Filtros IIR

May 29, 2024

1 Introducción

Los filtros IIR (respuesta infinita al impulso) cuentan con una parte recursiva y una no recursiva.

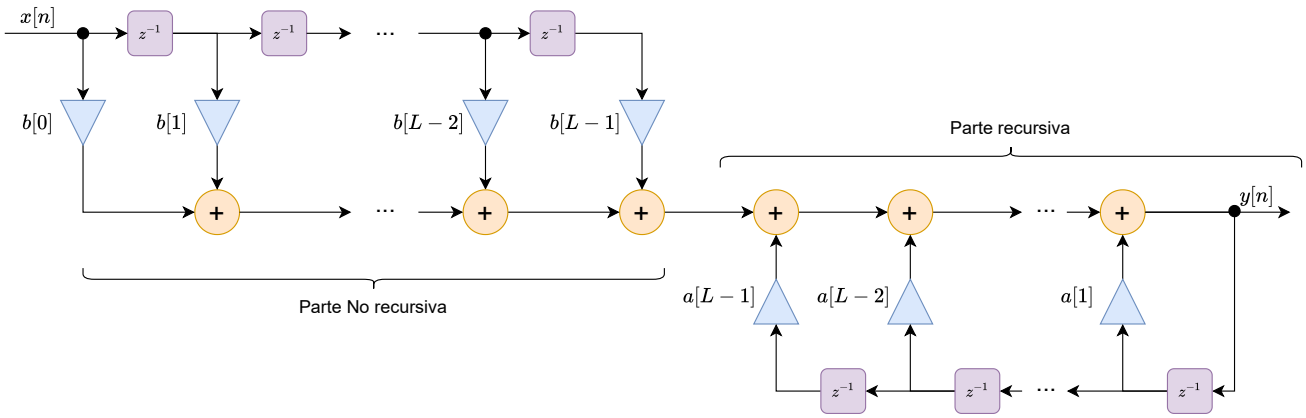

Figure 1: Filtro con Feedback

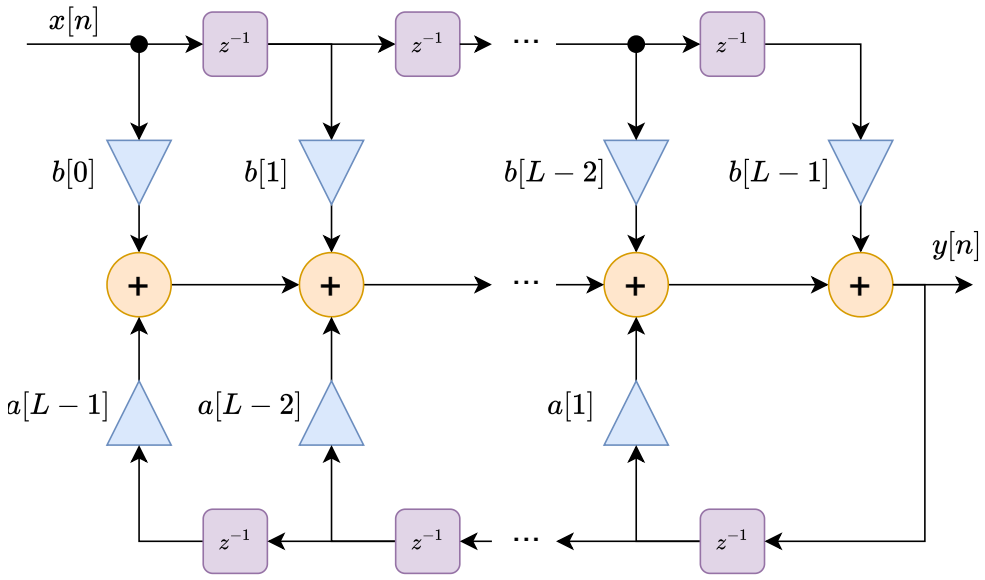

Figure 2: Filtro resultante de juntar las partes recursivas y no recursivas.

Ecuación en diferencias:

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

Función transferencia:

$$H(z) = \frac{\sum_{k=0}^{M} b_k z^{-k}}{1 + \sum_{k=1}^{N} a_k z^{-k}} \tag{2}$$

