5

Diseño de filtros en tiempo discreto

En este capítulo se trata el diseño de filtros digitales con respuesta infinita al impulso (RII). El diseño de estos filtros se basa en transformaciones que permiten convertir la función de transferencia de filtros analógicos $H(s)$, en filtros digitales o de tiempo discreto $H(z)$.

5.1. Diseño por solución directa de la ecuación en diferencias

Un sistema lineal en tiempo discreto e invariante al desplazamiento se puede caracterizar por su función de transferencia $H(z)$, o por una ecuación en diferencias.

Representando este sistema en tiempo discreto por su función de transferencia tenemos,

$$\sum_{k=0}^{M} b_k y_k = \sum_{k=0}^{N} a_k x_k; \quad M \leq N, \tag{5.1}$$

entonces, aplicando la transformada $\mathcal{Z}$,

$$H(z) = \frac{\sum_{k=0}^{N} a_k z^{-k}}{\sum_{k=0}^{M} b_k z^{-k}}, \tag{5.2}$$

cuya respuesta en frecuencia está dada por

$$H(e^{j\omega}) = \frac{\sum_{k=0}^{N} a_k e^{-j\omega k}}{\sum_{k=0}^{M} b_k e^{-j\omega k}}. \tag{5.3}$$

Podemos suponer que nuestro filtro digital actúa como un filtro analógico, desde el punto de vista de la entrada y salida analógicas del filtro, según

se presenta en la figura 5.1, donde $x_n = x_a(nT)$ y $y_n = y_a(nT)$ son las versiones muestreadas de $x_a(t)$ y $y_a(t)$, y H(z) es un sistema lineal en tiempo discreto.

Esto será nuestro punto de partida en el diseño de filtros digitales.

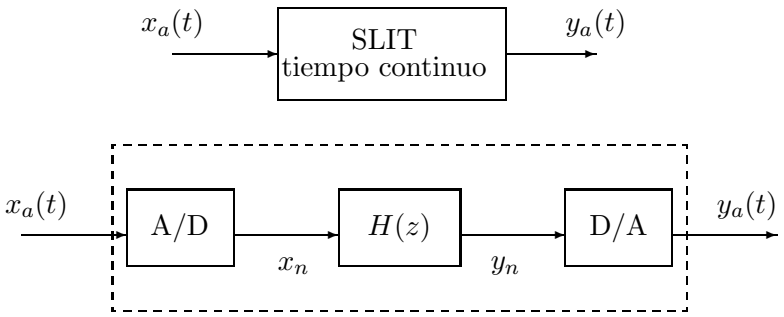


Figura 5.1: Sistema analógico equivalente en tiempo discreto.

Si nuestro filtro analógico está representado por la siguiente ecuación diferencial:

$$\sum_{k=0}^{N} p_k \frac{d^k}{dt^k} y_a(t) = \sum_{k=0}^{M} q_k \frac{d^k}{dt^k} x_a(t), \tag{5.4}$$

entonces, de la transformación de Laplace tenemos

$$H(s) = \frac{\sum_{k=0}^{N} q_k s^k}{\sum_{k=0}^{M} p_k s^k}. \tag{5.5}$$

Para trabajar en el dominio del tiempo discreto, podemos aproximar las derivadas por diferencias hacia atrás. Recuérdese la definición de derivada como

$$\frac{d}{dt} f(t) = \lim_{\Delta T \longrightarrow 0} \frac{f(t) - f(t - \Delta T)}{\Delta T}. \tag{5.6}$$

Entonces, para la primera derivada tenemos

$$\bigtriangledown^{(1)}[y(n)] = \frac{[y(n) - y(n-1)]}{T}, \tag{5.7}$$

donde $\bigtriangledown^{(k)}$ indica la primera diferencia. Para órdenes mayores tenemos

$$\bigtriangledown^{(k)}\{y[n]\} = \bigtriangledown^{(1)}\{\bigtriangledown^{(k-1)}\{y[n]\}\}. \tag{5.8}$$

Así, podemos escribir

$$\sum_{k=0}^{N} p_k \bigtriangledown^{(k)} \{y_a(nT)\} = \sum_{k=0}^{M} q_k \bigtriangledown^{(k)} \{x_a(nT)\}. \tag{5.9}$$

(5.9) es una aproximación numérica para obtener la versión muestreada de $y(t)$. Aplicando la transformada $\mathcal{Z}$ a la primera diferencia tenemos

$$\begin{aligned}\mathcal{Z}\{\bigtriangledown^{(1)}\{y[n]\}\} &= \mathcal{Z}\{\frac{y[n]-y[n-1]}{T}\} \\ &= \frac{1}{T}Y(z)(1-z^{-1}) \end{aligned} \tag{5.10}$$

y para la k-ésima diferencia,

$$\mathcal{Z}\{\bigtriangledown^{(k)}\{y[n]\}\} = Y(z)\Big[\frac{(1-z^{-1})}{T}\Big]^k. \tag{5.11}$$

Haciendo $T=1$, podemos escribir $y_a(nT)$ como $y[n]$, y $x_a(t)$ como $x[n]$, así,

$$\sum_{k=0}^{N} p_k \mathcal{Z}\{\bigtriangledown^{(k)}\{y[n]\}\} = \sum_{k=0}^{M} q_k \mathcal{Z}\{\bigtriangledown^{(k)}\{x[n]\}\},$$

$$\sum_{k=0}^{N} p_k \Big[\frac{(1-z^{-1})}{T}\Big]^k Y(z) = \sum_{k=0}^{M} q_k \Big[\frac{(1-z^{-1})}{T}\Big]^k X(z), \tag{5.12}$$

de modo que finalmente

$$H(z) = \frac{Y(z)}{X(z)} = \frac{\sum_{k=0}^{N} p_k [\frac{(1-z^{-1})}{T}]^k}{\sum_{k=0}^{M} q_k [\frac{(1-z^{-1})}{T}]^k}. \tag{5.13}$$

De (5.13) podemos ver, comparándola con (5.5), que $H(z)$ se puede obtener reemplazando s por $[(1-z^{-1})/T]$ en $H(s)$, esto es

$$H(z) = H(s)\Big|_{s=\frac{(1-z^{-1})}{T}}. \tag{5.14}$$

Lo anterior corresponde a una transformación (en algunos textos se le llama "mapeo") del plano-s al plano-z, de modo que

$$s = \frac{1-z^{-1}}{T}, \quad z = \frac{1}{1-sT} \tag{5.15}$$

La respuesta en frecuencia resultante se puede obtener haciendo $s = j\Omega$ en el dominio analógico, entonces

$$z = \frac{1}{1-j\Omega T}. \tag{5.16}$$

Se puede demostrar que con esta transformación el eje $j\Omega$, del plano-s, es transformado en un círculo de radio 0.5 centrado en $(x = 0.5,\ y = 0)$ sobre el plano-z, como se muestra en la figura 5.2.

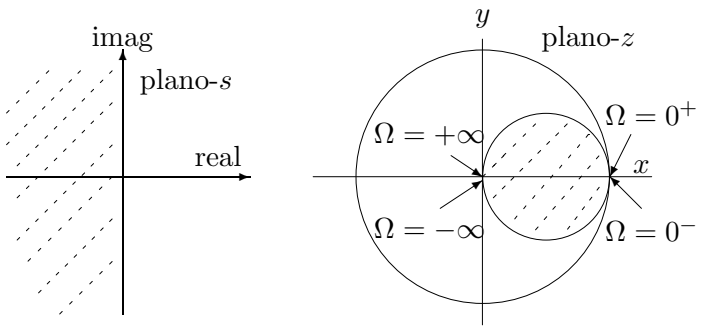


Figura 5.2: Transformación del plano-s al plano-z.

Así, los polos del semiplano izquierdo del plano-s estarán dentro del círculo unitario, lo que significa que esta transformación entregará un filtro digital estable a partir de un filtro analógico estable; sin embargo, la forma de la respuesta en frecuencia de $H(z)$ será diferente a la respuesta en frecuencia de $H(s)$. Recuérdese que obtenemos la respuesta en frecuencia del filtro digital evaluando $H(z)$ sobre el círculo unitario haciendo $z = e^{j\omega}$.

