sufra distorsión después de la reducción de la frecuencia de muestreo.

1.2.4 Representación en el dominio de la transformada z

Algunas veces es más conveniente representar las señales después del sub-muestreo, mediante la transformada z, la cual para una secuencia $x(n)$ está definida como

$$X(z) = \sum_{n=-\infty}^{\infty} x(n)z^{-n}, \tag{1.19}$$

donde z es una variable compleja determinada por $z = re^{j\omega}$.

Cuando la transformada de Fourier existe, la DTFT es simplemente

$$X(z)|_{z=e^{j\omega}} = X(e^{j\omega}). \tag{1.20}$$

De esta manera se puede expresar la ecuación (1.15) como;

$$Y(z^M) = X'(z) = \frac{1}{M}\sum_{k=0}^{M-1} X\left(ze^{-j2\pi k/M}\right) = \frac{1}{M}\sum_{k=0}^{M-1} X\left(zW_M^k\right), \tag{1.21}$$

donde

$$W_M^k = e^{-j2\pi k/M}, \quad k = 0, \ldots, M-1. \tag{1.22}$$

1.2.5 Propiedades del DS

El sub-muestreo de una señal, llevado a cabo por el DS, es una operación *lineal* y *variante en el tiempo*. Es lineal debido a que cumple con *el principio de superposición*. Es decir, el sub-muestreo de una señal que es la suma de dos o más señales escaladas es igual a la suma escalada de esas dos o más señales sub-muestreadas individualmente.

Para examinar su comportamiento variante en el tiempo se considera lo siguiente. En general si el retardo *D* de la señal de entrada resulta en el mismo retardo *D* de la señal de salida, la operación es invariante en el tiempo, de lo contrario es variante en el tiempo. De acuerdo con la ecuación (1.1) si la señal de entrada es retrasada por *D*,

$$x(mM - D), \tag{1.23}$$

la señal después del sub-muestreo será

$$y\left(\frac{mM - D}{M}\right) = y\left(m - \frac{D}{M}\right) \neq y(m - D). \tag{1.24}$$

Como se puede ver el retardo en la señal de entrada no resulta en el mismo retardo de la señal después del sub-muestreo, y se puede concluir que la operación de sub-muestreo (aquella que realiza el DS) es una operación variante en el tiempo. Cuando *D/M* no es un entero, la señal después del sub-muestreo tendrá un retardo fraccional como lo indica (1.24). Esto significa que la señal de salida tendrá una forma diferente a la señal original después del sub-muestreo.

Considérese el caso donde el retardo es múltiplo del factor de sub-muestreo *M*, es decir *D=kM*. A partir de (1.24) se puede ver que la señal después del sub-muestreo será

$$y\left(m - \frac{kM}{M}\right) = y(m - k). \tag{1.25}$$

En este caso si la señal de entrada es retrasada por *kM* muestras la señal resultante será una versión retrasada por *k* muestras de la señal después del sub-muestreo original. Esto significa que si el retardo en la señal de entrada es un múltiplo entero del factor de sub-

muestreo *M*, la correspondiente señal después del sub-muestreo tendrá la misma forma que la señal obtenida de la señal de entrada sin retardo.

1.2.6 Identidades multi-razón

Las *Identidades multi-razón* resumen las propiedades más importantes asociadas con el sub-muestreo de una señal. Estas sirven para resolver más fácilmente un sistema que involucra múltiples razones de muestreo.

La *Primera identidad* (Fig. 1.6), resume la propiedad de linealidad y establece que: el sub-muestreo de una señal que es la suma de dos o más entradas escaladas (Fig. 1.6 (a)) es equivalente a la suma escalada de esas dos o más señales sub-muestreadas individualmente (Fig. 1.6 (b)).

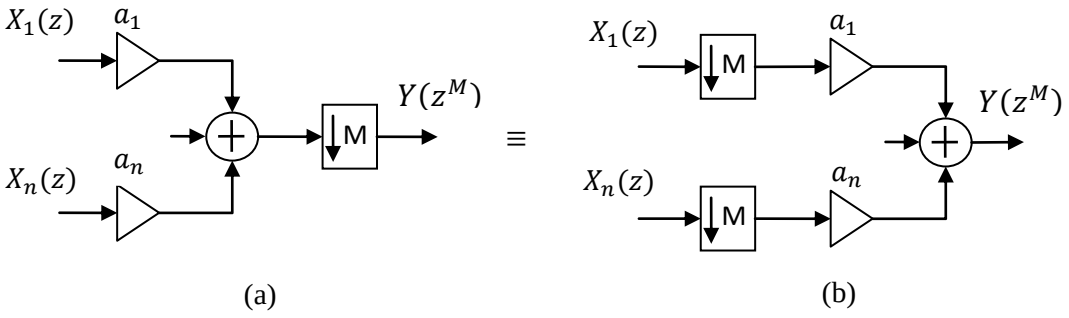


Fig. 1.6 Primera identidad multi-razón.

La *Segunda identidad* establece que un retardo por M muestras antes del DS es equivalente al retardo en una sola muestra después del DS (Fig. 1.7).

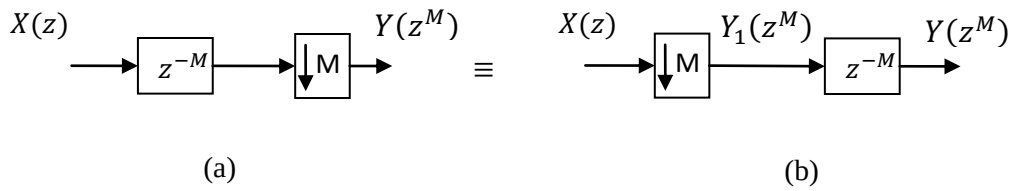


Fig. 1.7 Segunda identidad multi-razón.

Para verificar esto se calcula la salida de la estructura de la Fig. 1.7 (a) como

$$Y(z^M) = \frac{1}{M}\sum_{k=0}^{M-1} X\left(zW_M^k\right)\left(zW_M^k\right)^{-M}. \tag{1.26}$$

$$W_M^{-kM} = e^{-j2\pi k} = 1 \;\forall\; k. \tag{1.27}$$

$$Y(z^M) = \frac{z^{-M}}{M}\sum_{k=0}^{M-1} X\left(zW_M^k\right). \tag{1.28}$$

En la estructura de la Fig. 1.7 (b) la salida se determina como:

$$Y_1(z^M) = \frac{1}{M}\sum_{k=0}^{M-1} X\left(zW_M^k\right). \tag{1.29}$$

$$Y(z^M) = (z^M)^{-1}Y_1(z^M) = \frac{z^{-M}}{M}\sum_{k=0}^{M-1} X\left(zW_M^k\right). \tag{1.30}$$

Se puede ver que tanto (1.28), como (1.30) son iguales por lo que las estructuras de la Fig. 1.7 (a) y (b) son equivalentes.

La *Tercera identidad* es una generalización de la anterior, donde la función de filtrado por un filtro expandido $G(z^M)$ antes del DS es equivalente a tener primero el DS y después el filtro original $G(z)$. El filtro expandido $G(z^M)$ se obtiene reemplazando cada retardo z^{-1} por z^{-M} en el filtro original $G(z)$.

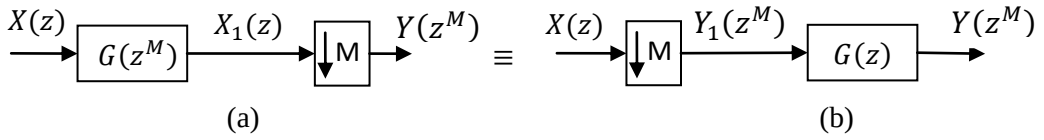


Fig. 1.8 Tercera identidad multi-razón.

Para verificar esto primero se calcula la salida de la estructura de la Fig. 1.8 (a) como:

$$Y(z^M) = \frac{1}{M}\sum_{k=0}^{M-1} X_1\left(zW_M^k\right) = \frac{1}{M}\sum_{k=0}^{M-1} X\left(zW_M^k\right)G\left(z^M W_M^{kM}\right)$$

$$= \frac{1}{M}\sum_{k=0}^{M-1} X\left(zW_M^k\right)G(z^M). \quad (1.31)$$

La salida para la estructura de la Fig. 1.8 (b) es:

$$Y_1(z^M) = \frac{1}{M}\sum_{k=0}^{M-1} X\left(zW_M^k\right). \quad (1.32)$$

$$Y(z^M) = Y_1(z^M)G(z^M) = \frac{1}{M}\sum_{k=0}^{M-1} X\left(zW_M^k\right) = \frac{1}{M}\sum_{k=0}^{M-1} X\left(zW_M^k\right)G(z^M) \quad (1.33)$$

Una vez más puede verse que (1.31) y (1.33) son equivalentes.

1.2.7 Decimación polifásica

En las secciones pasadas se pudo ver que en la estructura de un decimador el filtro antialiasing trabaja con la frecuencia de muestreo de la señal de entrada. Sin embargo, algunos términos de la convolución del filtro con la señal de entrada son descartados después del sub-muestreo por lo que son calculados innecesariamente. Por lo tanto una estructura más eficiente es aquella donde el filtrado es llevado a cabo a la frecuencia de muestreo más baja. Para hacer esto es necesario analizar el filtro FIR a través de sus *componentes polifásicas*.

1.2.7.1 Filtros polifásicos

Si el filtro FIR tiene N coeficientes, donde N es un múltiplo entero de M, se pueden obtener M diferentes componentes muestreadas discretamente de la respuesta al impulso. Esto significa que la transformada z correspondiente puede ser dividida en M sub-señales. Por ejemplo para un filtro con $N = 12$ y $M = 4$ se puede escribir su función de transferencia como:

$$H(z)\sum_{n=0}^{N-1} h(n)z^{-n} = h(0)z^0 + h(4)z^{-4} + h(8)z^{-8}$$

$$+h(1)z^{-1} + h(5)z^{-5} + h(9)z^{-9}$$

$$+h(2)z^{-2} + h(6)z^{-6} + h(10)z^{-10}$$

$$+h(3)z^{-3} + h(7)z^{-7} + h(11)z^{-11}$$

$$= z^{-0}[h(0)z^{0} + h(4)z^{-4} + h(8)z^{-8}]$$

$$+z^{-1}[h(1)z^{0} + h(5)z^{-4} + h(9)z^{-8}]$$

$$+z^{-2}[h(2)z^{0} + h(6)z^{-4} + h(10)z^{-8}]$$

$$+z^{-3}[h(3)z^{0} + h(7)z^{-4} + h(11)z^{-8}]. \qquad (1.34)$$

La ecuación (1.34) puede ser reescrita como

$$H(z) = \sum_{K=0}^{3} z^{-k}\, H_K(z^4), \qquad (1.35)$$

donde

$$H_K(z^4) = \sum_{n=0}^{2} h(4n + k)(z^4)^{-n}\,; \qquad k = 0,1,2,3. \qquad (1.36)$$

La expresión (1.35) se puede generalizar para obtener

$$H(z) = \sum_{K=0}^{M-1} z^{-k}\, H_K(z^M), \qquad (1.37)$$

donde

$$H_K(z^M) = \sum_{n=0}^{N/M-1} h(nM + k)(z^M)^{-n}\,; \qquad k = 0 \ldots M - 1. \qquad (1.38)$$

La implementación de $H(z)$ basada en la descomposición de (1.37) es llamada *implementación polifásica* y constituye la base para la decimación polifásica. A partir de (1.34) y (1.38) se puede ver que las componentes polifásicas se obtienen por el corrimiento a la izquierda de la respuesta al impulso por k muestras, seguidas por el muestreo discreto con periodo fundamental de M (como el paso 1 de sub-muestreo). La componente cero

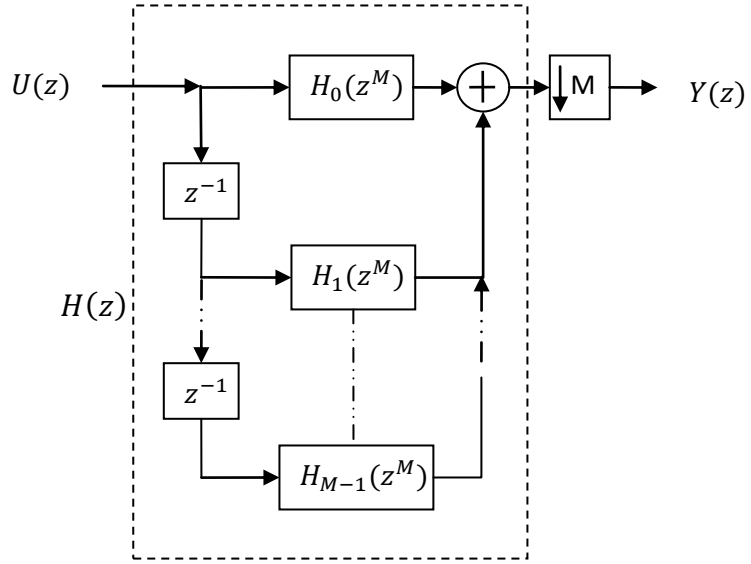
corresponde al corrimiento por $k = 0$, la primer componente por $k = 1$ y la $(M-1)$-ésima componente a $k = M - 1$.

Eliminando los ceros, introducidos por el muestreo discreto, de (1.38) (aplicando el paso 2 de sub-muestreo) se obtiene:

$$H_K(z) = \sum_{n=0}^{N/M-1} h(nM + k)z^{-n}; \qquad k = 0 \dots M - 1. \tag{1.39}$$

Se puede notar que las componentes polifásicas de (1.39) son simplemente las componentes polifásicas expandidas dadas en (1.38). Por lo tanto, esta última representación puede ser usada en la aplicación de la Tercera identidad multi-razón.

Utilizando la relación en (1.37) el decimador de la Fig. 1.1 puede representarse como el de la Fig. 1.9 (a). Usando la Primera y Tercera identidad multi-razón se obtiene una estructura más eficiente en la cual tanto el número de operaciones como la cantidad de memoria requerida se reduce en un factor de M. Esta estructura es mostrada en la Fig. 1.9 (b). Se puede ver que los sub-filtros son las componentes polifásicas del filtro antialiasing. También que las operaciones del filtro se realizan a una frecuencia de muestreo M veces menor que la de la señal de entrada.



(a)

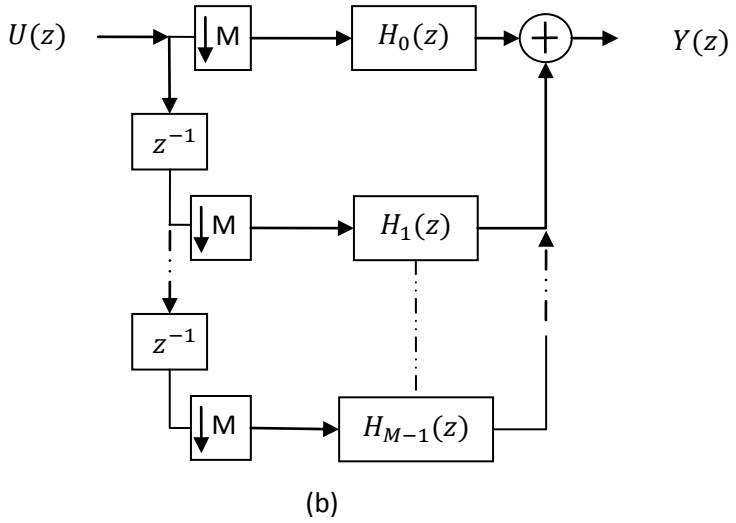


Fig. 1.9 Decimación polifásica.

De la Fig. 1.9 (b) se puede notar que la cadena de retardos y DS en conjunto tienen un efecto de distribuir muestras consecutivas entre los sub-filtros. Debido a esto, la estructura generalmente se representa usando un conmutador a la entrada tal y como se muestra en la Fig. 1.10.

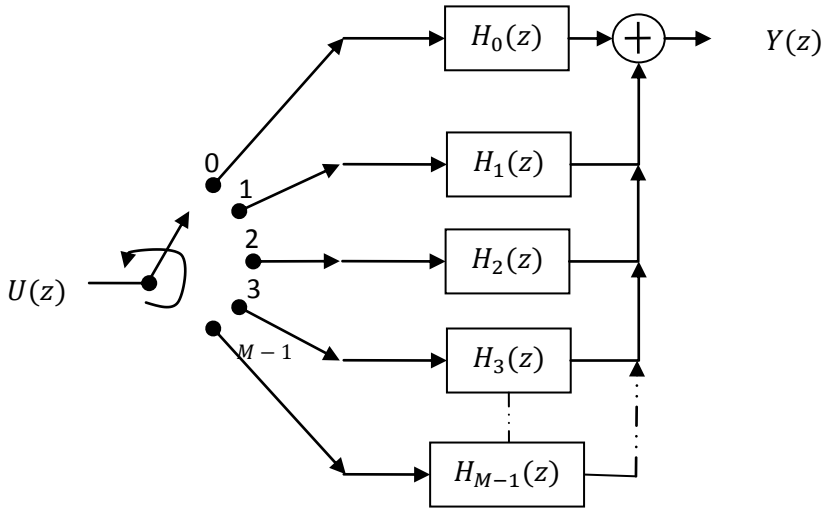


Fig. 1.10 Decimación polifásica con conmutador a la entrada.

1.3 Interpolación

Este es el procedimiento opuesto a la decimación y consiste en incrementar la frecuencia de muestreo de una señal. De forma similar a la decimación, consiste de dos etapas: *sobre-muestreo* y filtrado (mostrado en la Fig. 1.11).

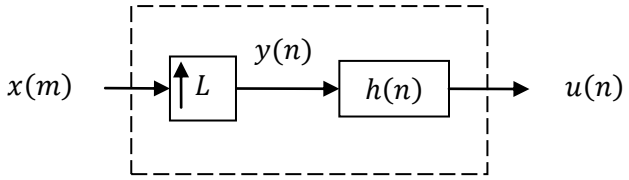


Fig. 1.11 Interpolación.

1.3.1 Sobre-muestreo

El *sobre-muestreo* incrementa la frecuencia de muestreo por un valor entero L, al introducir $L-1$ igualmente espaciados ceros entre cada par de muestras,

$$y(n) = \begin{cases} x(n/L) & para\ n = mL \\ 0 & otros\ casos \end{cases} \tag{1.40}$$

El símbolo para este elemento es presentado en la Fig. 1.12 y es conocido como *Upsampler* (US). Se puede ver que la frecuencia de muestreo de la señal de salida se ve incrementada por el factor de sobre-muestreo L.

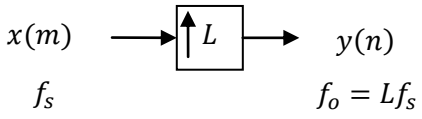


Fig. 1.12 Upsampler.

El sobre-muestreo no cambia el contenido de la señal de entrada, y solo introduce escalamiento en el tiempo por el factor L. El eje del tiempo es dividido por el factor L (se contrae). En el dominio de la frecuencia sucede lo contrario y se expande por el mismo factor. Para verificar esto se analiza su comportamiento en frecuencia a través de la DTFT. Usando (1.40) la transformada de Fourier es

$$Y(e^{j\omega}) = \sum_{n=-\infty}^{\infty} y(n)e^{-j\omega n} = \sum_{n=-\infty}^{\infty} x(n/L)e^{-j\omega nL/L}$$

$$= \sum_{n=-\infty}^{\infty} x(n/L)e^{-j\omega L(n/L)} = X(e^{j\omega L}). \quad (1.41)$$

A partir de (1.41) se puede ver que la señal no sufre escalamiento en magnitud como en el caso del sub-muestreo. Por otra parte, una frecuencia ω en $X(e^{j\omega})$ es transformada en una nueva frecuencia ω/L en $Y(e^{j\omega})$. Como resultado de esta operación se introducen $L-1$ imágenes indeseadas del espectro original en el intervalo de [0, 2π].

1.3.2 Imágenes

El sobre-muestreo introduce imágenes del espectro principal cada $2\pi/L$. Este fenómeno es llamado *imaging* ya que hay $L-1$ imágenes en el intervalo [0, 2π]. Estas imágenes deben ser eliminadas para lo cual se requiere de un filtro digital pasa-bajas inmediatamente después del US. Este filtro es llamado *filtro anti-imágenes* o *filtro de interpolación*. En el dominio del tiempo su efecto es que las muestras con valor cero, introducidas por el sobre-muestreo, son sustituidas por valores interpolados. Las especificaciones para este filtro están dadas por [1].

$$|H(e^{j\omega})| = \begin{cases} L & |\omega| \le \omega_c/L \\ 0 & \pi/L \le |\omega| \le \pi \end{cases}, \quad (1.42)$$

donde ω_c es la frecuencia más alta que se necesita preservar en la señal interpolada.

1.4 Cambio de la frecuencia de muestreo por un valor fraccional

Como se presentó anteriormente, la decimación permite reducir la frecuencia de muestreo por un múltiplo entero M, mientras la interpolación permite aumentar la frecuencia de muestreo por un valor entero L. Poniendo en cascada un decimador y un interpolador se puede obtener un cambio en la frecuencia de muestreo por un valor fraccional L/M.

Hay dos posibles conexiones dependiendo que se realice primero, decimación o interpolación, como se puede ver en las Fig. 1.13 (a) y (b). En la estructura donde se realiza primero la interpolación se puede ver que los filtros digitales antialiasing y anti-imágenes operan a la misma frecuencia de muestreo. Por lo tanto, pueden ser combinados en un solo filtro $G(z)$ como se muestra en la Fig. 1.13 (c). Este filtro tiene una frecuencia de corte normalizada en:

$$\omega_s = \min(\pi/L, \pi/M), \tag{1.43}$$

el cual elimina las imágenes causadas por el sobre-muestreo y al mismo tiempo evita la distorsión por aliasing causada por el sub-muestreo. Finalmente, con esta estructura es posible obtener cambios de frecuencia de muestreo fraccionarios. Por ejemplo para una señal muestreada a 48KHz que se desea reducir a 32KHz, se debe usar la estructura de la Fig. 1.13 (c) con $L = 2$ y $M = 3$, es decir, $L/M = 2/3$.

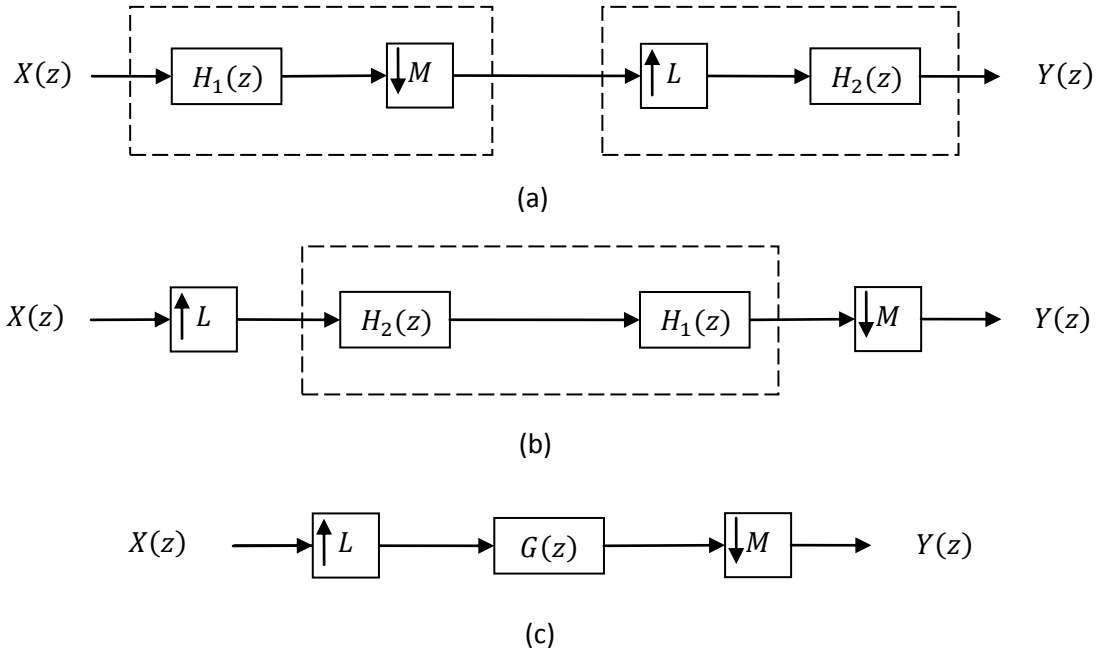


Fig. 1.13 Cambio de frecuencia de muestreo por un valor fraccionario.

Capítulo 2

Filtros CIC

En este capítulo se presentará el filtro CIC (Cascaded-Integrator-Comb), su función de transferencia, respuesta en frecuencia e implementación. Después se describirán brevemente los parámetros principales a considerar cuando es utilizado como filtro antialiasing para construir decimadores. Más adelante se presentará el filtro CIC no recursivo, su función de transferencia, ventajas frente a la representación recursiva e implementación. Por último se presentará la implementación en componentes polifásicas para el filtro CIC no recursivo y su estructura eficiente como parte de un decimador.

2.1 El filtro CIC

Este filtro es llamado de esta manera porque está formado por una etapa *integradora* en cascada con una etapa *comb* (vea esta estructura más adelante). Su propiedad de realizar filtrado sin multiplicadores los hace muy atractivos para trabajar con señales con frecuencias de muestreo muy altas, como por ejemplo en convertidores analógico a digital sigma-delta, donde la señal a convertir se muestrea a una frecuencia mucho mayor que la frecuencia de Nyquist [4].

El filtro CIC fue introducido por E. Hogenauer en 1981 [5], se deriva a partir del filtro MA (Moving Average), cuya ecuación de diferencias es;

$$y(n) = \frac{x(n) + x(n-1) + x(n-2) \ldots + x\big(n-(M-1)\big)}{M}, \qquad (2.1)$$

donde M es el orden del filtro y, como se verá más adelante, es igual al factor de sub-muestreo.

Su respuesta al impulso está determinada por

$$h[n] = \begin{cases} 1/M, & 0 \le n \le M-1 \\ 0, & otros\ casos \end{cases}. \tag{2.2}$$

La representación de este filtro en el dominio de la transformada z es

$$H(z) = \frac{1}{M}\left[z^0 + z^{-1} + z^{-2} \ldots + z^{-(M-1)}\right]. \tag{2.3}$$

El filtro de (2.3) puede implementarse mediante $M-1$ sumadores. Esta se llama forma no recursiva. Sin embargo, la ecuación (2.3) también se puede representar mediante la forma compacta de (2.4).

$$H(z) = \frac{1}{M}\sum_{k=0}^{M-1} z^{-k} = H(z) = \frac{1}{M}\frac{1-z^{-M}}{1-z^{-1}}. \tag{2.4}$$

La ecuación (2.4) es llamada forma recursiva y solo requiere de dos sumadores para su implementación.

2.1.1 Respuesta en magnitud

La respuesta en frecuencia se determina sustituyendo $z = e^{j\omega}$ en (2.4):

$$H\left(e^{j\omega}\right) = \frac{1}{M}\frac{1-e^{-j\omega M}}{1-e^{-j\omega}} = \frac{1}{M}\left[\frac{e^{-j\omega M/2}\left(e^{j\omega M/2} - e^{-j\omega M/2}\right)}{e^{-j\omega/2}\left(e^{j\omega/2} - e^{-j\omega/2}\right)}\right]. \tag{2.5}$$

Usando la identidad de Euler, $2jsin(x) = e^{jx} - e^{-jx}$, (2.5) se convierte en

$$H\left(e^{j\omega}\right) = \frac{1}{M}\left[\frac{e^{-j\omega M/2}}{e^{-j\omega/2}}\frac{2jsin(j\omega M/2)}{2jsin(j\omega/2)}\right] = \frac{1}{M}\frac{\sin\left(\frac{\omega M}{2}\right)}{\sin\left(\frac{\omega}{2}\right)}e^{j\omega[(M-1)/2]}. \tag{2.6}$$

De (2.6) puede verse que la respuesta en magnitud exhibe una respuesta $\sin(Mx)/\sin(x)$, es decir una característica $\mathrm{sinc}(Mx)$ y por esta razón a veces es llamado *sinc filter*. El término exponencial representa la fase del filtro la cual se tratará más adelante. Para poder

apreciar gráficamente la respuesta en frecuencia de este filtro consideremos, como ejemplo, un filtro CIC con $M = 10$. Su función de transferencia a partir de (2.4) es:

$$H(z) = \left(\frac{1}{10}\right)\frac{1 - z^{-10}}{1 - z^{-1}}, \tag{2.7}$$

y su correspondiente respuesta en magnitud es:

$$\left|H\left(e^{j\omega}\right)\right| = \left|\left(\frac{1}{10}\right)\frac{\sin\left(5\omega\right)}{\sin\left(\frac{\omega}{2}\right)}\right|. \tag{2.8}$$

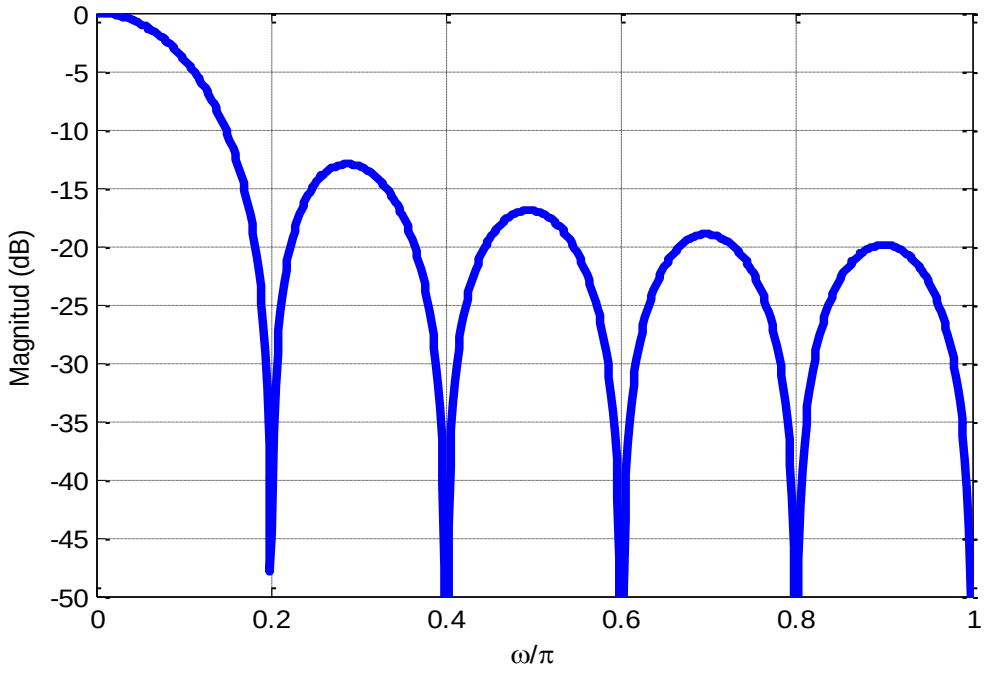


Fig. 2.1 Respuesta en magnitud para el filtro CIC con $M = 10$.

La gráfica para (2.8) se muestra en la Fig. 2.1 donde puede verse que la respuesta en magnitud tiene ceros en múltiplos enteros de las frecuencias $\omega = 2\pi/M$, los cuales generan los llamados dobleces de la banda. En el intervalo de $[0 - \pi]$ se encuentran $M/2$ ceros distribuidos en múltiplos enteros de $2/M$ para M par y $(M - 1)/2$ para M impar. En este ejemplo se encuentran 5 ceros en múltiplos de 0.2 en el intervalo $[0 - \pi]$, y por consiguiente la misma cantidad de dobleces en la banda. La repuesta en frecuencia presenta una característica pasa-baja, sin embargo tiene una banda de transición muy grande y una atenuación muy pequeña de aproximadamente 13dB´s en el primer lóbulo de la banda de

rechazo, lo cual no lo hace muy selectivo para su uso. Se pueden poner K filtros CIC en cascada con el objetivo de aumentar la atenuación en la banda de rechazo. Así (2.4) y (2.6) se convierten en (2.9) y (2.10), respectivamente.

$$H(z) = \left[\frac{1}{M}\frac{1-z^{-M}}{1-z^{-1}}\right]^K. \tag{2.9}$$

$$\left|H\left(e^{j\omega}\right)\right| = \left|\frac{1}{M^K}\left[\frac{\sin\left(\frac{\omega M}{2}\right)}{\sin\left(\frac{\omega}{2}\right)}\right]^K\right|. \tag{2.10}$$

Continuando con el ejemplo para $M = 10$, en la Fig. 2.2 puede verse la respuesta en magnitud para valores de K desde 1 hasta 4. Conforme este valor aumenta, la atenuación en los dobleces de la banda se incrementa, debido a que los ceros que los generan poseen una multiplicidad K (el polo de la parte recursiva presenta la misma multiplicidad). Por otra parte, existe una *caída en la banda de paso* (CBP) que provoca una inevitable distorsión de la magnitud. Esta caída aumenta en función del valor de la cascada K.

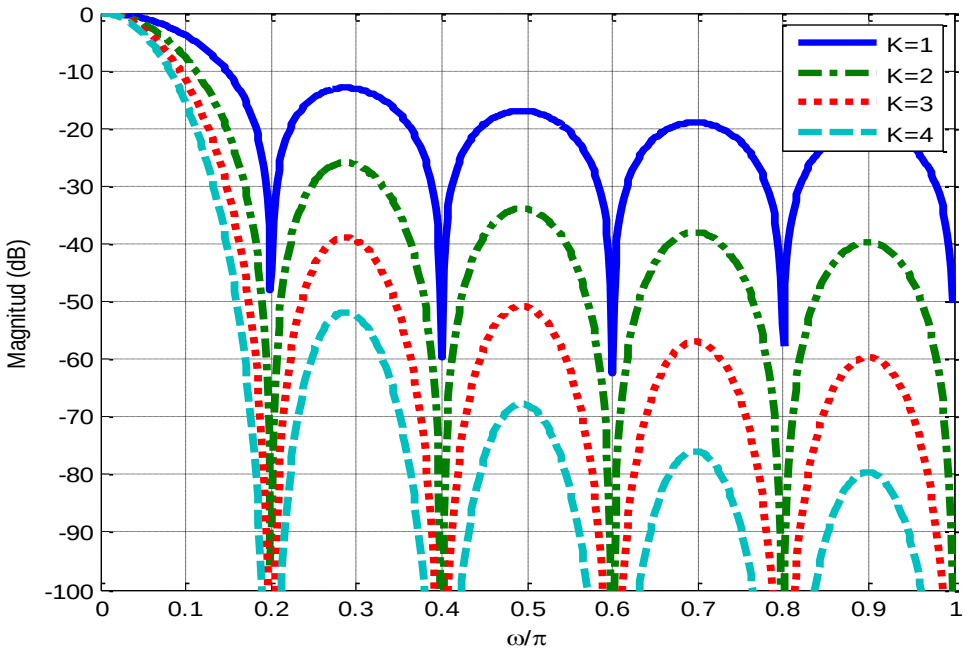


Fig. 2.2 Respuesta en magnitud para el filtro CIC con $M = 10$ y diferentes valores de K.

2.1.2 Respuesta en fase

Retomando la respuesta en frecuencia del filtro CIC en (2.6), el término $e^{j\omega[(M-1)/2]}$ representa la fase del filtro. Esta es lineal y el retardo de grupo constante

$$Group\ Delay = \frac{d(\omega[(M-1)/2])}{d\omega} = \frac{M-1}{2}. \tag{2.11}$$

Para filtros con mayor cascada K el retardo de grupo se incrementa a:

$$(Group\ Delay)^{\mathrm{K}} = \frac{(M-1)K}{2}. \tag{2.12}$$

Como se mencionó, la fase es lineal aun a pesar de que el filtro está formado por una parte recursiva (IIR) con un polo ubicado exactamente en el círculo unitario ($z = 1$). Este integrador es un sistema inestable sin embargo su polo es aniquilado en su totalidad por el cero de $z = 1$ de la parte comb.

En una implementación física los sistemas recursivos sufren de inestabilidad debido al redondeo de los coeficientes que pueden mover los polos fuera o en el límite del círculo unitario [1]. En la implementación CIC esto no sucede debido a que el coeficiente del integrador es unitario y no tiene problemas de redondeo. Sin embargo la retroalimentación unitaria presenta problemas de *overflow* de datos por lo que para una implementación correcta requiere que [5]:

1. La aritmética sea implementada en complemento a dos, o en cualquiera que permita ignorar el overflow en operaciones intermedias manteniendo correcto el resultado.
2. El rango de los números o la longitud de palabra del integrador sea mayor o igual que la esperada a la salida del filtro completo.

Lo último surge del hecho que para la correcta operación de cualquier filtro hay que considerar que la longitud de palabra necesaria para sus elementos (sumadores, multiplicadores y registros) debe ser mayor que la longitud de entrada en G bits, según lo determina la ecuación (2.13) [6].

$$G = \left\lfloor log_2\left(\sum_{k=0}^{L} h[k]\right)\right\rfloor. \tag{2.13}$$

Este aumento se hace para asegurar una suma de productos (convolución) sin *overflow*. Para el caso de los filtros CIC este incremento se simplifica como en (2.14) y se debe aplicar tanto al integrador como a la etapa comb.

$$G = log_2(M^K)\ bits. \tag{2.14}$$

2.2 El filtro CIC como filtro antialiasing

En el capítulo 1 se mostró que al hacer pasar una señal $x(n)$ a través de un DS con un factor de sub-muestreo M se introducen réplicas del espectro original $X(e^{j\omega})$ en las frecuencias múltiplos de $2\pi/M$ y que si la señal no está limitada en banda, mediante un filtro antialiasing, parte de las réplicas se introducen en la banda de interés de la señal y esta sufre distorsión por aliasing.