1.1 Lossy Integrator

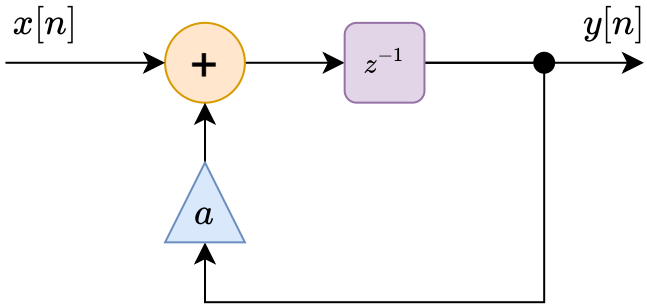

Figure 3: IIR de primer orden - Lossy Integrator

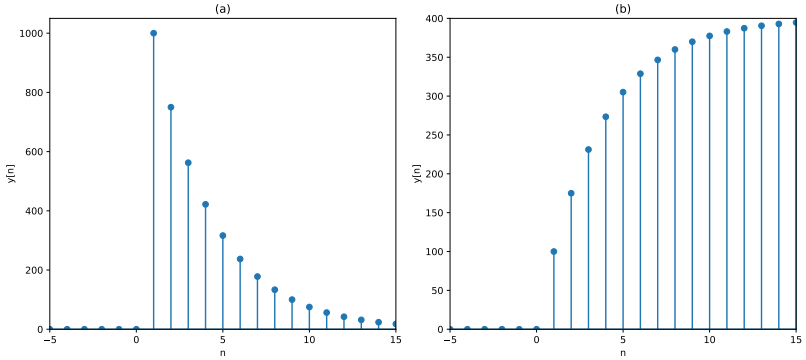

Figure 4: Simulación de un lossy integrator con $a = 3/4$. (a) Respuesta al impulso para $x[n] = 1000\delta[n]$. (b) Respuesta al escalón para $x[n] = 100\sigma[n]$

```
module iir #(parameter W = 14) (
    input  clk,                  // clock
    input  reset,                // async reset
    input  signed [W:0] x_in,    // data input
    output signed [W:0] y_out    // data output
);
    reg signed [W:0] x, y;
    // Usamos FFs para la entrada y la parte recursiva
    always @(posedge clk or posedge reset) begin
        if (reset) begin
            // Nota: signados
            x <= 0; y <= 0;
        end else begin
            x <= x_in;
            y <= x + (y >>> 1) + (y >>> 2);   // >>> menor uso de LEs
            // Alternativas de codigo:
            // y <= x + y / 'sd2 + y / 'sd4;  // similar al shifting
            // y <= x + y / 2 + y / 4;        // div, usa mas LEs
            // y <= x + {y[W],y[W:1]}+ {y[W],y[W],y[W:2]};
        end
    end
    assign y_out = y;   // Conectamos y al pin de salida
endmodule
```

Listing 1: IIR Filtro de orden 1 - Lossy Integrator

1.2 Técnicas de Diseño de IIR

Para diseñar los IIR y calcular sus coeficientes, se utilizan técnicas de diseño de filtros analogicos ya que:

- El diseño de filtros analógicos es un tema ampliamente desarrollado y estudiado en la literatura.
- Cuentan con procedimientos y fórmulas relativamente simples.
- Entre los distintos métodos podemos mencionar:
  - Invarianza al impulso
  - Transformación Bilineal
  - Aproximación de derivadas

Tipo	Banda de Paso	Banda de Transición	Banda de Rechazo	Orden
Butterworth	Plana (óptima)	Ancha	Plana	Alto
Chebyshev I	Equiripple	Moderada	Plana	Medio
Chebyshev II	Plana	Moderada	Equiripple	Medio
Elíptico	Equiripple	Angosta	Equiripple	Bajo

2 Estructuras Básicas

Al igual que en el caso de los sistemas FIR, existen varios tipos de estructuras o realizaciones, entre las que se incluyen:

- Estructuras en forma directa
- Transpuestas
- Cascada
- Lattice

Además, los sistemas IIR disponen de una realización en paralelo. Comenzaremos describiendo dos realizaciones en su forma directa.

2.1 Estructuras en Forma Directa

La función dada por 1 que caracteriza un sistema IIR puede interpretarse como dos sistemas conectados en cascada, es decir,

$$H(z) = H_1(z)H_2(z) \tag{3}$$

donde $H_1(z)$ consta de los ceros de $H(z)$ y $H_2(z)$ consta de los polos de $H(z)$,

$$H_1(z) = \sum_{k=0}^{M} b_k z^{-k} \tag{4}$$

y

$$H_2(z) = \frac{1}{1 + \sum_{k=1}^{N} a_k z^{-k}} \tag{5}$$

Conectado el sistema de sólo polos en cascada con $H_1(Z)$, que es un sistema FIR, obtenemos la forma directa I como se ve en la siguiente figura. Esta realización requiere $M+N+1$ multiplicaciones, $M+N$ sumas y $M+N+1$ elementos de memoria.