5.2. La transformación bilineal

Otra solución a la ecuación en diferencias está basada en la aplicación de la regla del trapecio para aproximar la integración.

Recordando la regla del trapecio tenemos,

$$\int_{t_1}^{t_2} x(\tau)d\tau = \frac{t_2 - t_1}{2}[x(t_1) + x(t_2)]. \tag{5.17}$$

Supóngase que tenemos una función de transferencia analógica dada por:

$$H_a(s) = \frac{1}{s}, \tag{5.18}$$

cuya respuesta al impulso es

$$h(t) = \mathcal{L}^{-1}H_a(s) = \begin{cases} 1, & \text{si } t \geq 0^+ \\ 0, & \text{si } t \leq 0^-, \end{cases} \tag{5.19}$$

la respuesta a una entrada arbitraria $x(t)$ es

$$y(t) = \int_0^t x(\tau)h(t-\tau)d\tau, \tag{5.20}$$

haciendo $t = t_1$ tenemos

$$y(t_1) = \int_0^{t_1} x(\tau)h(t_1-\tau)d\tau, \tag{5.21}$$

y si $t = t_2$

$$y(t_2) = \int_0^{t_2} x(\tau)h(t_2 - \tau)d\tau, \tag{5.22}$$

si además, $h(t_2 - \tau) = h(t_1 - \tau) = 1$, entonces

$$y(t_2) - y(t_1) = \int_{t_1}^{t_2} x(\tau)d\tau, \tag{5.23}$$

y aplicando la regla del trapecio para aproximar la integración

$$y(t_2) - y(t_1) \approx \frac{t_2 - t_1}{2}[x(t_1) + x(t_2)]. \tag{5.24}$$

Ahora, haciendo $t_2 = nT$ y $t_1 = nT - T$

$$y(nT) - y(nT - T) = \frac{T}{2}[x(nT - T) + x(nT)], \tag{5.25}$$

y de la transformada $\mathcal{Z}$

$$Y(z) - z^{-1}Y(z) = \frac{T}{2}[z^{-1}X(z) + X(z)], \tag{5.26}$$

de aquí que

$$H(z) = \frac{Y(z)}{X(z)} = \frac{T}{2}\frac{z+1}{z-1}. \tag{5.27}$$

Nuevamente, comparando $H(z)$ con $H_a(s)$ se puede ver que la función de transferencia en tiempo discreto se obtiene como

$$H(z) = H_a(s)\Big|_{s \to \frac{2}{T}\frac{z-1}{z+1}}. \tag{5.28}$$

La (5.28) es conocida como la transformación bilineal, y su efecto en la transformación del plano s al plano z se muestra en la figura 5.3. La transformación bilineal tiene las siguientes propiedades:

1. El eje $j\Omega$ del plano-s cae sobre el círculo unitario en el plano-z.
2. El semiplano izquierdo del plano-s cae dentro del círculo unitario.
3. No hay una relación lineal entre las frecuencias analógicas y las frecuencias del filtro digital.

Como resultado de la propiedad número 3, los filtros digitales tendrán la misma respuesta en amplitud sólo a bajas frecuencias ya que la relación entre ω y Ω es no lineal. Esto se puede demostrar haciendo $z = e^{j\omega}$ y $s = j\Omega$, de modo que

$$\begin{aligned} \Omega &= \frac{2}{T}\tan(\frac{\omega T}{2}) \\ \omega &= 2\tan^{-1}(\frac{\Omega T}{2}) \end{aligned} \tag{5.29}$$

A esta relación no lineal de las frecuencias ω y Ω se le llama el efecto de combadura *warping* de la transformada bilineal. Por simplicidad le llamaremos efecto de deformado.

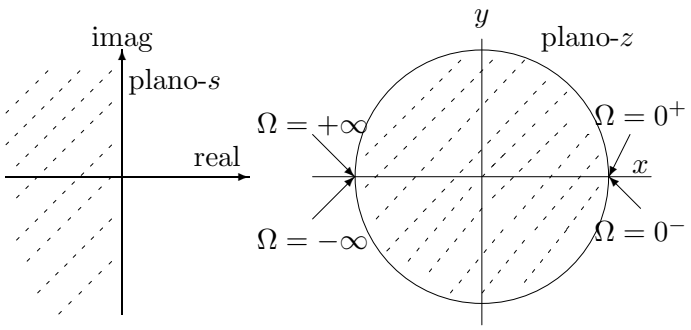


Figura 5.3: Transformación bilineal.

5.2.1. Diseño de filtros digitales por la transformación bilineal

Como se vio en la sección 5.2, al aplicar la transformación bilineal a un filtro analógico se presentará el efecto de deformación en las frecuencias digitales. Para obtener las frecuencias de corte deseadas debemos diseñar el filtro analógico con las frecuencias dadas por

$$\Omega_1 = \frac{2}{T}\tan(\frac{\omega T}{2}). \tag{5.30}$$

Esta operación se conoce como *pre-deformado*. Luego, diseñamos el filtro analógico con estas frecuencias pre-deformadas para, posteriormente, aplicarle la transformación bilineal para obtener la $H(z)$ deseada.

El procedimiento de diseño es:

1. Pre-deformar las especificaciones del filtro digital.
2. Diseñar el filtro analógico que satisfaga esas especificaciones analógicas.
3. Aplicar la transformación bilineal.

Para simplificar los cálculos en el diseño de filtros digitales por la transformación bilineal, se pueden hacer lo siguientes cambios sin que se afecten los resultados:

- Para el pre-deformado de las frecuencias digitales ω_i se empleará la siguiente expresión para obtener las frecuencias analógicas de diseño Ω_i:

$$\Omega_i = \tan(\frac{\omega_i}{2Fs}), \tag{5.31}$$

donde Fs es la frecuencia de muestreo.

- Para la transformación bilineal se empleará la expresión simplificada:

$$s = \frac{z-1}{z+1} \tag{5.32}$$

Note que se ha eliminado el término $2/T$ de ambas expresiones, obteniendo de este modo una simplificación en los cálculos.

Ejemplo 5.2.1 Diseñar un filtro digital de primer orden con frecuencia de corte de 1 rad/s. Considerar $T = 1$.
Solución:
Pre-deformando la frecuencia digital tenemos

$$\Omega_1 = \tan \frac{2\pi \times 0.1591}{2} = 0.5460.$$

Como se desea un filtro de primer orden, no hay necesidad de calcular n puesto que es igual a 1. Así que la función prototipo será $H(s) = 1/(s+1)$ con escalamiento en frecuencia

$$H(s) = \frac{1}{(s+1)_{s \to s/0.5460}} = \frac{1}{1.83123s + 1}.$$

Aplicando la transformada bilineal tenemos

$$H(z) = \frac{0.3532 + 0.3532z^{-1}}{1 - 0.29359z^{-1}}.$$

Ejemplo 5.2.2 Diseñar un filtro digital que satisfaga las siguientes especificaciones:

a) Banda de paso y de rechazo monotónicas.

b) Frecuencia de corte $\pi/2$ rad/s con -3 dB.

c) Atenuación de al menos 15 dB a una frecuencia de $3\pi/4$.