Si se usa un filtro CIC como filtro antialiasing, las réplicas del espectro debido al sub-muestreo se introducen en exactamente los dobleces de la respuesta en magnitud del filtro CIC donde existe un valor de atenuación muy alto. Sin embargo, la región alrededor de los dobleces (con menor atenuación) se introduce dentro de la banda de paso que se desea preservar ocasionando distorsión por aliasing. Para diseños prácticos los errores de aliasing pueden ser caracterizados por el máximo error sobre todos los dobleces de la banda. Para una gran cantidad de problemas de diseño este error máximo ocurre en el primer doblez de la banda [5].

A partir de lo anterior las características principales a considerar para un filtro CIC usado como filtro antialiasing son:

1. El *ancho de banda de paso* ω_p (AB), la cual depende del factor de sub-muestreo M y un segundo factor de sub-muestreo R de una etapa siguiente, está determinado por
$$\omega_p = \frac{2\pi}{MR}, \tag{2.15}$$
donde el mínimo valor para $R = 2$.
2. La *caída en la banda de paso* (CBP), depende del ancho de banda de la señal que se desea preservar, así como también del valor de la cascada K y se obtiene evaluando la respuesta en magnitud en la máxima frecuencia de ω_p.

3. La *atenuación en el primer doblez de la banda* (APDB), está en función del ancho de banda y del valor de la cascada K. Está determinado por la frecuencia a la que ocurre el primer doblez de la banda menos el ancho de la banda de paso. Este valor proporciona la menor atenuación posible contra el aliasing, que se introduce en la banda de paso, durante la reducción de la frecuencia de muestreo y generalmente es la característica más deseable de mejorar, y está determinado por:

$$\omega_A = \frac{2\pi}{M} - \omega_p. \tag{2.16}$$

4. El *factor de selectividad* $\emptyset$, está definido como la razón entre el valor de la magnitud en el límite de la banda de paso y la magnitud de la atenuación en el primer doblez de la banda.

$$\emptyset = \left| \frac{\sin\left(\pi(1/N - \omega_c/2)\right)}{\sin\left(\pi\omega_c/2\right)} \right|^K. \tag{2.17}$$

Las características anteriores han sido ampliamente adoptadas para comparar las diferentes mejoras aplicadas en filtros CIC [7] y son resumidas en la Fig. 2.3.

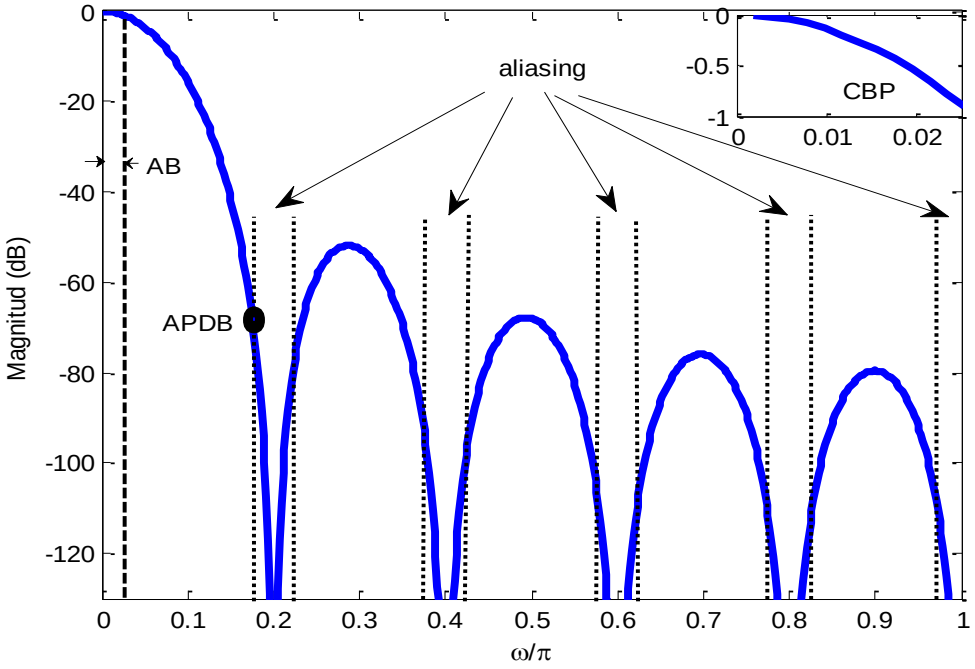


Fig. 2.3 Características principales de un filtro CIC usado como filtro antialiasing.

Para modificar la respuesta en magnitud de un filtro CIC solo se pueden usar dos parámetros de diseño, el orden del filtro M (el cual dependerá del factor de sub-muestreo) y el número de secciones en cascada K. No obstante, es necesario analizar los compromisos entre el ancho de banda de paso, la caída en la banda de paso y la atenuación en el primer doblez de la banda. Para comparar estos compromisos se supone un factor de sub-muestreo M lo suficientemente grande y se calcula la CBP y la APDB como función de la cascada K, para diferentes anchos de banda de paso. Un resumen de esto se encuentra en las Tablas 2.1 y 2.2 [5].

ω_p ↓	Caída en la banda de paso					
K →	1	2	3	4	5	6
1/128	0.00	0.00	0.00	0.00	0.00	0.01
1/64	0.00	0.01	0.01	0.01	0.02	0.02
1/32	0.01	0.03	0.04	0.06	0.07	0.08
1/16	0.06	0.11	0.17	0.22	0.28	0.34
1/8	0.22	0.15	0.67	0.90	1.12	1.35
1/4	0.91	1.82	2.74	3.65	4.56	5.47

Tabla 2.1 Caída en la banda de paso como función del número de cascada K.

ω_p ↓	Atenuación en el primer doblez de la banda					
K →	1	2	3	4	5	6
1/128	42.1	84.2	126.2	168.3	210.4	252.5
1/64	36.0	72.0	108.0	144.0	180.0	215.9
1/32	29.8	59.7	89.5	119.4	149.2	179.0
1/16	23.6	47.2	70.7	94.3	117.9	141.5
1/8	17.1	34.3	51.4	68.5	85.6	102.8
1/4	10.5	20.9	31.4	41.8	52.3	62.7

Tabla 2.2 Atenuación en el primer doblez de la banda como función del número de cascada K.

En estas tablas es posible apreciar que entre menor sea el ancho de banda de paso las características del filtro CIC son mejores. Esto se puede lograr haciendo que el filtro CIC

trabaje solo con una parte del factor de sub-muestreo total (según (2.15)) y deje el resto a otros decimadores *FIR* (Finite Impulse Response) más complejos.

2.3 Decimador CIC recursivo

El concepto básico de un decimador CIC recursivo se muestra en la Fig. 2.4 (a). Si la estructura se implementa como en la Fig. 2.4 (b) se puede aplicar la Tercera identidad multi-razón, entre la etapa comb y el DS, es decir, el DS queda entre las dos etapas como se muestra en la Fig. 2.4 (c). Esto último ocasiona que la etapa comb trabaje a una frecuencia de muestreo M-veces menor que la señal de entrada, lo que permite reducir el consumo de potencia, además la cantidad de retardos necesarios para su implementación se reducen de M a uno.

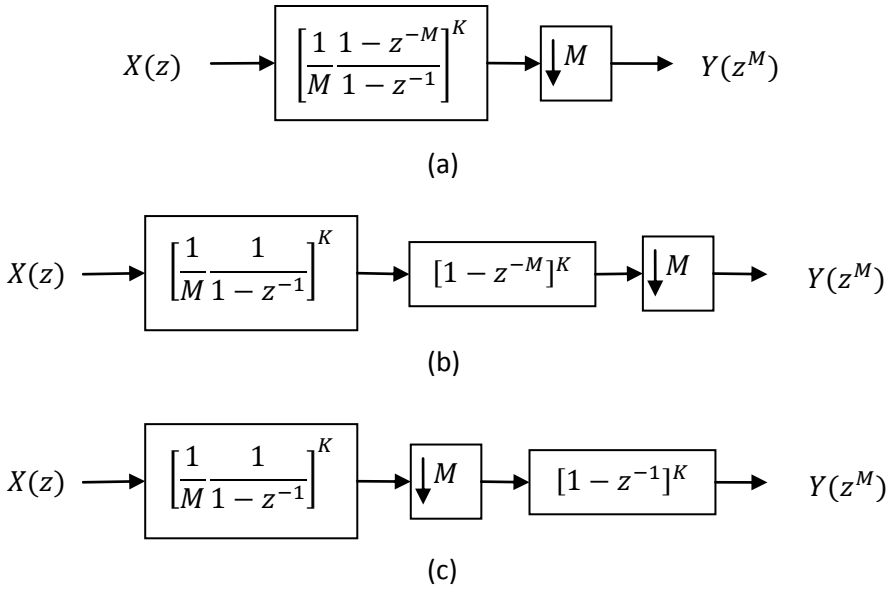


Fig. 2.4 Estructura decimador CIC.

Ejemplo 2.1: Se desea diseñar un decimador CIC recursivo para reducir la frecuencia de muestreo en un factor $M = 32$ y $M = 500$ ¿cómo será su función de transferencia y su implementación respectiva? La función de transferencia se presenta en (2.18) y (2.19) respectivamente, mientras que la implementación para ambos es la misma ya que al aplicar la Tercera identidad multi-razón la etapa comb, en ambos casos, se mueve adelante del DS

y queda como la estructura de la Fig. 2.4 (c), la única diferencia está en frecuencia de muestreo a la que trabaja la etapa comb.

$$H(z) = \left[\frac{1}{N}\frac{1 - z^{-32}}{1 - z^{-1}}\right]^4 . \tag{2.18}$$

$$H(z) = \left[\frac{1}{N}\frac{1 - z^{-500}}{1 - z^{-1}}\right]^4 . \tag{2.19}$$

Resumen de ventajas y desventajas de la estructura CIC recursiva

Ventajas:

- No requiere multiplicadores.
- No requiere almacenamiento de coeficientes.
- La cantidad intermedia de registros es reducida debido al uso de la tercera identidad multi-razón.
- Su estructura es muy regular consistiendo de solo dos bloques.
- Un mismo diseño puede ser usado para un gran número de de factores de decimación, ya que la estructura permanece invariable y solo requiere que las señales de reloj que activan la sección comb tengan la frecuencia de muestreo adecuada.

Desventajas:

- El tamaño de palabra necesario a la entrada de los integradores puede ser muy grande para factores de decimación grandes, lo que aumenta el consumo de potencia.
- La respuesta en magnitud está determinada solo por los parámetros M y K, lo que resulta en características muy limitadas y al tratar de mejorarlas generalmente se pierde la simplicidad.

2.4 Filtro CIC no recursivo

Los decimadores CIC recursivos son computacionalmente eficientes y simples de implementar. Sin embargo, el mayor de sus problemas es la necesidad del aumento de longitud de palabra para el integrador, el cual aumenta en función del factor de decimación. Este aumento y la alta frecuencia de muestreo provocan que el consumo de potencia sea elevado. Este problema puede ser evitado usando la forma no recursiva del filtro CIC de (2.3). Para el caso especial cuando el orden del filtro se puede expresar como una potencia de dos $M = 2^P$, o una potencia de tres $M = 3^P$, se obtienen las ecuaciones (2.20) y (2.21), respectivamente.

$$H(z) = (1 + z^{-1})^K \times (1 + z^{-2})^K \times (1 + z^{-4})^K \dots \times \left(1 + z^{-2^{P-1}}\right)^K. \quad (2.20)$$

$$H(z) = (1 + z^{-1} + z^{-2})^K \times (1 + z^{-3} + z^{-6})^K \dots \times \left(1 + z^{-3^{P-1}} + z^{-2*3^{P-1}}\right)^K. \quad (2.21)$$

Estas se pueden representar en forma compacta como:

$$H(z) = \left[\frac{1}{M}\sum_{k=0}^{M-1} z^{-k}\right]^K = 2^{-PK}\left[\prod_{i=0}^{P-1}\left(1 + z^{-2^i}\right)\right]^K, \quad (2.22)$$

$$H(z) = \left[\frac{1}{M}\sum_{k=0}^{M-1} z^{-k}\right]^K = 2^{-PK}\left[\prod_{i=0}^{P-1}\left(1 + z^{-3^i} + z^{-2*3^i}\right)\right]^K. \quad (2.23)$$

Estas factorizaciones eliminan la parte recursiva (IIR) en la implementación por lo que la estructura se vuelve más eficiente en cuanto a consumo de potencia. No obstante, muchas veces se le sigue llamando estructura CIC aun a pesar de la ausencia del integrador. Las ecuaciones (2.22) y (2.23) se pueden implementar según se muestra en la Fig. 2.5 (a) y (b), respectivamente.

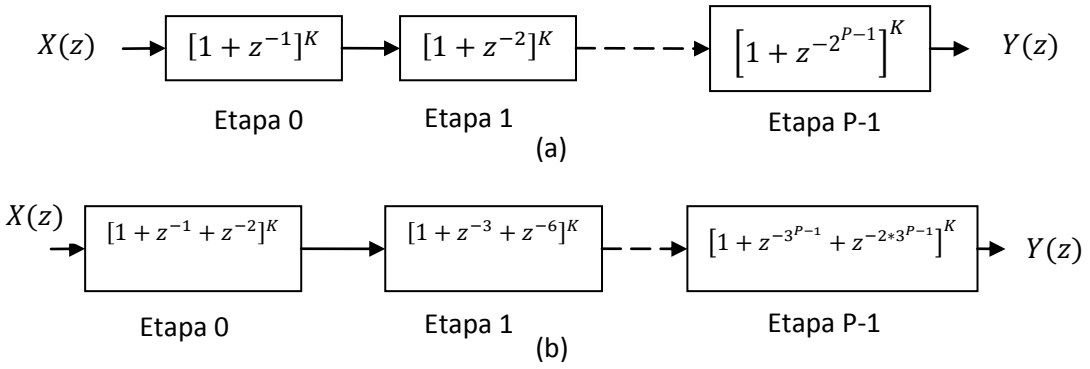


Fig. 2.5 Estructura filtro CIC no recursivo, (a) $M = 2^P$ y (b) $M = 3^P$.

2.4.1 Decimador CIC no recursivo

Usando las estructuras CIC no recursivas de la Fig. 2.5 como filtro antialiasing se obtienen los decimadores CIC no recursivos mostrados en la Fig. 2.6 (a) para $M = 2^P$ y (b) para $M = 3^P$. Al igual que en el decimador basado en filtros CIC recursivos, las estructuras de la Fig. 2.6, se vuelven más eficientes debido al uso de Tercera identidad multi-razón. Esto provoca que las etapas del filtro CIC no recursivo trabajen con diferentes frecuencias de muestreo que van disminuyendo en factores de dos o tres, según como se represente el factor de sub-muestreo. Al mismo tiempo los elementos de memoria se reducen a la forma básica de la primera etapa por lo que al final todas las etapas se implementan en la misma manera, es decir, se convierten en una celda básica que solo cambia su frecuencia de muestreo y la precisión necesaria de los elementos que la forman. De esta manera, se conserva la simplicidad en la implementación debido a que solo es necesaria una etapa y a partir de esta, iterarla $P-1$ veces para formar la estructura del decimador completo.

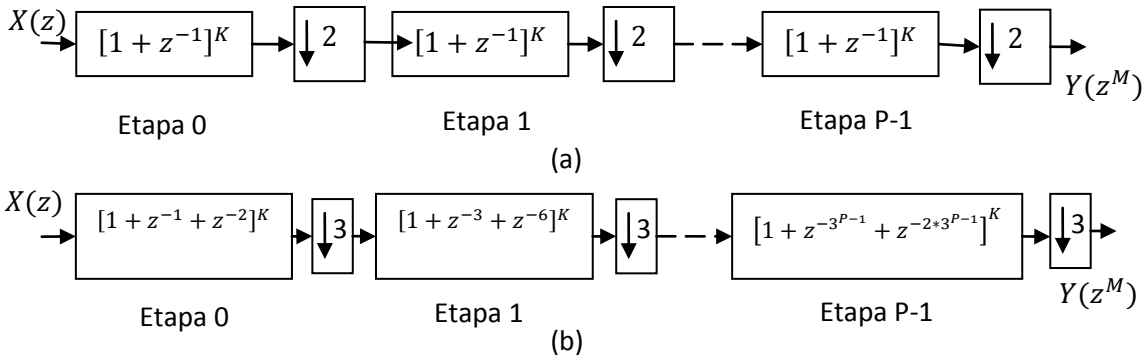


Fig. 2.6 Estructura decimador CIC no recursivo, (a) $M = 2^P$ y (b) $M = 3^P$.

Las estructuras para los decimadores CIC no recursivos presentan un menor consumo de potencia que su contraparte recursiva debido principalmente a [8]:

1. En la primera etapa la longitud de palabra necesaria para los componentes es la mínima posible y va incrementándose gradualmente, mientras la estructura recursiva posee toda la longitud de palabra en el integrador de entrada.
2. La etapa con la longitud de palabra más pequeña trabaja a la frecuencia de muestreo más alta, mientras que la que tiene la longitud de palabra más alta trabaja a la frecuencia de muestreo más baja.

Al representar el orden del decimador como potencias de dos o tres se presenta una limitación en cuanto al factor de sub-muestreo que se puede obtener para una aplicación específica. Estos valores están limitados a $2, 4, 8, 16 \ldots 2^P$ y $3, 9, 27, 81, 3^P$. Sin embargo es posible combinar las dos estructuras para generar una variedad más amplia. Para esto hay que descomponer el factor de sub-muestreo como múltiplos de dos y tres. Por ejemplo para un factor $M = 36$ se puede usar $M = 2 \times 2 \times 3 \times 3$, o para $M = 48 = 2 \times 2 \times 2 \times 2 \times 3$.

Finalmente, la estructura CIC no recursiva del caso especial $M = 2^P$ ha sido reportada como la de menor consumo de potencia [9].

2.4.2 Decimador CIC no recursivo en componentes polifásicas

Las etapas que permiten implementar el decimador CIC no recursivo son *sub-filtros* de tipo FIR, los cuales se pueden descomponer en sus componentes polifásicas. En el capítulo anterior se presentó esta descomposición mediante las ecuaciones (1.37) y (1.38). Para el caso de los decimadores CIC no recursivos esta descomposición se debe realizar para valores de $M = 2$ y $M = 3$, con el objetivo de poder implementar las etapas de la Fig. 2.6 mediante dos y tres sub-filtros, respectivamente, para así poder aplicar la Tercera identidad multi-razón una vez mas y reducir la frecuencia de operación de cada etapa. Para el caso de $M = 2^P$ esta descomposición se realiza mediante (2.24) y para $M = 3^P$ con (2.25).

$$H(z) = \sum_{m=0}^{1} z^{-m} H_m(z^2). \tag{2.24}$$

$$H(z) = \sum_{m=0}^{2} z^{-m} H_m(z^3). \tag{2.25}$$

Descomponiendo cada una de las etapas de la estructura de la Fig. 2.6 (a) mediante la ecuación (2.25) y las de la Fig. 2.6 (b) mediante (2.25) se obtienen las estructuras de la Fig. 2.7 (a) y (b), respectivamente.

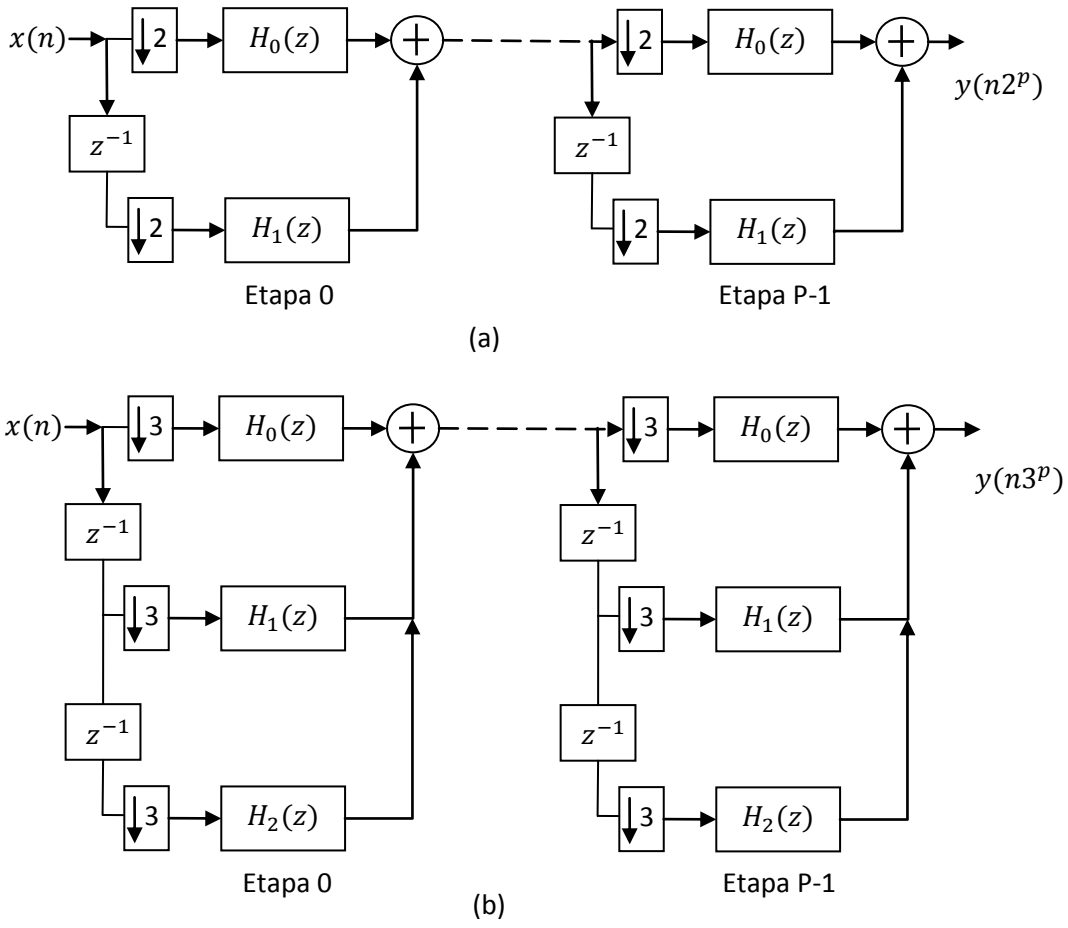


Fig. 2.7 Estructura decimador CIC no recursivo en componentes polifásicas, (a) $M = 2^P$ y (b) $M = 3^P$.

A partir de la Fig. 2.7 es posible ver cuatro aspectos fundamentales:

1. Cada etapa reduce su frecuencia de muestreo en un factor de dos o tres, respecto con las implementaciones de la Fig. 2.6, lo cual permite reducir aún más el consumo de potencia del decimador CIC no recursivo.

2. Permite un aumento en la frecuencia máxima de operación del decimador debido a que cada etapa trabaja a menor frecuencia. Por ejemplo, si una implementación particular hecha con la estructura de la Fig. 2.6 (a) tiene un límite de frecuencia de 100MHz entonces la implementación para la misma aplicación pero con la estructura de la Fig. 2.7 (a) tendrá una frecuencia máxima de operación de 200MHz debido a que cada etapa trabaja a la mitad de la frecuencia.
3. La simplicidad en la implementación del decimador se ve afectada debido a que para obtener las componentes polifásicas o sub-filtros de (2.24) y (2.25) se debe expandir la función de transferencia $(1 + z^{-1})^K$, y con esto se obtiene un sub-filtro FIR con coeficientes diferentes de uno.
4. La implementación de los coeficientes en las componentes polifásicas requiere el uso de multiplicadores digitales, lo cual tiene un impacto negativo en la cantidad de recursos para su implementación, es decir se incrementa el área requerida en comparación al decimador CIC recursivo y no recursivo.

El uso de la estructura polifásica es ideal en aplicaciones que requieren de un factor de decimación grande, debido a que por una parte el consumo de potencia es reducido, y por otra la frecuencia máxima de operación del circuito aumenta en un factor de dos o tres en comparación con la estructura CIC no recursiva, sin la necesidad de hacer optimización de hardware.

Capítulo 3

Revisión de métodos sobre mejoras a los filtros CIC

En este capítulo primero se presentarán, brevemente, algunos métodos que se han propuesto para mejorar la respuesta en frecuencia de los filtros CIC. Estos métodos se encuentran divididos en los destinados a mejorar la banda de paso, la banda de rechazo y ambas. A lo largo de la descripción de los métodos se presentarán ejemplos que permitan apreciar sus principales ventajas y desventajas. Después, se presentarán algunas alternativas para reducir el consumo de potencia en la implementación de filtros digitales, tales como la representación eficiente de coeficientes, su implementación en base a corrimientos y sumas/restas, y la compartición de sub-expresiones comunes. Finalmente, se presentará un método para decimadores CIC no recursivos, y los resultados de su implementación, que permite reducir el consumo de potencia.

3.1 Introducción

En los capítulos anteriores se mostró que debido a su simplicidad de implementación y eficiencia computacional los decimadores CIC son ideales para ser utilizados en la primera etapa de decimación, donde la frecuencia de muestreo es muy alta. Sin embargo, su respuesta en magnitud exhibe una caída *en la banda de paso* (CBP) y una baja atenuación en la *banda de rechazo* (BR). Debido a esto, han sido desarrolladas una gran cantidad de propuestas para mejorar estas características, algunas de las cuales serán analizadas en este capítulo. También se presentan algunas mejoras, a nivel hardware, que permiten reducir

el consumo de potencia en la implementación de filtros digitales.

3.2 Métodos para mejorar la respuesta en frecuencia

3.2.1 Caída en la banda de paso

Para mejorar la CBP del filtro CIC es necesario un filtro compensador, el cual debe tener la complejidad más baja posible para no contribuir significativamente en el consumo de potencia. Diferentes métodos han sido propuestos para lograr este objetivo. Estos se pueden clasificar en métodos de compensación para banda angosta ($R > 2$) y para banda ancha ($R = 2$). Los métodos presentados en [10] emplean técnicas de optimización, por lo que los filtros de compensación obtenidos requieren de multiplicadores para su implementación. Por otra parte, el método descrito en [11] presenta el diseño de un filtro compensador de segundo orden sin multiplicadores donde los coeficientes pueden representarse mediante *suma de potencias de dos* (SPD) y son calculados usando el algoritmo de búsqueda aleatoria. Más adelante en la sección 3.3 se presentará la forma de implementar estos coeficientes mediante corrimientos y sumas.

Las siguientes características son las más deseables para un compensador CIC:

- El filtro compensador debería trabajar en la frecuencia de muestreo más baja posible.
- Al igual que el filtro CIC no usar multiplicadores.
- Diseño simple que no requiera rediseñar el filtro para diferentes valores de M y K. Esta es una característica muy deseable porque de esta manera el compensador permanece invariable a los valores antes mencionados.

Compensador de banda angosta [12]

En este apartado se puede mencionar el filtro compensador propuesto en [12] porque este filtro satisface la mayoría de las características antes mencionadas.

Considérese un filtro con respuesta en magnitud

$$\left|G(e^{j\omega})\right| = \left|1 + 2^{-b} sin^2(\omega M/2)\right|, \tag{3.1}$$

donde b es un parámetro entero.

Usando la relación

$$sin^2(\alpha) = \frac{1 - \cos(2\alpha)}{2}, \tag{3.2}$$

la correspondiente función de transferencia de (3.1) puede ser expresada como

$$G(z^M) = -2^{-(b+2)}\left[1 - \left(2^{(b+2)} + 2\right)z^{-M} + z^{-2M}\right]. \tag{3.3}$$

Haciendo

$$A = -2^{-(b+2)}\,; B = \left(2^{(b+2)} + 2\right), \tag{3.4}$$

se llega a

$$G(z^M) = A[1 - Bz^{-M} + z^{-2M}]. \tag{3.5}$$

Este filtro compensador tiene el factor de escalamiento A y un único coeficiente B que requiere solo de un sumador para su implementación. Además, el compensador puede ser implementado después del sub-muestreo de la señal, por el factor M, al hacer uso de la Tercera identidad multi-razón (véase sección 1.2.6), convirtiéndolo en un filtro de segundo orden,

$$G(z^M) = A[1 - Bz^{-1} + z^{-2}]. \tag{3.6}$$

Como se puede ver, el filtro no depende del factor de sub-muestreo M pero si del valor b, el cual a su vez depende de la cascada K. Para la determinación del parámetro b, en [12], se proporciona la Tabla 3.1.

La función completa del filtro CIC más el compensador es

$$H(z) = H_{CIC}(z)G(z^M), \tag{3.7}$$

donde $H_{CIC}(z)$ y $G(z^M)$ están dadas por (2.9) y (3.5), respectivamente.

Ejemplo 3.1: Considérese el filtro CIC con parámetros $M = 16$ y $K = 5$. A partir de la Tabla 3.1 se tiene que $b = 0$, por lo que $A = -0.25$ y $B = 6$. En la Fig. 3.1 se puede encontrar la respuesta en magnitud para el filtro CIC, el compensador y la cascada.

Parámetro K	Parámetro b, $R = 8$
2	2
3	2
4	1
5	0
6	0

Tabla 3.1 Determinación del parámetro b para el método de [12]

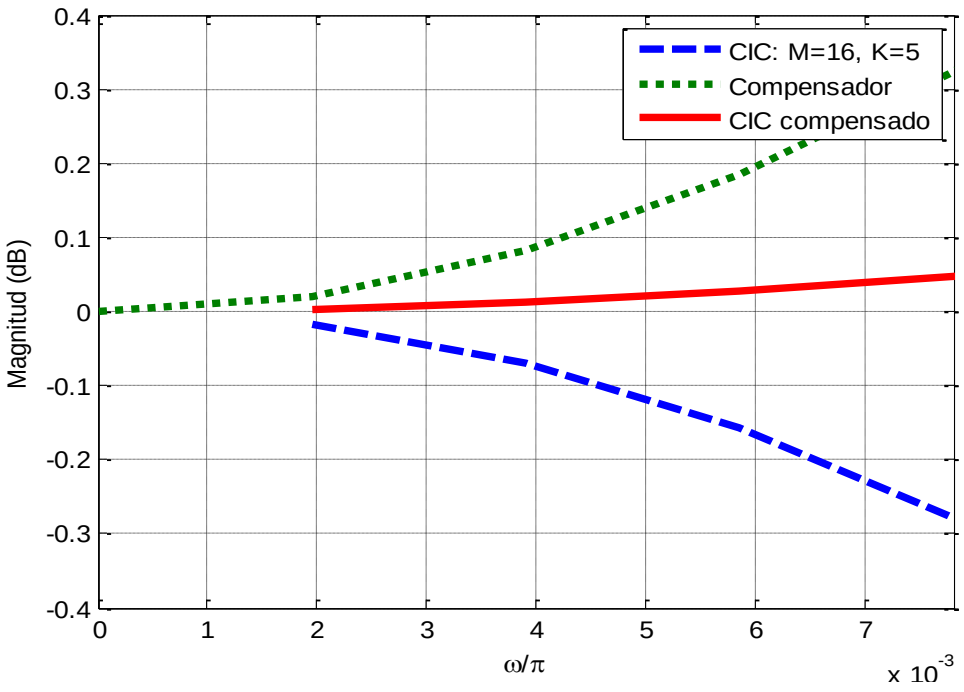


Fig. 3.1 Ilustración para el Ejemplo 3.1.

3.2.2 Atenuación en la banda de rechazo

En la sección 2.1 se pudo ver que la atenuación del filtro CIC en la banda de rechazo es muy pequeña y que para incrementarla lo único que se puede hacer es aumentar el valor de cascada K. No obstante, se han propuesto varios métodos que permiten incrementar esta

atenuación en base a una modificación del filtro CIC original, algunos de los cuales se presentan a continuación.

Filtro RS [13]

Con el objetivo de aumentar la atenuación y el ancho en los dobleces de la banda alrededor de los ceros del filtro CIC, Presti introdujo la rotación de los ceros y propuso el *filtro sinc rotado* (RS por sus siglas en inglés) [13]. Este se obtiene como se muestra a continuación.

Primero, se aplica una rotación de los ceros del filtro CIC, en el sentido de las manecillas del reloj, por β radianes y se obtiene

$$H_u(z) = \frac{1}{M}\frac{1 - z^{-M}e^{j\beta M}}{1 - z^{-1}e^{j\beta}}. \tag{3.8}$$

Una expresión equivalente a (3.8) se consigue al aplicar la misma rotación pero en dirección opuesta

$$H_d(z) = \frac{1}{M}\frac{1 - z^{-M}e^{-j\beta M}}{1 - z^{-1}e^{-j\beta}}. \tag{3.9}$$

Estos filtros (3.8) y (3.9) tienen coeficientes complejos, pero si se conectan en cascada se obtiene un filtro $H_r(z)$ con coeficientes reales

$$H_r(z) = H_u(z)H_d(z) = \frac{1}{M^2}\frac{1 - 2\cos(\beta M)\, z^{-M} + z^{-2M}}{1 - 2\cos(\beta)\, z^{-1} + z^{-2}}. \tag{3.10}$$

La cascada del filtro CIC con (3.10) es llamada por Presti como filtro RS y su función de transferencia es

$$H_R(z) = H_{CIC}(z)H_r(z), \tag{3.11}$$

donde $H_{CIC}(z)$ y $H_r(z)$ están dadas por (2.9) y (3.10), respectivamente.

La respuesta en magnitud para (3.11) está dada por:

$$\left|H_R\left(e^{j\omega}\right)\right| = \left|\frac{1}{M^3}\frac{\sin\left(\frac{\omega M}{2}\right)}{\sin\left(\frac{\omega}{2}\right)}\right|^K \left|\frac{\sin\left(\frac{(\omega+\beta)M}{2}\right)}{\sin\left(\frac{(\omega+\beta)}{2}\right)}\right|^K \left|\frac{\sin\left(\frac{(\omega-\beta)M}{2}\right)}{\sin\left(\frac{(\omega-\beta)}{2}\right)}\right|^K . \quad (3.12)$$

Ejemplo 3.2: Usando el método de [13], se diseña el filtro RS para $M = 16$, $K = 1$, y $\beta = 0.0184$. La respuesta en magnitud para el filtro RS se muestra en la Fig. 3.2 junto a la del filtro CIC con los mismos parámetros.

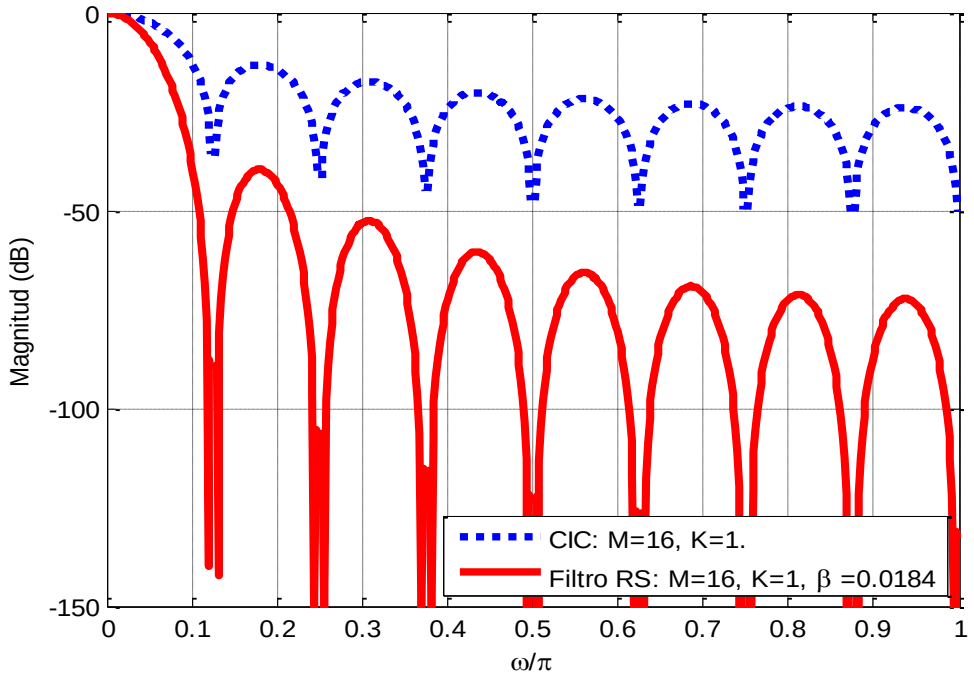


Fig. 3.2 Ilustración para el Ejemplo 3.2.

A partir de la Fig. 3.2 se puede notar lo siguiente:

- Los dobleces de la banda del filtro RS son más anchos y con una atenuación mayor que la correspondiente del filtro CIC. Sin embargo, la caída en la banda de paso también se incrementa.
- Para su implementación el filtro de (3.10) requiere dos multiplicadores, uno de ellos trabajando a la frecuencia de muestreo de entrada (el de la parte recursiva), lo cual incrementa el consumo de potencia.
- Al igual que en el filtro CIC el primer doblez de la banda es el punto más crítico, donde se tiene la menor atenuación para la distorsión por aliasing.