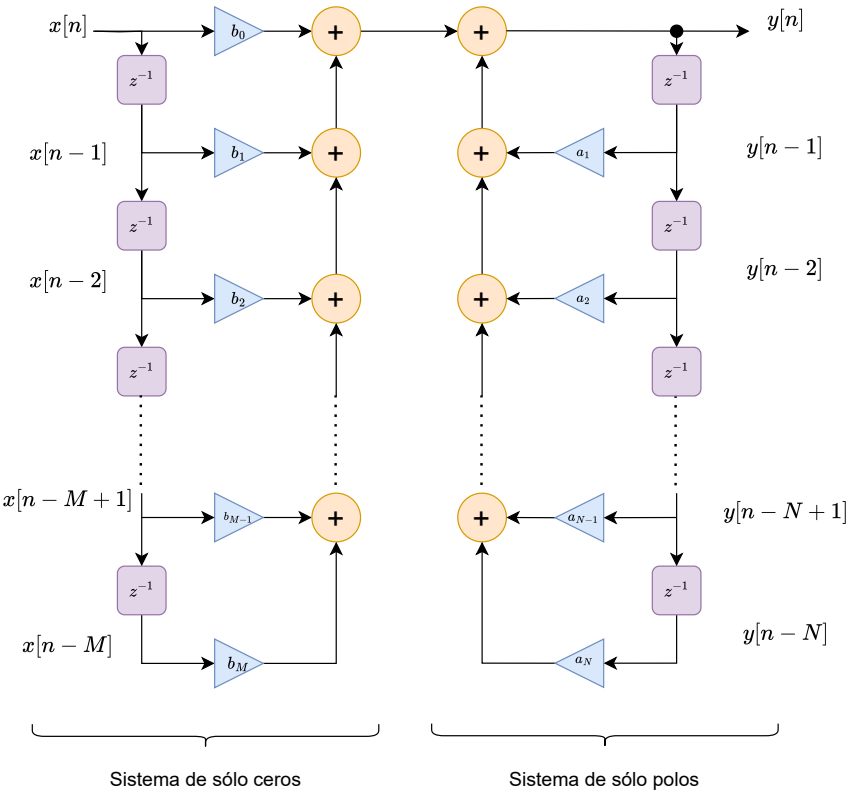

Figure 5: Estructura Forma Directa I

Si el filtro de sólo polos $H_2(z)$ se coloca antes que el filtro de sólo ceros $H_1(z)$, se consigue una estructura más compacta, llamada *forma directa II*.

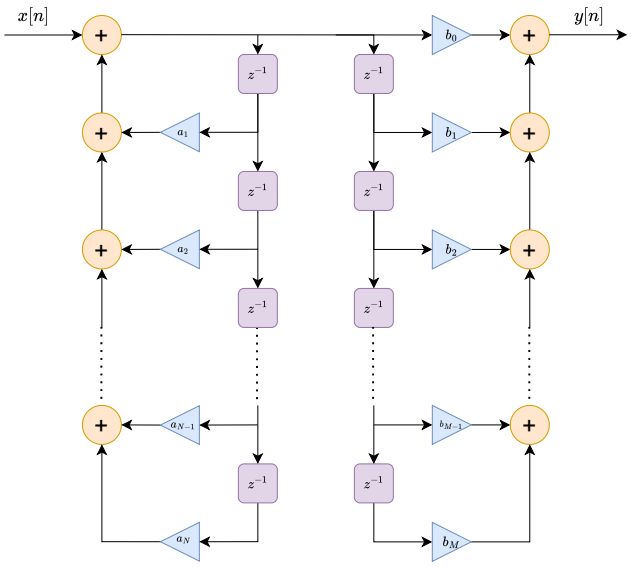

Figure 6: Reorganización de la forma directa I para llegar a la II

Sólo se necesita una única linea de delay o un único conjunto de registros para almacenar los valores pasados del sistema de solo polos dado por $H_2(z)$, por lo tanto podemos mergear estos registros, dando la forma directa II. Cómo esta forma minimiza el número de registros se dice que es *canónica*.

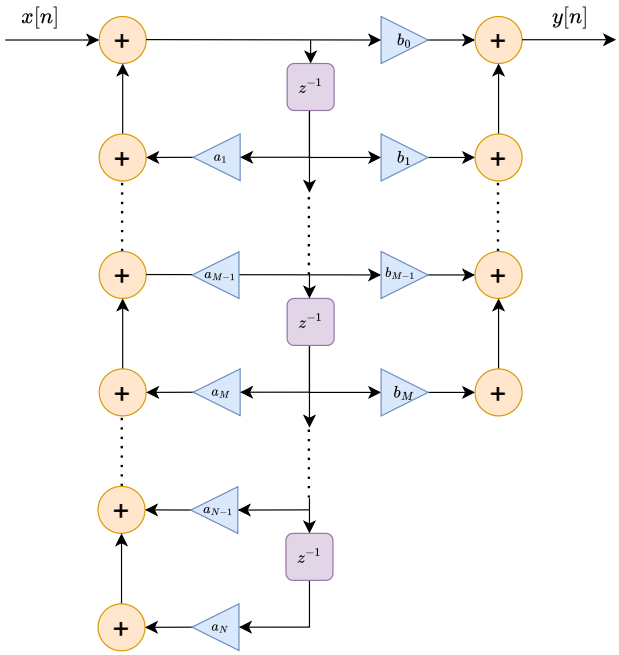

Figure 7: Forma Directa II

Una técnica que resulta útil para obtener estructuras nuevas para sistemas FIR e IIR resulta del *teorema de trasposición o teorema del diagrama de flujo inverso*. Este teorema simplemente establece que si invertimos las direcciones de todas las transmitancias de rama e intercambiamos la entrada y la salida en el diagrama de flujo, la función del sistema no varía. La estructura resultante se denomina estructura transpuesta o forma transpuesta.