Emplear la transformada bilineal y T=1.
Solución:
Predeformando las especificaciones de frecuencia digital para obtener las frecuencias analógicas tenemos

$$\begin{aligned} \Omega_1 &= \tan(\pi/4) = 1.000, \\ \Omega_2 &= \tan(3\pi/8) = 2.4142135. \end{aligned}$$

El orden del filtro analógico será

$$\begin{aligned} n &= \lceil \frac{\log\left[(10^{0.3}-1)/(10^{1.5}-1)\right]}{2\log(1/2.4142135)} \rceil \\ &= \lceil 1.94383 \rceil = 2, \end{aligned}$$

así que la función prototipo es

$$H(s) = \frac{1}{s^2 + \sqrt{2}s + 1}|_s \rightarrow s/1,$$

ya que

$$\Omega_r = \frac{1.000}{(10^{0.3} - 1)^{1/4}} = 1.00118.$$

Aplicando la transformada bilineal resulta

$$\begin{aligned} H(z) &= \frac{1}{s^2 + \sqrt{2}s + 1}|_{s \rightarrow \frac{z-1}{z+1}} \\ &= \frac{0.29289 + 0.58579z^{-1} + 0.29289z^{-2}}{1 + 0.17158z^{-2}} \end{aligned}$$

5.3. Diseño de filtros digitales RII usando transformaciones digital-a-digital

Se ha mostrado, en las secciones anteriores, que es posible encontrar una función de transferencia en tiempo discreto $H(z)$ para filtros digitales Butterworth normalizados, aplicando la transformación bilineal. Podemos construir, nuevamente, un catálogo de funciones de transferencia en tiempo discreto $H(z)$ para filtros Butterworth, Chebyshev, y elípticos. Ahora la pregunta es ¿podemos diseñar un filtro digital con algunas especificaciones dadas usando solamente funciones de transferencia en tiempo discreto normalizadas?

Sí, ya que, de la misma manera que se diseñaron filtros analógicos, usando escalamiento en frecuencia (transformaciones analógico-a-analógico), se puede definir un conjunto de transformaciones digital-a-digital. Partiendo de un filtro digital pasa-bajas normalizado podemos obtener filtros pasa-altas, pasa-banda, y rechazo de banda. Este juego de transformaciones, dado en la tabla 5.1, fue propuesto por Constantinides (1970), donde Pb significa pasa-bajas, PA pasa-altas, PB pasa-banda, y RB rechazo de banda.

5.3.1. Determinación del orden n para un filtro digital Butterworth

Para un filtro pasa-bajas, usualmente tenemos una frecuencia de corte ω_1 y una frecuencia de rechazo ω_2 con ganancias k_1 y k_2 respectivamente, donde

$$\begin{aligned} 0 \geq 20 \log |H(e^{j\omega_1})| \geq k_1 \\ 20 \log |H(e^{j\omega_2})| \leq k_2, \end{aligned} \tag{5.33}$$

entonces,

$$n = \frac{\log[(10^{-k_1/10} - 1)/(10^{-k_2/10} - 1)]}{2 \log[\frac{\tan(\omega_1 T/2)}{\tan(\omega_2 T/2)}]}, \tag{5.34}$$

Tabla 5.1: Transformaciones digital-a-digital.

Transformación:	Ecuaciones:
Pb θ_p–a–Pb ω_p	$z^{-1} = \frac{z^{-1}-\alpha}{1-\alpha z^{-1}}$ donde $\alpha = \frac{\text{sen}[(\theta_p-\omega_p)/2]}{\text{sen}[(\theta_p+\omega_p)/2]}$
Pb θ_p–a–PA ω_p	$z^{-1} = -\frac{z^{-1}+\alpha}{1+\alpha z^{-1}}$ donde $\alpha = -\frac{\cos[(\theta_p+\omega_p)/2]}{\cos[(\omega_p-\theta_p)/2]}$
Pb θ_p–a–PB ω_p	$z^{-1} = -\frac{z^{-2}-\frac{2\alpha k}{k+1}z^{-1}+\frac{k-1}{k+1}}{\frac{k-1}{k+1}z^{-2}-\frac{2\alpha k}{k+1}z^{-1}+1}$ donde $\alpha = \frac{\cos[(\omega_1+\omega_2)/2]}{\cos[(\omega_2-\omega_1)/2]}$ y $k = \cot[(\omega_2-\omega_1)/2]\tan(\theta_p T/2)$
Pb θ_p–a–RB ω_p	$z^{-1} = \frac{z^{-2}-\frac{2\alpha}{1+k}z^{-1}+\frac{1-k}{1+k}}{\frac{1-k}{1+k}z^{-2}-\frac{2\alpha}{1+k}z^{-1}+1}$ donde $\alpha = \frac{\cos[(\omega_2+\omega_1)/2]}{\cos[(\omega_2-\omega_1)/2]}$ y $k = \tan[(\omega_2-\omega_1)/2]\tan(\theta_p T/2)$

para satisfacer el requisito de ganancia k_1, seleccionamos una frecuencia crítica

$$\Omega_p = \frac{2\tan(\omega_1 T/2)}{(10^{-k_1/10}-1)^{1/2n}}, \tag{5.35}$$

tal que

$$\omega_p = 2\tan[(10^{-k_1/10}-1)^{-1/2n}\, 2\tan(\omega_1 T/2)] \tag{5.36}$$

y finalmente,

$$H(z) = H_{Bn}(z)\big|_{z^{-1}=\frac{z^{-1}-\alpha}{1-\alpha z^{-1}}}, \tag{5.37}$$

donde

$$\alpha = \frac{\text{sen}[(\theta_p-\omega_p)/2]}{\text{sen}[(\theta_p-\omega_p)/2]}. \tag{5.38}$$

Ejemplo 5.3.1 Se desea un filtro digital normalizado Butterworth de orden 1. Obtenga $H(z)$ *a)* por transformación bilineal y *b)* empleando una transformación digital-a-digital.

Solución:

a) Para aplicar la transformación bilineal, partiendo de una $H(s) = 1/(s+1)$, requerimos de un filtro analógico de primer orden pero con frecuencia de corte

$$\Omega = \tan(1/2) = 0.546302$$

$$H_a(s) = H(s)|_{s=s/0.546302} = \frac{1}{(\frac{s}{0.546302})+1} = \frac{0.546302}{s+0.546302}.$$

Por transformación bilineal

$$H(z) = H_a(s)\big|_{s=\frac{z-1}{z+1}} = \frac{0.546302}{\frac{z-1}{z+1}+0.546302}$$

$$= \frac{(z+1)0.546302}{z-1+0.546302+0.546302} = \frac{0.546302(z+1)}{1.546302z-0.4536975}$$
$$= \frac{0.3532957896(1-z^{-1})}{1-0.293408z^{-1}}.$$

b) Si, en cambio, aplicamos directamente la transformación bilineal sobre el filtro analógico $H(s) = 1/(s+1)$, obtendremos una $H(z)$ con una frecuencia de corte que no corresponde a la del filtro digital normalizado. Entonces, aplicaremos una transformación digital-a-digital para obtener la $H(z)$ deseada.

5.4. Realización de filtros digitales RII

Por realización se entiende la traducción de la función de transferencia $H(z)$, primero en una estructura y después en una ecuación en el domino del tiempo, de modo que permita su implantación en el lenguaje propio de un procesador de señales digitales. Para el caso de filtros RII se tienen varias estructuras posibles.

5.4.1. Forma directa I

Sea

$$H(z) = \frac{\sum_{k=0}^{M} b_k z^{-k}}{1+\sum_{k=1}^{N} a_k z^{-k}}, \quad M \leq N, \tag{5.39}$$

entonces, de la transformada $\mathcal{Z}$ inversa tenemos

$$y(n) = -\sum_{k=1}^{N} a_k y(n-k) + \sum_{k=0}^{M} b_k x(n-k). \tag{5.40}$$

Una realización del filtro RII, usando (5.40) es llamada forma directa I. La figura 5.4 muestra tal realización, donde se puede observar que el número de bloques de retardo z^{-1} es igual a $M+N$.