- Debe existir una cancelación perfecta entre los polos y lo ceros. Durante la cuantización de los coeficientes esta cancelación se puede perder resultando en inestabilidad.

Filtro comb generalizado [14]

Este método trata de conseguir una implementación del método [13] en la que no se requiera el uso de multiplicadores, con el objetivo principal de obtener una implementación más eficiente y una perfecta cancelación entre polos y ceros. Su función de transferencia es la misma que la de (3.11) y se presenta nuevamente en (3.13).

$$H_3(z) = \frac{1 - z^{-M}}{1 - z^{-1}} \frac{1 - az^{-M} + z^{-2M}}{1 - bz^{-1} + z^{-2}}, \tag{3.13}$$

donde

$$a = 2\cos(\beta M), \tag{3.14}$$

$$b = 2\cos(\beta). \tag{3.15}$$

Si el parámetro $\beta = 0$ la función de transferencia de (3.13) es equivalente a un filtro CIC de tercer orden ($K = 3$). En este método se utiliza el valor óptimo para $\beta = 0.038779$, derivado en [7].

El punto clave de este método es representar los coeficientes de (3.14) y (3.15) solamente como función del término común $\cos(\beta)$. Esto con el objetivo de poder cuantizar $\cos(\beta)$ y asegurar la perfecta cancelación de polos y ceros, así como también la estabilidad del filtro.

Para convertir el término de (3.14) a una función de $\cos(\beta)$ se le aplica la siguiente ecuación recursiva

$$\cos(\beta M) = 2\cos\big(\beta(M-1)\big)\cos(\beta) - \cos\big(\beta(M-2)\big). \tag{3.16}$$

Una aproximación útil para representar $\cos(\beta)$, en el resultado de (3.16), es mediante una *suma de potencias de dos* (SPD). Al hacer esto, las multiplicaciones se realizan como corrimientos y sumas/restas (véase más adelante la Sección 3.3). Para encontrar la SPD del término $\cos(\beta)$, este se cuantiza como

$$\cos(\beta) = 2^{-k} I, \tag{3.17}$$

donde I es un entero apropiado que también puede ser expresado como SPD.

Lo siguiente es encontrar un valor adecuado del parámetro k con el que se consiga la mejor representación del término $\cos(\beta)$ en (3.17). Considerando que $\cos(\beta) \leq 1, \ \forall \beta$, se obtiene

$$\cos(\beta) = 2^{-k} I, \quad I_{max} = 2^k - 1, \tag{3.18}$$

Por lo tanto

$$\cos(\beta) \leq 2^{-k}(2^k - 1) = 1 - 2^{-k}. \tag{3.19}$$

Resolviendo (3.19) se obtiene

$$k = \lfloor -log_{10}(1 - \cos(\beta))/log_{10}(2) \rfloor. \tag{3.20}$$

Considerando el parámetro óptimo $\beta = 0.038779$, se obtiene $k = 11$ por lo que el término $\cos(\beta)$ se cuantiza como

$$\cos(\beta) = 1 - 2^{-11}. \tag{3.21}$$

Finalmente, el diseño de este decimador es muy similar al presentado en [13] solo que la implementación final se lleva a cabo con el coeficiente a expresado solo como función del término $\cos(\beta)$, al usar la ecuación (3.16), donde a su vez el término $\cos(\beta)$ se cuantiza como muestra (3.21). Esto da lugar a una implementación sin multiplicadores y una perfecta cancelación de polos y ceros. Sin embargo, este método sigue teniendo el problema de poca atenuación en el primer doblez de la banda como método de [13].

Filtro RS simplificado [15]

Para solucionar el problema de poca atenuación en el primer doblez de la banda en [13] y [14], el método de [15] propone realizar la rotación de los ceros solo en el primer doblez de la banda, a lo que los autores llaman el término de rotación del cero (ZTR por sus siglas en inglés). Este está definido como

$$H_{ZRT}(z) = k\left(1 - z^{-1}e^{-j\beta}\right)\left(1 - z^{-1}e^{j\beta}\right) = k(1 - 2\cos(\beta)\, z^{-1} + z^{-2}), \quad (3.22)$$

donde k es la constante de normalización introducida para asegurar que la respuesta en magnitud es unitaria para frecuencias $\omega = 0$. Considerando que $R = 2$, el ancho de banda de paso para un filtro CIC según (2.15), está definido por la frecuencia de corte de $\pi\ /2M$. El cero que se introduce deberá estar cerca del punto de *atenuación en el primer doblez de la banda* (APDB) definido en (2.16), de esta manera

$$\beta = \frac{\pi}{2M} - \frac{\pi}{(\beta_0 + 2)M}, \quad (3.23)$$

donde β_0 es el término que mueve ligeramente el cero desde la APDB hacia la derecha, dentro del primer doblez de la banda. El valor típico para $\beta_0 = 0.99$. La constante de normalización k es

$$k = \frac{1}{2 - \cos\left(\frac{\pi}{2M} - \frac{\pi}{(\beta_0 + 2)M}\right)}, \quad (3.24)$$

De esta manera, la función de transferencia para esta estructura es

$$H_{CIC,ZRT}(z) = \frac{1}{M}\frac{1 - z^{-M}}{1 - z^{-1}} \times k\left(1 - 2\cos\left(\frac{\pi}{2M} - \frac{\pi}{(\beta_0 + 2)M}\right)z^{-1} + z^{-2}\right). \quad (3.25)$$

Ejemplo 3.3: Se diseña un filtro CIC para $M = 12$, $K = 6$, y el término ZTR correspondiente con $\beta = 0.99$. La respuesta en magnitud para estos filtros de forma individual y en cascada se muestra en la Fig. 3.3.

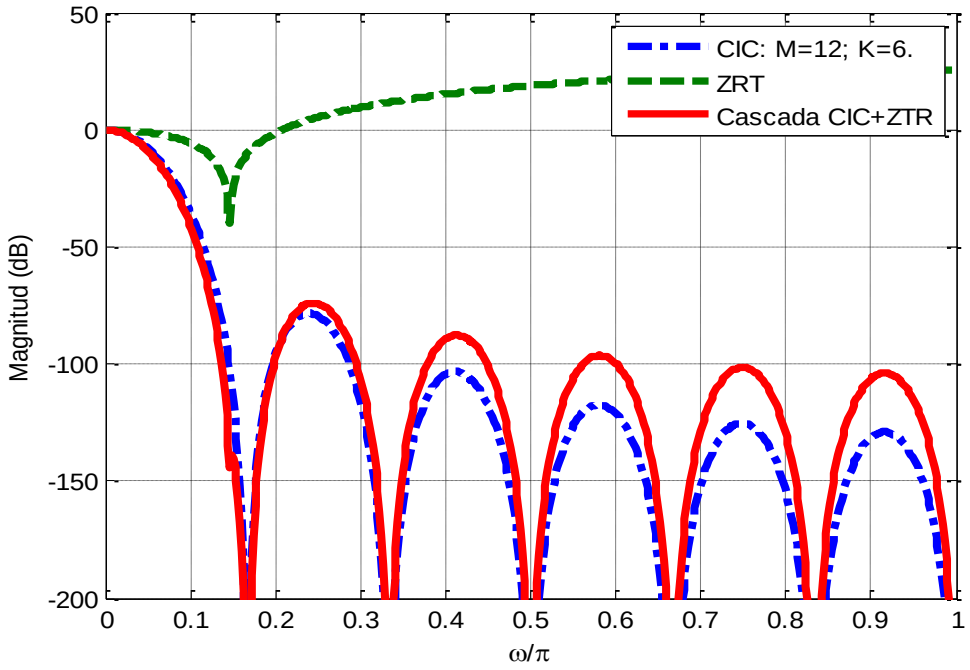


Fig. 3.3 Ilustración para el Ejemplo 3.3.

De la Fig. 3.3 se puede notar que el primer doblez de la banda es más ancho para el filtro CIC en cascada con el término ZTR que para filtro CIC original. Sin embargo, las regiones alrededor de los demás ceros se incrementan así como también la caída en la banda de paso. Para incrementar la atenuación en los demás dobleces se propone el uso de filtros coseno en cascada, cuya función de transferencia es

$$H_{COS}(z) = \prod_{k} \left[\frac{1}{2}(1 + z^{N_k})\right]^{K_k}. \tag{3.26}$$

La elección de los factores de la ecuación anterior dependerá del compromiso entre incremento en complejidad e incremento en atenuación.

De esta manera, la función de transferencia para el filtro RS simplificado es

$$H_{CIC,ZTR,COS}(z) = H_{CIC}(z)H_{ZTR,}(z)H_{COS}(z)\,. \tag{3.27}$$

Ejemplo 3.4: Se diseña el filtro RS modificado para $M = 12$, $K = 5$, $\beta = 0.99$, y el filtro CIC con $K = 5\ y\ 6$. La respuesta en magnitud para estos filtros es presentada en la Fig. 3.4, donde también puede encontrarse una ampliación para el primer doblez de la banda.

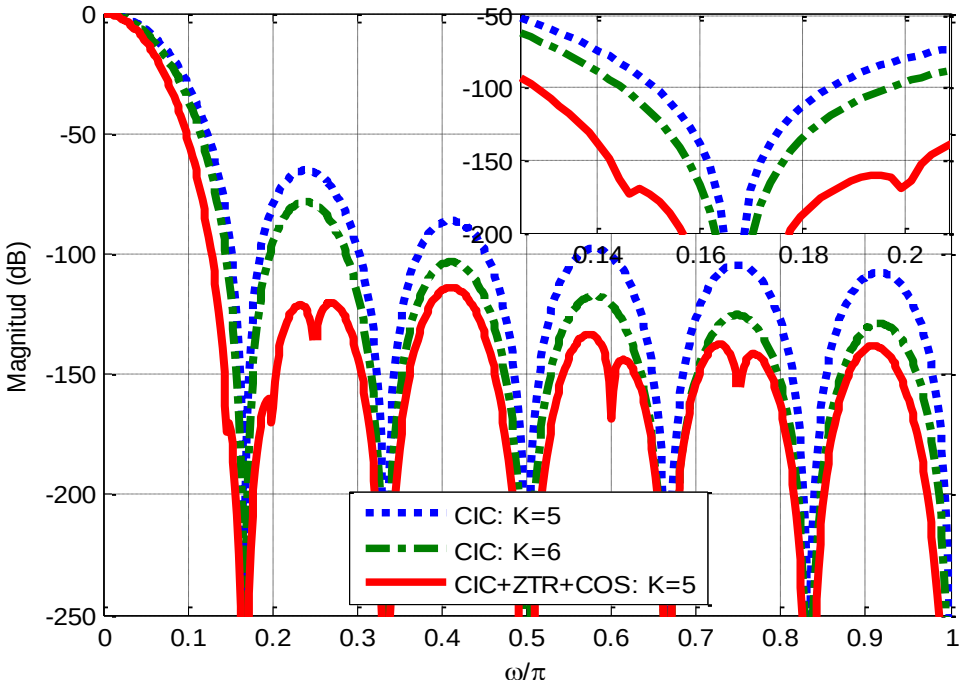


Fig. 3.4 Ilustración para el Ejemplo 3.4.

En la Fig. 3.4 se puede ver que la respuesta en magnitud para el filtro RS simplificado es mejor que para el filtro CIC original. Sin embargo, el término ZTR requiere del uso de un multiplicador, que a pesar de poder ser implementado mediante sumas y corrimientos, eligiendo la constante de redondeo adecuada, no puede ser trasladado hacia una frecuencia de muestreo baja, lo que contribuye al incremento en el consumo de potencia. Además, no todos los términos coseno se pueden trasladar a baja frecuencia solo aquellos que tengan un orden que sea múltiplo del factor de sub-muestreo, siempre y cuando este último se descomponga adecuadamente.

3.2.3 Caída en la banda de paso y atenuación en la banda de rechazo

En las sub-secciones anteriores fueron presentados algunos métodos para mejorar la caída en la banda de paso y la atenuación en la banda de rechazo de forma individual. Sin embargo, existe un método que permite mejorar ambas características al mismo tiempo. En esta sub-sección se presenta este método y una de sus modificaciones más importantes.

Sharpened CIC [16]

Sharpening es una técnica que permite mejorar la banda de paso y la banda de rechazo de un filtro digital, en una manera tal que mejora la banda de paso e incrementa la atenuación en la banda de rechazo simultáneamente. Este método fue originalmente concebido por Kaiser y Hamming en 1977 [17].

El primer trabajo que aplico la técnica sharpening a los filtros CIC fue presentado por Kwentus y Wilson en 1997 [16]. Ellos utilizaron el polinomio

$$H_{sh} = 3H^{2K_1} - 2H^{3K_1}, \tag{3.28}$$

donde H es la respuesta en magnitud para el filtro CIC y K_1 es el número de cascadas.

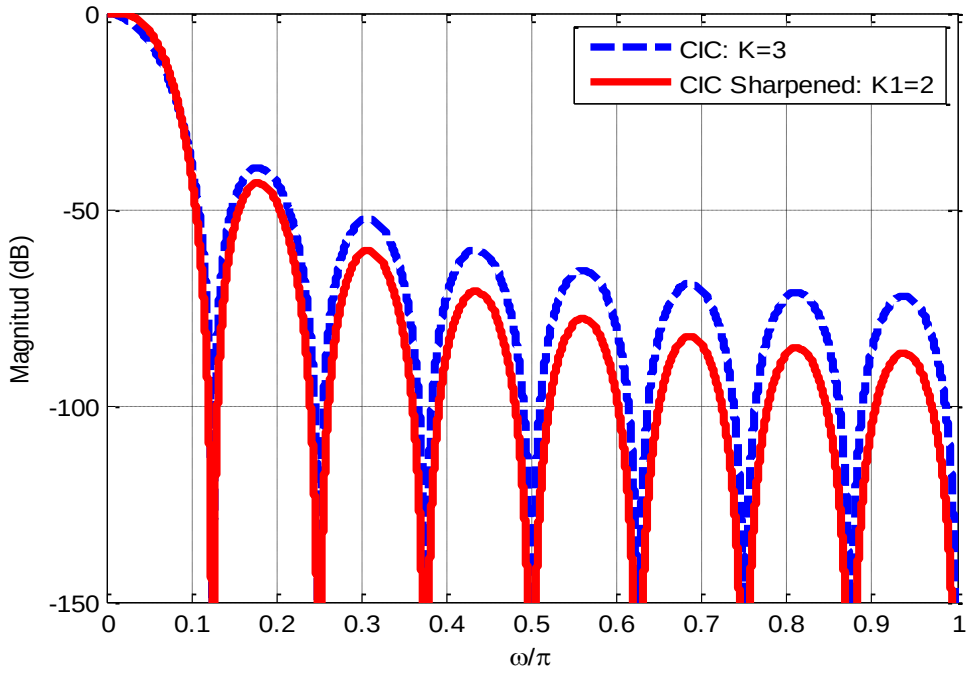
La respuesta en magnitud correspondiente al filtro *Sharpened CIC* (*SCIC* por sus siglas en inglés) es

$$|H_{shCIC}(z)| = \left| 3\left(\frac{1}{M}\frac{\sin\left(\frac{\omega M}{2}\right)}{\sin\left(\frac{\omega}{2}\right)}\right)^{2K_1} - 2\left(\frac{1}{M}\frac{\sin\left(\frac{\omega M}{2}\right)}{\sin\left(\frac{\omega}{2}\right)}\right)^{3K_1} \right|. \tag{3.29}$$

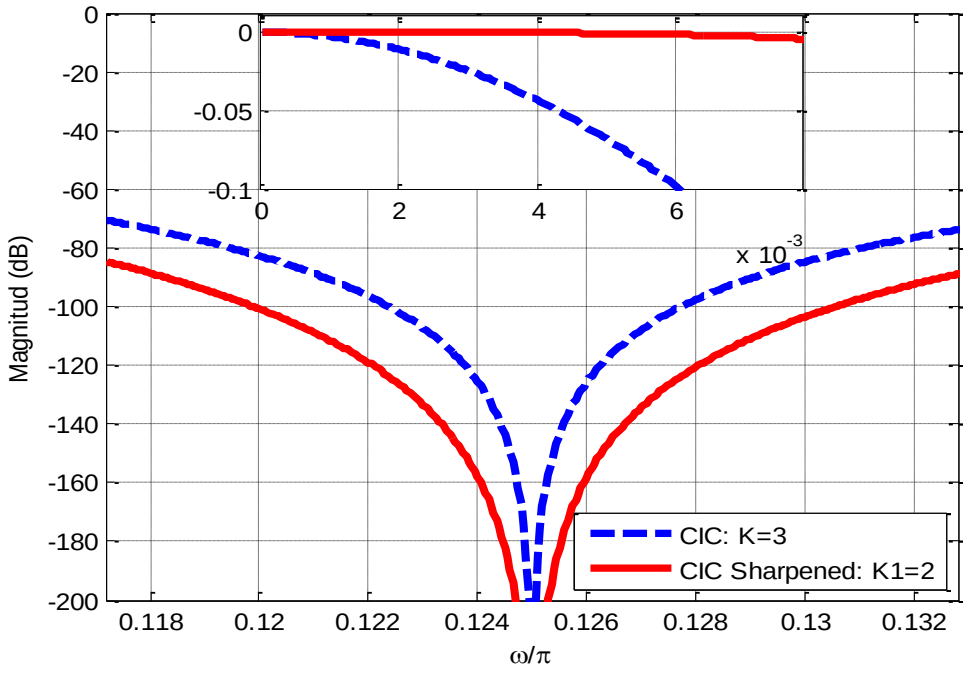
Como se mencionó antes la respuesta en magnitud de (3.29) permite mejorar la respuesta del filtro CIC mejorando la atenuación y disminuyendo la caída en la banda de paso, para poder apreciar esto considérese el siguiente ejemplo.

Ejemplo 3.5: Se diseña el filtro SCIC con parámetros $K_1 = 2$, $M = 16$ y se compara con el filtro CIC con $K = 3$. La Fig. 3.5 (a) muestra la respuesta en magnitud para ambos filtros y la Fig. 3.5 (b) una ampliación para el primer doblez de la banda y la banda de paso.

De la Fig. 3.5 es notable ver que el filtro SCIC mejora las dos características más críticas de la respuesta en magnitud del filtro CIC. Sin embargo, presenta el mismo problema que el filtro CIC recursivo, que las etapas integradoras no pueden ser trasladadas a una baja frecuencia de muestreo para reducir su consumo de potencia.



(a)



(b)

Fig. 3.5 Ilustración para el Ejemplo 3.5, (a) respuesta en magnitud, (b) ampliación para la banda de paso y PDB.

Sharpened CIC modificado [18]

Diversos autores han desarrollado métodos para mejorar las características de respuesta en magnitud del filtro SCIC como las presentadas en [7, 18-22]. En esta sección se presentan los resultados obtenidos en [18], debido a que este método integra varias de las ventajas propuestas en [7, 19-21] e introduce el filtro *SCIC modificado* (SCICM) que opera a una frecuencia de muestreo menor que el SCIC original. Además propone el uso de un compensador para la CBP que se obtiene a partir de los parámetros de diseño del SCICM.

Este método divide el factor de sub-muestreo en dos etapas

$$M = M_1 M_2, \tag{3.30}$$

con el objetivo de que la parte sharpening opere a una baja frecuencia de muestreo.

La primer etapa es un filtro CIC simple ($M_1 < M$) y puede ser implementada en forma recursiva o no recursiva

$$H_1(z) = \frac{1}{M_1}\frac{1 - z^{-M_1}}{1 - z^{-1}} = \sum_{i=0}^{M_1 - 1} z^{-i}. \tag{3.31}$$

La segunda etapa es un filtro CIC de la forma

$$H_2(z) = \frac{1}{M_2}\frac{1 - z^{-M_1 M_2}}{1 - z^{-M_1}}, \tag{3.32}$$

a la cual se le aplica la técnica sharpening.

De esta manera, la función de transferencia para este método está determinada por:

$$H(z) = H_1^{K_1}(z) Sh\{H_2^{K_2}(z^{M_1})\}, \tag{3.33}$$

donde $Sh\{\,.\,\}$ significa sharpening de $\{\,.\,\}$, y

$$K_1 \geq 2K_2. \tag{3.34}$$

La correspondiente respuesta en magnitud para (3.33) es

$$\left|H_{shCICmod}\left(e^{j\omega}\right)\right| =$$

$$\left|\left|\frac{1}{M_1}\frac{\sin\left(\frac{\omega M_1}{2}\right)}{\sin\left(\frac{\omega}{2}\right)}\right|^{K_2}\left[3\left(\frac{1}{M_2}\frac{\sin\left(\frac{\omega M}{2}\right)}{\sin\left(\frac{\omega M_1}{2}\right)}\right)^{2K_1} - 2\left(\frac{1}{M_2}\frac{\sin\left(\frac{\omega M}{2}\right)}{\sin\left(\frac{\omega M_1}{2}\right)}\right)^{3K_1}\right]\right|. \qquad (3.35)$$

Ejemplo 3.6: Se diseña el filtro SCICM con parámetros $M = 16$, $K_1 = 5$ y $K_2 = 2$, y se compara con el filtro SCIC original con $K = 3$. La Fig. 3.6 (a) muestra la respuesta en magnitud para estos filtros, así como también una aplicación en la banda de paso.

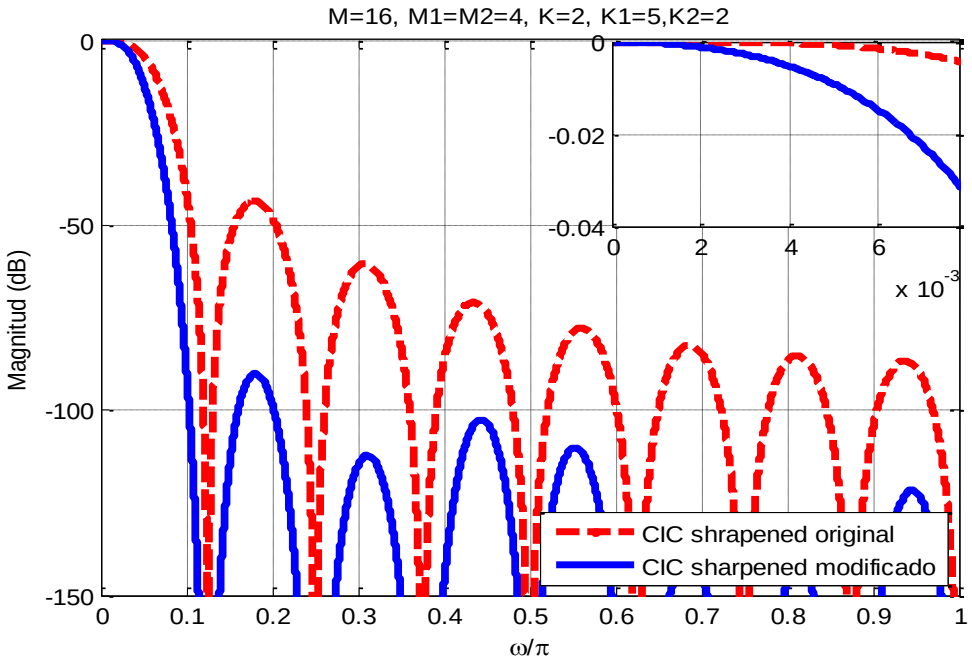


Fig. 3.6 Ilustración para el Ejemplo 3.6.

De la Fig. 3.6 se puede ver que el método de [18] permite mejorar la atenuación en la banda de rechazo, pero deteriora la caída en la banda de paso en relación con el filtro SCIC.

3.3 Métodos para disminuir el consumo de potencia

En este apartado se pueden encontrar métodos que tratan de mejorar a nivel hardware el consumo de potencia en la implementación de filtros digitales. También, se presenta un método para reducir el consumo de potencia en decimadores CIC no recursivos.

3.3.1 Métodos generales [23-33]

Multiplicaciones en base a sumas y corrimientos [23-25]

Para implementar los coeficientes de un filtro digital es necesario utilizar multiplicadores digitales, los cuales representan el mayor consumo de potencia/área en una implementación [23]. Existen varios algoritmos para realizar multiplicaciones en forma eficiente [24,25]. Tal es el caso del algoritmo de Höner presentado en [25], donde si la base usada es 2 (números binarios), la multiplicación se reduce a una determinada cantidad de corrimientos y sumas efectuadas sobre el multiplicando. Para realizar esto es necesario seguir los siguientes pasos:

1. Representar el coeficiente como una *suma de potencias de dos* (SPD). Esto se puede obtener a partir de la representación binaria del coeficiente.

2. Para cada término 2^n, presente en la SPD, aplicar un corrimiento en $n - bits$ a la izquierda sobre el multiplicando.

3. Sumar todos los corrimientos del multiplicando obtenidos del paso 2. El número de sumas necesarias será $b - 1$, donde b es el número de bits diferentes de cero de la representación binaria del coeficiente.

Ejemplo 3.7: Para implementar la multiplicación, en base a corrimientos y sumas, de un multiplicando cualquiera (x) por el coeficiente 14, primero se representa el coeficiente como SPD

$$14x = 2^3x+2^2x+2^1x. \tag{3.36}$$

Los términos $2^3, 2^2, 2^1$ indican aplicar un corrimiento a la izquierda por 3, 2 y 1 bits sobre el multiplicando x. Al final, se suman todos los corrimientos mediante dos sumadores.

Representación eficiente para los coeficientes [6]

De la sub-sección anterior podemos notar que la representación del coeficiente determina el número de operaciones de suma realizadas para su implementación como corrimientos y sumas. Debido a esto, existen representaciones eficientes que permiten reducir la cantidad de dígitos no cero en la representación de un coeficiente.

Una de ellas es la representación en formato *CSD* (*Canonical Signed Digit*), la cual forma parte de una representación denominada *SD* (*Signed Digit*) [6]. Estas difieren de la representación binaria común por estar formadas por la terna $\{\bar{1}, 0,1\}$, lo cual conduce hacia implementaciones más eficientes. Un coeficiente representado en CSD tendrá el mínimo número de dígitos no cero (mínima cantidad de $\{\bar{1}, 1\}$). Un coeficiente de *B* bits, solo puede tener un máximo de *B/2* bits no-cero. De esta manera, la representación CSD requiere de un máximo de (*B/2 – 1*) sumadores/restadores para su implementación.

Un método sencillo para obtener la representación en CSD es presentado en [6] y requiere seguir los siguientes pasos.

1. Representar el coeficiente en forma binaria sin signo.

2. En la representación binaria iniciando por la derecha, sustituir la primera cadena de unos como la de la parte izquierda de (3.37) por la cadena de la parte derecha de (3.37).

$$011 \dots 11 = 100 \dots 0\bar{1} \tag{3.37}$$

3. Concatenar la cadena de bits sustituidos en el paso anterior con los que no fueron sustituidos.

4. Repetir el paso dos desde el primer bit de la cadena que no ha sido sustituida y finalizar cuando ya no se pueda realizar la sustitución o se hayan cambiado todos los bits.

5. Si el coeficiente es negativo se siguen los pasos del 1 al 4 y al final se cambian todos los 1 por $\bar{1}$ y viceversa.

Ejemplo 3.8: Para implementar la multiplicación por el coeficiente 14 de forma eficiente, primero se convierte a su representación CSD como en (3.38) y luego se representa como SPD en (3.39)

$$14\,|_{10} = 1110\,|_2 = \; 100\bar{1}0\,|_{CSD} \tag{3.38}$$

$$14x = 2^4x - 2^1x. \tag{3.39}$$

Los términos 2^4, 2^1 indican aplicar un corrimiento a la izquierda por 3 y 1 bits sobre el multiplicando x. Al final, se restan los corrimientos en la forma indicada por (3.39), mediante un restador, el cual representa una complejidad similar a la de un sumador, debido a que los datos en un filtro generalmente se representan en complemento a dos [6].

Como punto final se puede mencionar que para hacer uso, tanto de la representación CSD como de la implementación de multiplicadores mediante corrimientos y sumas/restas, es necesario que los coeficientes del filtro sean números constantes y enteros. Los algoritmos de diseño generalmente arrojan coeficientes fraccionarios. Para estos casos se debe utilizar un factor de redondeo apropiado, α, para convertir los coeficientes en enteros y así poder aplicar corrimientos y sumas/restas para implementar los multiplicadores [26].

Métodos para la eliminación de sub-expresiones comunes [27-33]

Existen varios métodos para minimizar la cantidad de sumadores en la implementación de un filtro mediante la *compartición de sub-expresiones comunes* (CSC) [27-33]. Estos métodos utilizan diferentes representaciones eficientes para los coeficientes, como la CSD, y en base a un algoritmo buscan sub-expresiones comunes que permitan reutilizar hardware en la implementación de varios coeficientes que interactúan simultáneamente con la misma señal de entrada (multiplicando).

Ejemplo 3.9: Si los coeficientes 14 y 28 interactúan simultáneamente con la misma señal de entrada x, se puede implementar la multiplicación por el coeficiente 14 según lo descrito en el Ejemplo 3.8 y aplicándole un corrimiento en un bit a la izquierda se puede obtener la multiplicación por $28 = 14 \times 2$. De esta manera, se aprovecha el hardware usado para generar el 14 (sub-expresión común), haciendo esto se logra reducir tanto el consumo de potencia como de área usada en una implementación.

3.3.2 Descomposición polifásica eficiente para filtros CIC no recursivos [34]

Como se mostró en el capítulo 2, la estructura CIC no recursiva en componentes polifásicas tiene el menor consumo de potencia en comparación con las otras implementaciones. Sin embargo, este consumo puede reducirse aún más eligiendo correctamente los factores de sub-muestreo y el tipo de sumadores, tal y como se propone en [34]. Ese trabajo se basa en la descomposición del decimador CIC no recursivo polifásico en una primera etapa $H_1(z)$ con factor de sub-muestreo M_1, seguido por la cascada de filtros FIR $(1+z^{-1})^K$ con factor de sub-muestreo igual a 2. La razón de escoger este esquema se basa en el hecho de poder reducir la frecuencia de muestreo de la primera etapa tanto como sea posible. Las siguientes etapas se mantienen con el mínimo factor de sub-muestreo igual a 2. A continuación se presenta una breve descripción del trabajo presentado en [34].

La ecuación (2.9) puede ser escrita como

$$H(z) = H_1(z)H_2(z), \tag{3.40}$$

donde

$$H_1(z) = \left(\sum_{i=0}^{M_1-1} z^{-i} \right)^K, \tag{3.41}$$

$$H_2(z) = \prod_{i=0}^{(log_2(M/M_1))-1} \left(1 + z^{-2^i}\right)^K. \tag{3.42}$$

La expansión de $H_1(z)$ resulta en un filtro FIR de orden $K(M_1-1)$

$$H_1(z) = \sum_{n=0}^{K(M_1-1)} h(n)z^{-n}. \tag{3.43}$$

Los coeficientes de (3.43) son enteros y simétricos por lo que se obtiene la estructura de la Fig. 3.7 (a), donde se puede ver que el filtrado se lleva a cabo después del sub-muestreo, por lo tanto las sumas y multiplicaciones necesarias se realizan a una frecuencia de muestreo M_1-veces menor a la frecuencia de entrada.

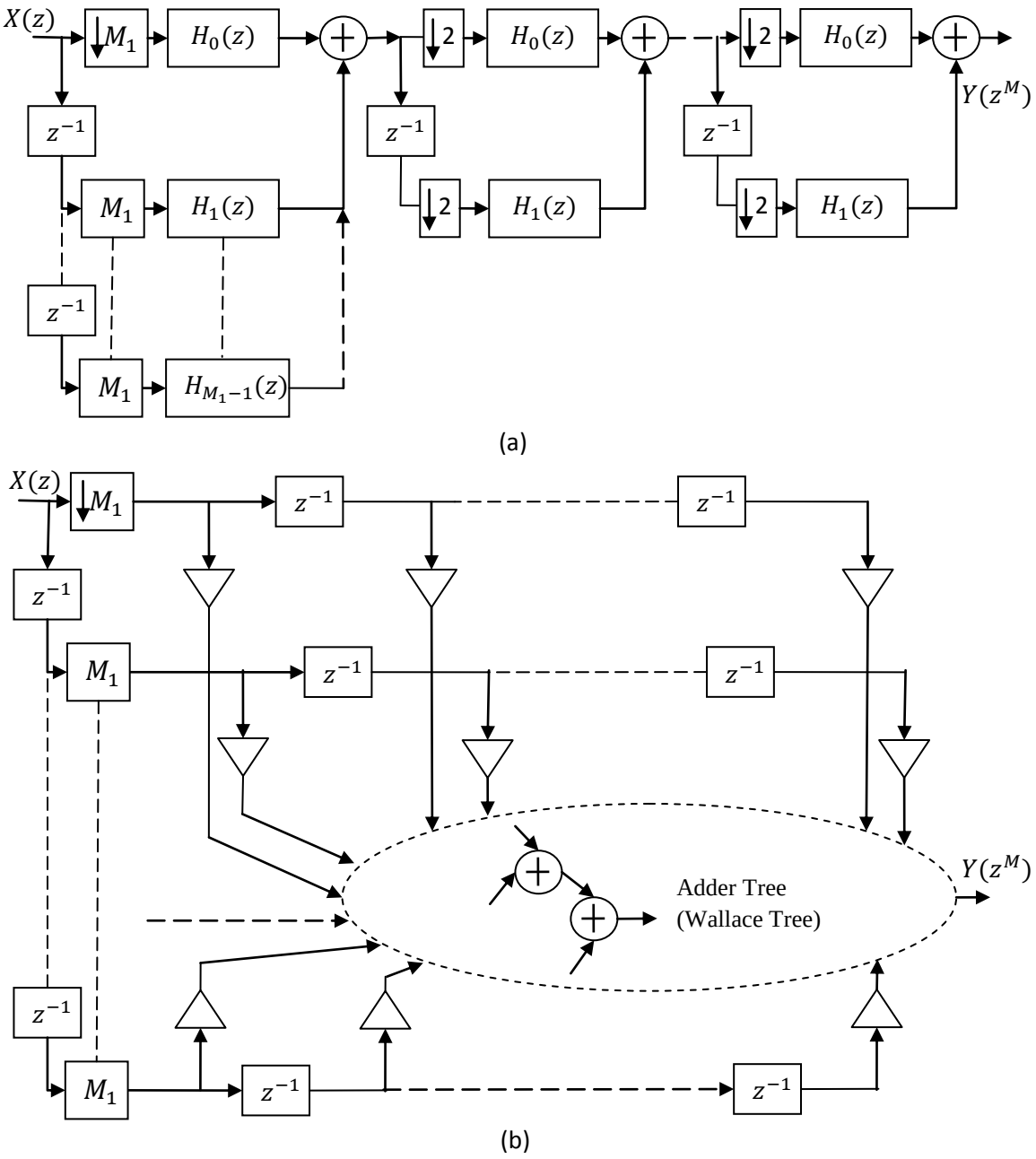


Fig. 3.7 Estructura decimador CIC no recursivo para bajo consumo de potencia propuesta en [34], (a) completa y (b) primera etapa eficiente.

Altos valores de M_1 reducirán significativamente la frecuencia de muestreo de la primer etapa, lo cual implica un menor consumo de potencia para las etapas subsecuentes. Por otra parte, a partir de (3.41) se puede ver que valores grandes de M_1 incrementarán el orden del filtro de la primera etapa lo que significa mayor complejidad y mayor número de productos

parciales. También se debe notar que la longitud de palabra de la entrada incrementará en $Klog_2(M_1)$, según (2.13).

Para implementar la primera etapa se utiliza un sumador en árbol eficiente (Wallace Tree) para la suma de los productos parciales de los sub-filtros. Este permite reducir la cantidad de área necesaria en la implementación, el cual es mostrado en la Fig. 3.7 (b).

Para evaluar las características de esta estructura los autores implementaron y simularon el consumo de potencia y área usada para varios decimadores con un factor de sub-muestreo $M = 32$ y $K = 5$, implementados en tecnología CMOS de 0.35µm. Probaron con diferentes combinaciones para el factor de sub-muestreo de la primera etapa $M_1 = 2, 4, 8, 16\ y\ 32$, y diferentes longitudes de palabra de la entrada.

Para decimadores con entrada *mono-bit* el mínimo consumo de potencia lo consiguieron eligiendo $M_1 = 16$. El peor rendimiento con $M_1 = 2$, el cual no es más que la estructura CIC polifásica. Comparando las dos implementaciones para $M_1 = 16$ y $M_1 = 2$ el consumo de potencia es reducido en un 30% y el área en 20%.

Para decimadores *multi-bit* los resultados no son muy relevantes y el mejor rendimiento se obtiene para $M_1 = 4$.

Capítulo 4

Método propuesto

En este capítulo primero se presentará el filtro coseno y a partir de sus características en frecuencia se derivará la estructura propuesta. Luego, se presentarán los parámetros de diseño para el decimador propuesto y de forma matemática se derivarán los criterios para la elección de sus valores. Finalmente, se presentarán dos diseños que usan la estructura propuesta y que satisfacen dos diferentes condiciones de diseño.