- Dibujamos el grafo de flujo de señal de forma directa II.
- Intercambiamos input por output.
- Cambiamos la dirección de todos los paths.
- Intercambiamos sumadores por branching.

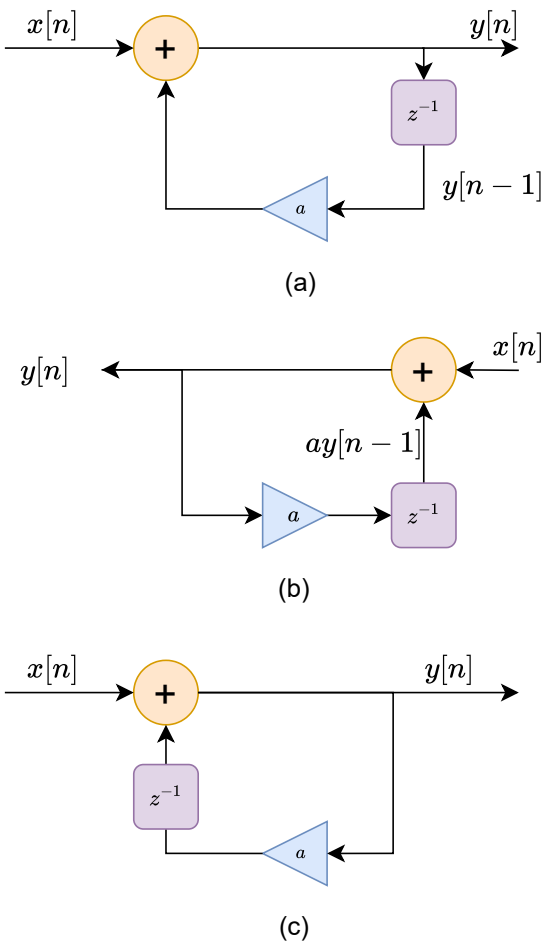

Figure 8: Formas traspuestas. (a) Aplicamos teorema de transposición. (b) Reordenamos el gráfico para obtener la transformación deseada.

2.1.1 Forma Directa I Traspuesta

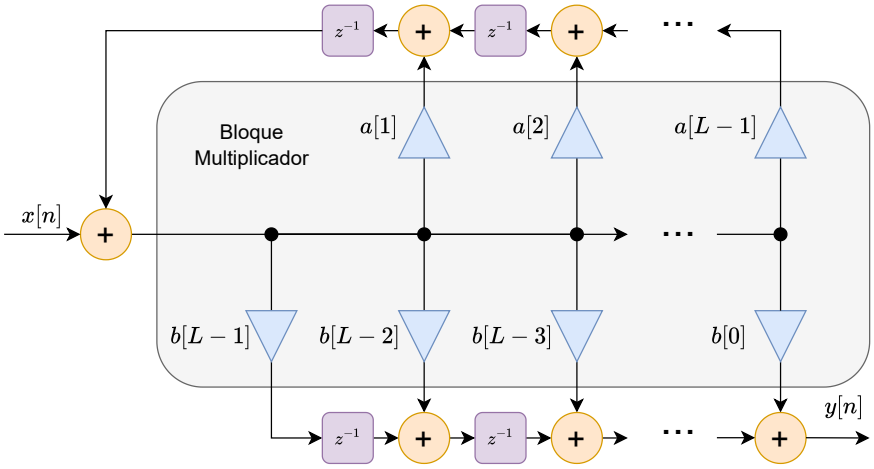

Figure 9: Forma Directa I Traspuesta

2.1.2 Forma Directa II Traspuesta

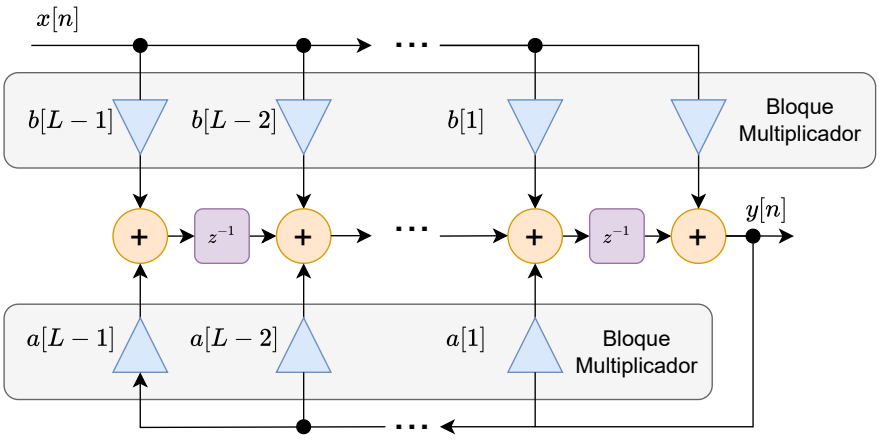

Figure 10: Forma Directa II Traspuesta

2.2 BiQuad

Un filtro IIR bicuadrático es un filtro de segundo orden. El término "biquad" proviene de "biquadrático", indicando que la función de transferencia del filtro es una razón de dos polinomios cuadráticos. Estos filtros son versátiles y pueden implementar filtros pasa-bajos, pasa-altos, pasa-banda y rechaza-banda, entre otros.