5.4.2. Forma Directa II

Es posible obtener otra realización descomponiendo $H(z)$ en dos funciones: $H_1(z)$ y $H_2(z)$; donde $H_1(z)$ contiene sólo al denominador o polos de $H(z)$, y $H_2(z)$ contiene sólo al numerador o ceros de $H(z)$. Así, como se muestra en la figura 5.5

$$H(z) = H_1(z)H_2(z) = \frac{Y(z)}{X(z)}, \tag{5.41}$$

donde

$$H_1(z) = \frac{1}{1+\sum_{k=1}^{N} a_k z^{-k}} \text{ y} \tag{5.42}$$

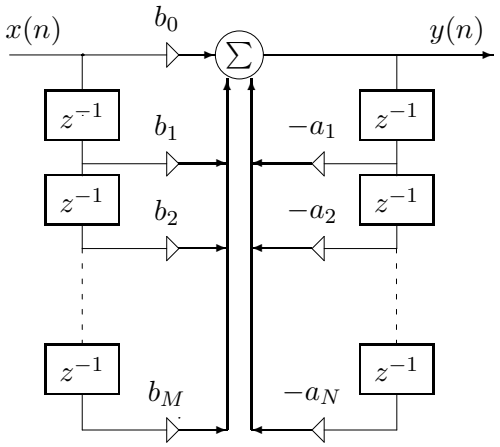


Figura 5.4: Forma directa I.

$$H_2(z) = \sum_{k=0}^{N} b_k z^{-k}. \tag{5.43}$$

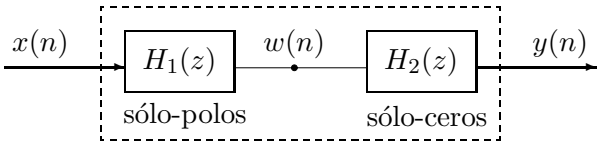


Figura 5.5: Descomposición de $H(z)$.

La salida se obtiene calculando el resultado intermedio $w(n)$ por:

$$W(z) = H_1(z)X(z) = \frac{1}{1 + \sum_{k=1}^{N} a_k z^{-k}} X(z) \tag{5.44}$$

y

$$Y(z) = H_2(z)W(z) = \sum_{k=0}^{N} b_k z^{-k} W(z) \tag{5.45}$$

Tomando las transformadas $\mathcal{Z}$ inversas

$$w(n) = x(n) - \sum_{k=1}^{N} a_k w(n-k) \tag{5.46}$$

y

$$y(n) = \sum_{k=0}^{M} b_k w(n-k). \tag{5.47}$$

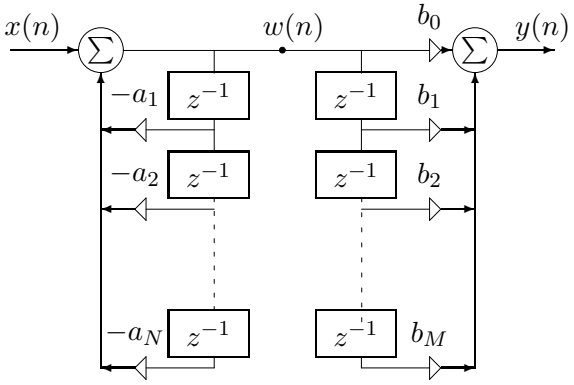


Figura 5.6: Forma directa II.

En forma gráfica tenemos las estructuras que se muestran en la figura 5.6.

Puede verse que las dos ramas de los elementos de retardo se pueden combinar en una sola, ya que se refieren a la misma versión retrasada de $w(n)$. Entonces, después de simplificar, tenemos la forma directa II simplificada o canónica en la figura 5.7.

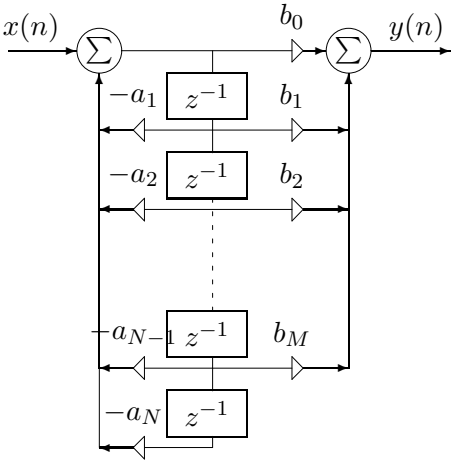


Figura 5.7: Forma (canónica) directa II.

En este caso, el número de bloques de retardo es igual a N, es decir, es igual al orden del filtro. N es el mínimo número de bloques de retardo. Esta realización es sólo una de las muchas que contienen el número mínimo de bloques de retardo, también como multiplicadores y sumadores. Ésa es una consideración importante para prevenir o reducir errores de redondeo y de cuantización.

Un caso especial e importante ocurre cuando $N = M = 2$, pues

$$H(z) = \frac{b_0 + b_1 z^{-1} + b_2 z^{-2}}{1 + a_1 z^{-1} + a_2 z^{-2}} = \frac{b_0(1 + b'_1 z^{-1} + b'_2 z^{-2})}{1 + a_1 z^{-1} + a_2 z^{-2}}, \tag{5.48}$$

donde $b'_1 = b_1/b_0$ y $b'_2 = b_2/b_0$. Entonces tenemos las secciones bi-cuadráticas, comúnmente llamadas secciones bi-cuads, compuestas de dos polinomios de segundo orden como se muestran en las figuras 5.8 y 5.9. La forma directa II alternativa es útil para escalamiento de la amplitud de $x(n)$ y mejora el desempeño del filtro.

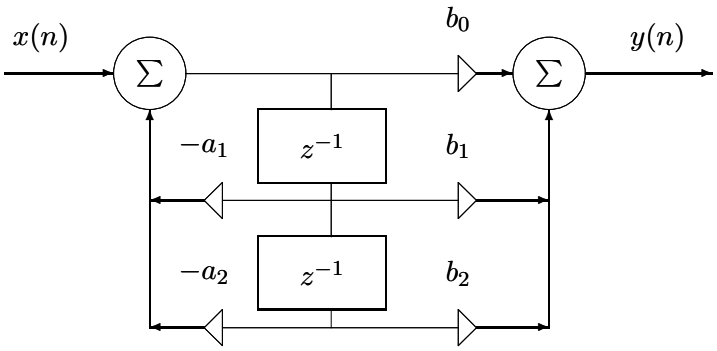


Figura 5.8: Sección bi-cuad.

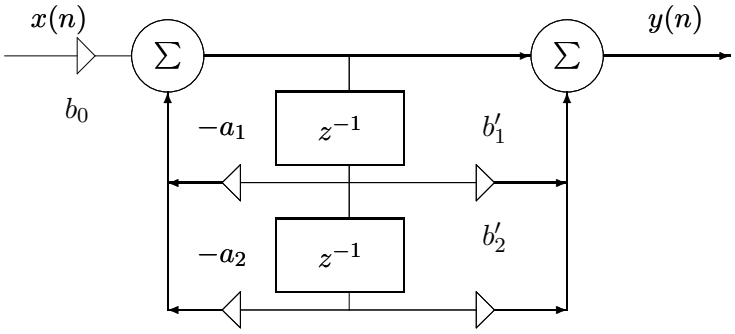


Figura 5.9: Sección bi-cuad alternativa.

5.4.3. Conexión de estructuras en cascada

La $H(z)$ total puede ser factorizada como

$$H(z) = CH_1(z)H_2(z)\cdots H_i(z). \tag{5.49}$$

Y en términos de secciones de segundo orden,

$$H(z) = \prod_{i=1}^{N/2} \frac{b_{0_i} + b_{1_i}z^{-1} + b_{2_i}z^{-2}}{1 + a_{1_i}z^{-1} + a_{2_i}z^{-2}}, \tag{5.50}$$

donde el subíndice i representa el número de la sección bi-cuad.

En esta conexión en cascada la salida de una sección es la entrada de la siguiente sección, y el código necesario en su implantación toma en cuenta este hecho al calcular la salida una sección para emplearla como entrada para la siguiente.

5.4.4. Conexión de estructuras en paralelo

Para la conexión en paralelo, $H(z)$ se representa por

$$H(z) = C + H_1(z) + H_2(z) + \cdots + H_i(z), \tag{5.51}$$

y en términos de secciones de segundo orden,

$$H(z) = C + \sum_{i=1}^{N/2} \frac{b_{0_i} + b_{1_i}z^{-1} + b_{2_i}z^{-2}}{1 + a_{1_i}z^{-1} + a_{2_i}z^{-2}}. \tag{5.52}$$

Así, la salida se obtiene como

$$y(n) = Cx(n) + \sum_{i=1}^{N/2} y_i(n), \tag{5.53}$$

donde $y_i(n)$ representa la salida de la i-ésima sección bi-cuad.