4.1 Filtro coseno

El filtro coseno tiene la función de transferencia presentada en (4.1), se llama de esta manera debido a que su respuesta en magnitud exhibe una característica coseno como la presentada en la Fig. 4.1 (a).

$$H_{cos}(z) = 0.5(1 + z^{-1}). \tag{4.1}$$

La ecuación (4.1) puede ser expandida por un factor N dando origen a

$$H_{cos}(z^N) = 0.5(1 + z^{-N}). \tag{4.2}$$

La respuesta en magnitud para (4.2) es

$$\left|H_{cos}\left(e^{j\omega N}\right)\right| = |cos(\omega N/2)|. \tag{4.3}$$

Al hacer la expansión en (4.2) el espectro original de (4.1) se repite $N-1$ veces en el intervalo de $[0,2\pi]$. Un ejemplo para $N = 4$ se presenta en la Fig. 4.1 (b).

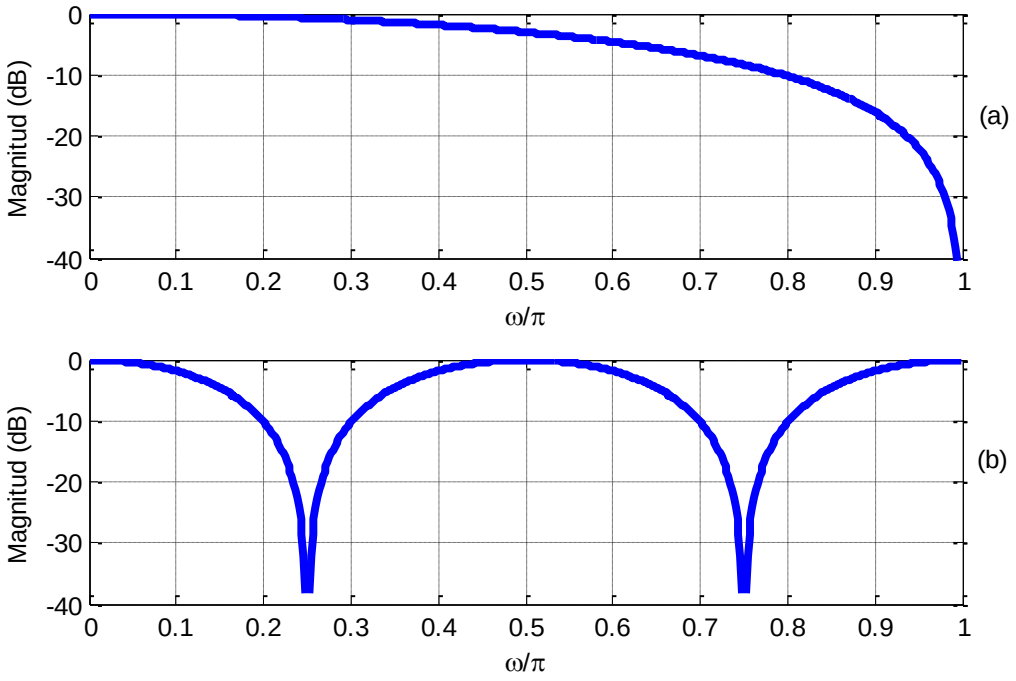


Fig. 4.1 Respuesta en frecuencia para (a) filtro coseno (b) coseno expandido $N = 4$.

A partir de (4.3) se puede ver que, de manera similar al filtro CIC, el filtro coseno expandido tiene ceros ubicados en frecuencias múltiplos de π/N. Si se conecta en cascada el filtro CIC no recursivo con el filtro coseno y se hace que $N = M/2$, se obtiene un cero adicional en el primer doblez de la banda del filtro CIC no recursivo. Como se mencionó a través de los capítulos 2 y 3, esto es muy útil porque permite aumentar *la atenuación en el primer doblez de la banda* (APDB), la cual es menor en relación con los demás dobleces. Para que N sea un valor entero se requiere que el factor de sub-muestreo sea un número par, lo cual se satisface para todos los casos donde el factor de sub-muestreo puede representarse como $M = 2^P$. De esta manera, la propuesta de este trabajo es aplicable a decimadores CIC no recursivos.

Como ejemplo de esto en la Fig. 4.2 se presenta la respuesta en magnitud para un filtro CIC no recursivo con $M = 8$ y $K = 4$, el filtro coseno correspondiente con $N = M/2 = 4$ y la cascada de ambos. También se presenta una ampliación en el *primer doblez de la banda* (PDB), donde se puede notar un aumento en la anchura y por consiguiente un aumento en la APDB.

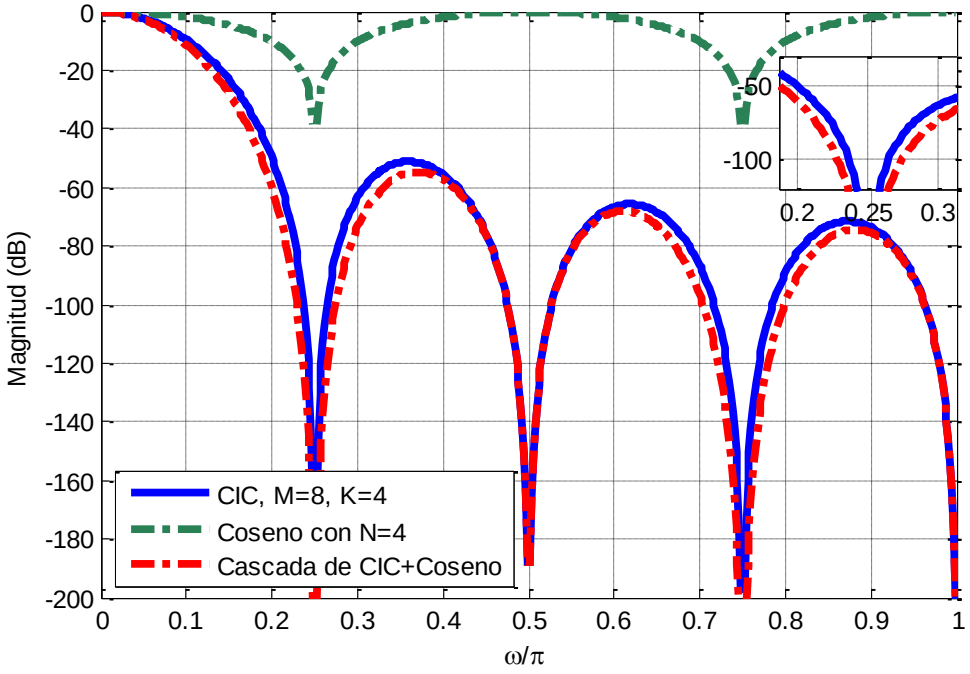


Fig. 4.2 Filtro CIC con $M = 8$ y $K = 4$, coseno con $N = 4$, y cascada.

4.2 Estructura propuesta

El filtro propuesto se obtiene a partir de (2.22) y (4.2), considerando que el factor de sub-muestreo se puede expresar como una potencia de dos $M = 2^p$, y $N = M/2$. De esta manera, la función de transferencia para el filtro propuesto es

$$H_p(z) = \left[\prod_{i=0}^{P-1} \frac{1}{2}\left(1 + z^{-2^i}\right)\right]^{K_1} \left[\frac{1}{2}\left(1 + z^{-2^{(P-1)}}\right)\right]^{K_2}, \tag{4.4}$$

donde los parámetros K_1 y K_2 son los parámetros de diseño, y como se verá más adelante, dependerán del objetivo de diseño. La respuesta en frecuencia para (4.4) es

$$\left|H_p\left(e^{jw}\right)\right| = \left|\left[\frac{1}{M}\frac{\sin\left(\frac{\omega M}{2}\right)}{\sin\left(\frac{\omega}{2}\right)}\right]^{K_1} \left[cos\left(\frac{\omega M}{4}\right)\right]^{K_2}\right|. \tag{4.5}$$

La estructura propuesta se presenta en la Fig. 4.3 (a), la cual se obtiene usando el filtro propuesto de (4.4) como parte de un decimador (filtro antialiasing), donde se ha aplicado la Tercera identidad multi-razón, debido a que el factor de sub-muestreo es $M = 2^P$.

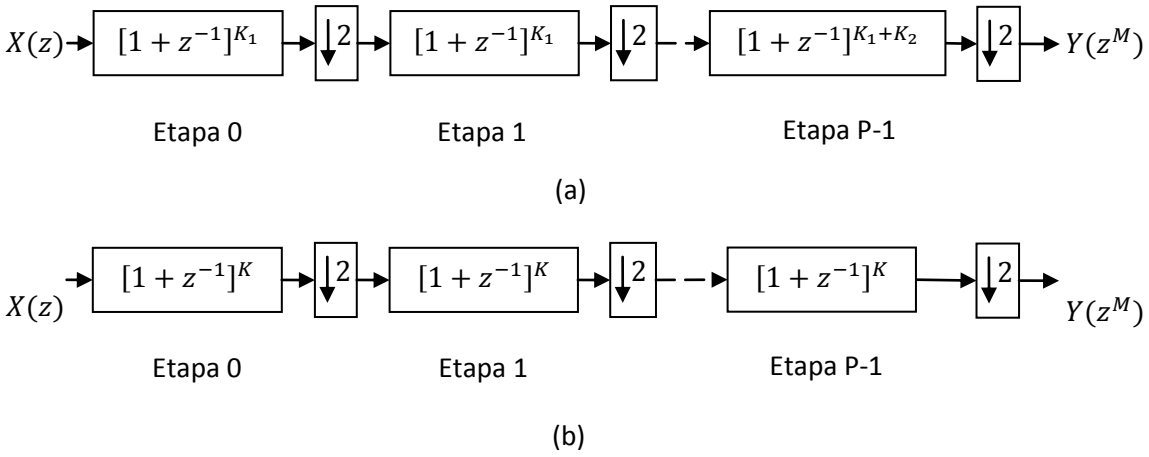


Fig. 4.3 (a) Estructura propuesta, (b) estructura CIC no recursiva.

En la Fig. 4.3 (b), por conveniencia, se vuelve a presentar la estructura del decimador CIC no recursivo, donde se puede ver que la estructura propuesta tiene únicamente una cascada adicional de K_2 filtros en la última etapa, lo que provee una atenuación adicional en todos los dobleces de la banda que son impares (dentro de los cuales se encuentra el primer doblez). Como ejemplo de esto considérese la Fig. 4.4, donde se presenta la respuesta en magnitud para la estructura propuesta con parámetros $M = 8$, $K_1 = 3$ y $K_2 = 3$ junto a la correspondiente de los decimadores CIC no recursivos con $K = 3$ y $K = 4$.

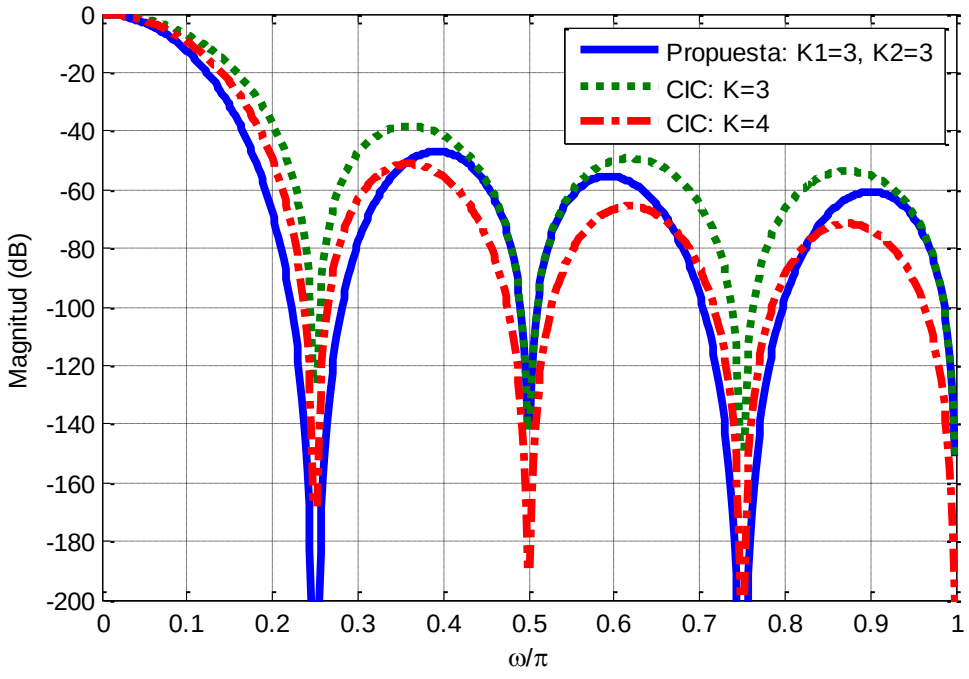


Fig. 4.4 Respuesta en magnitud para la estructura propuesta con $M = 8$, $K_1 = 3$ y $K_2 = 3$.

En la Fig. 4.4 se puede ver que la estructura propuesta tiene una mejor atenuación en el primer doblez de la banda, y en los subsecuentes dobleces impares, que los decimadores CIC no recursivos con $K = 3$ y $K = 4$. Sin embargo, el decimador CIC no recursivo con $K = 4$ tiene mejor atenuación en todos los dobleces de la banda que son pares. De esta manera, para que la estructura propuesta tenga la mínima anchura en los dobleces que son pares en comparación con la estructura CIC no recursiva, es necesario establecer el parámetro $K_1 = K$.

Finalmente, el número de *sumas por muestra de salida* (SPMS) para el decimador propuesto de la Fig. 4.3 (a) está definido por

$$SPMS_{Propuesta} = K_1(2^P + 2^{P-1} \dots 2) + 2K_2 = 2K_1(2^P - 1) + 2K_2, \quad (4.6)$$

mientras que para el decimador CIC no recursivo de la Fig. 4.3 (b) es

$$SPMS_{CIC\ no\ recursivo} = K(2^p + 2^{p-1} \dots + 2) = 2K(2^p - 1). \quad (4.7)$$

De (4.6) y (4.7) se puede ver un incremento en $2K_2$ SPMS de la estructura propuesta en comparación con la estructura CIC no recursiva.

El siguiente paso es la correcta elección de los parámetros K_1 y K_2 para satisfacer un objetivo de diseño especifico, lo cual será presentado en las secciones siguientes.

4.3 Elección de los parámetros de diseño

Como se pudo ver en la sub-sección anterior el filtro propuesto de (4.4) permite mejorar la APDB del filtro CIC no recursivo, para obtener una APDB deseada y/o requerida es necesario calcular el orden para el filtro CIC no recursivo y para el filtro coseno, K_1 y K_2, respectivamente. Esto se hace calculando la respuesta en magnitud del filtro CIC no recursivo y del filtro coseno en la frecuencia de (2.16), para obtener la APDB del filtro propuesto. Para el filtro CIC no recursivo este valor es

$$H_{CIC}\left(e^{j\omega_A}\right) = \left[\frac{\sin\left(\frac{\pi(2R-1)}{RM}\frac{M}{2}\right)}{\mathrm{M} * \sin\left(\frac{\pi(2R-1)}{RM}\frac{1}{2}\right)}\right]^{K_1}. \quad (4.8)$$

Considerando $M \gg R$

$$\sin(x) \approx x, \tag{4.9}$$

la ecuación (4.8) se puede aproximar a

$$H_{CIC}\left(e^{j\omega_A}\right) \approx \left[\frac{\sin\left(\frac{\pi(2R-1)}{RM}\frac{M}{2}\right)}{\frac{\pi(2R-1)}{2R}}\right]^{K_1}. \tag{4.10}$$

De forma similar la respuesta en magnitud en la frecuencia (2.16) para el filtro coseno es

$$H_{cos}\left(e^{j\omega_A}\right) = \left[\cos\left(\frac{\pi(2R-1)}{RM}\frac{M}{4}\right)\right]^{K_2} = \left[\cos\left(\frac{\pi(2R-1)}{4R}\right)\right]^{K_2}. \tag{4.11}$$

Finalmente, la APDB (en la frecuencia (2.16)) para el filtro propuesto es

$$H_p\left(e^{j\omega_A}\right) \approx \left[\frac{\sin\left(\frac{\pi(2R-1)}{RM}\frac{M}{2}\right)}{\frac{\pi(2R-1)}{2R}}\right]^{K_1}\left[\cos\left(\frac{\pi(2R-1)}{4R}\right)\right]^{K_2}. \tag{4.12}$$

La ecuación (4.10) demuestra que la APDB en el filtro CIC no recursivo prácticamente no depende del factor de sub-muestreo M, solo de K y R, de forma similar para el filtro propuesto en (4.12) la APDB solo depende de los valores para K_1 , K_2 y R.

Considerando $R = 2$ y usando (4.10) se puede encontrar la APDB para el filtro CIC no recursivo en función del parámetro K, y de (4.12) se puede encontrar la APDB para el filtro propuesto en función del parámetro K_1 para diferentes valores de K_2. La Fig. 4.5 muestra la APDB para el filtro CIC no recursivo con $R = 2$ y valores de $K = 1, \dots, 6$, junto a la del filtro propuesto con $R = 2$, $K_1 = K$ y diferentes valores para $K_2 = 1, \dots, 6$. Como se verá más adelante, esta figura constituye la principal referencia para la determinación de los parámetros de diseño según sea el objetivo de diseño.

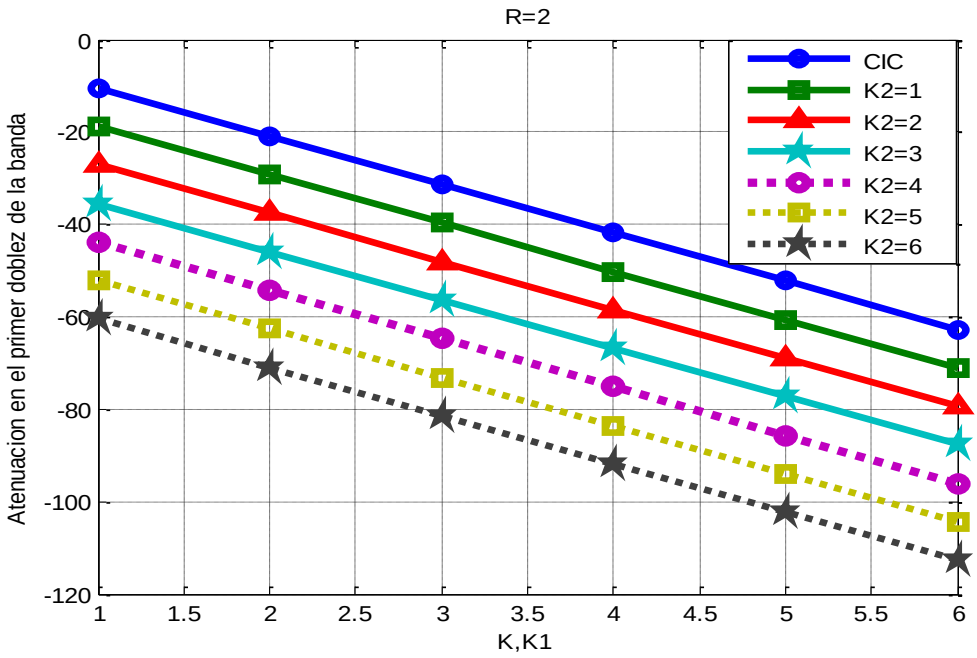


Fig. 4.5 Atenuación en el primer doblez de la banda para el filtro CIC y el filtro propuesto.

4.4 Mejora de atenuación en el primer doblez de la banda:

DISEÑO 1

Para mejorar la atenuación en el primer doblez de la banda de un decimador CIC no recursivo con una cascada K determinada, la solución es la estructura propuesta de la Fig. 4.3 (a), donde $K_1 = K$ y K_2 se obtiene de la Fig. 4.5 en la intersección de K_1 con la mínima APDB deseada. Por ejemplo, un decimador CIC no recursivo con $K = 4$ tendrá una APDB de -40dB. Sin embargo, si se desea una APDB de -50dB, se usa la estructura propuesta, manteniendo $K_1 = K = 4$ y eligiendo $K_2 = 1$, según la intersección de K_1 con la APDB en la Fig. 4.5. Así mismo, si se desea una mínima APDB de -70dB se debe elegir $K_2 = 4$, etc.

Considérese un decimador CIC no recursivo con parámetros $M = 32$ y $K = 5$, el cual tiene una APDB de -50dB. Sin embargo, se desea mejorar la APDB a un valor mínimo de -70dB. Para lograr este objetivo se utiliza la estructura propuesta haciendo $K_1 = K = 5$. Luego, en la Fig. 4.5 se busca la intersección de $K_1 = 5$ y -70dB, y se obtiene $K_2 = 3$. La Tabla 4.1

resume las características y parámetros para este decimador derivado de la estructura propuesta con $K_1 = 5$ y $K_2 = 3$, el cual de ahora en adelante será referido como **Diseño 1**.

Decimador	M	K_1	K_2	APDB
CIC no recursivo	32	5	---	-50dB
Diseño 1	32	5	3	-70dB

Tabla 4.1 Resumen de características y parámetros del decimador CIC no recursivo y **Diseño 1**.

La estructura para el decimador CIC no recursivo con $M = 32$ y $K = 5$ se muestra en la Fig. 4.6 (a), donde se puede ver que consta de cinco etapas, cada una con un valor de cascada $K = 5$. La estructura para el **Diseño 1** se obtiene a partir de la estructura propuesta de la Fig. 4.3 (a), y es mostrada en la Fig. 4.6 (b), donde las primeras cuatro etapas tienen un valor de cascada $K_1 = 5$ y la última tiene $K_1 + K_2 = 8$.

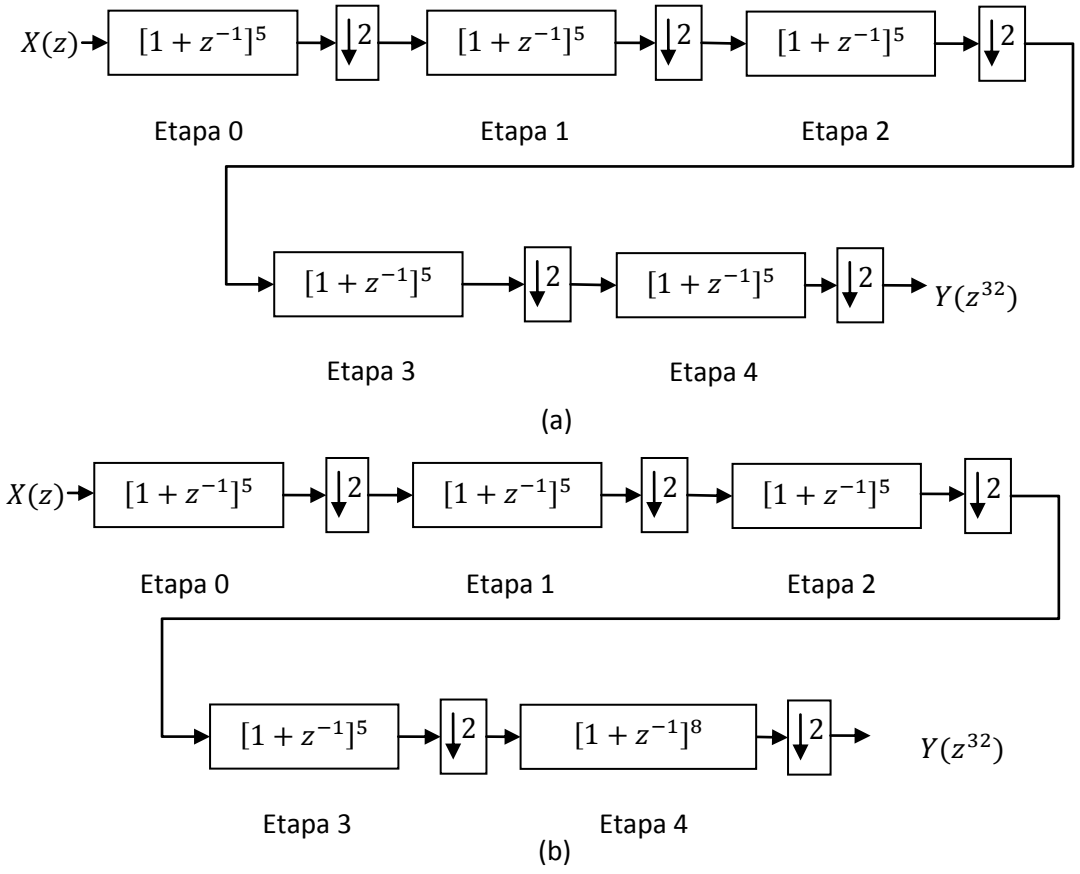


Fig. 4.6 Estructura para el decimador (a) CIC no recursivo con $M = 32$ y $K = 5$ y (b) **Diseño 1**.

En las ecuaciones (4.13) y (4.14) se muestra la cantidad de SPMS del decimador CIC no con $M = 32$ y $K = 5$ y del **Diseño 1**, de la Fig. 4.6 (a) y (b), respectivamente.

$$SPMS_{CIC \text{ con } M=32,K=5} = 2 \times 5(2^5 - 1) = 310. \quad (4.13)$$

$$SPMS_{Diseño\ 1} = 2 \times 5(2^5 - 1) + 2 \times 3 = 316. \quad (4.14)$$

La Fig. 4.7 muestra la respuesta en magnitud para el decimador CIC no recursivo con $M = 32$, $K = 5$ y el **Diseño 1**, y una ampliación en el primer doblez de la banda, donde se puede comprobar el aumento en APDB del **Diseño 1**. Como se mencionó con anterioridad, el primer doblez de la banda del **Diseño 1** es más amplio y con un incremento en atenuación. Lo mismo ocurre para aquellos dobleces impares. Sin embargo, los dobleces que son pares tienen la misma anchura tanto en el decimador CIC no recursivo con $M = 32$, $K = 5$ como en el **Diseño 1,** pero la atenuación en el propuesto es ligeramente mayor.

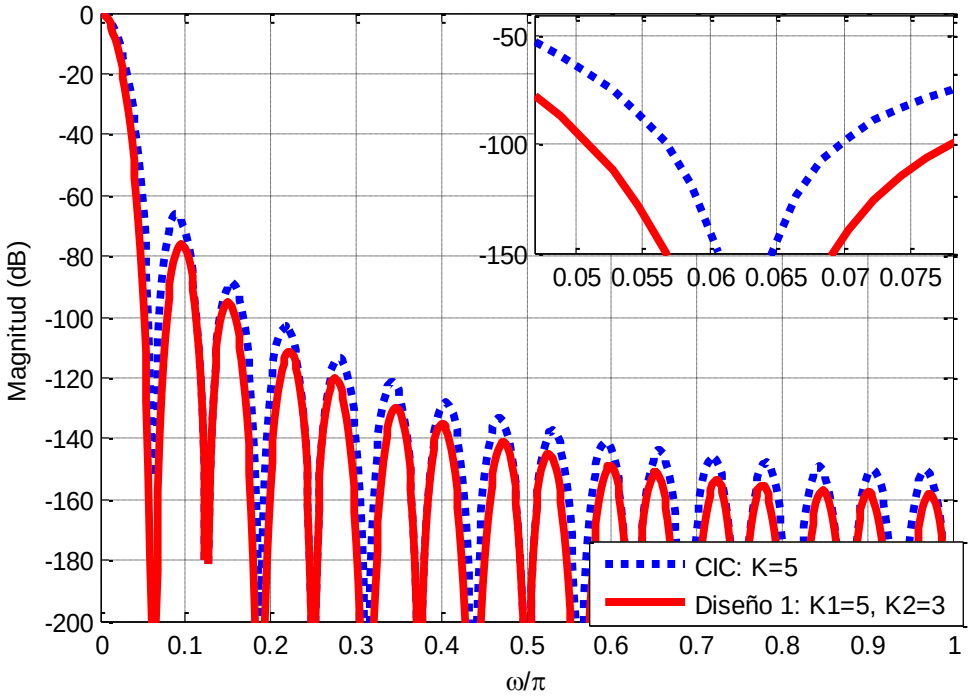


Fig. 4.7 Respuesta en magnitud para el decimador CIC no recursivo con $M = 32$, $K = 5$ y el **Diseño 1**.

Comparando las respuestas en magnitud y las estructuras del decimador CIC no recursivo con $M = 32$, $K = 5$ y del **Diseño 1** con $M = 32$, $K_1 = 5$ y $K_2 = 3$, es posible observar que el **Diseño 1** consigue un considerable incremento en la APDB a la vez que mantiene la

estructura básica del decimador CIC no recursivo, donde el único cambio es en la cantidad de cascada de la última etapa, lo cual resulta en un ligero aumento, del 1.94%, en la cantidad de SPMS del **Diseño 1** en comparación al decimador CIC no recursivo, por lo cual, en una implementación, podemos esperar un aumento en el consumo de potencia relativamente pequeño.

4.4.1 Comparación con otros métodos

En el capítulo 3 se pueden encontrar diversos métodos para mejorar las características de respuesta en magnitud de los decimadores CIC. El **Diseño 1**, realiza una mejora en la atenuación de la banda de rechazo, por lo cual podemos compararlo con los métodos presentados en [14] y [15], de la Sección 3.2.2.

Para realizar la comparación del **Diseño 1** con los métodos presentados en [14] y [15], se diseñan los decimadores con características similares al **Diseño 1**, es decir, para $M = 32$ y una APDB de al menos -70dB.

A. Método de [14]

Este método, además del decimador, utiliza un filtro compensador para la CBP, sin embargo éste no será considerado, debido a que el método propuesto de este trabajo de tesis no utiliza compensación para la CBP.

Para el método de [14] con un factor de sub-muestreo $M = 32$, el coeficiente a se expresa como

$$a = 2\cos(32\beta) \tag{4.15}$$

donde el valor optimo para β es 0.038779. Después, se desarrolla este término mediante la ecuación (3.16) para que solo dependa del término $\cos(\beta)$.

Finalmente, se cuantiza el término $\cos(\beta)$ como

$$\cos(\beta) = 1 - 2^{11}, \tag{4.16}$$

La estructura para este método, que se obtiene usando (4.15) y (4.16), es demasiado grande para ser mostrada por lo que sólo se presenta como bloques de la función de transferencia, lo cual puede verse en la Fig. 4.8.

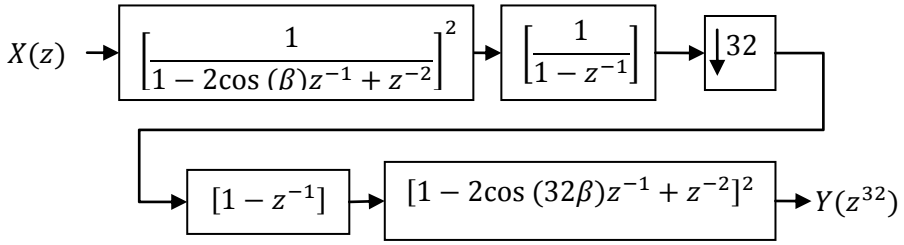


Fig. 4.8 Estructura para el decimador del método [14] con características similares al **Diseño 1**.

En la Fig. 4.9 se puede encontrar la comparación de la respuesta en magnitud para el **Diseño 1** y el método de [14], así como también una ampliación en el PDB.

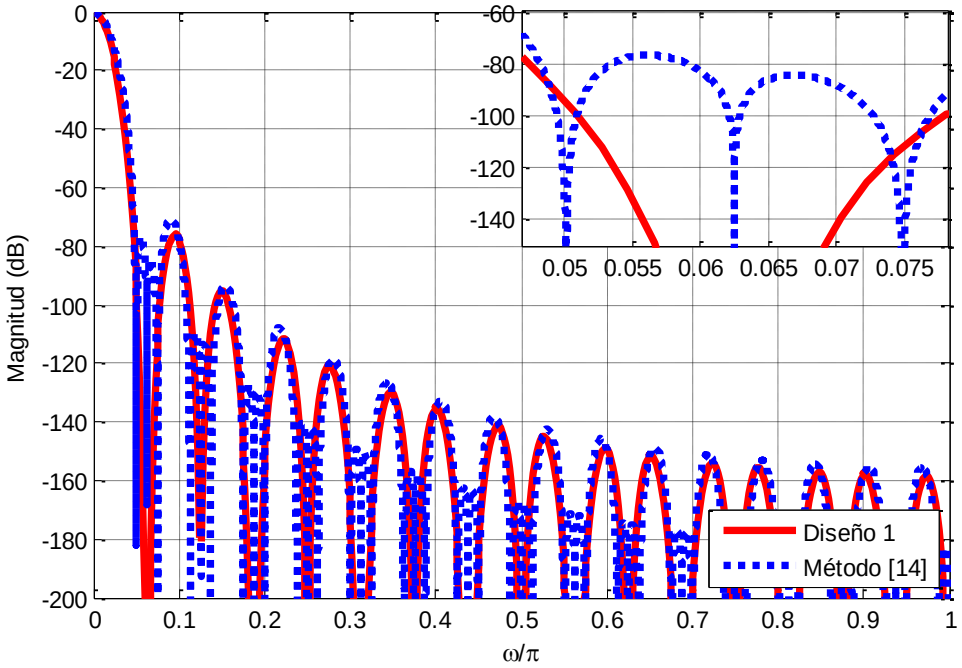


Fig. 4.9 Respuesta en magnitud para el **Diseño 1** y el método de [14].

A partir de la Fig. 4.9, se puede ver que el método [14] tienen una respuesta en magnitud similar a la obtenida con el **Diseño 1.** Sin embargo, su complejidad de diseño e implementación en comparación con el decimador CIC no recursivo se ve considerablemente incrementada (Fig. 4.8), debido a que el término $2\cos(32\beta)$, se debe

sustituir por todas las sumas/restas obtenidas mediante (3.16) (las cuales no han sido mostradas aquí, pero basta saber que generan términos $\cos^{n}(\beta)$, donde n = 0,2,4…32). Además, este método no puede ser implementado mediante la estructura CIC no recursiva a pesar de que $M = 2^P$. En este sentido, el **Diseño 1** representa una mejor opción, ya que el diseño e implementación se mantiene tan simple como en el decimador CIC no recursivo.

B. Método de [15]

Para el método de [15], con características similares a las del **Diseño 1**, se obtienen los parámetros

$$\beta = \frac{2\pi}{M} - \frac{\pi}{(\beta_0 + 2)M} = \frac{2\pi}{32} - \frac{\pi}{(0.99 + 2)32} = 0.1635, \qquad (4.17)$$

$$k = \frac{1}{2 - 2\cos(\beta)} = 37.48. \qquad (4.18)$$

De esta manera, el término ZTR se expresa como:

$$H_{ZTR}(z) = k(1 - 1.9733z^{-1} + z^{-2}). \qquad (4.19)$$

Para conseguir la APDB de al menos -70dB se debe usar un valor de $K = 6$. Este método si permite la implementación mediante la estructura no recursiva para cuando $M = 2^P$, debido a que el término ZTR que se introduce no interfiere de manera directa con la estructura del decimador CIC no recursivo. No obstante, el término ZTR trabaja a la frecuencia de muestreo de entrada, lo cual se puede ver en su estructura presentada en la Fig. 4.10.

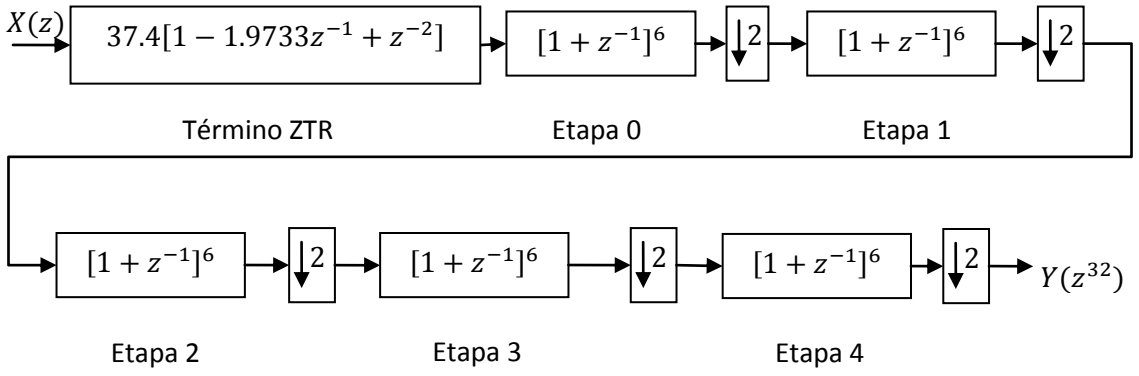


Fig. 4.10 Estructura para el decimador del método [15] con características similares al **Diseño 1**.

La Fig. 4.11 presenta la respuesta en magnitud para el **Diseño 1** y el decimador del método [15], y una ampliación en el primer doblez de la banda, donde se puede ver que ambos presentan una APDB muy similar, solo que el **Diseño 1** tiene un ligero aumento en la atenuación de los demás dobleces.