Función de Transferencia

La función de transferencia general $H(z)$ de un filtro IIR bicuadrático está dada por:

$$H(z) = \frac{b_0 + b_1 z^{-1} + b_2 z^{-2}}{1 + a_1 z^{-1} + a_2 z^{-2}}$$

Donde:

- b_0, b_1, b_2 son los coeficientes del polinomio del numerador.
- a_1, a_2 son los coeficientes del polinomio del denominador (con $a_0 = 1$).

Ecuación de Diferencias

La ecuación de diferencias correspondiente, que relaciona la señal de entrada $x[n]$ con la señal de salida $y[n]$, es:

$$y[n] = b_0 x[n] + b_1 x[n-1] + b_2 x[n-2] - a_1 y[n-1] - a_2 y[n-2]$$

Esta ecuación describe cómo la salida actual $y[n]$ es una combinación de los valores actuales y pasados de la entrada $x[n], x[n-1], x[n-2]$ y de los valores pasados de la salida $y[n-1], y[n-2]$.

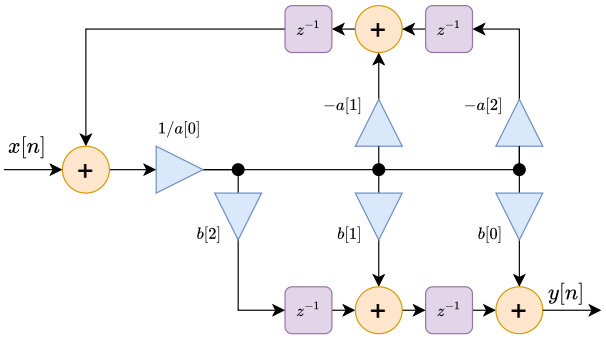

Figure 11: BiQuad Transpuesta directa I, con función de transferencia $F(z) = (b[0]+b[1]z^{-1}+b[2]z^{-2})/(a[0]+a[1]z^{-1}+a[2]z^{-2})$