5.5. El triángulo de estabilidad

Antes de introducir la implantación de filtros digitales RII, en los procesadores de señales de la familia DSP56k, se analizan las propiedades de los coeficientes de las estructuras bi-cuad.

El numerador y el denominador de las secciones bi-cuad tienen la forma:

$$1 + C_1z^{-1} + C_2z^{-2} = (1 - q_1z^{-1})(1 - q_2z^{-1}), \tag{5.54}$$

donde las raíces q_1 y q_2 son los polos y ceros de $H(z)$ en el plano-z, dadas por

$$q_{1,2} = \frac{-C_1 \pm \sqrt{C_1^2 - 4C_2}}{2}. \tag{5.55}$$

Las raíces pueden ser reales o complejas, pero en cualquier caso

$$\begin{aligned} C_1 &= -(q_1 + q_2), \text{ y} \\ C_2 &= q_1 q_2. \end{aligned}$$

Si $C_1^2 \geq 4C_2$, las raíces son reales, mientras que si $C_1^2 < 4C_2$, las raíces son complejas conjugadas.

Específicamente para el denominador podemos obtener condiciones sobre los coeficientes a_1 y a_2 de modo que se asegure la estabilidad. Sea el polinomio del denominador dado por

$$D(z) = 1 + a_1 z^{-1} + a_2 z^{-1} = (1 - p_1 z^{-1})(1 - p_2 z^{-1}), \tag{5.56}$$

donde p_1 y p_2, para asegurar la estabilidad, deben caer dentro del círculo unitario, es decir,

$$|p_1| \text{ y } |p_2| < 1.$$

Como $C_2 = q_1 q_2$, $|a_2| = |p_1 p_2| < 1$ y como $C_1 = -(q_1 + q_2)$, entonces $|a_1| = -(p_1 + p_2)$ de donde se puede ver que $|a_1| < 1 + a_2$. Así, mientras a_2 puede variar entre -1 y +1, a_1 puede variar entre -2 y 2. Esto se ilustra en la figura 5.10.

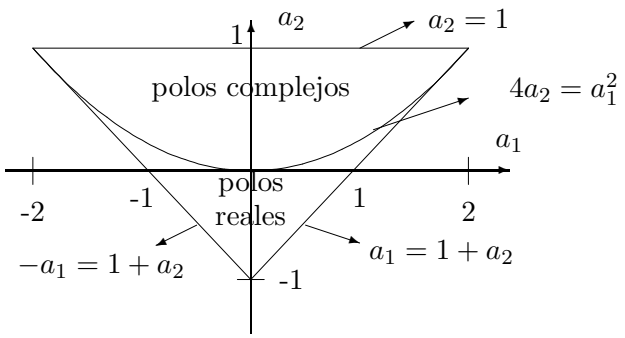


Figura 5.10: Triángulo de estabilidad.

Para una sección de segundo orden se cumplirá el requisito de estabilidad si y sólo si a_1 y a_2 definen un punto dentro del triángulo de estabilidad.

5.6. Código de una sección de segundo orden

Para el procesador de señales DSP56002, necesitaremos que tanto a_1 como a_2 (igualmente b_1 y b_2) sean menores a 1, debido a la aritmética fraccionaria empleada por este procesador.

La sección de segundo orden más general se presenta cuando a_1 y a_2 son menores a 1, entonces tenemos el siguiente juego de ecuaciones:

$$\begin{aligned} w(n) &= \frac{1}{a_0}[x(n) - a_1 w(n-1) - a_2 w(n-2)] \\ y(n) &= b_0 w(n) + b_1 w(n-1) + b_2 w(n-2) \\ w(n-2) &= w(n-1) \\ w(n-1) &= w(n) \end{aligned} \tag{5.57}$$

En caso de que $a_1 > 1$ se puede factorizar un 2 en el denominador. El núcleo del código, también llamado *kernel* en los textos extranjeros, requiere seis ciclos de instrucciones para su ejecución. Antes de entrar al núcleo de instrucciones, supondremos que se han hecho los siguientes movimientos: la muestra más reciente $x(n)$ ha sido cargada al acumulador a, el valor en $w(n-2)$ ha sido cargado en el registro de datos x0, y el coeficiente a_2 ha sido cargado en el registro de datos y0.

```
mac  -x0,y0,a  x:(r0)+,x1  y:(r4)+,y0; R1=x(n)-a_2 w(n-2)
macr -x1,y0,a  x1,x:(r0)+  y:(r4)+,y0; R2=R1-a_1 w(n-1)
asl   a                              ; R3=2 R2
mpy   x0,y0,a  a,x:(r0)    y:(r4)+,y0; R4=R3+b_2 w(n-2)
mac   x1,y0,a  x:(r0)+,x0  y:(r4)+,y0; R5=R4+b_1 w(n-1)
macr  x0,y0,a  x:(r0)+,x0  y:(r4)+,y0; R5=ecuación
```

5.6.1. Código para la forma directa II transpuesta

Una realización alternativa a la forma directa II mostrada en la figura 5.7 es la forma transpuesta. Esta forma está incluida en el paquete de diseño que se describe en la siguiente sección.

Para obtener la forma transpuesta se siguen los pasos siguientes: Considérese la sección de segundo orden mostrada en la figura 5.8.

- Invertir las direcciones de todas las ramas, e
- invertir el papel de la entrada y la salida, como se muestra en la figura 5.11.
- Volver a dibujar la estructura con la entrada a la izquierda y la salida a la derecha, para obtener la forma transpuesta que se muestra en la figura 5.12, con las siguientes ecuaciones:

$$\begin{aligned} u[n] &= b_2 x[n] - a_2 y[n] \\ v[n] &= b_1 x[n] - a_1 y[n] + u[n-1] \\ y[n] &= b_0 x[n] + v[n-1] \end{aligned} \tag{5.58}$$

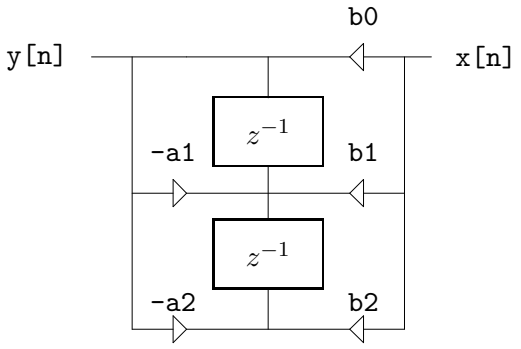


Figura 5.11: Inversión de ramas para la forma transpuesta.

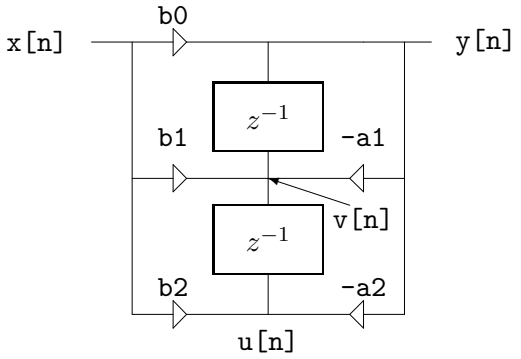


Figura 5.12: Forma directa II transpuesta.