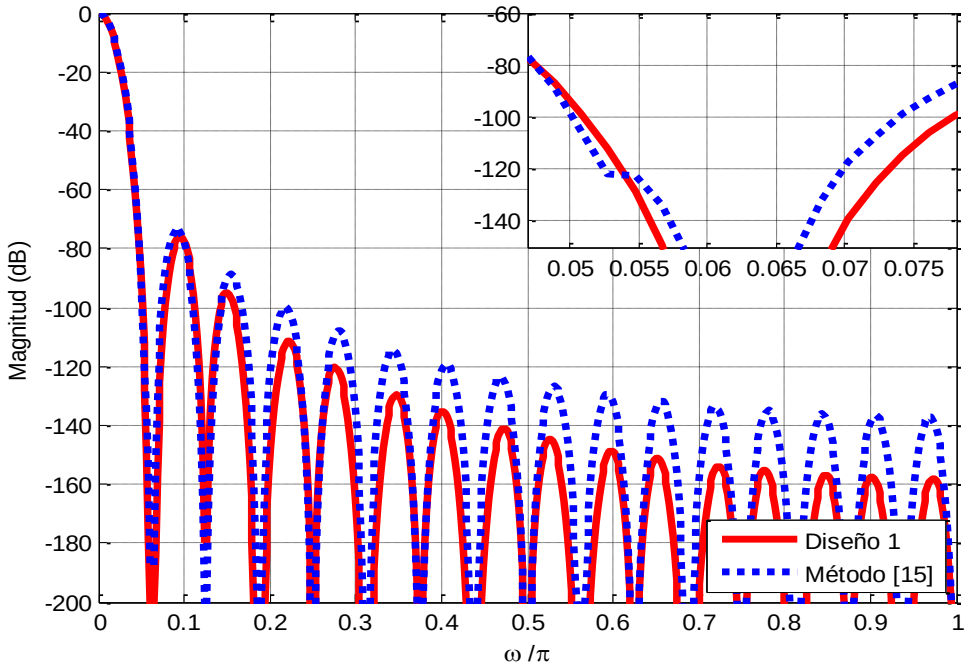


Fig. 4.11 Respuesta en magnitud para el **Diseño 1** y el método de [15].

A partir de la Fig. 4.11 es posible ver que la respuesta en magnitud del método [15] es muy similar a la del **Diseño 1**. Sin embargo, la estructura para el método [15] también cambia la forma básica del decimador CIC no recursivo al agregar el término ZTR, el cual requiere el uso de una constante de redondeo, α, para aproximar el coeficiente de (4.19) mediante corrimientos y sumas/restas, para evitar el uso de multiplicadores. En contraste, el **Diseño 1** conserva la misma forma básica de la estructura CIC no recursiva y solo modifica el valor de la cascada en la última etapa. Por esta razón, en los capítulos posteriores sobre implementación y simulación, la comparación del método propuesto se enfatizará contra los decimadores CIC no recursivos, porque conserva la misma estructura básica pero permite mejorar las características en frecuencia mediante un ligero cambio en el número de cascada de la última etapa.

4.5 Ahorro de consumo de potencia: DISEÑO 2

En esta estructura el objetivo es mantener el consumo de potencia tan bajo como sea posible manteniendo una mínima atenuación en todos los dobleces de la banda alrededor de los ceros del decimador CIC no recursivo. Para lograr este objetivo nuevamente se utiliza la estructura propuesta de la Fig. 4.3 (a). Luego, en la Fig. 4.5 se localiza la APDB deseada y de forma horizontal, a este punto, se buscan todas las posibles intersecciones para los parámetros K_1 y K_2 que proporcionen este valor. Para que la mínima atenuación deseada ocurra en todos los dobleces de la banda es necesario establecer la condición de que la *atenuación en el segundo doblez de la banda* (ASDB) debe ser menor ó igual a la APDB.

La frecuencia para la ASDB está determinada por

$$\omega_{A2} = \frac{4\pi}{M} - \frac{\pi}{RM}. \tag{4.20}$$

De esta manera la condición a satisfacer es

$$\left|H_P\left(e^{j\omega_{A2}}\right)\right| \leq \left|H_P\left(e^{j\omega_A}\right)\right|. \tag{4.21}$$

De (4.12) se conoce el término de la derecha para (4.21), mientras que el término de la izquierda se calcula sustituyendo (4.20) en (4.5), de esta manera la condición (4.21) se convierte en

$$\left|\left[\frac{\sin\left(7\pi/4\right)}{\left(7\pi/4\right)}\right]^{K_1}\left[\cos(7\pi/8)\right]^{K_2}\right| \leq \left|\left[\frac{\sin\left(3\pi/4\right)}{\left(3\pi/4\right)}\right]^{K_1}\left[\cos(3\pi/8)\right]^{K_2}\right|. \tag{4.22}$$

Despejando los valores para los parámetros de diseño K_1 y K_2 se llega a

$$K_1 \geq 1.063K_2. \tag{4.23}$$

Una vez más a partir de (4.22) se puede ver que la condición no depende del factor de sub-muestreo M, y debido a que la respuesta en magnitud del decimador de la estructura propuesta (Fig. 4.3 (a)) exhibe una característica decadente, solo es necesario verificar (4.23) y elegir aquellas parejas de K_1 y K_2 que cumplan esta condición.

Considérese un decimador CIC no recursivo con los parámetros $M = 16$ y $K = 6$, es decir, con una APDB mínima de -60dB. Sin embargo se desea disminuir el consumo de potencia manteniendo ese valor de APDB de -60dB. Para cumplir este objetivo se usa la estructura propuesta, de la Fig. 4.3 (a). Luego, en la Fig. 4.5 se ubica el punto de atenuación de -60dB y se recorre de manera horizontal para hallar todas las posibles combinaciones para K_1 y K_2. Estas combinaciones son mostradas en la Tabla 4.2 y resaltadas aquellas que cumplen la condición (4.23).

K_1	K_2	¿Satisface condición (4.23)?	APDB (dB)	SPMS
2	5	No	---	---
3	4	No	---	---
4	**3**	**Si**	**-66.8477**	**126**
5	**1**	**Si**	**-60.6158**	**152**

Tabla. 4.2 Elección de los parámetros K_1 y K_2 para ahorro de potencia.

Las dos combinaciones de la Tabla 4.2 que cumplen con la condición (4.23), de ahora en adelante, podrán ser referidas como **Diseño 2**. En la Tabla. 4.3 se puede encontrar un resumen de sus características así como también para el decimador CIC no recursivo correspondiente.

Decimador	M	K_1	K_2	APDB
CIC no recursivo	16	6	---	-60dB
Diseño 2 – Caso 1	16	5	1	-60dB
Diseño 2 – Caso 2	16	4	3	-66dB

Tabla 4.3 Resumen de características y parámetros del decimador CIC no recursivo y **Diseño 2**.

La estructura que permite implementar el decimador CIC no recursivo con $M = 16$ y $K = 6$ se muestra en la Fig. 4.12 (a). Las estructuras de bajo consumo de potencia que se obtienen a partir de la estructura propuesta, sustituyendo los valores de los parámetros K_1 y K_2 del **Diseño 2 – Caso 1** y **2, s**e muestran en la Fig. 4.12 (b) y (c), respectivamente.

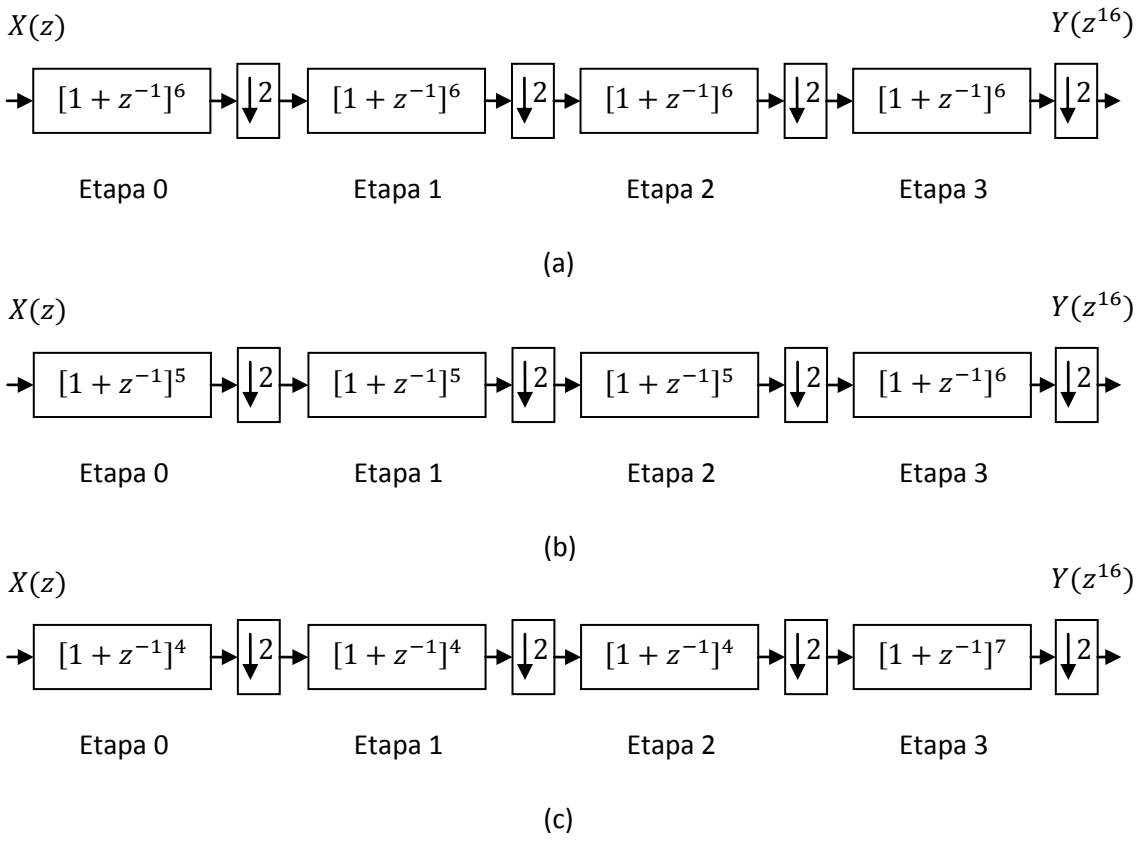


Fig. 4.12 (a) Decimador CIC no recursivo con parámetros $M = 16$ y $K = 6$, (b) **Diseño 2 – Caso 1**, (c) **Diseño 2 – Caso 2.**

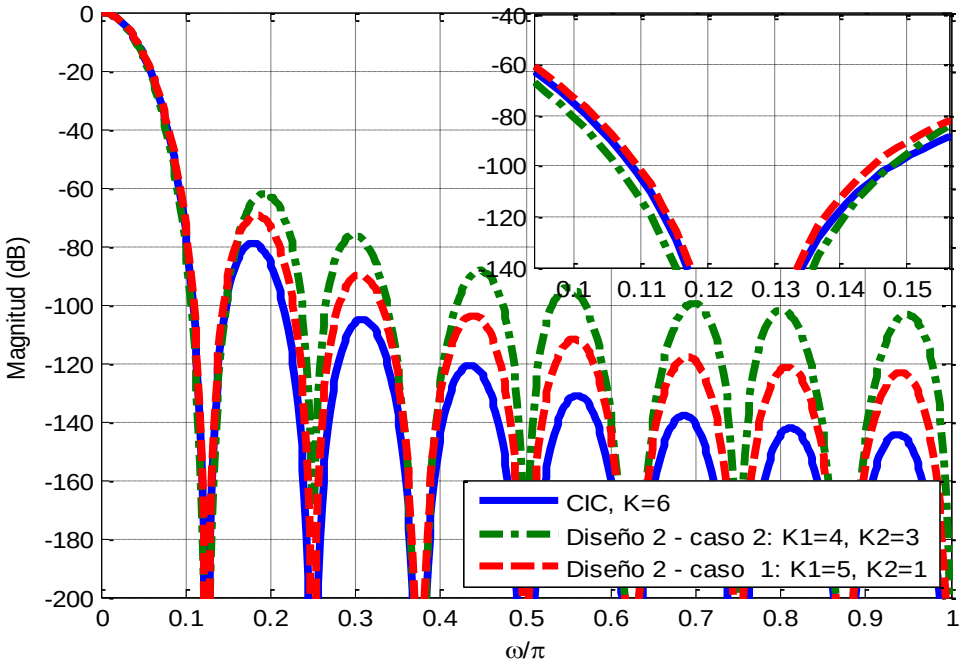
La cantidad de SPMS para el decimador CIC no recursivo con $M = 16$ y $K = 6$, y el **Diseño 2 – Caso 1 y 2** se pueden encontrar en (4.24) - (4.26).

$$SPMS_{CIC \text{ con } M=16 \text{ y } K=6} = 2 \times 6(2^4 - 1) = 180. \quad (4.24)$$

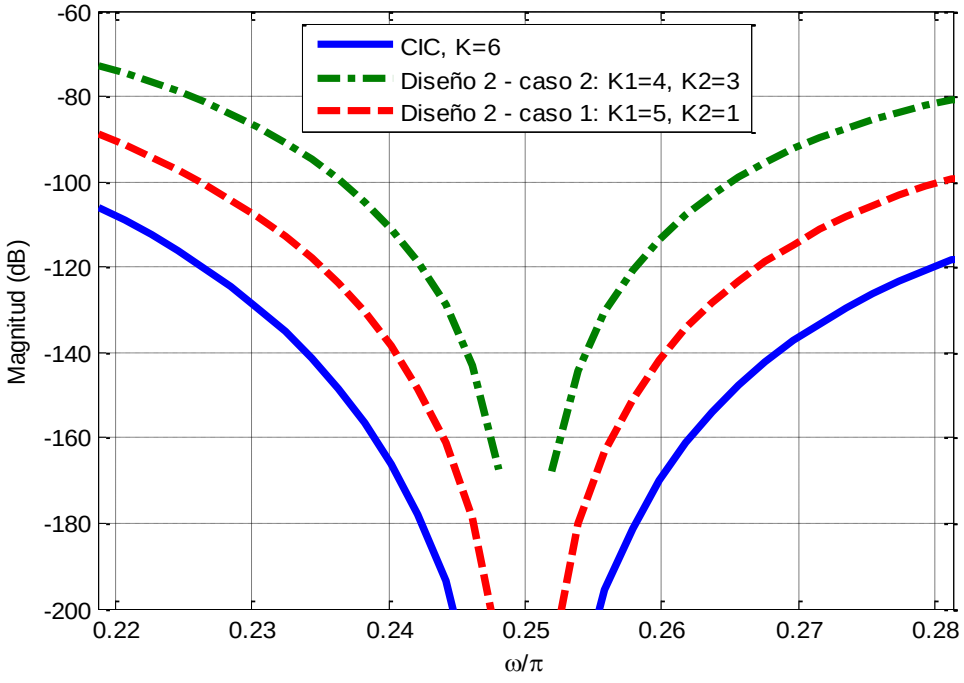
$$SPMS_{Caso\ 1} = 2 \times 5(2^4 - 1) + 2 \times 1 = 152. \quad (4.25)$$

$$SPMS_{Caso\ 2} = 2 \times 4(2^4 - 1) + 2 \times 3 = 126. \quad (4.26)$$

En la Fig. 4.13 (a) se muestra la respuesta en magnitud para el decimador CIC no recursivo con $M = 16$ y $K = 6$, junto a la del **Diseño 2 - Caso 1** y **2**, además de una ampliación en el primer doblez de la banda, el cual permite comprobar que la APDB cumple con el valor mínimo establecido de -60dB. En la Fig. 4.13 (b) se muestra una ampliación para el segundo doblez de la banda que permite comprobar una mayor ASDB con respecto a la APDB.



(a)



(b)

Fig. 4.13 (a) Respuesta en magnitud para el decimador CIC no recursivo con $M = 16$, $K = 6$ y el **Diseño 2 – Caso 1** y **2**, (b) ampliación en el segundo doblez de la banda.

Comparando las respuestas en magnitud y las estructuras del decimador CIC no recursivo con $M = 16$ y $K = 6$, del **Diseño 2 – Caso 1** con $M = 16, K_1 = 5$ y $K_2 = 1$, y del **Diseño 2 – Caso 2** con $M = 16, K_1 = 4$ y $K_2 = 3$, es posible observar que ambas estructuras de bajo consumo de potencia consiguen la mínima APDB requerida de -60dB a la vez que mantienen la estructura básica del decimador CIC no recursivo, donde el único cambio es en los dos parámetros de diseño K_1 y K_2, lo cual reduce considerablemente la cantidad de SPMS requeridas, por lo que, en una implementación, se puede esperar una reducción considerable en el consumo de potencia.

4.5.1 Comparación con otros métodos

A. Método de [14] y [15]

Para la comparación del **Diseño 2 – Caso 1** y **2** con los métodos de [14] y [15], se diseñan los decimadores de los métodos [14] y [15] con $M = 16$ y una APDB mínima de -60dB, las estructuras correspondientes se muestran en la Fig. 4.14 (a) y (b), respectivamente.

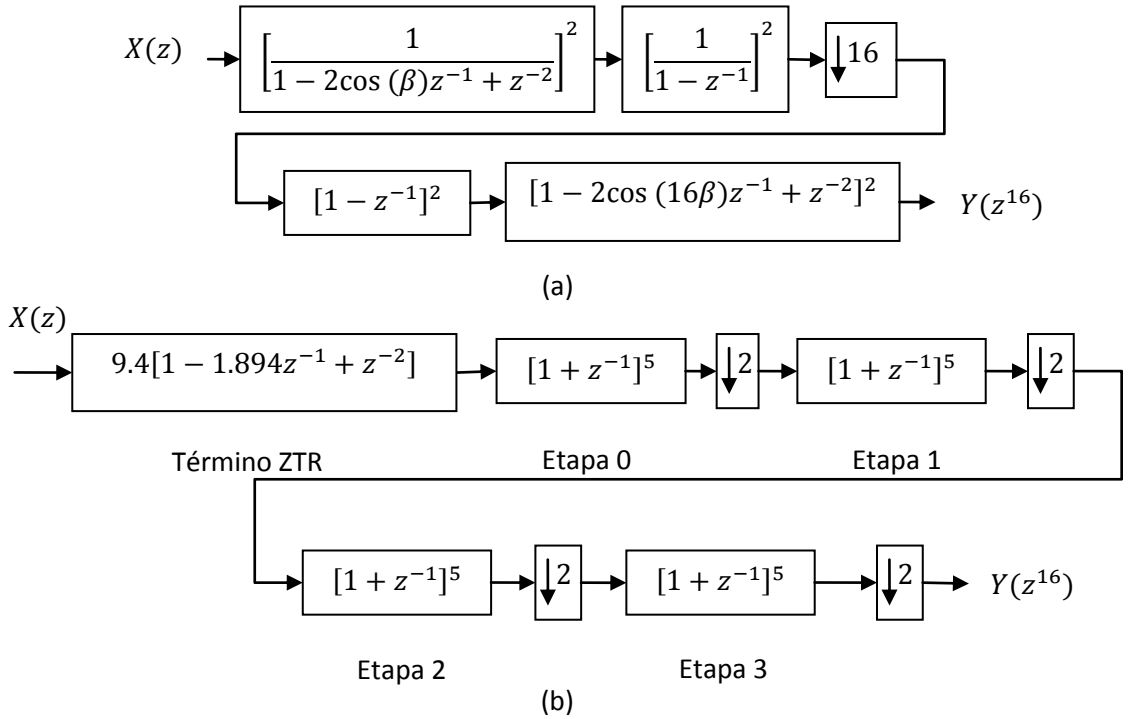
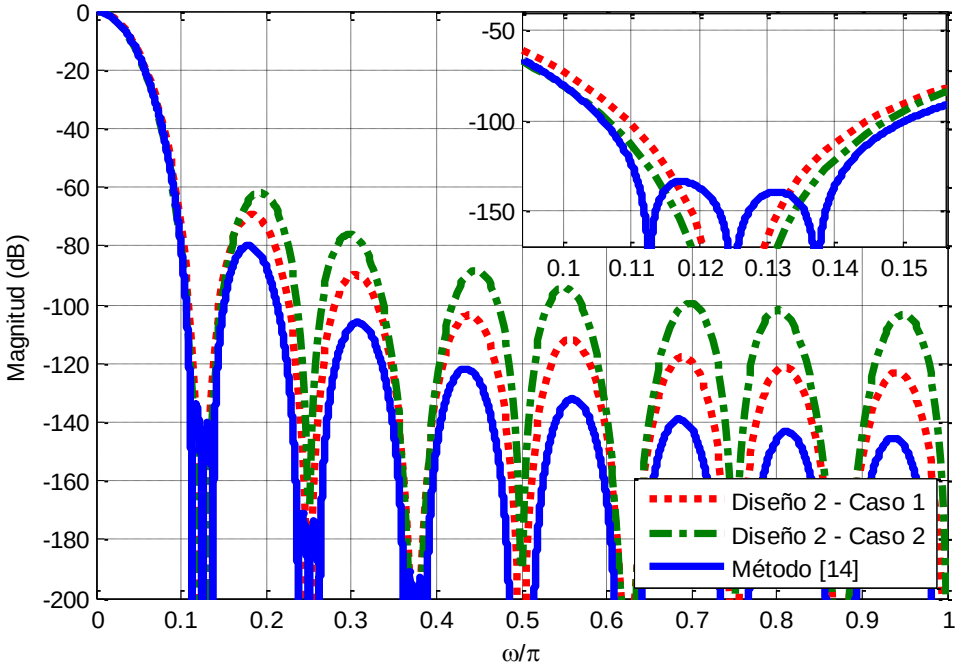
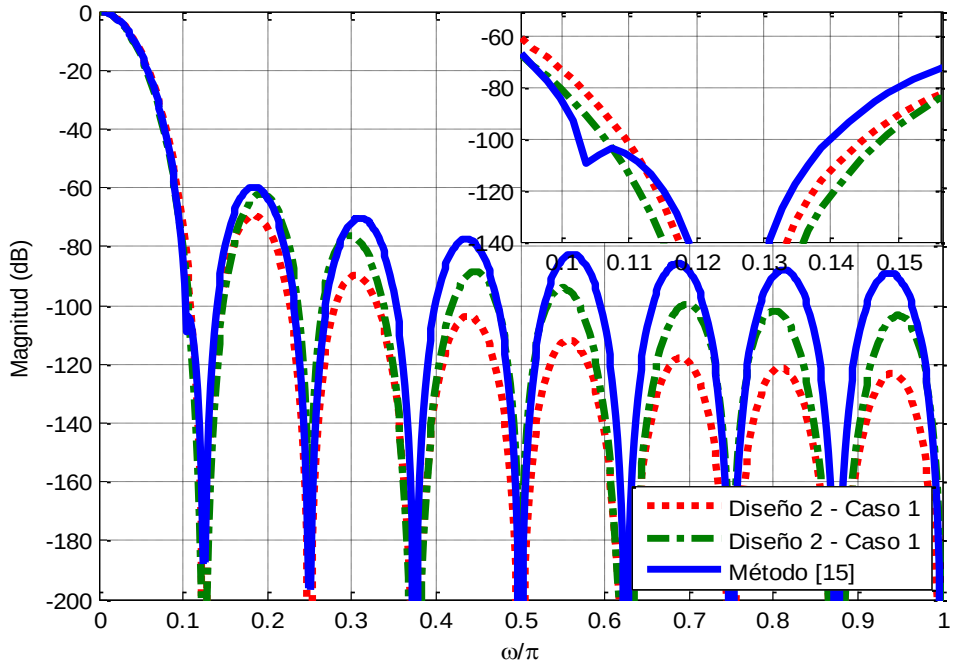


Fig. 4.14 Estructuras con características similares al **Diseño 2** (a) método [14] y (b) método [15].

En la Fig. 4.15 (a) y (b) se puede encontrar la respuesta en magnitud, y una ampliación del PDB, para el **Diseño 2 – Caso 1** y **2** frente al método de [14] y [15], respectivamente.



(a)



(b)

Fig. 4.15 Respuesta en magnitud para el **Diseño 2 – Caso 1** y **2** junto al método (a) [14] y (b) [15].

A partir de la Fig. 4.15 (a) y (b) se puede ver que el **Diseño 2**, ambos casos, cumple con la mínima atenuación requerida en todos los dobleces de la banda, por lo cual en cuanto a respuesta en magnitud representa una buena opción de diseño frente a los métodos de [14] y [15]. Por otra parte, en la Fig. 4.14, se puede observar que los métodos de [14] y [15] siguen modificando la estructura básica del decimador CIC no recursivo, a diferencia de la estructura propuesta que en esencia es la misma estructura básica del decimador CIC no recursivo, y solo modifica el valor de cascada de las etapas, para tener una menor cantidad de SPMS. Por tal motivo, en los próximos capítulos se presentará la implementación, simulación y comparación del **Diseño 2 – Caso 1** y **2** contra el decimador CIC no recursivo con $M = 16, K = 5$.

Capítulo 5

Introducción a la implementación del decimador CIC no recursivo

En este capítulo, primero se presentará el diseño de un circuito divisor de frecuencias, necesario para generar las diferentes frecuencias de muestreo a las que trabajan las etapas de los decimadores CIC no recursivos. Después se mostrará como implementar los DS y las etapas del decimador a través de procesos en VHDL (Very High Speed Integrated Circuit Hardware Description Language). Esta implementación se hará en forma directa y mediante componentes polifásicas. En esta última se dará una breve explicación sobre las técnicas usadas para obtener implementaciones eficientes. Finalmente, se presentarán ejemplos que resumen lo aquí mostrado.

5.1 Descripción de circuitos digitales mediante VHDL

Los circuitos o sistemas digitales, generalmente, están formados por miles de transistores, o millones según sea la complejidad. Por tal motivo su diseño e implementación se realizan con un nivel de abstracción mayor al transistor, como a nivel de compuertas lógicas y/o bloques completos como comparadores, multiplexores, etc. Los filtros digitales requieren de bloques aritméticos como el sumador y el multiplicador, y de elementos de memoria como los *Flip-Flops*. Para facilitar la tarea de implementación de un filtro, cada uno de sus elementos se describe mediante el lenguaje VHDL, y luego se interconectan en la forma adecuada para realizar el filtrado de una señal digital.

La descripción de filtros en VHDL puede ser automatizada mediante herramientas como el

HDL CODER de MATLAB, o de forma manual *RTL* (Register Transfer Level). La primera goza de una mayor rapidez en el diseño y la comprobación del correcto funcionamiento del filtro, pero su inconveniente es que solo permite describir filtros con algoritmos de diseño y estructuras predefinidas. Por otra parte, la descripción manual brinda mayor libertad al diseñador para implementar estructuras propuestas que permitan mejorar ciertas características de las implementaciones tradicionales. Por tal motivo los decimadores de este trabajo se describen de forma manual (RTL) en el entorno de desarrollo Xilinx ISE 9.1.

La codificación manual requiere conocer dos aspectos fundamentales del leguaje de descripción VHDL:

- Su naturaleza es concurrente, lo cual quiere decir que todos los elementos que se describen en la plantilla trabajan de manera independiente como si de sub-circuitos se tratara.
- Para generar máquinas de estado y/o elementos de memoria como Flip-Flops, es necesario utilizar funciones denominadas procesos. En la descripción de filtros estos sirven para generar los retardos que se necesitan para realizar la suma de la entrada actual con entradas anteriores.

5.2 Diseño del divisor de frecuencias

En la implementación de filtros para aplicaciones *multirate*, especialmente aquellos que usan decimación multi-etapa, es de suma importancia el bloque divisor de la frecuencia de muestreo de la señal de entrada. Este tiene como objetivo principal generar de manera sincronizada submúltiplos de la frecuencia de muestreo para activar los DS y en conjunto con el filtro antialiasing conseguir la reducción en la frecuencia de muestreo. Como se mostró en la sección 2.4, para los casos especiales del decimador CIC no recursivo estas frecuencias son sub-múltiplos de dos o tres. Para el caso $M = 2^p$ el divisor se puede implementar mediante un contador binario, el cual es un divisor de frecuencias por dos en cada bit de salida, donde el LSB (Bit menos significativo) produce una frecuencia $f_{LSB0} = f_{in}/2$, el segundo LSB una frecuencia $f_{LSB1} = f_{in}/4$, etc. Sin embargo, las formas de onda de salida de un contador binario no son adecuadas para sincronizar las diferentes etapas del

decimador CIC no recursivo. Esto es muy importante debido a que el proceso de decimación es *variante en el tiempo*. Además del inconveniente antes mencionado, con un contador binario no es posible obtener división de frecuencias para números impares como 3, el cual también es utilizado en la implementación de decimadores CIC no recursivos. En la Figura 5.1 se pueden ver las formas de onda de salida para un contador binario de 5 bits.

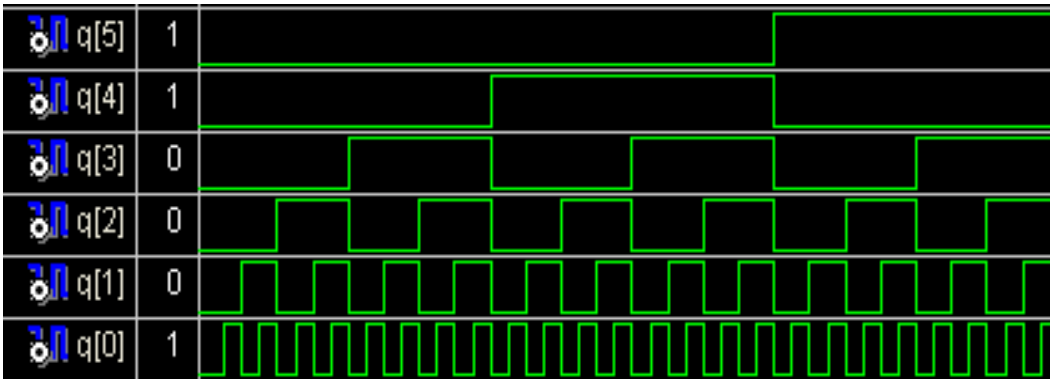


Fig. 5.1 Formas de onda de salida para un contador binario de 5 bits, donde q(0) es el LSB.

Una alternativa para este bloque es diseñar un circuito que funcione con base en una señal de entrada (con cierta frecuencia de muestreo), y que a partir de ésta genere las diferentes salidas (sub-múltiplos de la frecuencia de muestreo) sincronizadas con el flanco de subida de la señal de entrada. Para diseñar este circuito se considera el diagrama de la Fig. 5.2, que es una simplificación de la estructura CIC no recursiva de la Fig. 2.6 (a). Solo se considera la etapa 0 que trabaja a la misma frecuencia de muestreo que la señal de entrada, por conveniencia también llamada R_0, y los DS ($M = 2$) que están representados por un Flip-Flop tipo D llamado R_n. La señal digital de entrada $x(n)$ consta de una determinada secuencia de muestras, por ejemplo la salida de un modulador sigma-delta, y la salida $y(m) = x(mn)$ presenta solo aquellas muestras de entrada con un índice que es múltiplo entero de el factor de sub-muestreo acumulado a través de los registros R_n.

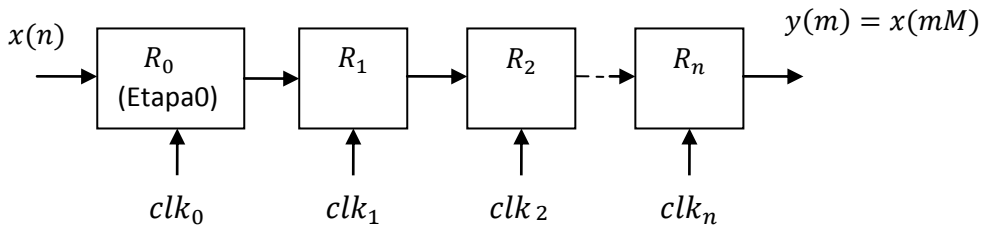


Fig. 5.2 Estructura simplificada de un decimador CIC no recursivo donde solo se presentan los DS.

Como se mencionó anteriormente la etapa 0 del decimador trabaja a la misma frecuencia que la señal de entrada $x(n)$ por lo que la señal clk_0 debe tener esta misma frecuencia para que las muestras $x(n)$ que ingresan al registro R_0 no sean sub-muestreadas. Para que un sub-muestreo por dos en R_1, por cuatro en R_2, por ocho en R_3, etc. se realice es necesario que las señales clk_1, clk_2, clk_n se encuentren sincronizadas con la señal de referencia clk_0 de la siguiente manera:

- En el momento que ocurre la primer transición de clk_0 de un estado bajo (‘0’ lógico) a un estado alto (‘1’ lógico), $x(0)$ sea almacenado en R_0.

- Cuando clk_0 efectúa su segunda transición se almacena $x(1)$ en R_0 y clk_1 deberá iniciar, simultáneamente, su primera transición hacia un estado alto para almacenar $x(0)$ en R_1.

- Cuando clk_0 realiza su tercer transición se almacena $x(2)$ en R_0 y clk_1 debe permanecer en estado bajo para no almacenar $x(1)$.

- Cuando clk_0 realiza su cuarta transición se almacena $x(3)$ en R_0, simultáneamente, clk_1 debe realiza su segunda transición para almacenar $x(2)$ (múltiplo de $M = 2$) en R_1, al mismo tiempo debe ocurrir la primera transición de clk_2 almacenándose $x(0)$ en R_2.

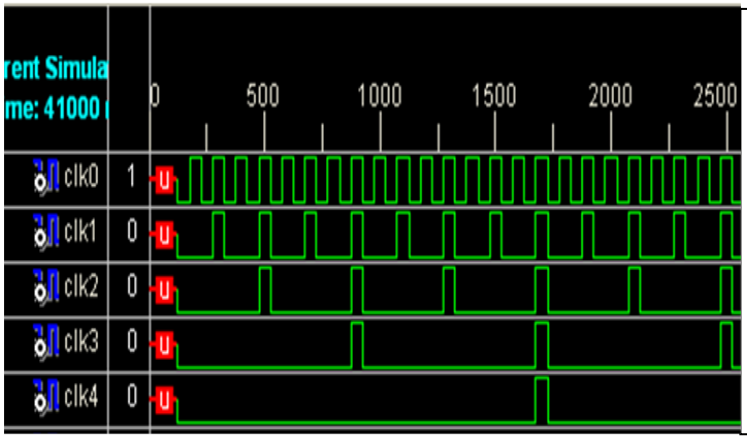
Siguiendo un razonamiento similar se puede ver que la señal clk_1 continúa realizando sus transiciones simultáneas cada dos transiciones de clk_0, mientras que clk_2 las realiza cada cuatro transiciones, clk_3 cada ocho transiciones hasta llegar a clk_n que deberá efectuar su primer transición en:

$$x = 2^n. \tag{5.1}$$

En la ecuación (5.1) "x" también representa el factor de sub-muestreo acumulado hasta R_n y el límite de transiciones para todas las señales clk, ya que después de esto el circuito divisor de frecuencias se debe reiniciar y comenzar de nuevo, como un proceso cíclico que le permita funcionar de forma indefinida.

Ejemplo 5.1: Para el divisor de frecuencias aquí planteado, ¿cómo deben ser las formas de onda para $n = 4$?

La señal clk_1 debe iniciar su primer transición en la segunda de clk_0 y repetirse cada dos hasta llegar a $x = 2^4 = 16$. clk_2 debe iniciar su primer transición en la cuarta de clk_0 y repetirse cada cuatro hasta 16. clk_3 debe iniciar en la octava de clk_0 y repetirse cada ocho hasta 16. Finalmente, clk_4 debe iniciar y realizar solo una transición en la décimo sexta de clk_0. Después de esto el circuito se reinicia de forma automática y empieza nuevamente el ciclo. La Fig. 5.3 muestra la simulación para este ejemplo así como también parte de su código en VHDL que lo genera.




```
DIVISOR: process
variable cont: integer
range 0 to 16; begin
wait until clk='1';
cont:=cont + 1; if (cont= 2
or cont=4… or cont=16) then
clk1<='1'; else clk1<='0';
end if; if (cont= 4 or
cont=8…… or cont=16) then
clk2<='1'; else clk2<='0';
end if;……; if (cont=16)
then clk4<='1'; else
clk4<='0'; end if; if
(cont=16) then cont:=0; end
if; end process DIVISOR;
```

Fig. 5.3 Simulación del divisor de frecuencia para clk_n con *n=4* y parte de su código de VHDL.

5.3 Implementación en forma directa

La función de transferencia $(1 + z^{-1})$ se puede implementar mediante la estructura directa de la Fig. 5.4, la cual es la celda básica para la construcción de una etapa del decimador CIC no recursivo de la Fig. 2.6 (a). La etapa completa se implementa poniendo K celdas básicas en cascada. Un ejemplo para $K = 4$ se presenta en la Fig. 5.5.

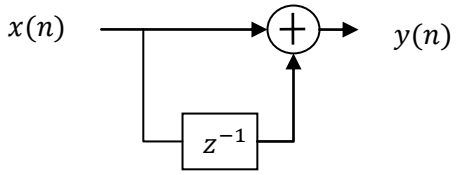


Fig. 5.4 Implementación de $(1 + z^{-1})$ mediante la estructura directa.

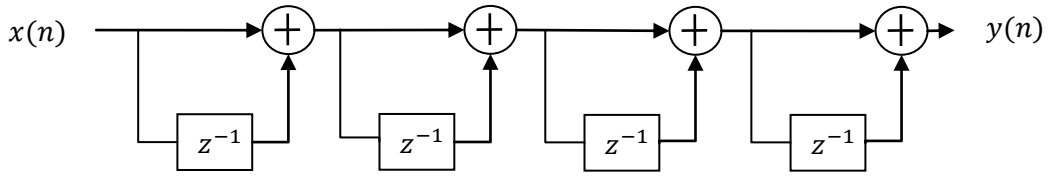


Fig. 5.5 Implementación de $(1 + z^{-1})^4$.

Para los decimadores propuestos **Diseño 1** y **2** se utiliza la misma celda básica de la Fig. 5.4 para la implementación de cada una de las etapas. La única diferencia, en comparación con los decimadores CIC no recursivos, radica en la cantidad de celadas básicas K_1 y K_2 que se deben conectar por cada etapa.