Formas de Implementación

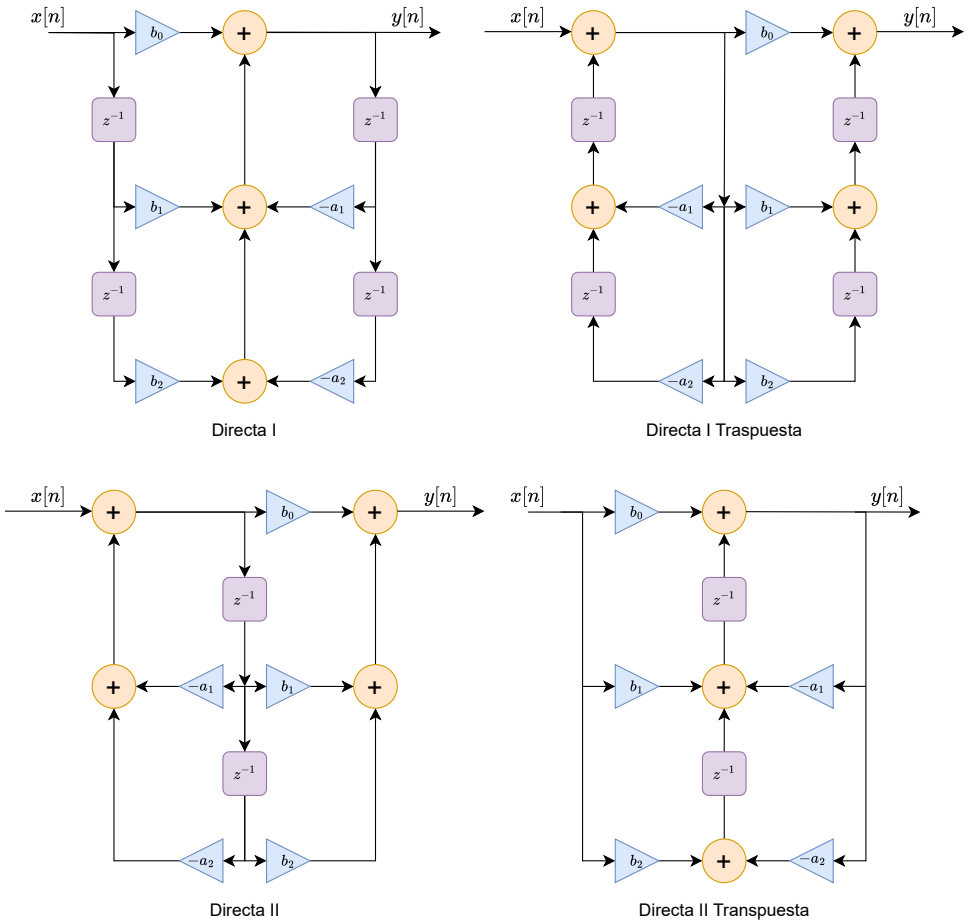

Figure 12: IIR Biquad Formas de implementación

Por qué los Filtros Bicuadráticos son Útiles

- **Flexibilidad**: Los filtros bicuadráticos pueden configurarse para realizar una variedad de tareas de filtrado (pasa-bajos, pasa-altos, pasa-banda, etc.) eligiendo adecuadamente los coeficientes b_0, b_1, b_2, a_1, a_2.
- **Estabilidad y Rendimiento**: Las secciones de segundo orden son generalmente más fáciles de estabilizar y pueden diseñarse para tener buenas características de rendimiento, como una banda pasante plana o un corte agudo.
- **Diseño Modular**: Los filtros complejos pueden construirse encadenando múltiples filtros bicuadráticos. Esta modularidad permite construir filtros de mayor orden con las especificaciones deseadas.
- **Eficiencia**: Los filtros bicuadráticos ofrecen un buen equilibrio entre la complejidad computacional y el rendimiento. Requieren menos recursos en comparación con los filtros IIR de mayor orden, proporcionando capacidades de filtrado efectivas.

2.2.1 Ejemplo: BiQuad Filter

Usando $Fs = 500KHz$, es decir, una muestra cada $2\mu s$, utilizamos los métodos clásicos para calcular los coeficientes de un filtro BiQuad de $25KHz$ de frecuencia de corte -3db

$$F(z)_{25KHz} = 0.019531250\frac{(1 + 2z^{-1} + z^{-2})}{1 - 1.5625z^{-1} + 0.640625z^{-2}} \tag{6}$$

Se puede apreciar que al utilizar $[1, 2, 1]$ como coeficientes del numerador, se reduce la complejidad del sistema, ya que solo tenemos una multiplicación $\times 2$, que se resuelve mediante shifteo. Para representar los coeficientes del denominador, usamos el sizing fixed point *fixdt(0,7,6)* y un ajuste de escala dado por $2^{-8} = 0.019531250$

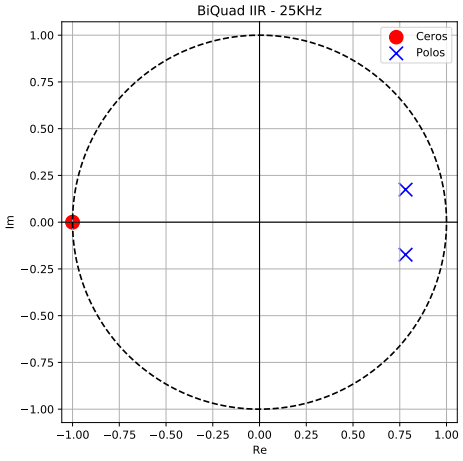

Figure 13: Diagrama polos y ceros del filtro de 25KHz

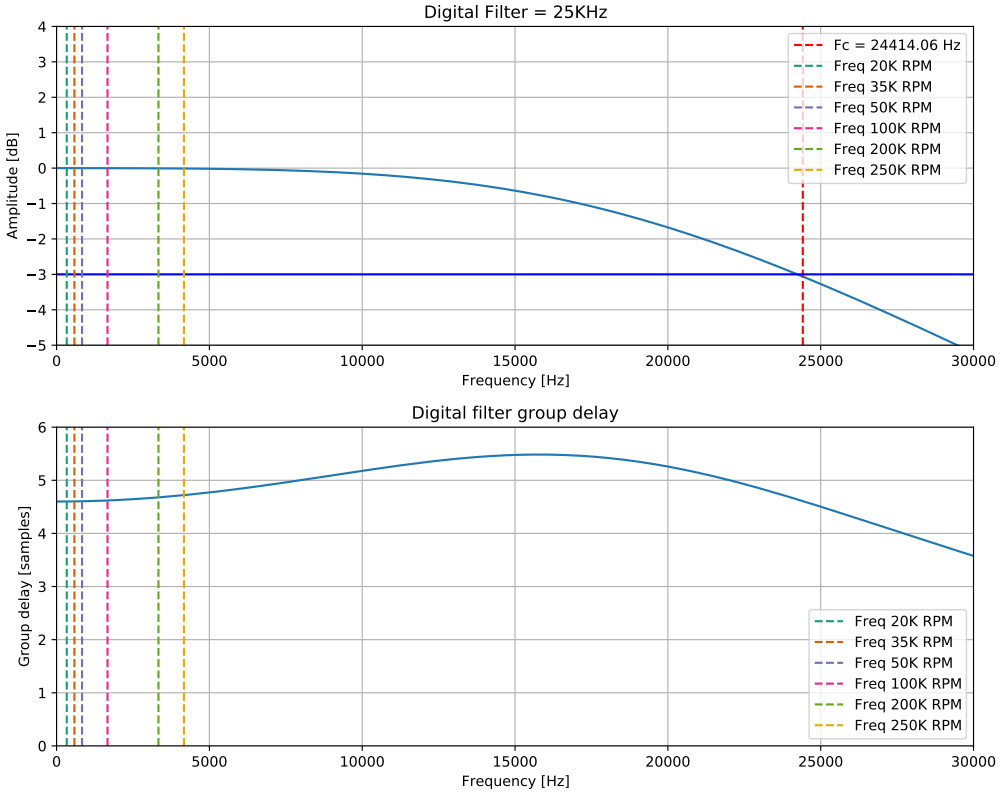

Figure 14: Bode y Group Delay

2.3 Estructura en Cascada

Descomponemos el sistema en filtros de menor orden:

$$H(z) = \prod_{k=1}^{K} H_k(z) \tag{7}$$

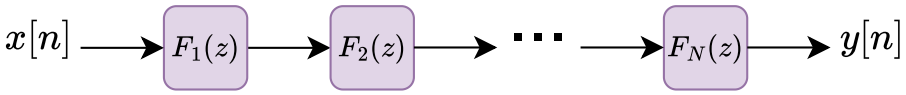

Figure 15: Estructura en Cascada

Descomposición en productos de primer y segundo orden:

$$H(z) = A\frac{\prod_{k=1}^{M_1}(1 - f_k z^{-1}) \prod_{k=1}^{M_2}(1 - g_k z^{-1})(1 - g_k^* z^{-1})}{\prod_{k=1}^{N_1}(1 - c_k z^{-1}) \prod_{k=1}^{N_2}(1 - d_k z^{-1})(1 - d_k^* z^{-1})} \tag{8}$$

Siendo: $M = M_1 + 2M_2$ y $N = N_1 + 2N_2$

Ejemplo: Agrupamiento en filtros de segundo orden:

$$H(z) = \prod_{k=1}^{N_s} \frac{b_{0k} + b_{1k} z^{-1} + b_{2k} z^{-2}}{1 - a_{1k} z^{-1} - a_{2k} z^{-2}} \tag{9}$$

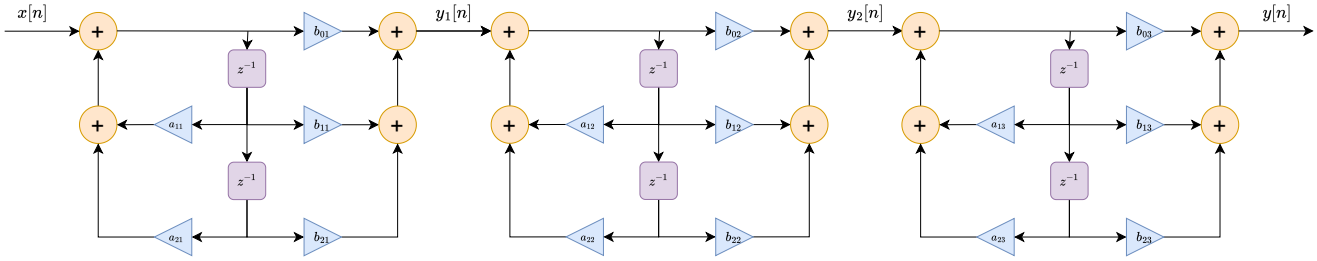

Figure 16: Cascada de filtros de forma directa II

El diseño e implementación de filtros IIR de banda estrecha utilizando cascadas de sistemas de primer y segundo orden ofrecen varias ventajas:

2.3.1 Reducción del Ruido de Cuantización

- Diseñar pares de polos/ceros en un BiQuad permite elegir la ganancia de manera que el ruido de cuantización se minimice sin que la ganancia entera se vuelva excesivamente grande.

2.3.2 Combinación y Ordenación de Secciones

- Una consideración clave es qué polos/ceros combinar en cada sección y cómo ordenarlos.
- Jackson [7] compiló una serie de reglas entre las que se destacan:
  - Los polos más cercanos al círculo unitario tienen las mayores ganancias y producen más ondulaciones.
  - Comenzar con el polo más cercano al círculo unitario combinado con el cero más cercano para reducir la ganancia y el riesgo de overflow aritmético.
  - La sección del polo más cercano al círculo unitario se coloca al final de la cascada para minimizar las ondulaciones, ya que estos son los que producen mayor ripple.
  - O dicho de otra manera, el polo mas lejano al círculo unitario, se implementa primero.
  - La funcion de Scipy tf2sos proporciona las secciones de segundo orden en el orden recomendado.