Debe notarse que la (5.58) equivale a la ecuación en diferencias dada por

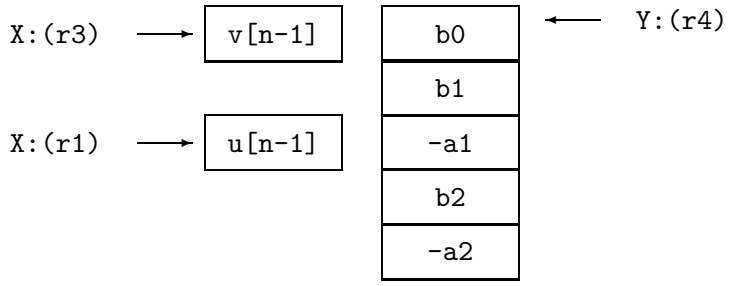
$$\begin{aligned} y[n] &= b_0x[n] + b_1x[n-1] + b_2x[n-2] \\ &= -a_1y[n-1] - a_2y[n-2] \\ &= \sum_{k=0}^{M} b_kx[n-k] - \sum_{k=1}^{N} a_ky[n-k]. \end{aligned} \tag{5.59}$$

Si suponemos que todos los coeficientes son divididos entre dos, entonces podemos escribir

$$\begin{aligned} y[n] &= 2\{b_0x[n] + \frac{1}{2}v[n-1]\} \\ v[n] &= 2\{b_1x[n] - a_1y[n] - a_2y[n-1] + \frac{1}{2}u[n-1]\} \\ u[n] &= 2\{b_2x[n] - a_2y[n]\}. \end{aligned} \tag{5.60}$$

A continuación se muestra la construcción del código en ensamblador, con las siguientes suposiciones: antes de entrar al lazo `do`, el registro `y1`

contiene el dato de entrada `x[n]`, el registro `y0` contiene el valor del coeficiente `b0`, el acumulador `a` es igual a `1/2 v[n-1]`, y el registro `r4` apunta al coeficiente `b1`. El arreglo de memoria será el siguiente:



Con estos preparativos el código es el siguiente:

```
   do    #nsec,_fin
   macr  y0,y1,a   x:(r3),b   y:(r4)+,y0 ;a=y[n], b=u[n-1]
                                         ;y0=b1/2, r4->-a1
   asr   b         a,x0                  ;b=1/2 u[n-1], x0=y[n]
   mac   y0,y1,b             y:(r4)+,y0 ;b=(b1 x[n]+1/2 u[n-1])
                                         ;y0=-a1, r4 ->b2
   macr  y0,x0,b             y:(r4)+,y0 ;b=v[n], y0=b2, r4->-a2
   mpy   y0,y1,b   b,x:(r1)+ y:(r4)+,y0 ;b=b2 x[n], guarda v[n]
                                         ;y0=-a2, r4-> b0
   macr  y0,x0,b   x:(r1),a  a,y1       ;b=u[n], prepara a=v[n-1]
   asr   a         b,x:(r3)+ y:(r4)+,y0 ;a=1/2 v[n-1],u[n]->u[n-1]
                                         ;y0=b0, r4->b1
                                         ;y1=y[n] salida
_fin
```

donde `#nsec` es el número de secciones de segundo orden del filtro, y `_fin` es la etiqueta que indica el final del código hasta donde se ejecuta la instrucción `do`.

5.7. Descripción de FDI

FDI es una colección de archivos M para el ambiente de programación MATLAB. En el desarrollo de esta herramienta se ha hecho uso de las capacidades gráficas de este poderoso ambiente de trabajo. Al hacer uso de la interfase gráfica para usuario (GUI), se ha obtenido un ambiente de diseño e implantación de filtros digitales muy amigable.

Se tienen dos versiones de FDI, una para el DSP56002EVM y una segunda versión para el DSP56303EVM. En las siguientes figuras se muestra paso a paso el empleo de FDI. Para comenzar, se debe crear una carpeta en el directorio raíz, `C:\FDI2k2`, o `C:\FDI563x`. Dentro de esta carpeta se crea

otra carpeta: `transposeM`, la cual contiene los archivos M que deberán ejecutarse desde MATLAB. En la primera se copian los archivos de los programas ensamblador, otros programas ejecutables y *macros*.

Dentro del espacio de trabajo de MATLAB se da inicio a la herramienta FDI escribiendo `mainfdi`. Después de la pantalla de presentación aparecerá una pantalla con el menú principal, como se muestra en la figura 5.13. Se tiene la opción para el diseño de filtros RII o FIR, la cual se puede seleccionar desde el menú principal.

Figura 5.13: Presentación del menú principal.

5.7.1. Opción diseñar RII

Una vez seleccionada la opción RII (ver figura 5.14), aparece el menú, para elegir el tipo de filtro a diseñar, es decir, pasa-bajas (Lowpass), pasa-altas (Highpass) o pasa-banda (Bandpass). Al seleccionar el tipo de filtro, pulsando con el ratón el botón apropiado del menú, aparece enseguida el menú de selección de la función de aproximación a usar: Butterworth, Chebyshev, o Elíptico, como se muestra en la figura 5.15.

Figura 5.14: Presentación del menú tipo de filtro.

Las especificaciones de diseño se dan cuando el programa regresa al ambiente de trabajo de MATLAB, solicitando el ingreso de la frecuencia de corte, la frecuencia de rechazo, y la atenuación deseada, en dB, para cada una. En la figura 5.16 se muestra el ingreso de los datos para un filtro pasa-bajas.

Figura 5.15: Presentación del menú tipo de aproximación.

Una vez ingresados los datos se puede obtener la gráfica de la respuesta en

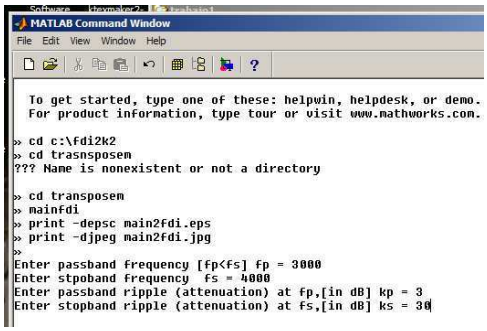


Figura 5.16: Entrada de especificaciones del filtro deseado.

frecuencia, y para el caso de filtros RII se obtiene, además, el diagrama de ceros y polos en el plano-z. La figura 5.17 muestra el trazo de la respuesta en frecuencia, y además se pueden ver las otras opciones que aparecen en ese momento, `Write ASM` para escribir el archivo con los coeficientes del filtro en lenguaje ensamblador, y `Assemble` para llamar al ensamblador, y finalmente `Back to main` para regresar al menú principal.

Una última opción, antes de terminar con el diseño del filtro y la generación del código ensamblador, es la selección de la tasa de muestreo que empleará el procesador de señales. Para esto aparece un menú como el que se muestra en la figura 5.18.

Cabe hacer notar que sólo la versión de FDI para el DSP56002EVM permite seleccionar diferentes tasas de muestreo; para el DSP56303EVM se ha dejado una tasa de muestreo fija de 48 kHz.

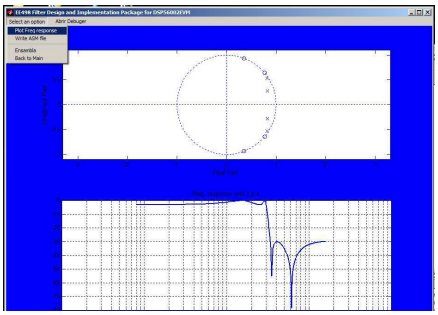


Figura 5.17: Respuesta en frecuencia del filtro diseñado.

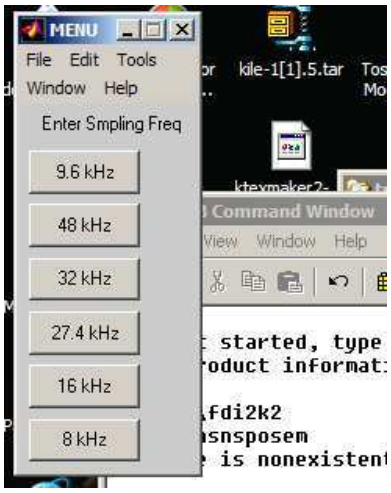


Figura 5.18: Menú para la selección de la frecuencia de muestreo.