5.3.1 Descripción en VHDL

La implementación en VHDL, a nivel RTL, de las etapas se realiza mediante un proceso, el cual describe el flujo de los datos a través de las celdas básicas, de tal forma que se cumpla con la función de transferencia.

El DS también se implementa a través de un proceso debido a que en el diseño del divisor de frecuencias este fue considerado como un Flip-Flop tipo D. Sin embargo, este Flip-Flop (R_n) realiza el sub-muestreo por dos, gracias a la señal clk_n. Así que, puede ser eliminado y simplemente implementar las etapas como un proceso en VHDL que se activa con la señal clk_n. De esta manera la etapa-*n* solo procesa (convoluciona) las muestras con índice múltiplo de 2 de la etapa *n-1*.

En resumen, la etapa-n y el DS n-1 se describen en un solo proceso de VHDL el cual contiene la descripción del flujo de datos de la etapa-n trabajando con la señal clk_n, tal y como se muestra en la Fig. 5.6. El último DS se implementa de manera individual debido a que no hay una etapa enfrente con la cual pueda simplificarse.

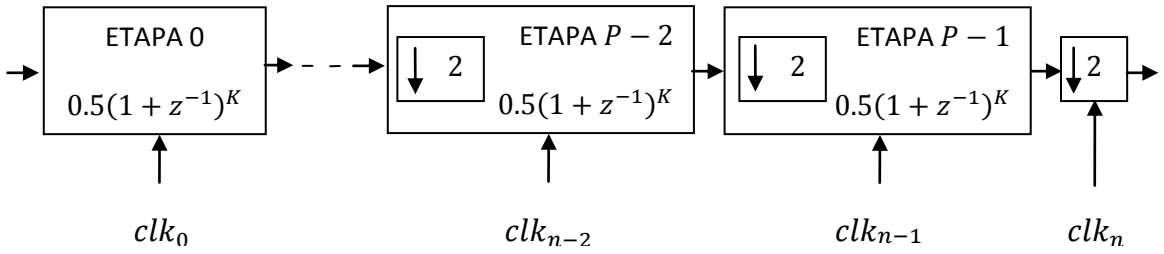
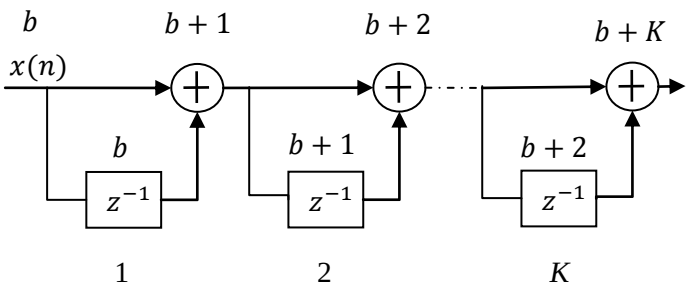


Fig. 5.6 Estructura para implementar una etapa y un DS en un único proceso VHDL.

Con base en (2.13), la implementación de la celda básica de la Fig. 5.4 requiere el aumento en la longitud de palabra en un bit según (5.2). De hecho, este aumento solo es necesario aplicarlo al sumador, debido a que la suma de dos números de *B* bits producen un resultado de (*B* + 1) bits. De esta forma, los sumadores de las celdas básicas que forman la estructura de la Fig. 5.5 van aumentando su longitud de palabra hasta llegar al último sumador el cual debe aumentar en $G = K$ bits, mientras el retardo correspondiente a cada celda básica se queda con la misma longitud de palabra que el dato de su entrada.

$$G = \left\lfloor log_2\left(\sum_{k=0}^{1} h[k]\right)\right\rfloor = \lfloor log_2(1+1)\rfloor = 1. \tag{5.2}$$

La Fig. 5.7 muestra una forma generalizada para la implementación de una etapa en la que se especifica cómo se debe incrementar el tamaño de la palabra para los sumadores y los retardos. Esta misma figura también muestra un código de VHDL genérico que permite describir este circuito. En dicho código la señal clk_n sirve para activar los elementos de memoria y los sumadores del proceso VHDL cada que sucede una transición de estado bajo a alto, siempre y cuando la señal de reset (rst) se encuentre en un estado lógico bajo de lo contrario se efectúa un restablecimiento asíncrono para la etapa.




```
Etapa_n: PROCESS (clk_n, rst) begin
if (rst='1') then ff_nK<=0; sum_nK<=0;
elsif (clk_n'event and clk_n='1') then
ff_n1<= x; sum_n1 <= x + ff_n1;
ff_n2<=sum_n1; sum_n2<= ff_n2 +
sum_n1; ..... ff_nK<=sum_n(K-1);
sum_nK<= ff_nK + sum_n(K-1); end if;
end PROCESS Etapa_n;
```

Fig. 5.7 Estructura y código VHDL para implementar $(1 + z^{-1})^K$.

Ejemplo 5.2: En este ejemplo se implementa en forma directa el decimador CIC no recursivo con $M = 32$ y $K = 5$. Para este caso son necesarias cinco etapas con $K = 5$ celdas básicas cada una. Para obtener la división de la frecuencia de muestreo por un factor de 32 el divisor de frecuencias se diseña hasta clk_5. La longitud de palabra a la entrada para todos los decimadores, del presente trabajo, se considera de 1bit, suponiendo que se usan para decimar la señal de un modulador sigma-delta mono-bit.

Con base en la Fig. 5.7 se implementa la etapa 0, donde el primer sumador requiere una longitud de palabra de 2 bits, el segundo de 3 bits, hasta llegar al quito sumador que requiere $1 + K = 6$ bits, tal y como lo muestra la Fig. 5.8. La etapa 1 requiere un aumento similar en la longitud de palabra obteniéndose 11 bits a su salida. Finalmente, las salidas correspondientes a las etapas 2, 3 y 4 son de 16, 21 y 26 bits respectivamente.

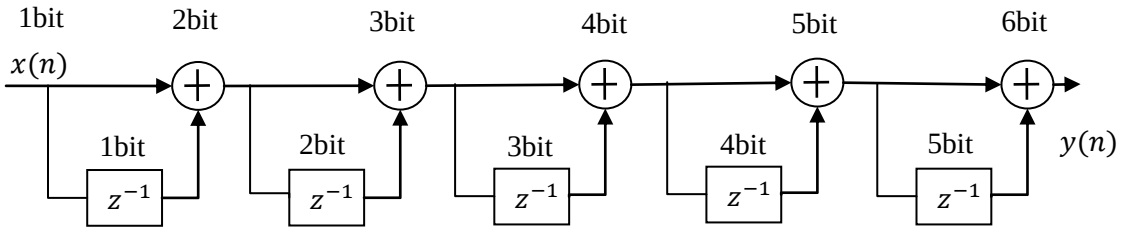


Fig. 5.8 Estructura para implementar en forma directa una etapa del decimador CIC no recursivo con $M = 32$ y $K = 5$.

Ejemplo 5.3: Para implementar en forma directa el decimador CIC no recursivo con $M = 16$ y $K = 6$, son necesarias cuatro etapas por lo que el divisor de frecuencias se diseña hasta clk_4. Una vez más, con base en la Fig.5.7 se implementa la etapa 0, donde el primer sumador tiene una longitud de palabra de 2 bits y el ultimo de 7 bits, debido a que $K = 6$, tal y como se muestra en la Fig. 5.9, donde la precisión de los componentes corresponde a la etapa 0. La longitud de palabra a la salida para las etapas 1, 2 y 3 es de 13, 19, 25 bits, respectivamente.

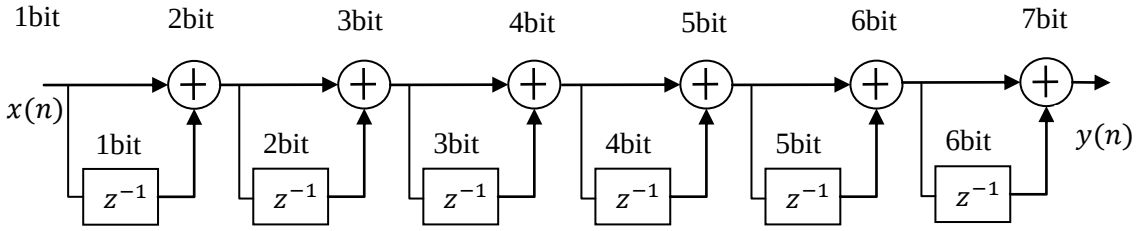


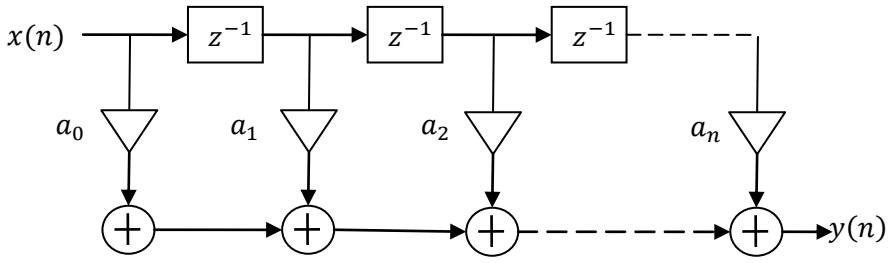
Fig. 5.9 Estructura para implementar en forma directa una etapa del decimador CIC no recursivo con $M = 16$ y $K = 6$.

5.4 Implementación polifásica

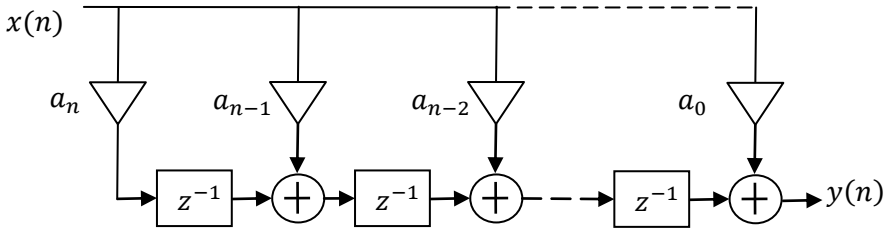
Los decimadores considerados en este trabajo, CIC no recursivo y **Diseño 1** y **2**, al ser implementados en componentes polifásicas reducen a la mitad la frecuencia de muestreo de

cada etapa con respecto a la implementación directa. Sin embargo, el proceso de implementación requiere de mayor atención debido a que es necesario implementar los coeficientes de los sub-filtros mediante corrimientos, sumas/restas y compartimiento de sub-expresiones comunes con el objetivo de obtener implementaciones eficientes (véase Sección 3.3).

Las funciones de transferencia de los sub-filtros $H_0(z^2)$ y $H_1(z^2)$ de la Fig. 2.7 (a), se pueden implementar a través de la estructura directa o la estructura directa transpuesta. La primera, mostrada en la Fig. 5.10 (a), tiene la principal desventaja de que todos los sumadores conectados en cascada forman la trayectoria crítica que impone el límite en la frecuencia máxima de operación, la cual disminuye a medida que aumenta el orden del sub-filtro.



(a)



(b)

Fig. 5.10 (a) Estructura directa y (b) directa transpuesta.

Por otra parte, la estructura directa transpuesta, Fig. 5.10 (b), es más eficiente desde el punto de vista de circuito debido a que posee un retardo entre cada sumador. Esto permite que la trayectoria crítica se reduzca al retardo de un multiplicador (coeficiente) más el retardo de un solo sumador, como en una estructura pipeline, permitiendo operar el sub-filtro a una frecuencia mayor. La técnica pipeline se utiliza para el diseño de circuitos

lógicos donde la latencia no es un tema importante pero la velocidad de los datos de salida si lo es [35]. No obstante, la latencia que experimentan las señales que procesa el filtro, se mantiene igual que para la estructura directa debido a que no se aplica intencionalmente la técnica pipeline solo se aprovecha la forma de la estructura. Además, en la estructura directa transpuesta todos los coeficientes interactúan simultáneamente con la señal de entrada, lo cual posibilita el compartimiento de sub-expresiones comunes para obtener implementaciones eficientes.

Con base en la estructura directa transpuesta, una etapa para el decimador polifásico de la Fig. 2.7 (a) se implementa mediante la estructura de la Fig. 5.11, donde a_n y b_n son los coeficientes de $H_0(z^2)$ y $H_1(z^2)$, respectivamente.

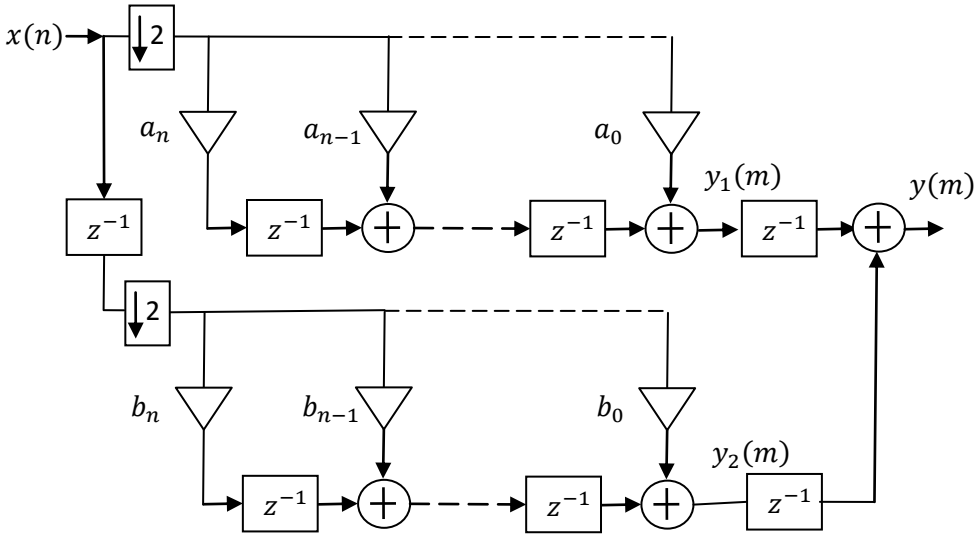


Fig. 5.11 Estructura polifásica general para implementar las etapas del decimador CIC no recursivo.

Se ha agregado un elemento de memoria o retardo a la salida y_1 y y_2 que permite tener todos los sumadores de la estructura entre dos retardos para conservar la trayectoria crítica como el retardo de un multiplicador más un sumador. Este último retardo no afecta la correcta ejecución de la suma de productos del sub-filtro, sin embargo perjudica en cuanto a un ligero aumento de área y aumento en la latencia por un ciclo de reloj en cada etapa, debido a que se ha agregado intencionalmente y no forma parte de la estructura. No obstante, este puede ser eliminado si el incremento en latencia no se tolera en alguna aplicación.

5.4.1 Descripción en VHDL

La descripción en VHDL, a nivel RTL, se hace de manera similar a la implementación directa. Los elementos de memoria, corrimientos y sumadores/restadores se describen dentro de un proceso el cual tiene en su lista de sensibilidad a la señal clk_{n+1} debido a que ahora la etapa-n trabaja a una frecuencia que es la mitad de la que usaba en la implementación directa.

La longitud de palabra para los sumadores y registros de cada etapa, para garantizar que no ocurra *overflow*, se determina mediante la ecuación (2.13), sustituyendo los coeficientes de la función de transferencia expandida para una etapa. Como se verá más adelante el aumento de la longitud de palabra también es de $G = K$ bits, similar que para la implementación directa, la diferencia es que ahora se debe considerar para todos los sumadores/restadores y registros, debido a que en esta estructura la señal de entrada es multiplicada y sumada varias veces, por lo cual hay que asegurar que se tiene la precisión necesaria para representar estas operaciones.

Ejemplo 5.4: El decimador CIC no recursivo con $M = 32$ y $K = 5$, se implementa en componentes polifásicas. Para este decimador se requiere de cinco etapas con función de transferencia $H(z) = (1 + z^{-1})^5$. Para la implementación polifásica primero es necesario expandir esta función como se aprecia en (5.3). Las componentes polifásicas se obtienen a partir de la ecuación (2.24) como se muestra en (5.4), y así los sub-filtros quedan determinados por (5.5) y (5.6).

$$H(z) = (1 + z^{-1})^5 = 1z^0 + 5z^{-1} + 10z^{-2} + 10z^{-3} + 5z^{-4} + 1z^{-5}. \tag{5.3}$$

$$H(z) = H_0(z^2) + H_1(z^2) \tag{5.4}$$

$$H_0(z^2) = 1z^0 + 10z^{-2} + 5z^{-4} \tag{5.5}$$

$$H_1(z^2) = z^{-1}[5z^0 + 10z^{-2} + 1z^{-4}] \tag{5.6}$$

Una vez se obtienen las componentes polifásicas, lo siguiente es encontrar las representaciones CSD para los coeficientes de (5.5) y (5.6). Estas se presentan en (5.7) y (5.8).

$$5|_{10} = 101|_2 = \ 101\ |_{CSD}. \tag{5.7}$$

$$10|_{10} = 1010|_2 = \ 1010\ |_{CSD}. \tag{5.8}$$

Sin embargo, en este ejemplo, las representaciones en CSD no conducen a una implementación más simple que la que se obtiene con su representación binaria debido a que solo se requiere de un sumador/restador por cada coeficiente, el cual no puede ser reducido.

Luego, a partir de su representación binaria o CSD se expresan los coeficientes como sumas de potencias de dos, para su implementación como corrimientos y sumas/restas, lo cual se puede ver en las ecuaciones (5.9) y (5.10).

$$5x = \ 2^2x + 2^0x. \tag{5.9}$$

$$10x = \ 2^3x + 2^1x. \tag{5.10}$$

En estas ecuaciones se puede ver que el coeficiente 5 se implementa como el corrimiento a la izquierda en 2 bits de la señal de entrada, que equivale a una multiplicación por 4, más la señal original. La multiplicación por el coeficiente 10 se obtiene con un corrimiento a la izquierda en 3 bits de la señal de entrada, equivalente a multiplicar por 8, más la señal de entrada recorrida un bit a la izquierda, equivalente a multiplicar por 2.

Después, es necesario determinar si existe una sub-expresión común a los coeficientes, que permita aprovechar el compartimiento de hardware. Para este ejemplo, se puede ver que el coeficiente 10 es un múltiplo de 5, por lo que este ultimo representa una sub-expresión común, debido a que puede generar el coeficiente 10 a través de una multiplicación por dos o un corrimiento en un bit a la izquierda. Haciendo esto se logra ahorrar un sumador al implementar los coeficientes 10 y 5.

Finalmente, utilizando la estructura de la Fig.5.11 y sustituyendo los multiplicadores por los corrimientos, sumas/restas y compartición de hardware derivados con anterioridad se obtiene la estructura de la Fig. 5.12, la cual permite implementar una etapa del decimador CIC no recursivo con $M = 32$ y $K = 5$. Los elementos que forman esta etapa deben tener una longitud de palabra 5 bits mayor a la del dato de entrada, según la ecuación (5.11).

$$G = \left\lfloor log_2\left(\sum_{k=0}^{5} h[k]\right)\right\rfloor = \lfloor log_2(1+5+10+10+5+1)\rfloor = \lfloor log_2(32)\rfloor = 5. \quad (5.11)$$

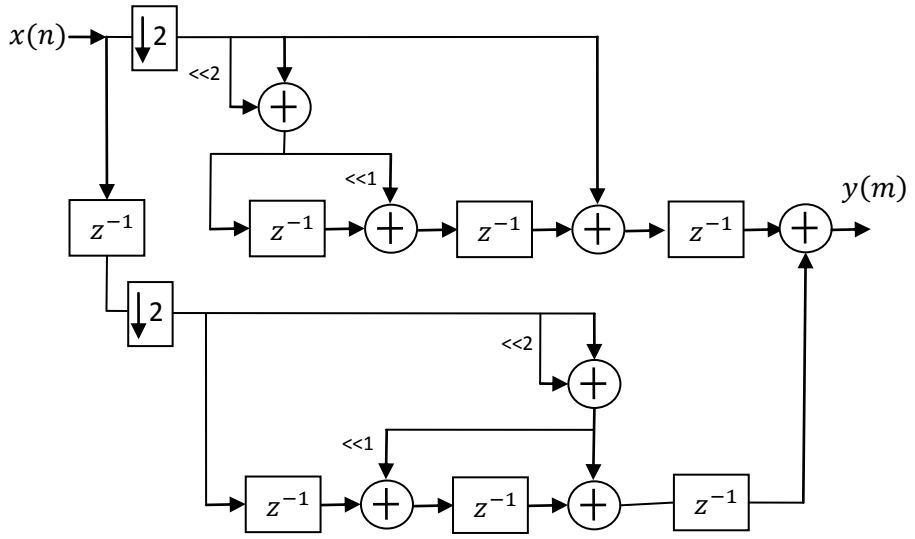


Fig. 5.12 Estructura polifásica eficiente para implementar las etapas del decimador CIC no recursivo con $M = 32$ y $K = 5$.

Una vez que se obtiene la estructura para una etapa, se describen las cinco etapas en VHDL. Para formar el decimador CIC no recursivo con $M = 32$ y $K = 5$ la etapa 0 trabaja con la frecuencia de clk_1 y su longitud de palabra aumenta de 1 a 6 bits. La etapa 1 trabaja con clk_2 y su longitud de palabra aumenta a 11 bits. Las etapas 2, 3, 4 trabajan con clk_{3-5} y su longitud de palabra es de 16, 21 y 26 bits respectivamente.

Ejemplo 5.5: Se implementa en componentes polifásicas el decimador CIC no recursivo con $M = 16$ y $K = 6$. Para este decimador se requiere de cuatro etapas con función de transferencia $H(z) = (1 + z^{-1})^6$. Como en el ejemplo anterior primero es necesario expandir la función de transferencia como en (5.12) y obtener las componentes polifásicas a partir de (2.24), y así los sub-filtros quedan determinados por (5.13) y (5.14).

$$H(z) = (1 + z^{-1})^6 = 1z^0 + 6z^{-1} + 15z^{-2} + 20z^{-3} + 15z^{-4} + 6z^{-5} + 1z^{-6} \quad (5.12)$$

$$H_0(z^2) = 1z^0 + 15z^{-2} + 15z^{-4} + 1z^{-6} \quad (5.13)$$

$$H_1(z^2) = z^{-1}[6z^0 + 20z^{-2} + 6z^{-4}] \quad (5.14)$$

Luego, se obtienen las representaciones en CSD para los coeficientes de (5.13) y (5.14), mismas que se presentan en (5.15) - (5.17).

$$15\,|_{10} = 1111\,|_2 = \ 1000\bar{1}\,|_{CSD}. \tag{5.15}$$

$$6\,|_{10} = 110\,|_2 = \ 10\bar{1}0\,|_{CSD}. \tag{5.16}$$

$$20\,|_{10} = 10100\,|_2 = \ 10100\,|_{CSD}. \tag{5.17}$$

El coeficiente 15 en su representación binaria requiere del uso de 3 sumadores/restadores. Esto, sin duda, no resulta en una implementación eficiente, debido a que tanto el consumo de área como el de potencia es similar a usar un multiplicador. Por otra parte, su representación en CSD resulta en una implementación eficiente al requerir solo un sumador/restador. El coeficiente 6 tiene una representación CSD que no disminuye su complejidad en comparación con la binaria y puede ser implementado en cualquiera de las dos formas. El coeficiente 20 posee la misma representación en CSD y binario.

Después, considerando las características CSD mencionadas con anterioridad, los coeficientes se representan como SPD mediante las ecuaciones (5.18) - (5.20).

$$15x = 2^4x - 2^0x. \tag{5.18}$$

$$6x = 2^2x + 2^1x. \tag{5.19}$$

$$20x = 2^4x + 2^2x. \tag{5.20}$$

El coeficiente 15 se implementa mediante un corrimiento a la izquierda en 4 bits del valor de entrada menos el valor original, así mismo el 6 como un corrimiento en dos bits más el corrimiento en un bit, y el 20 como el corrimiento en 4 bits más el corrimiento en 2 bits.

Lo siguiente es la búsqueda de sub-expresiones comunes, de (5.14) se ve que puede usarse el coeficiente 6 para generar 20 a través de $20 = 6 \times 2 + 8$. Sin embargo, se sigue utilizando la misma cantidad de sumadores/restadores que al implementar 20 en forma directa a través de (5.20). A pesar de que no se puede usar una sub-expresión común, los coeficientes de (5.13) y (5.14) presentan simetría, por lo que se ahorra la mitad de sumadores/restadores al implementar 15 y 6.

Finalmente, la implementación de una etapa para el decimador CIC no recursivo con $M = 16$ y $K = 6$ se hace con base en la estructura de la Fig. 5.11, con los coeficientes de (5.13) y (5.14) representados como corrimientos, sumas/restas en (5.18) - (5.20) y aprovechando la simetría de los coeficientes. La estructura resultante es presentada en la Fig. 5.13 y permite implementar una de las cuatro etapas del decimador CIC no recursivo con $M = 16$ y $K = 6$, donde la única diferencia radica en la frecuencia de operación y la longitud de palabra de cada una, que aumenta en 6 bits según (5.21).

$$G = \left\lfloor log_2\left(\sum_{k=0}^{5} h[k]\right)\right\rfloor = \lfloor log_2(1 + 6 + 15 + 20 + 15 + 6 + 1)\rfloor$$

$$= \lfloor log_2(64)\rfloor = 6. \qquad (5.21)$$

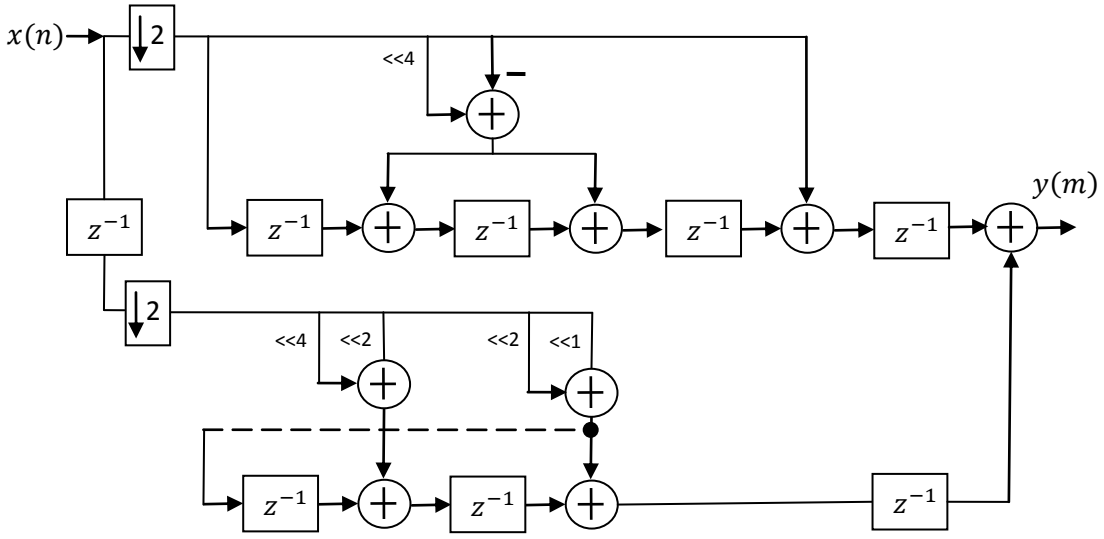


Fig. 5.13 Estructura polifásica eficiente para implementar las etapas del decimador CIC no recursivo con $M = 16$ y $K = 6$.

Una vez que se obtiene la estructura para una etapa se describen las cuatro etapas en VHDL. La etapa 0, del decimador CIC no recursivo con $M = 16$ y $K = 6$, trabaja con la frecuencia de clk_1 y su longitud de palabra aumenta de 1 a 7 bits. La etapa 1 trabaja con clk_2 y su longitud de palabra aumenta de 7 a 13 bits. Las etapas 2, 3 trabajan con clk_{3-4} y su longitud de palabra es de 19 y 25 bits respectivamente.

5.5 Síntesis a nivel transistor

Cuando se diseñan circuitos digitales y se describen mediante VHDL, es posible implementarlos de forma física en un FPGA (Field Programable Gate Array) o en un ASIC (Application Specific Integrated Circuit) según la aplicación a la cual vayan dirigidos. Para los decimadores aquí considerados una de sus aplicaciones principales es en convertidores sigma-delta, los cuales son implementados en un ASIC más que en un FPGA. Por este motivo el código de VHDL que describe a los decimadores, presentados en este trabajo de tesis, es sintetizado en transistores de tecnología CMOS de 0.18um, con el objetivo de obtener mediciones para el consumo de potencia y el área utilizada.

Para realizar la síntesis es necesario el uso de tres herramientas del Software Mentor Graphics:

- ***Leonardo Spectrum***: En esta herramienta el código VHDL que describe a los decimadores, es transformado en un código que aun a pesar de seguir siendo de alto nivel (VHDL o Verilog), describe el decimador mediante celdas estándar como son inversores, compuertas y Flip-Flops principalmente, es decir, con un nivel de abstracción menor. En esta misma herramienta se pueden optimizar aspectos como velocidad en las trayectorias críticas y área en el *layout* (patrón geométrico). Sin embargo, para los decimadores de este trabajo no se utilizan estas opciones para no afectar las comparaciones que se presentarán más adelante.

- ***Design Architect***: Esta herramienta trabaja con el código generado por Leonardo y genera un diagrama esquemático a nivel compuerta y/o bloque. También, tiene la función de hacer el mapeo de puertos de entrada y salida para una posterior verificación LVS (Layout vs Schematic). Además, permite generar un archivo en formato SPICE del decimador a nivel transistor que puede ser utilizado para simular el consumo de potencia bajo ciertas condiciones en la señal de entrada.

- ***IC Studio***: Esta herramienta permite generar el layout del decimador y utilizarlo para la fabricación de un ASIC. Sin embargo, en este trabajo solo fue usado para obtener un valor aproximado del área total utilizada para su implementación.

En el capítulo 7 se presentarán las comparaciones en relación al consumo de potencia y área usada en la implementación de los decimadores una vez han sido sintetizados en tecnología CMOS de 0.18um.

Capítulo 6

Implementación de los decimadores propuestos

En este capítulo se mostrará como implementar los decimadores propuestos en el capítulo 4, en forma directa y polifásica, en base a lo presentado en el capítulo anterior a fin de obtener estructuras eficientes.

6.1 Propuesta: Diseño 1

6.1.1 Implementación directa

Para implementar en forma directa el decimador **Diseño1** con los parámetros $M = 32$, $K_1 = 5$ y $K_2 = 3$, se requiere de cinco etapas, donde las primeras cuatro tienen un valor de cascada de $K_1 = 5$ y se implementan mediante la estructura de la Fig. 6.1 (a), donde la precisión de los componentes mostrados corresponde a la etapa 0.

La última etapa requiere una cascada de $K_1 + K_2 = 8$, para lo cual solo es necesario agregar $K_2 = 3$ celdas básicas a la estructura de la Fig. 6.1 (a). De esta manera, la última etapa se implementa como muestra la Fig. 6.1 (b), donde se considera el aumento en la longitud de palabra acumulada por las primeras cuatro etapas (21 bits).

Como se puede ver la complejidad en la implementación directa del **Diseño 1**, que mejora la atenuación en el primer doblez de la banda, permanece similar a la implementación del decimador CIC no recursivo con $M = 32$ y $K_1 = 5$ descrito en el Ejemplo 5.2, debido a que solo es necesario modificar la cascada de la última etapa agregando $K_2 = 3$ celdas básicas.

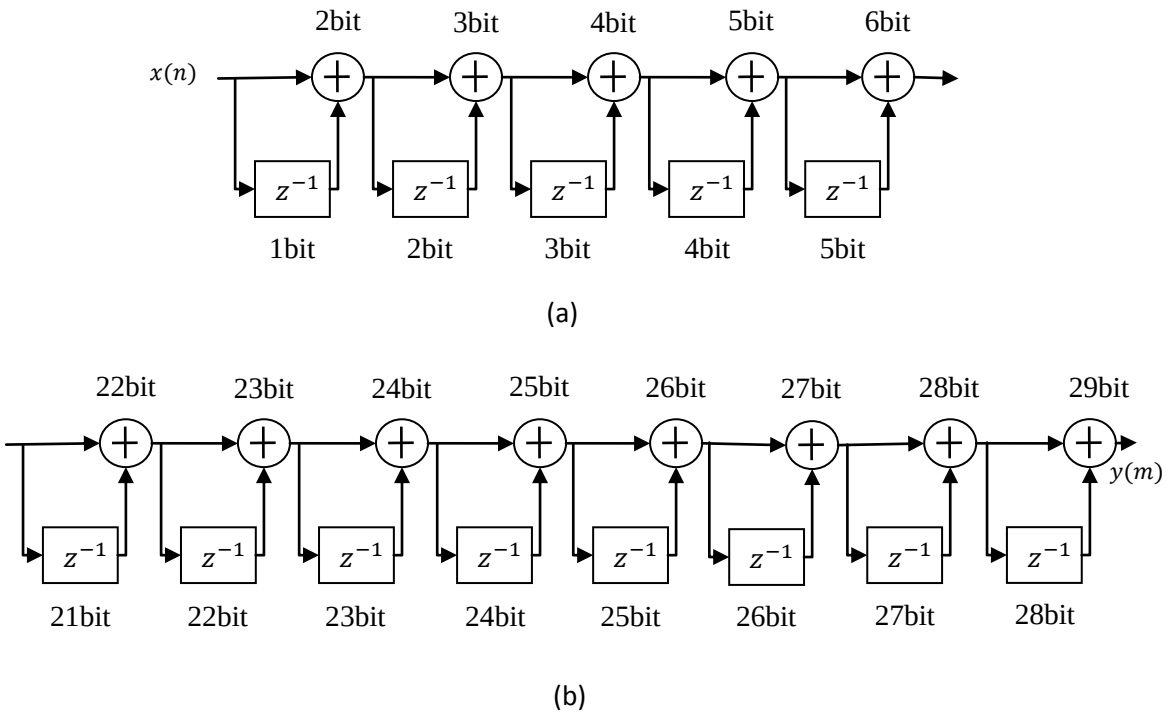


Fig. 6.1 Estructura para implementar en forma directa el decimador **Diseño 1** (a) primeras cuatro etapas, (b) última etapa.

6.1.2 Implementación Polifásica

Para implementar en forma polifásica el decimador **Diseño1** con $M = 32$, $K_1 = 5$ y $K_2 = 3$, también se requiere de cinco etapas. De forma similar a la implementación directa, las primeras cuatro etapas tienen un valor de cascada de $K_1 = 5$, por lo cual se implementan con la estructura de la Fig. 6.2 (a) que es la misma usada para todas las etapas en el decimador CIC no recursivo con $M = 32$, $K = 5$ del Ejemplo 5.4. Sin embargo, para la última etapa que requiere una cascada de $K_1 + K_2 = 8$, no es posible reusar la estructura de las primeras etapas, debido a que las componentes polifásicas son especificas para cada valor de cascada. Por tal motivo, es necesario re-expandir la función de transferencia tal y como se muestra en (6.1) y mediante (2.24) determinar las componentes polifásicas o sub-filtros, mostrados en (6.2) y (6.3).

$$H(z) = (1 + z^{-1})^8 = 1z^0 + 8z^{-1} + 28z^{-2} + 56z^{-3} + 70z^{-4}$$

$$+56z^{-5} + 28z^{-6} + 8z^{-7} + 1z^{-8}, \qquad (6.1)$$

$$H_0(z^2) = 1z^0 + 28z^{-2} + 70z^{-4} + 28z^{-6} + 1z^{-8}, \qquad (6.2)$$

$$H_1(z^2) = z^{-1}[8z^0 + 56z^{-2} + 56z^{-4} + 8z^{-6}]. \quad (6.3)$$

Lo siguiente es obtener la representación CDS para cada coeficiente de (6.2) y (6.3), los cuales son mostrados en (6.4) - (6.7).

$$28\,|_{10} = 11100\,|_2 = 100\bar{1}00\,|_{CSD}. \quad (6.4)$$

$$70\,|_{10} = 1000110\,|_2 = 1001010\bar{1}0\,|_{CSD}. \quad (6.5)$$

$$56\,|_{10} = 111000\,|_2 = 100\bar{1}000\,|_{CSD}. \quad (6.6)$$

$$8\,|_{10} = 1000\,|_2 = 1000\,|_{CSD}. \quad (6.7)$$

De estas últimas ecuaciones se puede ver que para los coeficientes 28 y 56 es más conveniente utilizar su representación CSD porque la cantidad de sumadores/restadores para su implementación se reduce en uno. Por otra parte, para 70 la complejidad es la misma y finalmente 8 no representa consumo de hardware debido a que solo es un corrimiento. De esta manera, los coeficientes representados como sumas de potencias de dos se muestran en (6.8) - (6.11).

$$28x = 2^5x - 2^2x. \quad (6.8)$$

$$70x = 2^6x + 2^3x - 2^2x. \quad (6.9)$$

$$56x = 2^6x - 2^3x. \quad (6.10)$$

$$8x = 2^3x. \quad (6.11)$$

Para $H_0(z^2)$ de (6.2) puede usarse el coeficiente 28 como sub-expresión. No obstante, para generar el coeficiente 70 como $70 = 28 \times 2 + 14$, se necesita la misma cantidad de sumadores/restadores que al implementar 70 de forma directa mediante (6.9), debido a que para generar $14\,|_{10} = 1110\,|_2 = 100\bar{1}0\,|_{CSD}$ se requiere un sumador/restador, y un segundo sumador/restador para adicionarlo con 28×2. En $H_1(z^2)$ de (6.3) puede usarse el coeficiente 8 como sub-expresión para generar el 56 a partir de $56 = 7 \times 8$, pero para generar $7|_{10} = 111|_2 = 100\bar{1}\,|_{CSD}$ es necesario utilizar un sumador/restador que es equivalente a generar el 56 de forma directa mediante (6.10). A pesar de que se pueden compartir sub-expresiones, en la última etapa, estas no conducen a una implementación

más eficiente. Sin embargo, las componentes polifásicas presentan simetría en los coeficientes lo cual permite ahorrar la mitad de sumadores/restadores al implementar los coeficientes 28 y 56.

Finalmente, la estructura para la implementación de esta última etapa del **Diseño 1** es mostrada en la Fig. 6.2 (b), la cual se obtiene con base en la estructura de la Fig. 5.11, los coeficientes de (6.2) y (6.3) representados como corrimientos, sumas/restas en (6.8) - (6.11) y aprovechando la simetría de los coeficientes.

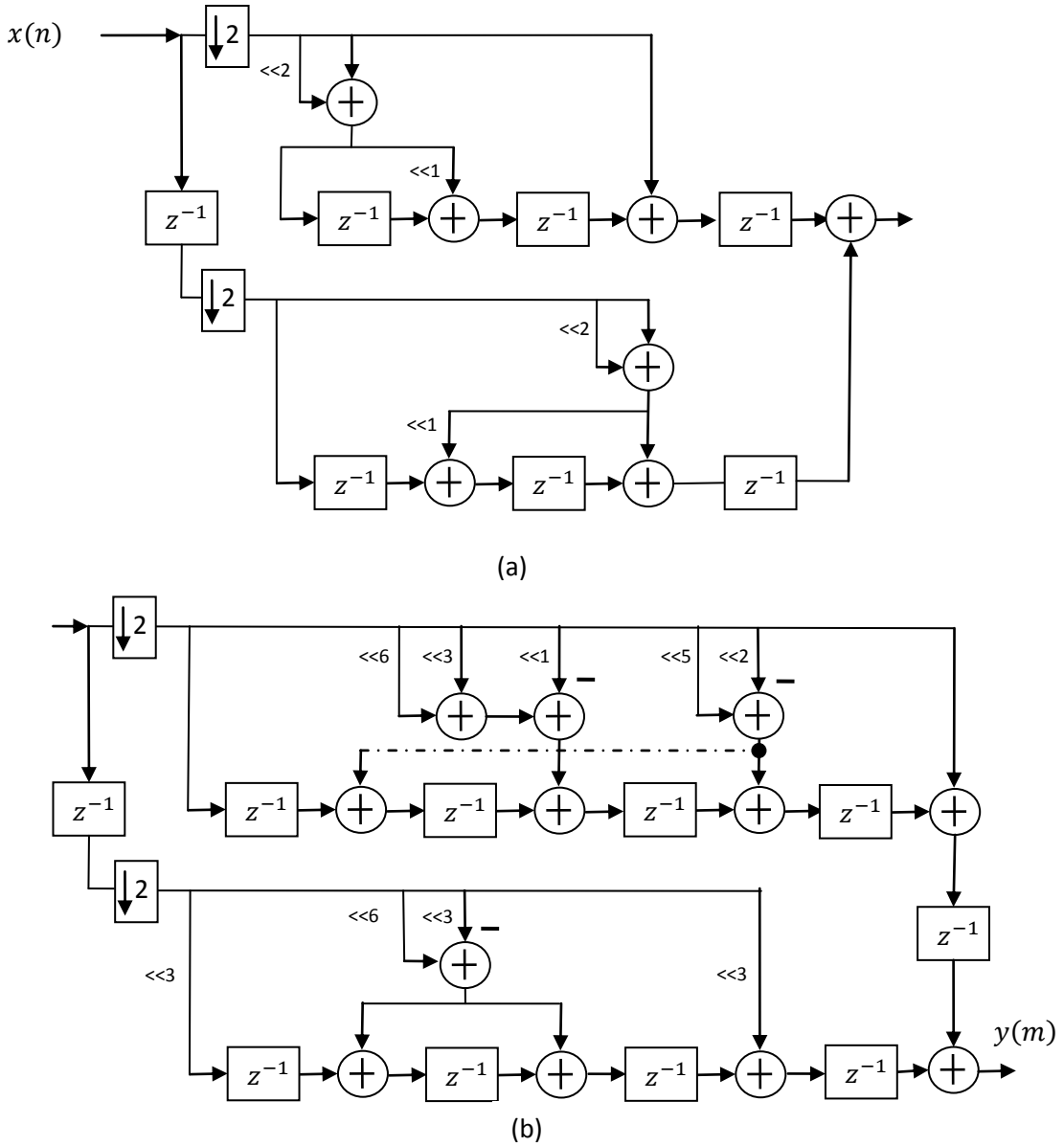


Fig. 6.2 Estructura polifásica eficiente para implementar el **Diseño 1** (a) primeras cuatro etapas, (b) última etapa.

Los componentes de la última del **Diseño 1**, de la Fig. 6.2 (b), requieren de un aumento general en la longitud de palabra en 8 bits, tal y como lo expresa (6.12).

$$G = \left\lfloor log_2\left(\sum_{k=0}^{8} h[k]\right)\right\rfloor = \lfloor log_2(1+8+28+56+70+56+28+8+1)\rfloor$$
$$= \lfloor log_2(256)\rfloor = 8. \quad (6.12)$$

Como se puede apreciar en la Fig. 6.2 (b) la última etapa ha aumentado su complejidad de implementación respecto del decimador CIC no recursivo con $M = 32$, $K = 5$ en componentes polifásicas presentado en el Ejemplo 5.4 (mayor número de coeficientes y sumadores/restadores), por lo que requiere de un mayor esfuerzo para su implementación. Sin embargo, siguiendo los pasos mostrados en este capítulo y el anterior la implementación se hace de una manera muy rápida y sencilla. De hecho, al tener diseños previos para otros valores de K, se pueden reutilizar las estructuras previamente implementadas a fin de obtener una nueva.

6.2 Propuesta: Diseño 2

6.2.1 Implementación directa

6.2.1.1 Caso 1: $M = 16$, $K_1 = 5$ y $K_2 = 1$

El decimador del **Diseño 2 – Caso 1** con parámetros $M = 16$, $K_1 = 5$ y $K_2 = 1$, requiere de cuatro etapas para su implementación, las primeras tres se implementan como muestra la Fig. 6.3 (a), donde la precisión de los componentes corresponde a la etapa 0. Para las etapas 1 y 2 la estructura es la misma, de la Fig. 6.3 (a), únicamente cambia la precisión necesaria para los sumadores y registros.

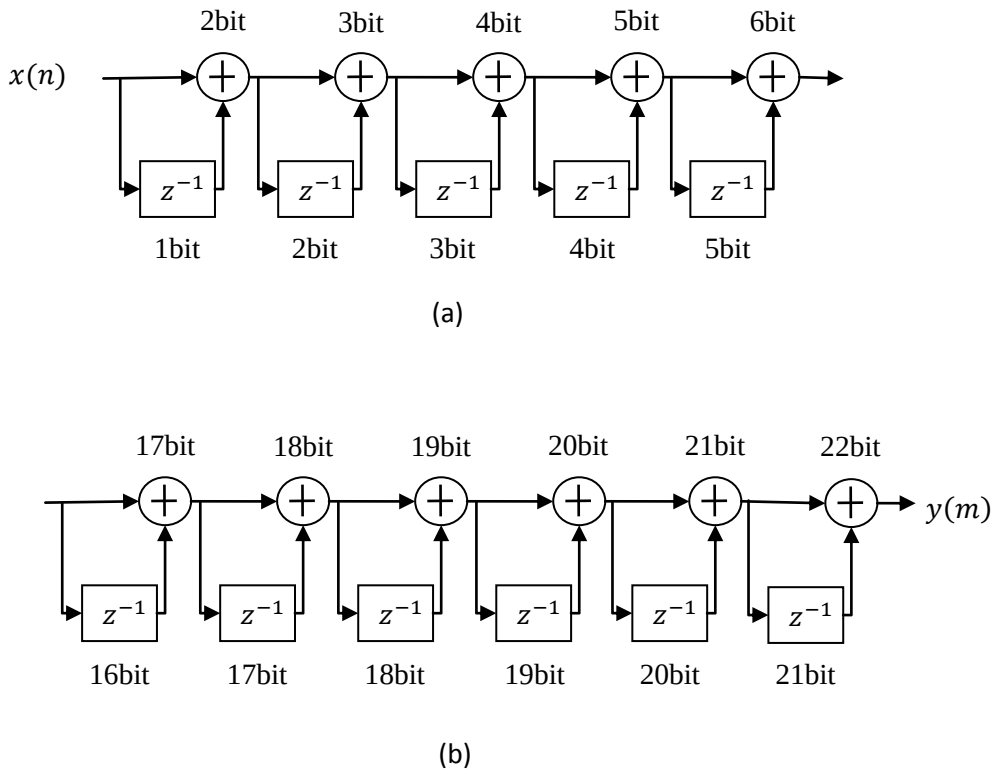


Fig. 6.3 Estructura para implementar en forma directa el **Diseño 2 – Caso1** (a) primeras tres etapas, (b) última etapa.

La última etapa requiere una cascada de $K_1 + K_2 = 6$, para lo cual solo es necesario agregar $K_2 = 1$ celda básica a la estructura de las primeras tres etapas. De esta manera la última etapa se implementa como muestra la Fig. 6.3 (b), donde se considera el aumento en la longitud de palabra acumulada por las primeras tres etapas (16 bits).

6.2.1.2 Caso 2: $M = 16$, $K_1 = 4$ y $K_2 = 3$

Para la implementación directa del decimador **Diseño 2 – Caso 2** con parámetros $M = 16$, $K_1 = 4$ y $K_2 = 3$ son necesarias cuatro etapas. Las etapas 0, 1 y 2 se forman con $k_1 = 4$ celdas básicas, como muestra la Fig. 6.4 (a), donde la longitud de palabra presentada es la correspondiente a la etapa 0. La última etapa requiere de $K_1 + K_2 = 7$, por lo que se implementa como en la Fig. 6.4 (b), donde se considera el aumento en la longitud de palabra acumulada por las primeras tres etapas (13 bits). Así pues, la precisión de los sumadores finales de la etapa 0, 1, 2 y 3 es de 5, 9, 13 y 20 bits, respectivamente.

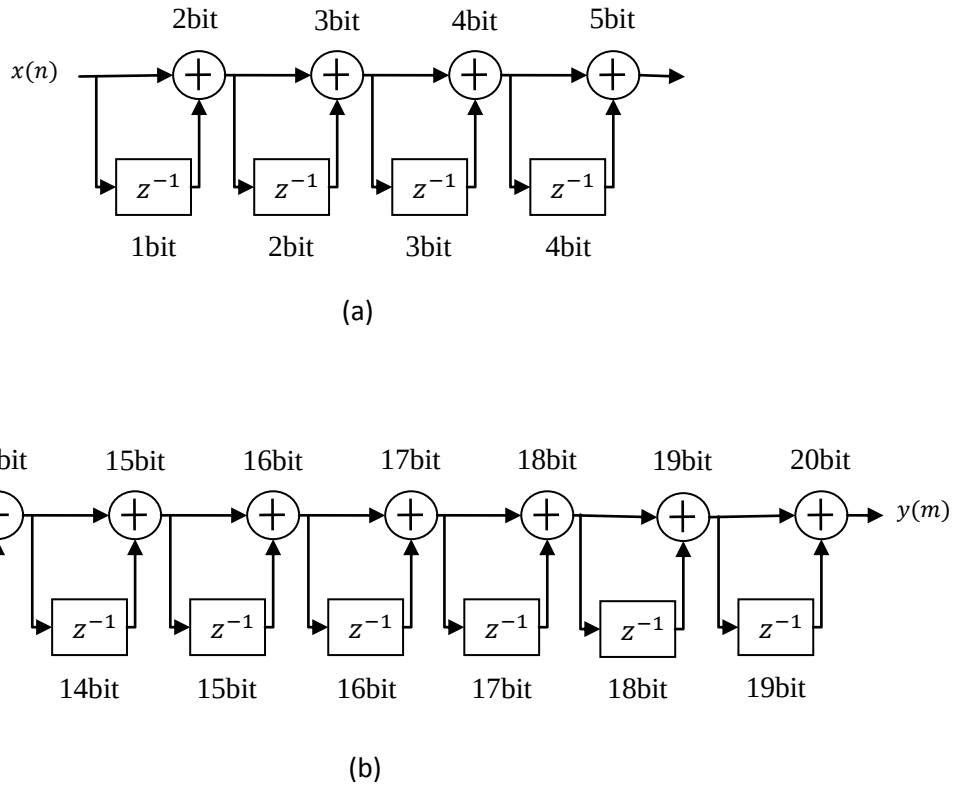


Fig. 6.4 Estructura para implementar en forma directa el **Diseño 2 – Caso 2** (a) primeras tres etapas, (b) última etapa.

6.2.2 Implementación polifásica

6.2.2.1 Caso 1: $M = 16$, $K_1 = 5$ y $K_2 = 1$

Este decimador se implementa mediante cuatro etapas. Las tres primeras son como las de la estructura del decimador CIC no recursivo con $M = 32$ y $K = 5$ del Ejemplo 5.4. La última es como la del decimador CIC no recursivo con $M = 16$ y $K = 6$ del Ejemplo 5.5. Ambas se repiten en la Fig. 6.5 (a) y (b) respectivamente. La longitud de palabra necesaria para las etapas 0, 1, 2 y 3 es de 6, 11, 16 y 22 bits, respectivamente. Cabe recordar, que el aumento en longitud de palabra es general para todos y cada uno de los sumadores y registros que se utilizan en la etapa en cuestión, con el objetivo de evitar el *overflow* de los datos y asegurar una correcta suma de productos (convolución).

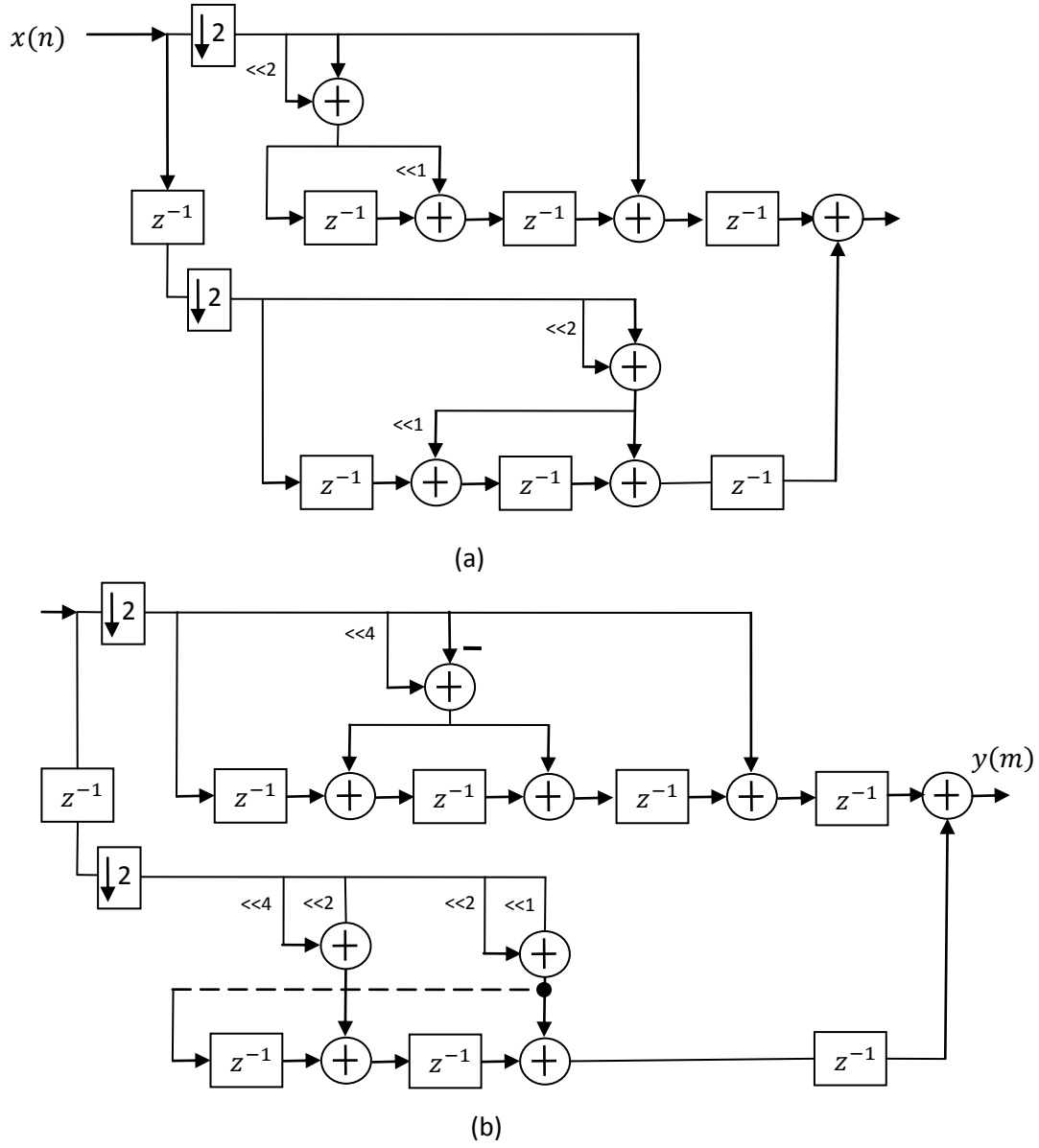


Fig. 6.5 Estructura polifásica eficiente para implementar el **Diseño 2 –Caso 1** (a) primeras cuatro etapas, (b) última etapa.

6.2.2.2 Caso 2: $M = 16$, $K_1 = 4$ y $K_2 = 3$

Este decimador está formado por cuatro etapas, donde las primeras tres requieren la expansión de la función de transferencia con $K_1 = 4$ y la última con $K_1 + K_2 = 7$. Debido a que con anterioridad no se han generado componentes polifásicas para estos valores es necesario expandir las dos funciones con su valor respectivo. La expansión y obtención de las componentes polifásicas para $K_1 = 4$ se puede encontrar en (6.13) - (6.15).

$$H(z) = (1 + z^{-1})^4 = 1z^0 + 4z^{-1} + 6z^{-2} + 4z^{-3} + 1z^{-4} \quad (6.13)$$

$$H_0(z^2) = 1z^0 + 6z^{-2} + 1z^{-4} \quad (6.14)$$

$$H_1(z^2) = z^{-1}[4z^0 + 4z^{-2}] \quad (6.15)$$

Para los coeficientes de (6.13) y (6.14) no es necesario obtener su representación en CSD debido a que el 6 se implementa mediante (5.19), mientras que 4 por ser una potencia de 2 se implementa como un corrimiento en dos bits a la izquierda del valor original. Estos coeficientes, al ser muy pocos, no es posible obtener una sub-expresión común que permita reducir la cantidad de hardware usada en su implementación. La simetría de los coeficientes solo se puede aprovechar en 4, pero dado que este no requiere de ningún recurso no podemos considerarlo como un ahorro de hardware. La estructura que permite implementar las primeras tres etapas se muestra en la Fig. 6.6 (a).

Para la última etapa se expande la función de transferencia con los valores de $K_1 + K_2 = 7$ como en (6.16) y se obtienen las componentes polifásicas como en (6.17) y (6.18).

$$H(z) = (1 + z^{-1})^7 = 1z^0 + 7z^{-1} + 21z^{-2} + 35z^{-3}$$

$$+35z^{-4} + 21z^{-5} + 7z^{-6} + 1z^{-7} \quad (6.16)$$

$$H_0(z^2) = 1z^0 + 21z^{-2} + 35z^{-4} + 7z^{-6} \quad (6.17)$$

$$H_1(z^2) = z^{-1}[7z^0 + 35z^{-2} + 21z^{-4} + 1z^{-6}] \quad (6.18)$$

Las representaciones en CSD para los coeficientes de (6.17) y (6.18) son mostradas en (6.19) - (6.21).

$$7\,|_{10} = 111\,|_2 = 100\bar{1}\,|_{CSD} \quad (6.19)$$

$$21\,|_{10} = 10101\,|_2 = 10101\,|_{CSD}. \quad (6.20)$$

$$35\,|_{10} = 100011\,|_2 = 10010\bar{1}\,|_{CSD}. \quad (6.21)$$

La única representación en CSD eficiente es para 7 porque para 21 y 35 se necesita la misma cantidad recursos que en la representación binaria. La representación de estos

coeficientes como suma de potencias de dos, para su implementación mediante corrimientos y sumas/restas, se presenta en (6.22) - (6.24).

$$7x = 2^3x - 2^0x. \tag{6.22}$$

$$21x = 2^4x + 2^1x + 2^0x. \tag{6.23}$$

$$35x = 2^5x + 2^1x + 2^0x. \tag{6.24}$$

Para implementar los coeficientes de (6.17) y (6.18) se requiere de 5 sumadores/restadores según las ecuaciones (6.22) a (6.24). Sin embargo, si se utiliza el coeficiente 7 como una sub-expresión común, se puede generar 21 y 35 como 7×3 y 7×5, respectivamente. Haciendo esto para generar 7, se utiliza un sumador/restador, para generar la multiplicación por 3 es necesario un sumador/restador, y la multiplicación por 5 también requiere un solo sumador/restador. De esta manera, se generan todos los coeficiente usando solo tres sumadores/restadores. Al implementar los coeficientes en esta forma se logra el ahorro de cuatro sumadores/restadores en la última etapa, cuya estructura es mostrada en la Fig. 6.6 (b).

Finalmente, la longitud de palabra para las tres primeras etapas está determinada por (6.25) y para la última por (6.26). Asumiendo una entrada de un bit, la precisión de los elementos para la etapa 0, 1, 2 es 5, 9, 13 bits y para la última es de 20 bits.

$$G = \left\lfloor log_2\left(\sum_{k=0}^{4} h[k]\right)\right\rfloor = \lfloor log_2(1+4+6+4+1)\rfloor = \lfloor log_2(16)\rfloor = 4, \tag{6.25}$$

$$\begin{aligned} G = \left\lfloor log_2\left(\sum_{k=0}^{7} h[k]\right)\right\rfloor &= \lfloor log_2(1+7+21+35+35+21+7+1)\rfloor \\ &= \lfloor log_2(128)\rfloor = 7. \end{aligned} \tag{6.26}$$

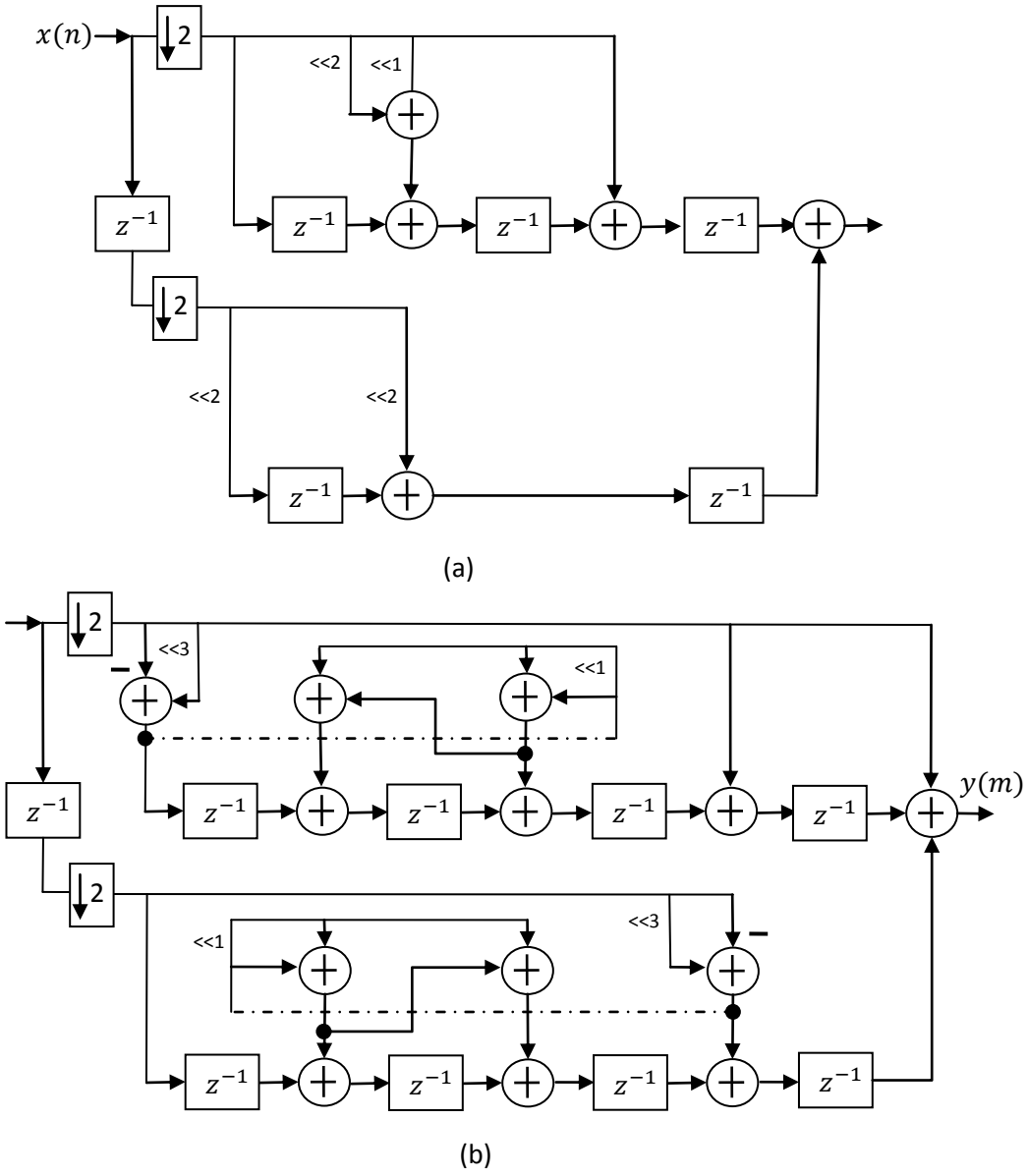


Fig. 6.6 Estructura polifásica eficiente para implementar el **Diseño 2- Caso2** (a) primeras tres etapas y (b) última etapa.

Capítulo 7

Simulación del funcionamiento de los decimadores

En este capítulo primero se presentará la simulación de la respuesta al impulso de los decimadores. Después se definirá la forma de onda y frecuencia de muestreo de la señal de entrada para simular el consumo de potencia de los decimadores. Luego, se presentará la cantidad de área usada por los decimadores al ser implementados en una tecnología CMOS de 0.18um. Finalmente, se presentarán comparaciones con base en los resultados de consumo de potencia y área usada.

7.1 Simulación

7.1.1 Respuesta al impulso

Para verificar el funcionamiento lógico de los circuitos descritos en VHDL se utiliza el simulador MODELSIM incluido en el entorno Xilinx ISE 9.1. Este permite agregar señales síncronas y asíncronas, definidas por el usuario, a las entradas del circuito con el objetivo de obtener transiciones lógicas a la salida y verificar el funcionamiento. La correcta descripción VHDL de los filtros digitales se puede comprobar obteniendo su respuesta al impulso [6]. Esto se logra aplicando una entrada asíncrona de valor 1, que provoca que las muestras de salida del filtro sean los coeficientes del polinomio característico de la función de transferencia.

Para los filtros que se utilizan como filtro antialiasing en decimadores, los coeficientes del

polinomio que se presentan a la salida son solo aquellos que corresponden a múltiplos enteros de z^{-M}, además de z^0. Por ejemplo para cuando $M = 16$ los coeficientes a la salida son aquellos correspondientes a z^0, $z^{-16}, z^{-32}, z^{-48}$…etc. Para los decimadores CIC no recursivos, ya sea en implementación directa o en componentes polifásicas, el polinomio característico se obtiene expandiendo todas las etapas por su valor de cascada K y multiplicándolas entre sí. No importa que las etapas se encuentren intercaladas con los DS, los coeficientes de salida también serán los correspondientes a múltiplos enteros de z^{-M}.

Cuando existen diferentes factores de sub-muestreo es necesario sincronizar la salida de las diferentes etapas, de no hacerlo el resultado puede ser inconsistente debido a que un DS es una función *variante en el tiempo*. En los decimadores aquí presentados esto se logra simplemente sincronizando la etapa 0 con la etapa 1, y las demás se sincronizan automáticamente debido a que en esta forma fue diseñado el divisor de frecuencias en la Sección 5.2.

Para la sincronización de la etapa 0 y 1 en la implementación directa, es necesario conocer la latencia en ciclos de reloj que tiene cada etapa la cual viene determinada por el valor K. Una vez se conoce este valor, el impulso se debe introducir en la K-ésima transición de clk_0. Por ejemplo, un decimador con $K = 6$ tiene una latencia de seis periodos de reloj por cada etapa así que el impulso debe ingresar a la etapa 0 en la sexta transición de clk_0.

Para las implementaciones en componentes polifásicas, la latencia de la etapa no depende del valor de K, debido a que la estructura directa transpuesta permite tener una latencia de solo un ciclo de reloj. Sin embargo, en la estructura usada en este trabajo (Fig. 5.11) se han implementado dos registros adicionales, por lo que se obtiene una latencia de dos ciclos de reloj por cada etapa. De esta manera, para la sincronización, el impulso se debe introducir en la segunda transición de la señal clk_0. Esto se hace para todos los decimadores polifásicos debido a que esta latencia no depende de la complejidad del sub-filtro.

La respuesta al impulso para los decimadores en implementación directa y polifásica es la misma, debido a que en ambos casos la función de transferencia es equivalente y el factor de sub-muestreo también. La única diferencia radica en la rapidez de la respuesta que para

el caso de la implementación directa es menor debido a que la latencia depende del valor de K, mientras que en la polifásica siempre es igual.

A. Decimador CIC no recursivo con $M = 32, K = 5$

En la Fig. 7.1 se puede observar la respuesta al impulso para este decimador. La salida $y[25:0]$ presenta los coeficientes de la función de transferencia correspondientes a z^0, $z^{-32}, z^{-64}, z^{-96}$…etc. Estos requieren que la longitud de palabra de salida sea 26 bits, tal y como se describió en la implementación. También se pueden ver las señales generadas por el divisor de frecuencias y que las muestras de salida tienen la misma frecuencia que clk_5 que es 32 veces menor que la de clk_0.

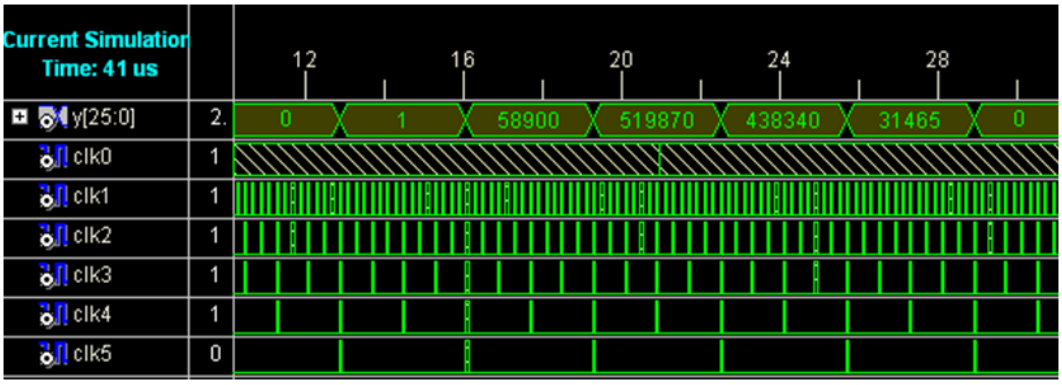


Fig. 7.1 Respuesta al impulso para el decimador CIC no recursivo con $M = 32$, $K = 5$.

B. Diseño 1 con $M = 32, K_1 = 5, K_2 = 3$

En la Fig. 7.2 se puede observar la respuesta al impulso para el **Diseño 1** con $M = 32$, $K_1 = 5$ y $K_2 = 3$. La salida $y[28:0]$ tiene una longitud de palabra de 29 bits y presenta los coeficientes de la función de transferencia correspondientes a z^0, $z^{-32}, z^{-64}, z^{-96}$…etc. a la misma frecuencia que clk_5. A diferencia de la Fig. 7.1 la cantidad de muestras de salida es mayor debido a que el aumento del valor de la cascada en la última etapa, para mejorar la atenuación, provoca que el polinomio característico sea de mayor longitud.

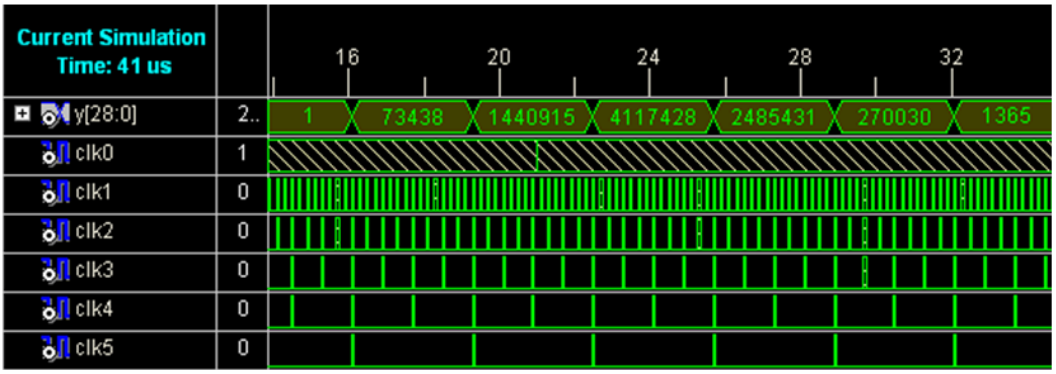


Fig. 7.2 Respuesta al impulso para el decimador **Diseño 1**.

C. Decimador CIC no recursivo con $M = 16,\ K = 6$

Su respuesta al impulso es presentada en la Fig. 7.3. La salida $y[24:0]$ tiene una longitud de palabra de 25 bits y presenta los coeficientes de la función de transferencia correspondientes a $z^0,\ z^{-16}, z^{-32}, z^{-48}$, etc. a la misma frecuencia que clk_4, la cual es dieciséis veces menor que la frecuencia de clk_0.

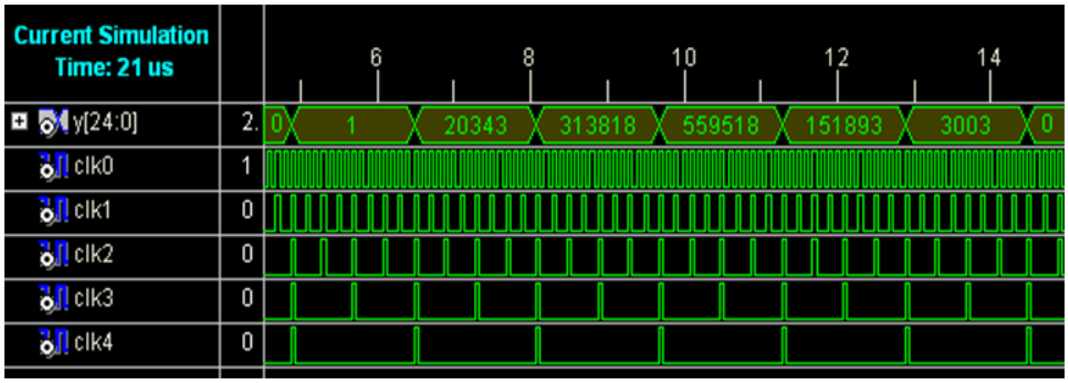


Fig. 7.3 Respuesta al impulso para el decimador CIC no recursivo con $M = 16$ y $K = 6$.

D. Diseño 2 - Caso 1 con $M = 16, K_1 = 5, K_2 = 1$

En la Fig. 7.4 se puede observar la respuesta al impulso para este **Caso 1**. La salida $y[21:0]$ tiene una longitud de palabra de 22 bits y presenta los coeficientes de la función de transferencia correspondientes a $z^0,\ z^{-16}, z^{-32}, z^{-48}$, etc. a la misma frecuencia que clk_4.

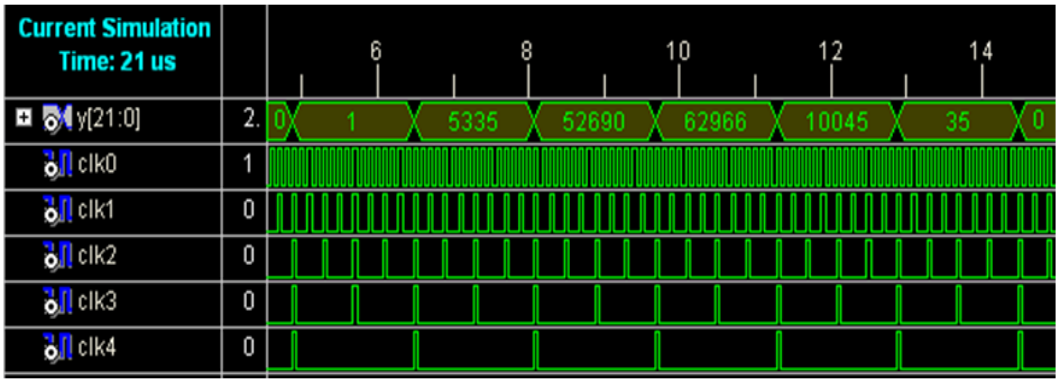


Fig. 7.4 Respuesta al impulso para el decimador **Diseño 2 - Caso 1**.