Una vez que se ha obtenido el diseño del filtro, es necesario generar el código y ejecutar el programa ensamblador para obtener el archivo `.cld`. Para la generación automática de código se emplean los archivos provistos por Motorola, los cuales con modificaciones mínimas han sido construidos como *macros* que son llamados desde el archivo que contiene los coeficientes del filtro. A continuación se muestra el código resultante para el DSP56303EVM del diseño de un filtro pasa-altas elíptico cuya frecuencia de corte es de 4 kHz. Puede notarse que el filtro se compone de dos secciones bicuads, definidas en la constante `nsec equ 2`.

```
;*******************************************
;*   Digital Signal Processing             *
;*  COEF_file for IIR filters on the DSP56303 *
;*.........................................*
;*         Gerardo Miramontes de Leon      *
;*      Universidad Autonoma de Zacatecas  *
;*                                         *
```

```
;*           gmiram@cantera.reduaz.mx         *
;********************************************
;         Filter specifications:
;         iirHPtr.ASM
;         Elliptic
;    Sampling Frequency  48000.00  [Hz]
;    Passband Frequency  4000.00  [Hz]
;********************************************
nsec   equ      2
 org x:$400
statesw1left ds  nsec
statesw2left ds  nsec
 org x:$400+2*nsec
statesw1right    ds  nsec
statesw2right    ds  nsec
 org x:$400+4*nsec+1
templ    ds  1
tempr    ds  1
      org  Y:$400
coefs
                        ;=== Biquad 1 ===
     dc   0.475399/2    ;  b(1,0)/2
     dc   -0.475399/2   ;  b(1,1)/2
     dc   0.093364/2    ; -a(1,1)/2
     dc   0.000000/2    ;  b(1,2)/2
     dc   0.000000/2    ; -a(1,2)/2
                        ;=== Biquad 2 ===
     dc   1.000000/2    ;  b(2,0)/2
     dc   -2.000000/2   ;  b(2,1)/2
     dc   1.572226/2    ; -a(2,1)/2
     dc   1.000000/2    ;  b(2,2)/2
     dc   -0.857434/2   ; -a(2,2)/2
    INCLUDE  'mactrans.asm'

 mactrans nsec,coefs,statesw1left,statesw2left,
          statesw1right,statesw2right
```

En la figura 5.19 se muestra la pantalla resultante una vez que se ha seleccionado el botón para ensamblar.

Como se dijo antes, en la construcción del *macro* se han empleado y modificado algunos archivos dados por Freescale Semiconductor, Inc., como `ioequ.asm`, `intequ.asm` `ada_equ.asm` y `vectors.asm`. El *macro* es el archivo que contiene tanto la programación del codec como la implantación del filtro, empleando una estructura RII transpuesta. A continuación se muestra, en varias partes, el código del *macro* `mactrans.asm`. En la primera parte se muestra el inicio del *macro* donde se realiza la inicialización del DSP56303 y los ajustes del codec:

```
mactrans macro nsec,coefs,statesw1left,statesw2left,
         statesw1right,statesw2right
```

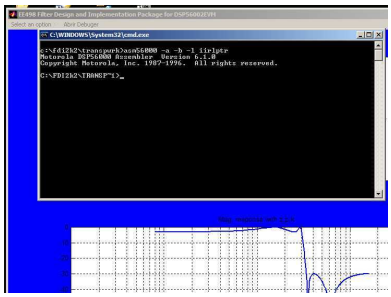


Figura 5.19: Ejecución del programa ensamblador ASM56000 (o ASM56300).

```
;*************************************************
      nolist
      include 'ioequ.asm'
      include 'intequ.asm'
      include 'ada_equ.asm'
      include 'vectors.asm'
    list
;*************************************************
;---Buffer for talking to the CS4218

       org    x:$0
RX_BUFF_BASE    equ     *
RX_data_1_2 ds  1   ; data time slot 1/2
                    ; for RX ISR (left audio)
RX_data_3_4 ds  1   ; data time slot 3/4
                    ; for RX ISR (right audio)
TX_BUFF_BASE    equ *
TX_data_1_2 ds  1   ; data time slot 1/2
                    ; for TX ISR (left audio)
TX_data_3_4 ds  1   ; data time slot 3/4
                    ; for TX ISR (right audio)
RX_PTR      ds   1   ; Pointer for rx buffer
TX_PTR      ds   1   ; Pointer for tx buffer
CTRL_WD_12  equ  MIN_LEFT_ATTN+MIN_RIGHT_ATTN+LIN2+RIN2
CTRL_WD_34  equ  MIN_LEFT_GAIN+MIN_RIGHT_GAIN

        org   p:$100
START
main
    movep   #$040006,x:M_PCTL  ;PLL 7 X 12.288 = 86.016MHz
    ori     #3,mr              ;mask interrupts
    movec   #0,sp              ;clear hardware stack pointer
    move    #0,omr             ;operating mode 0
    move    #$40,r7            ;initialize stack pointer
    move    #-1,m7             ;linear addressing
    jsr     ada_init           ;initialize codec
```

```
; Initialize Filter Parameters
    move    #statesw1left,r1   ;point to filter state1
    move    #statesw2left,r3   ;point to filter state2
    move    #coefs,r4
    move    #5*nsec-1,m4       ;addressing modulo 5*nsec
    clr     a                  ;initialize internal
                               ;state storage
    rep     #nsec
    move    a,x:(r3)+
    rep     #nsec
    move    a,x:(r1)+
    move    #statesw1right,r1  ;point to filter state1
    move    #statesw2right,r3  ;point to filter state2
    rep     #nsec
    move    a,x:(r3)+
    rep     #nsec
    move    a,x:(r1)+
loop
    jset    #3,x:M_SSISR0,*    ;wait for rx frame sync
    jclr    #3,x:M_SSISR0,*    ;wait for rx frame sync
    clr a
    clr b
    move    x:RX_BUFF_BASE,a    ;receive left
    move    x:RX_BUFF_BASE+1,b  ;receive right
    jsr     save_left
    jsr     save_right

    jsr     process_stereo

    jsr     get_left_In_a
    jsr     get_right_In_b
    move    a,x:TX_BUFF_BASE    ;transmit left
    move    b,x:TX_BUFF_BASE+1  ;transmit right
    jmp     loop

    include 'ada_init.asm'    ; used to include codec
                              ; initialization routines
```

Como puede observarse, dentro del lazo (`loop`) se hace el llamado a varias subrutinas: `process_stereo`, la cual llama a `left_ch` y a `save_left`, y luego a `right_ch` y `save_right`. A continuación se muestra la subrutina `process_stereo` en la cual se carga en dos ocasiones al apuntador `r4` con la dirección `#coeffs` ya que la operación de filtrado se aplica primero al canal izquierdo y después al canal derecho:

```
process_stereo
        jsr     get_left_In_a
        move    #coefs,r4
        move    #statesw1left,r1
```

```
        move    #statesw2left,r3

        jsr     left_ch
        jsr     save_left
        jsr     get_right_In_b

        move    #coefs,r4
        move    #statesw1right,r1
        move    #statesw2right,r3
        jsr     right_ch
        jsr     save_right
        rts
```

El las subrutinas `left_ch` y `right_ch` se hace el llamado a la rutina de filtrado, es decir, en estas subrutinas se preparan los apuntadores y los registros apropiados para el filtro. Note que cada subrutina termina con la instrucción `rts` (*return from subroutine*). En el siguiente listado se muestran también las subrutinas `save_left`, `save_right`, entre otras:

```
left_ch                             ;
        ori     #$08,mr  ;set scaling mode
        move    a,y1     ;load left signal
        move    x:(r1),a y:(r4)+,y0
        asr     a
        jsr     filtro   ; Call cascade biquad routine.
        move    y1,a
        andi    #$f7,mr  ;disable scaling mode?
        rts
right_ch
        ori     #$08,mr  ;set scaling mode
        move    x:(r1),a   y:(r4)+,y0
        asr     a
        move    b,y1     ;load left signal

        jsr    filtro    ; Call cascade biquad routine.

        move   y1,b
        andi    #$f7,mr  ;disable scaling mode?
        rts
save_right
        move  #tempr,r1
        nop
        move  b,x:(r1)   ;save temporally right channel
        rts
save_left
        move    #templ,r1
        nop
        move     a,x:(r1) ;save temporally left channel
        rts
get_left_In_a
```

```
        move   #templ,r1
        nop
        move   x:(r1),a   ;get temporally right channel
       rts
get_right_In_b
        move   #tempr,r1
        nop
        move    x:(r1),b  ;get temporally right channel
        rts
```