E. Diseño 2 - Caso 2 con $M = 16, K_1 = 4, K_2 = 3$

En la Fig. 7.5 se puede observar la respuesta al impulso para el **Caso 2**. Su salida $y[19:0]$ tiene una longitud de palabra de 20 bits y presenta los coeficientes de la función de transferencia correspondientes a z^0, $z^{-16}, z^{-32}, z^{-48}$, etc. a la misma frecuencia que clk_4.

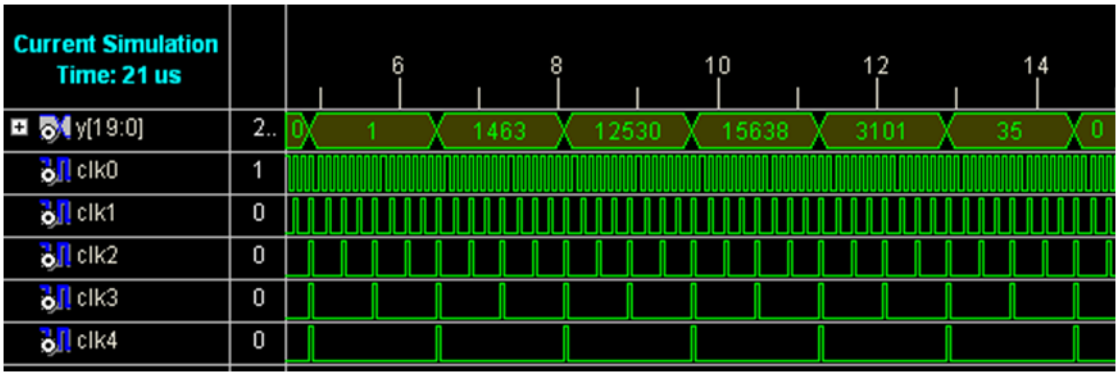


Fig. 7.5 Respuesta al impulso para el decimador **Diseño 2 - Caso 2**.

7.1.2 Consumo de potencia

Los circuitos CMOS digitales tienen dos componentes que contribuyen al consumo de potencia, el dinámico y el estático [36]. Sin embargo, en muchos casos de circuitos síncronos, que trabajan a una determinada frecuencia, se puede aproximar su consumo total mediante la potencia dinámica

$$P_D = AFCV_{DD}^2, \tag{7.1}$$

donde A es el factor de actividad, F es la frecuencia de operación, C es la capacitancia, y V_{DD} es el voltaje de alimentación.

Los decimadores analizados en esta tesis, tienen etapas que trabajan con diferentes frecuencias de muestreo, debido a que existen bloques DS entre cada etapa, por lo que para aproximar su consumo de potencia mediante (7.1) es necesario modificarla a

$$P_T = V_{DD}^2 \left[F_s C_{S_0} + \frac{F_s C_{S_2}}{2} + \frac{F_s C_{S_3}}{4} \ldots + \frac{F_s C_{S_{P-1}}}{2^{P-I}} \right]$$

$$= V_{DD}^2 F_s \left[C_{S_0} + \frac{C_{S_2}}{2} + \frac{C_{S_3}}{4} \ldots + \frac{C_{S_{P-1}}}{2^{P-I}} \right], \qquad (7.2)$$

donde I es un entero de valor 0 o 1 según se trate de una implementación polifásica o directa, respectivamente, C_{S_i} representa la capacitancia de la *i-ésima* etapa y F_s la frecuencia de muestreo de la señal de entrada. Para simplificar la ecuación (7.2) el factor de actividad se ha considerado como *A=1*, sin embargo, en la práctica tiene un valor diferente.

A partir de la ecuación (7.2) se puede ver que la etapa 0 es la que aporta un mayor consumo de potencia, debido a que trabaja con la frecuencia de muestreo más alta. Conforme se avanza sobre las demás etapas el tamaño de la palabra necesaria para los sumadores y registros va en aumento por lo cual se incrementa la capacitancia. Sin embargo, si se considera que la capacitancia es función del parámetro K de cada etapa resulta predominante la reducción de la frecuencia de muestreo, y entre mayor sea el término 2^{P-1} de una etapa menor será el impacto en el aumento de consumo de potencia. Por esta razón, se puede esperar un incremento muy pequeño en el consumo de potencia de la estructura propuesta, la cual agrega K_2 cascadas en la última etapa que opera a la frecuencia $F_s/2^{P-I}$.

Para medir el *consumo de potencia dinámica promedio* de los decimadores implementados en tecnología CMOS de 0.18µm, primero se describió un modulador sigma-delta de primer orden, ideal, mediante el lenguaje *Verilog-A* (analógico), el cual trabaja con una frecuencia de muestreo de 10MHz y se le aplica una señal de entrada senoidal de 78.125KHz, es decir, realiza un sobre-muestreo de la señal de entrada por un factor de 128, la Fig. 7.6 ilustra esto. Luego, usando el *netlist* de los decimadores, en el programa HSPICE se simuló el

consumo de potencia dinámica promedio, con el comando *".meas tran pot avg*" [36], durante $12.8\mu s$, tiempo equivalente a un periodo de la señal senoidal de entrada. En la Tabla 7.1 se muestra un resumen de los resultados, tanto para la implementación directa como en componentes polifásicas.

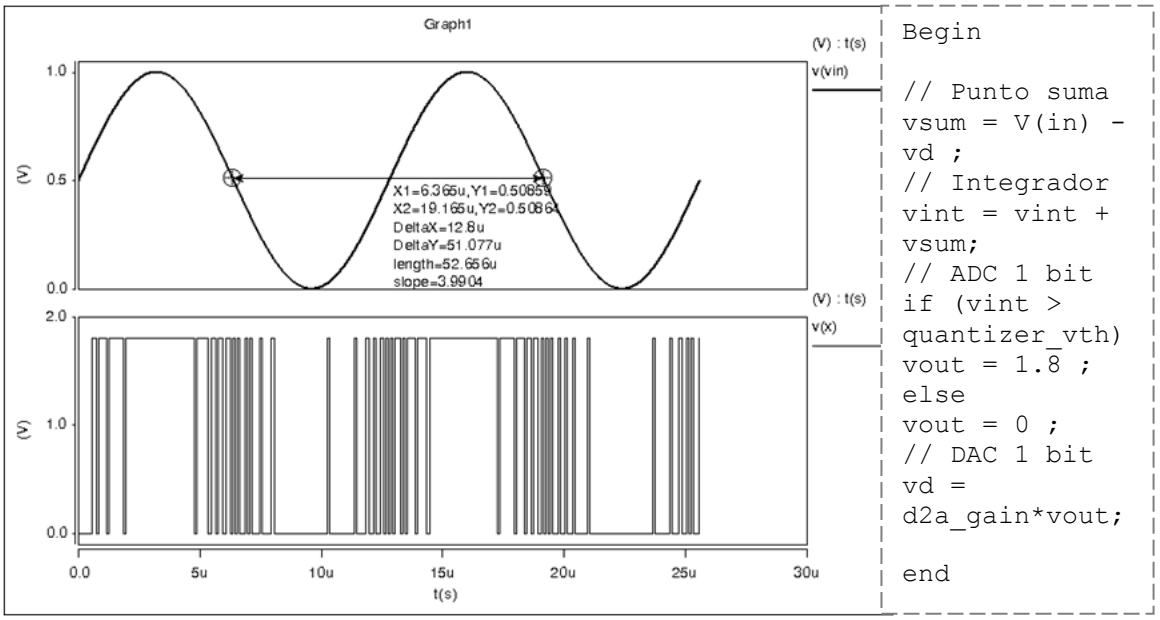


Fig. 7.6 Señal de entrada (arriba) y salida (abajo) del modulador sigma-delta, y su descripción en Verilog-A.

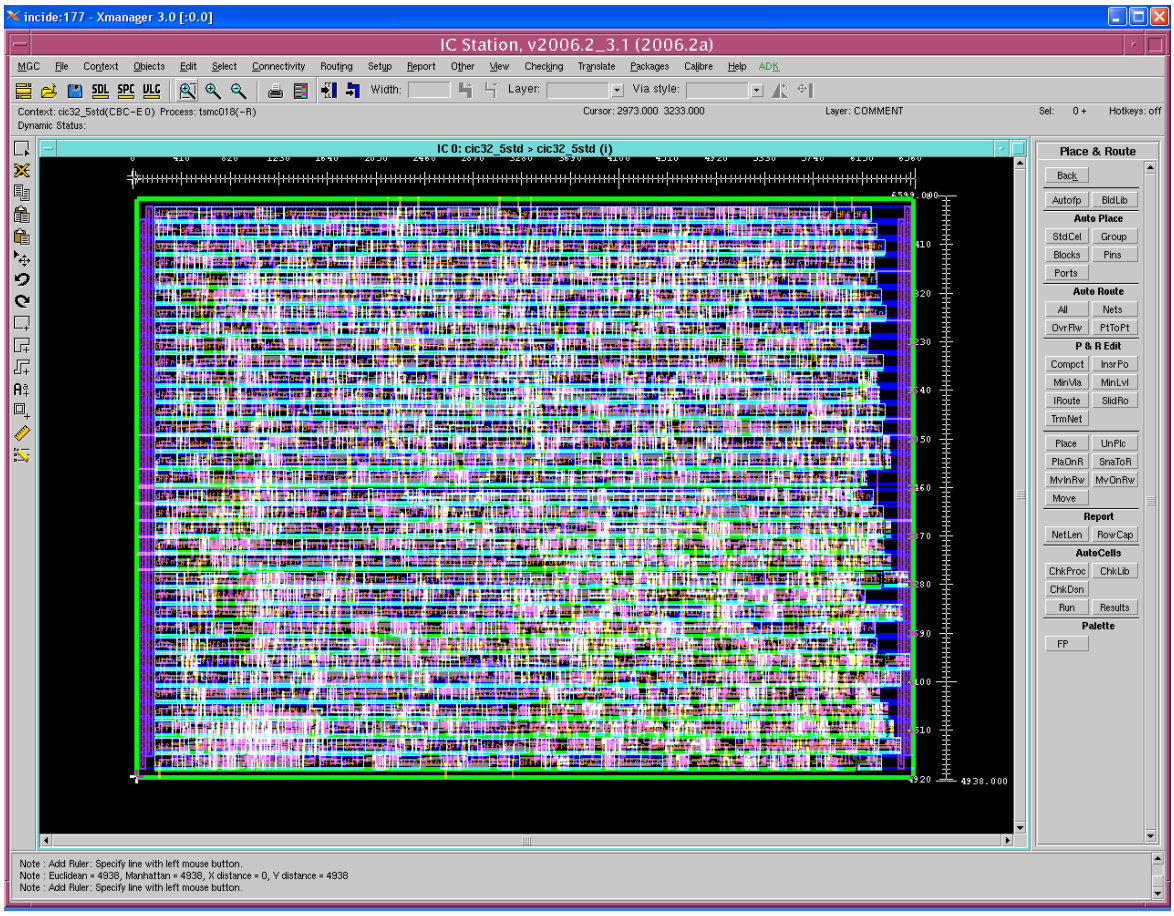
Decimador	F_{in}	F_{out}	Directa	Polifásica
CIC no recursivo: $M = 32, K = 5$	$10MHz$	$3.125KHz$	409.7 μW	281.5 μW
Diseño 1: $M = 32$, $K_1 = 5, K_2 = 3$	$10MHz$	$3.125KHz$	447.7 μW	305.1 μW
CIC no recursivo: $M = 16, K = 6$	$10MHz$	$625KHz$	501.5 μW	358.8μW
Diseño 2 - Caso 1: $M = 16, K_1 = 5$, $K_2 = 1$	$10MHz$	$625KHz$	390.4 μW	262.8 μW
Diseño 2 - Caso 1: $M = 16, K_1 = 4$, $K_2 = 3$	$10MHz$	$625KHz$	299 μW	216.3 μW

Tabla. 7.1 Resultados de consumo de potencia para los decimadores implementados en tecnología CMOS de 0.18µm, con una señal de entrada como la de la curva inferior de la Fig. 7.6.

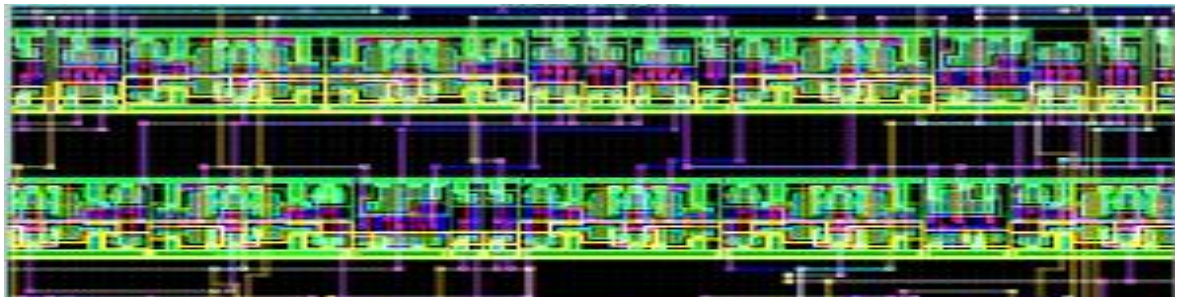
Como puede verse en la Tabla 7.1 la implementación en componentes polifásicas, para un decimador determinado, es más eficiente que la implementación directa, debido a la menor frecuencia de operación de cada etapa.

7.1.3 Área utilizada

El área utilizada por cada uno de los decimadores se obtuvo a partir de su layout, realizado en tecnología CMOS de 0.18µm. A manera de ilustrar esto en la Fig. 7.7 (a) se muestra el layout para el decimador CIC no recursivo con $M = 32$, $K = 5$ y en la Fig. 7.7 (b) una ampliación del mismo.



(a)



(b)

Fig. 7.7 Layout del decimador CIC no recursivo con $M = 32$, $K = 5$ (a) Completo y (b) ampliación.

En la Tabla 7.2 puede encontrarse un resumen del área utilizada por los decimadores aquí considerados, tanto para su implementación directa como para la polifásica.

Decimador	Implementación directa	Implementación polifásica
CIC no recursivo: $M = 32, K = 5$	$264,330.00\ \mu m^2$	$329{,}820.00\ \mu m^2$
Diseño 1: $M = 32$, $K_1 = 5, K_2 = 3$	$329{,}820.00\ \mu m^2$	$370{,}548.00\ \mu m^2$
CIC no recursivo: $M = 16, K = 6$	$243{,}097.00\ \mu m^2$	$260{,}820.00\ \mu m^2$
Diseño 2 - Caso 1: $M = 16, K_1 = 5$, $K_2 = 1$	$198{,}158.00\ \mu m^2$	$203{,}614.00\ \mu m^2$
Diseño 2 - Caso 1: $M = 16, K_1 = 4$, $K_2 = 3$	$164{,}943.00\ \mu m^2$	$185{,}887.00\ \mu m^2$

Tabla. 7.2 Resultados del área utilizada por los decimadores implementados en tecnología CMOS de 0.18µm.

En la Tabla 7.2, se puede ver que para un decimador determinado su implementación polifásica utiliza mayor cantidad de área que su implementación directa, debido al mayor número de sumadores requeridos para implementar los coeficientes de los sub-filtros. De hecho, si los coeficientes no hubiesen sido implementados mediante corrimientos, sumas/restas y compartición de sub-expresiones comunes los resultados de área usada para las implementaciones polifásicas hubieran sido mayores, debido a que un multiplicador requiere una gran cantidad de área para su implementación [23].

7.2 Comparación

7.2.1 Propuesta: Diseño 1

En este apartado se compara el decimador CIC no recursivo con $M = 32$ y $K = 5$ que tiene una atenuación en el primer doblez de la banda de -50 dB, frente al Diseño 1 con $M = 32$, $K_1 = 5$, $K_2 = 3$, el cual mejora la APDB a -70dB agregando únicamente una cascada $K_2 = 3$ en la última etapa. Esta comparación se hace en términos de consumo de potencia y área utilizada. Primero para la implementación directa y después para la polifásica.

A. Implementación directa

En la Tabla 7.3 se muestra un resumen de los resultados de consumo de potencia y área usada en la implementación directa del decimador CIC no recursivo con $M = 32$, $K = 5$ y el **Diseño 1**, así como también el aumento relativo en consumo de potencia y área del **Diseño 1** con respecto al decimador CIC no recursivo con $M = 32$, $K = 5$.

Decimador	APDB	Potencia total	Potencia extra	Área total utilizada	Área extra
CIC no recursivo: $M = 32; K = 5$	-50dB	$409.7\ \mu W$	0%	$264,330.00\ \mu m^2$	0%
Diseño 1: $M = 32;$ $K_1 = 5;\ K_2 = 3$	-70dB	$447.7\ \mu W$	8.5%	$334,194.00\ \mu m^2$	20.9%

Tabla 7.3 Comparación del consumo de potencia y área utilizada, en implementación directa, para el decimador CIC no recursivo con parámetros $M = 32, K = 5$ y el **Diseño 1**.

Respecto del consumo de potencia, a partir de la Tabla 7.3, se puede ver que el **Diseño 1** tienen un incremento del 8.5% relativo al decimador CIC no recursivo con $M = 32$, $K = 5$. Este aumento es muy pequeño, debido a que la cascada $K_2 = 3$ que se agrega en la última etapa trabaja en la frecuencia de muestreo más baja por lo que según la ecuación (7.2) esto no supone un gran impacto en el consumo de potencia total, aún a pesar del aumento de la capacitancia de la última etapa.

Respecto del área usada, a partir de la Tabla 7.3, se puede ver un aumento relativo de 20.9% para el **Diseño 1**. Esto sucede debido que los $K_2 = 3$ sumadores que se agregan en la última etapa tienen la longitud de palabra más alta 27, 28 y 29 bits, que al ser comparados con los de las primeras etapas constituyen un incremento considerable en la cantidad de área usada para su implementación.

Estos resultados comprueban que la estructura propuesta **Diseño 1** con $M = 32$, $K_1 = 5$, $K_2 = 3$ permite mejorar considerablemente la APDB del decimador CIC no recursivo con $M = 32$ y $K = 5$, con un ligero aumento de 8.5% en el consumo de potencia y un pequeño incremento de 20.9% en el área utilizada.

B. Implementación polifásica

En la Tabla 7.4 se muestran los resultados de consumo de potencia y área usada para la implementación polifásica del decimador CIC no recursivo con parámetros $M = 32$, $K = 5$ y el **Diseño 1,** donde también puede encontrarse el aumento relativo del **Diseño 1** con respecto al decimador CIC no recursivo con $M = 32$ y $K = 5$.

Decimador	APDB	Potencia total	Potencia extra	Área total utilizada	Área extra
CIC no recursivo: $M = 32; K = 5$	-50dB	281.5 μW	0%	329,820.00 μm^2	0%
Diseño 1: $M = 32;$ $K_1 = 5;\ K_2 = 3$	-70dB	305.1μW	8.6%	370,548 .00 μm^2	10.9%

Tabla 7.4 Comparación del consumo de potencia y área utilizada, en implementación polifásica, para el decimador CIC no recursivo con $M = 32, K = 5$ y el **Diseño 1**.

En componentes polifásicas el **Diseño 1** tiene un aumento en el consumo de potencia de 8.6% relativo al decimador CIC no recursivo con $M = 32$, $K = 5$, como puede observarse en la Tabla 7.4. En realidad, el incremento pudo haber sido mayor, debido al aumento en complejidad de la última etapa (mayor número de sumadores y registros). Sin embargo, debido a la implementación eficiente mediante corrimientos, sumas/restas, la simetría que presentan los coeficientes y sobre todo a la menor frecuencia de muestreo para esta etapa, el incremento relativo en consumo de potencia es comparable con el de la implementación directa.

En componentes polifásicas el **Diseño 1** tiene un aumento en el área usada de 10.9%, relativo al decimador CIC no recursivo con $M = 32$, $K = 5$. Este incremento es algo menor que en la implementación directa, debido a que cada etapa utiliza una longitud de palabra fija para todos sus elementos, y el aumento en $K_2 = 3$ bits en la longitud de palabra de la última etapa no supone un gran incremento en relación con las demás etapas. Además que el incremento en el número de sumadores de la última etapa con respecto a las primeras no es muy grande, gracias a las implementaciones de los coeficientes realizadas (véase la Fig. 6.2).

A partir de estos resultados se puede ver que también en implementación polifásica el **Diseño 1** cumple con su objetivo de incrementar la APDB a la vez que aumenta ligeramente su consumo de potencia (8.6%). No obstante, como se presentó en la Sección 6.1.2, la complejidad de implementación es algo mayor, debido a que es necesaria una estructura polifásica para las primeras etapas con valor de cascada K_1 y otra para la última etapa con $K_1 + K_2$, la cual siempre tendrá más coeficientes que las primeras.

7.2.2 Propuesta: Diseño 2

En este apartado se compara el decimador CIC no recursivo con $M = 16$ y $K = 6$, que tiene una APDB de -60 dB, frente al **Diseño 2 – Caso 1** y **Caso2** que surgen de la estructura propuesta de bajo consumo de potencia con APDB de -60.6 y -66.6dB, respectivamente. Esta comparación se hace en términos de consumo de potencia y área utilizada. Primero para la implementación directa y después para la polifásica.

A. Implementación directa

Los resultados de consumo de potencia y área usada en la implementación directa del decimador CIC no recursivo con $M = 16$ y $K = 6$, el **Diseño 2 – Caso 1** y **2**, se muestran en la Tabla 7.5, así como también el ahorro relativo de las estructuras de bajo consumo.

Decimador	APDB	Potencia total	Ahorro potencia	Área total utilizada	Ahorro área
CIC no recursivo: $M = 16;\ K = 6$	-60dB	$501.5\ \mu W$	0%	$243{,}097.00\ \mu m^2$	0%
Diseño 2 - Caso 1: $M = 16;$ $K_1 = 5;\ K_2 = 1$	-60.6dB	$390.4 \mu W$	22.5%	$198{,}158.00\ \mu m^2$	18.4%
Diseño 2 - Caso 2: $M = 16;$ $K_1 = 4;\ K_2 = 3$	-66.8dB	$299 \mu W$	40.3%	$164{,}943.00\ \mu m^2$	32.1%

Tabla 7.5 Comparación del consumo de potencia y área utilizada, en implementación directa, para el decimador CIC no recursivo con $M = 16$, $K = 6$ y el **Diseño 2 – Caso 1** y **Caso2**.

En términos de consumo de potencia, a partir de la Tabla 7.5, puede verse que el **Diseño 2 - Caso 1** tiene un ahorro en su consumo relativo al decimador CIC no recursivo con $M = 16$, $K = 6$ de aproximadamente 21.5%. Esto se debe a la reducción de la cascada de las tres primeras etapas que trabajan a mayor frecuencia de muestreo. Para el **Diseño 2 - Caso 2** la cantidad de cascada de las primeras etapas se reduce aún más y a pesar de que la última etapa aumenta su número de cascada el consumo de potencia relativo se reduce considerablemente hasta un valor de 40.3%. Esto sucede debido a que el aumento de la cascada en la última etapa no supone un gran incremento en el consumo de potencia, mientras que un menor número de cascada en las primeras etapas tiene un gran impacto en la reducción del consumo de potencia, como lo demuestra la ecuación (7.2).

De forma similar a la reducción en consumo de potencia existe una reducción en el área usada por los decimadores del **Diseño2 - Caso 1** y **2** en comparación con el decimador CIC no recursivo con $M = 16$ y $K = 6$, debido a la menor cantidad de celdas básicas necesarias en la implementación de las primeras etapas. Estas reducciones son del 18.4% y 32.1%, respectivamente.

A partir de estos resultados se puede ver que la estructura de bajo consumo de potencia **Diseño2 - Caso 1** y **2** cumple con su objetivo de mantener la mínima atenuación en el primer doblez de la banda a la vez que reducen considerablemente el consumo de potencia. Además, esta estructura de bajo consumo de potencia también presenta una reducción en la cantidad de área necesaria para su implementación por lo que la convierte en buen candidato para usarse en aplicaciones móviles donde se requiere bajo consumo de potencia y poca área en la implementación. También puede verse que para obtener el mínimo consumo de potencia posible hay que elegir aquella combinación de K_1 y K_2 que satisfacen la condición (4.23) donde K_1 es el número menor posible, tal y como lo demuestra el **Diseño2 - Caso 2.**

B. Implementación polifásica

La Tabla 7.6 muestra los resultados de consumo de potencia y área usada en la implementación polifásica del decimador CIC no recursivo con $M = 16$ y $K = 6$ y el **Diseño 2 – Caso 1** y **2**.

Decimador	APDB	Potencia total	Ahorro potencia	Área total utilizada	Ahorro área
CIC no recursivo: $M = 16;\ K = 6$	-60dB	$358.8\mu W$	0%	$260{,}820.00\ \mu m^2$	0%
Diseño 2 - Caso 1: $M = 16;$ $K_1 = 5;\ K_2 = 1$	-60.6dB	$282.8\mu W$	21.1%	$203{,}614.00\ \mu m^2$	21.9%
Diseño 2 - Caso 2: $M = 16;$ $K_1 = 4;\ K_2 = 3$	-66.8dB	$216.3\mu W$	39.7%	$185{,}887.00\ \mu m^2$	28.8%

Tabla 7.6 Comparación del consumo de potencia y área utilizada, en implementación polifásica, para el decimador CIC no recursivo con $M = 16$, $K = 6$ y el **Diseño 2 – Caso 1** y **Caso2**.

En componentes polifásicas también se obtiene un menor consumo de potencia para los decimadores del **Diseño 2 – Caso 1** y **2,** como se muestra en la Tabla 7.6, donde puede verse que para el **Diseño 2 - Caso 1** el ahorro es del 21.1% y para el **Caso 2** del 39.7 %, en relación con el decimador CIC no recursivo con $M = 16$ y $K = 6$.

Es posible notar que en componentes polifásicas el ahorro de potencia es algo menor que en las implementaciones directas, y es debido al incremento en la complejidad de cada etapa, mayor número de sumadores/restadores. Sin embargo, estos resultados no distan demasiado de los obtenidos en la implementación directa, debido a que el aumento en la complejidad es similar en las primeras etapas y no presupone un incremento considerable en el consumo de potencia.

De forma similar a como sucedió en la implementación directa, en la tabla 7.6, se puede ver que en implementación polifásica las dos estructuras de baja potencia **Diseño 2 – Caso 1 y 2** requieren de una menor cantidad de área para su implementación en comparación con el decimador CIC no recursivo con $M = 16$ y $K = 6$.

Para el **Caso 1** puede apreciarse un ahorro en el área ligeramente mayor que en la implementación directa. Por otra parte, el ahorro de área para el **Caso 2** es ligeramente menor que el obtenido en la implementación directa. Esto sucede principalmente por la no homogeneidad de las estructuras polifásicas para las primeras y última etapa. Por ejemplo para el **Caso 1** la última etapa tiene un incremento en dos sumadores con respecto a las primeras cuatro etapas, que si bien es mayor al incremento en un sumador en la implementación directa, se hace con una longitud de palabra fija que tiene un menor impacto en la cantidad de área total (véase la Fig. 6.5). Para el **Caso 2** la última etapa tiene un incremento en ocho sumadores/restadores con respecto a las primeras tres etapas mientras que en la implementación directa el aumento es de sólo tres sumadores, por esta razón el área final es mayor en la implementación polifásica y por lo tanto se obtiene un menor ahorro. De hecho, si la implementación de esta última etapa no se hubiese hecho en forma eficiente mediante el uso de corrimientos, sumas/restas y compartición de sub-expresiones comunes, el resultado podría diferir considerablemente del obtenido en la implementación directa.

A partir de estos resultados se puede ver que el ahorro en consumo de potencia de las estructuras de bajo consumo **Diseño 2 – Caso 1 y 2** en comparación con el decimador CIC no recursivo con $M = 16$ y $K = 6$, en las implementaciones polifásicas, permanece muy cercano a las implementaciones directas. No obstante, el ahorro de área presenta ligeras variaciones debido a la no homogeneidad de las componentes polifásicas.

Conclusiones

El objetivo principal de este trabajo era mejorar la atenuación en la banda de rechazo del decimador CIC no recursivo con un mínimo incremento en la complejidad de diseño e implementación de la estructura con $M = 2^P$.

El objetivo se cumplió proponiendo una estructura eficiente que logra satisfacer dos diferentes objetivos de diseño.

El primer diseño consiste en mejorar la atenuación en el primer doblez de la banda a costa de un ligero incremento en la complejidad de diseño, implementación, consumo de potencia y área usada en comparación con el decimador CIC no recursivo.

El segundo diseño presenta una estructura de bajo consumo de potencia que permite tener la mínima atenuación en todos los dobleces de la banda. De forma similar al caso anterior, el incremento en la complejidad de diseño e implementación en comparación con el decimador CIC no recursivo es mínimo y se obtiene un gran ahorro en el consumo de potencia y área usada.

De tal manera que la misma estructura propuesta se puede utilizar para dos diferentes objetivos de diseño, lo cual le da una mayor flexibilidad comparándola con la estructura CIC no recursiva.

Sobre las implementaciones se concluye que para mantener los resultados de consumo de potencia y área usada, similar entre la implementación directa y polifásica, es necesario obtener estructuras con coeficientes representados en CSD para su implementación como corrimientos, sumas/restas.

Para facilitar la implementación polifásica de la estructura propuesta, si el exponente de la función de transferencia $(1 + z^{-1})$ es un número impar buscar la compartición de sub-expresiones comunes, y si es un número par aprovechar la simetría de los coeficientes.

Finalmente, la estructura propuesta puede ser una alternativa útil, donde un bajo consumo de potencia sea el requisito más importante a satisfacer, mientras se mejora la atenuación para aliasing en el primer doblez de la banda y/o se mantiene una atenuación mínima en todos los dobleces de la banda.

Trabajo a futuro

Con base en los resultados obtenidos se proponen los puntos siguientes como trabajo a futuro:

- Implementar las estructuras propuestas para diferentes valores de longitud de palabra, frecuencia de muestreo, factor de sub-muestreo etc., para obtener las figuras de merito correspondientes e identificar cuando es preferible la implementación directa o polifásica.

- Buscar una mayor reducción en el consumo de potencia de la estructura propuesta, modificando independientemente los valores de cascada de cada etapa.

Lista de figuras

Capitulo 1

Capitulo 2

Capitulo 5

Lista de tablas

Referencias

[1] Mitra, S. K. (2006), *Digital signal processing: A computer based approach*. 3rd edition. New York, NY: The McGraw-Hill.

[2] Jovanovic-Dolecek, G.(2002), *Introduction to multirate systems*, Idea Group Publishing.

[3] Oppenheim, A. V., & Schafer, R. W. (1989), *Discrete-time signal processing*, 3rd edition. London: Prentice-Hall International.

[4] Maloberti F. (2007), *Data Converters*, 1st edition, Springer International Edition.

[5] Hogenauer, E. (1981). "An economical Class of Digital Filters for Decimation and Interpolacion", *IEEE Transactions Acoustic, Speech and Signal Processing*, Vol.ASSP-29, (Apr.1981), pp.155-162.

[6] Meyer-Baese U. (2007), *Digital Signal Processing with Field Programmable Gate Arrays*, 3rd edition, Springer International Edition.

[7] Laddomada, M. (2007b), "Comb-Based Decimation Filters for ΣΔ A/D Converters: Novel Schemes and Comparisons", *IEEE Transactions on Signal Processing*, vol.55, No. 5, Part 1, pp 1769-1779.

[8] Shahana T. K., Rekha K. (2007), "Polyphase Implementation of Non-recursive Comb Decimators for Digma-Delta A/D Converters", *Electron Devices and Solid State Circuits. IEEE Conference*, Issue 20-22 Dec. 2007, pp. 825-828.

[9] M. Abbas, O. Gustafsson, and L. Wanhammar, (2010) "Power Estimation of Recursive and Non-Recursive CIC Filters Implemented in Deep-Submicron Technology," *IEEE Int. Conf. Green Circuits Syst*., Shanghai, China, June 21-23, 2010.

[10] Kim, S., et al., (2006), "Design of CIC Roll-off Compensation Filter in a W-CDMA Digital Receiver", *Digital Signal Processing*, (*Elsevier*), Vol. 16, No.6, (November 2006), pp.846-854.

[11] Yeung, K. S. & Chan, S. S. (2004), "The Design and Multiplier-less Realization of Software Radio Receivers with Reduced System Delay", *IEEE Transactions on Circuits and Systems-1: Regular papers*, Vol.51, No.12, (December 2004), pp.2444-2459.

[12] Jovanovic Dolecek, G. & Mitra, S. K. (2008), "Simple Method For Compensation of CIC Decimation Filter", *Electronics Letters*, vol.44, No.19, (Sempember 11, 2008), pp . 1270-1272.

[13] Lo Presti, L. (2000), "Efficient modified-sinc filters for sigma-delta A/D converters", *IEEE Trans. Circuits Syst. II, Analog and Digital Signal Processing*, vol.47, No.11, pp 1204-1213.

[14] Jovanovic Dolecek G. and Laddomadda M. (2010), "An Economical Class of Droop-Compensated Generalized Comb Filters: Analysis and Design", *IEEE Transactions on Circuits and Systems II: Express Brief*, Vol.51, Issue 4, pp.275-279.

[15] Jovanovic Dolecek, G. (2010a), "Simplified Rotated SINC (RS) Filter for Sigma-Delta A/D Conversion", Proceedings of *International Conference on green Circuits and Systems* ICGCS 2010, Shangai, China, (June 21-23 2010), pp.283-288.

[16] Kwentus A. & Willson, Jr. A, (1997), "Application of Filter Sharpening to Cascaded Integrator-Comb Decimation Filters", *IEEE Transactions on Signal Processing*, Vol.45, No.2, (February 1997), pp.457-467.

[17] Kaiser, F. & Hamming, R.W. (1977), "Sharpening the Response of a Symmetric Nonrecursive Filter by Multiple Use of the Same Filter", *IEEE Transactions Acoustic, Speech and Signal Processing*, Vol. 25, No. 5, (October 1977), pp. 415-422.

[18] G. Jovanovic Dolecek and S. K. Mitra, "Two-Stage CIC Based Decimator with Improved Characteristics" *IET Signal Processing*, February 2010, vol.4, Issue 1, pp.22-29.

[19] Saramäki T., & Ritoniemi, T. (1997), "A modified comb filter structure for decimation", *Proc. IEEE International Symp. Circuits and Systems – ISCAS*, pp. 2353-2356.

[20] Abu-Al-Saud, W.A. & Stuber, G.L. (2006) "Efficient sample rate conversion for software radio systems". *IEEE Transactions on Signal Processing*, vol.54, No.3, pp. 932 - 939.

[21] Laddomada, M., & Mondin, M. (2004), "Decimation schemes for ΣΔ A/D converters based on Kaiser and Hamming sharpened filters", *IEE Proceedings - Vision, Image and Signal Processing*, vol.151, No.5, pp. 287-296.

[22] Stephen, G., & Stewart, R.W. (2004), "High-speed sharpening of decimating CIC filter", *Electronics Letters*, vol.40, No.21, pp. 1383-1384.

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

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

[25] 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.

[26] 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.

[27] Hartley RI (1996), "Subexpression sharing in filters using canonic signed digit multipliers", *IEEE Trans. Circuits Syst. II*, vol.43, No.10, pp.677–688.

[28] Park I-C and Kang H-J (2002), "Digital filter synthesis based on an algorithm to generate all minimal signed digit representations", *IEEE Trans. Computer-Aided Design* vol.21, No.12: pp. 1525–1529.

[29] Dempster AG and Macleod MD (2004), "Digital filter design using subexpression eliminiation and all signed-digit representations", *Proc. IEEE Int. Symp. on Circuits and Systems* 3: pp. 169–172.

[30] Yao C-Y, Chen H-H, Lin T-F, Chien C-J and Hsu C-T (2004), "A novel common subexpression elimination method for synthesizing fixed-point FIR filters", *IEEE Trans. Circuits Syst. I*, vol.51, No.11, pp. 2215–2221.

[31] Macleod MD and Dempster AG (2005), "Multiplierless FIR filter design algorithms", *IEEE Signal Processing Lett*, vol.12, No.3, pp. 186–189.

[32] Wang Y and Roy K (2005), "CSDC: A new complexity reduction technique for multiplierless implementation of digital FIR filters", *IEEE Trans. Circuits Syst. I*, vol.52, No.9, pp.1845–1853.

[33] Flores P, Monteiro J and Costa E. (2005), "An exact algorithm for the maximal sharing of partial terms in multiple constant multiplications", *Proc. IEEE Intl. Conf. Computer-Aided Design*, San Jose, CA, pp. 13–16, Nov. 6–10

[34] Aboushady, H. et al. (2001), "Efficient Polyphase Decomposition of Comb Decimation Filters in Sigma Delta Analog-to Digital Converters", *IEEE Transactions on Circuits and Systems II*, Vol.48, No.10, pp.898-905, (October 2001).

[35] Martin K. (2000), *Digital Integrated Circuit Design*, New edition, Oxford University Press.

[36] Weste N., Harris D. and Banerjee A., *CMOS VLSI Design*, 3rd edition, Pearson International Edition, 2005.