Finalmente se muestra el código que realiza la operación de filtrado incluyendo, en el campo de los comentarios, una breve descripción del proceso:

```
filtro
   do    #nsec,_end_filter
   macr  y0,y1,a  x:(r3),b  y:(r4)+,y0 ;a=x(n)*bi0/2+Vi/2,
                                       ;b=Ui,y0=bi1/2
   asr   b        a,x0                 ;b=Ui/2,x0=y(n)
   mac   y0,y1,b            y:(r4)+,y0 ;b=x(n)*bi1/2+Ui/2,
                                       ;y0=ai1/2
   macr  x0,y0,b            y:(r4)+,y0 ;b=b+y(n)*ai1/2,
                                       ;y0=bi2/2
   mpy   y0,y1,b  b,x:(r1)+ y:(r4)+,y0 ;b=x(n)*bi2/2,save
                                       ;Vi,y0=ai2
   macr  x0,y0,b  x:(r1),a  a,y1       ;b=b+y(n)*ai2/2,
                                       ;a=next iter wi1,
   asr   a        b,x:(r3)+ y:(r4)+,y0 ;y1=output of
                                       ;section i
                                       ;a=next iter wi1,save
                                       ;wi2,y0=next iter bi0
_end_filter
        rts                            ;Return from
                                       ;filter routine.
```

Para aplicar el filtro en tiempo real es necesario cargarlo al módulo que contiene al procesador de señales haciendo uso del programa de comunicación con él. Para esto se tiene la opción `Abrir Debugger` con la cual se llama al programa EVM56kw (o EVM30xW) de Domain Technologies, cuya ventana aparece en la figura 5.20.

Bibliografía

[1] Antoniou, A., *Digital Filters Analysis and Design,* McGraw-Hill, 1979.

[2] El-Sharkawy, M., *Digital Signal Processing Applications with the Motorola's DSP56002 Processor*, Prentice Hall PTR, 1996.

[3] Gold, B., Rader, C. M., *Digital Processing of Signals*, McGraw-Hill, 1969.

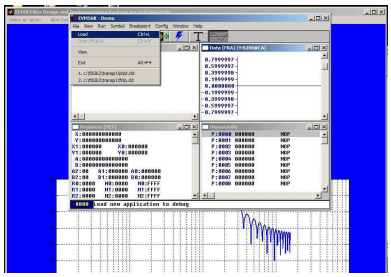


Figura 5.20: Ventana de la aplicación EVM56kw.

[4] Hamming, R. W., *Digital Filters*, Prentice-Hall, 1977.

[5] Jackson, L. B., *Digital Filters and Signal Processing*, 2nd Ed., Kluwer,1992.

[6] Mitra, S. K., *Digital Signal Processing: a computer based approach*, McGraw-Hill, 1998.

[7] Freescale Semiconductor, Inc., *DSP56000/56001 Digital Signal Processor User's Manual,*
DSP56000UM/AD Rev 2. Motorola Inc. 1990.

[8] Freescale Semiconductor Inc., *DSP56002 Digital Signal Processor User's Manual,*
DSP56002UM/AD Rev 1. Motorola Inc. 1993.

[9] Stanley, W. D., *Digital Signal Processing*, Reston Publishing Co., 1975.

Problemas

Problema 5.1 Un filtro Butterworth analógico normalizado de primer orden tiene una función de transferencia

$$H(s) = \frac{1}{s+1}$$

a) Aplique la transformación bilineal para obtener H(z).
b) ¿Cuál es la frecuencia de corte de $H(z)$?
c) Usando la $H(z)$ obtenida en a), aplique una transformación digital-a-digital para obtener un filtro digital Butterworth de primer order normalizado.

Problema 5.2 Una función de transferencia digital se puede obtener aplicando la transformación

$$s \longrightarrow \frac{1-z^{-1}}{T},$$

o equivalentemente

$$z = \frac{1}{1 - sT}.$$

Demuestre que el eje $j\Omega$ del plano-s se transforma a un círculo de radio 1/2 en el plano-z, como se muestra en la figura 5.2.

Problema 5.3 Trabajando sólo en el dominio digital, encuentre el orden y muestre los pasos necesarios para obtener la función de transferencia $H(z)$ a partir de las siguientes especificaciones: repuesta en frecuencia máximamente plana con -3 dB a una frecuencia de corte de 15 rad/s, y al menos 25 dB de atenuación a frecuencias mayores que 30 rad/s.

Problema 5.4 Encuentre $H(z)$, aplicando la transformación bilineal, si las especificaciones son las siguientes: frecuencia de corte de 100 Hz con atenuación de 3 dB, frecuencia de rechazo de 300 Hz con al menos 25 dB de atenuación y una frecuencia de muestreo de 1000 Hz.

Problema 5.5 Diseñe un filtro digital por el método de la transformada bilineal que cumpla con las especificaciones dadas en la figura 5.21.

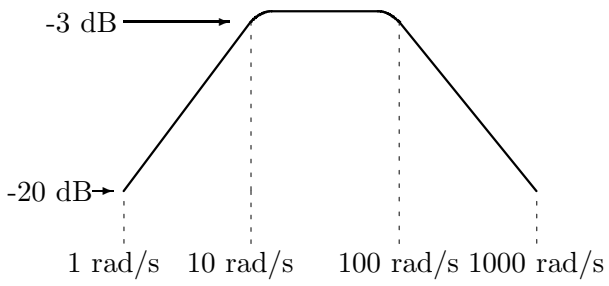


Figura 5.21: Respuesta deseada del filtro para el problema 5.5.

Problema 5.6 Diseñe un filtro digital Chebyshev tipo 1, empleando la transformación bilineal, de modo que cumpla con -3 dB a 20 Hz, y -30 dB a -120 Hz. Proponga la Fs adecuada.

Problema 5.7 Diseñe un filtro digital Butterworth, empleando la transformación bilineal, de modo que cumpla con -3 dB a 20 Hz, y -30 dB a 120 Hz. Proponga la Fs adecuada.

Problema 5.8 Se tiene un filtro analógico Butterworth normalizado de segundo orden, es decir,

$$H_a(s) = \frac{1}{s^2 + \sqrt{2}s + 1}.$$

Realizar las transformaciones necesarias para obtener un filtro digital que cumpla con la especificación de una frecuencia de corte de 100 Hz. Suponga una Fs=1000.

Problema 5.9 Diseñe un filtro digital pasa-bajas Chebyshev tipo 1, por la transformación bilineal, con las siguientes especificaciones:
a) Frecuencia de corte de 1000 Hz, con -2 dB de rizo.
b) Frecuencia de rechazo de 5kHz con -30 dB de atenuación.
c) Suponga Fs=15 kHz.

Problema 5.10 Partiendo de un filtro Butterworth digital de primer orden normalizado, obtenga un filtro Butterworth digital con frecuencia de corte de 100 Hz. Suponga Fs = 1000 Hz.

Problema 5.11 La función de transferencia de un filtro digital Butterworth normalizado está dada por

$$H(z) = \frac{0.353295(1 + z^{-1})}{1 - 0.293408z^{-1}}.$$

Aplique las transformaciones necesarias para obtener una nueva función de transferencia $H'(z)$ que tenga una frecuencia de corte ω=2 rad/s con una respuesta pasa-bajas.

Problema 5.12 Se tiene un filtro analógico Butterworth normalizado de primer orden, es decir,

$$H_a(s) = \frac{1}{s + 1}$$

Su respuesta al impulso es muestreada con una frecuencia de muestreo suficientemente alta para evitar el traslape, es decir F_s=100 Hz.

Obtenga $H(z)$ por el método de la invariancia al impulso (sin corrección). NOTA: $h[n] = Th_a(nT)$.