2.3.3 Ejercicio: Implementación Cascada

Realizar la implementación en cascada del sistema caracterizado por la siguiente función transferencia:

$$H[z] = \frac{2(z+2)}{z(z-0.1)(z+0.5)(z+0.4)} \tag{10}$$

Multiplicamos numerador y denominador por z^{-4} y obtenemos la forma standard:

$$\begin{aligned} H[z] &= \frac{2z^{-3}(1+2z^{-1})}{(1-0.1z^{-1})(1+0.5z^{-1})(1+0.4z^{-1})} \\ H[z] &= 2z^{-3}\frac{(1+2z^{-1})}{(1-0.1z^{-1})(1+0.5z^{-1})(1+0.4z^{-1})} \\ H[z] &= 2 \cdot H_1[z] \cdot H_2[z] \cdot H_3[z] \cdot H_4[z] \end{aligned} \tag{11}$$

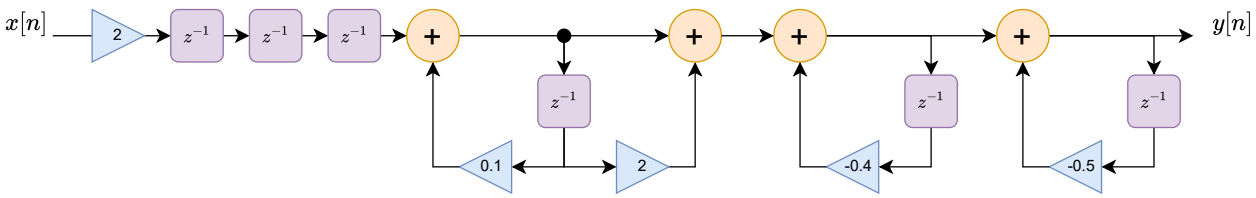

Figure 17: Implementación en Cascada

3 Cuantización

Durante el proceso de cuantización del filtro, es necesario verificar que siga cumpliendo con las especificaciones. Tenemos dos grupos de cuantizaciones:

- Aritmética
  - Ancho de palabra.
  - Escalado en diferentes etapas del filtro.
  - Truncado / Redondeo.
  - Wrapping / Saturación.
  - Estabilidad: cercanía de polos al círculo unitario.
- Coeficientes
  - Sensibilidad
  - Estabilidad

3.1 Sensibilidad frente a la cuantización de los polos

Dado un filtro IIR expresado según (2), se consideran que al cuantizar los coeficientes se obtiene [8]:

$$\begin{aligned} \overline{b_k} &= b_k + \Delta b_k \\ \overline{a_k} &= a_k + \Delta a_k \end{aligned} \tag{12}$$

siendo Δa_k y Δb_k las perturbaciones (errores) de cuantización. Por lo tanto la nueva función transferencia del filtro es:

$$\overline{H(Z)} = \frac{\sum_{i=1}^{n} \overline{b_i} z_i}{1 + \sum_{i=1}^{n} \overline{a_i} z_i} \tag{13}$$

Los polos de la trasnferencia sin cuantificar están dados por

$$D(z) = 1 + \sum_{i=1}^{n} a_i z^{-i} = \prod_{i=1}^{n} (1 - p_k z^{-1}) \tag{14}$$

y tras la cuantización:

$$\begin{aligned} \overline{D(z)} &= \prod_{i=1}^{n} (1 - \overline{p_k} z^{-1}) \\ \overline{p_k} &= p_k + \Delta p_k \end{aligned} \tag{15}$$

Para relacionar Δp_k con Δa_k, se expresa que la variación total experimentada por un polo es igual a la suma de las variaciones de dicho polo respecto de cada uno de los coeficientes del filtro:

$$\Delta p_i = \sum_{k=1}^{n} \frac{\partial p_i}{\partial a_k} \Delta a_k \tag{16}$$

Expresión que se puede calcular a partir de $D(z)$ aplicando la regla de la cadena:

$$\begin{aligned} \frac{\partial D(z)}{\partial a_k}\Big|_{z=p_i} &= \frac{\partial D(z)}{\partial z}\Big|_{z=p_i} \frac{\partial z}{\partial a_z}\Big|_{z=p_i} \Rightarrow \\ \frac{\partial p_i}{\partial a_k} &= \frac{\frac{\partial D(z)}{\partial a_k}\Big|_{z=p_i}}{\frac{\partial D(z)}{\partial z}\Big|_{z=p_i}} \end{aligned} \tag{17}$$

Calculándo el numerador y el denominador de esta última expresión y operando sobre los resultados obtenemos la expresión buscada

$$\Delta p_i = \sum_{k=1}^{n} \frac{p_i^{N-k}}{\prod\limits_{\substack{j=1 \\ j \neq i}}^{N} p_i - p_j} \Delta a_k \tag{18}$$

Esta expresión proporciona una medida de la sensibilidad del polo i-ésimo a cambios en el coeficiente a_k. A partir de esta expresión se pueden obtener las siguientes conclusiones:

- Como $|p_i| < 1 \Rightarrow |p_i|^N << |p_i|^0$ se observa que a_N es el coeficiente más sensible y a_0 es el coeficiente menos sensible.
- Cuanto mayor sea el orden más será la sensibilidad.
- La sensibilidad aumenta si los polos están muy cercanos, ya que el término $(p_i - p_j)$ es la distancia entre el polo considerado y los restantes. Se utilizan descomposiciones en cascada para disminuir el orden. Idealmente de orden 1, pero para evitar aritmética compleja en la práctica se suele agrupar en etapas de orden 2, además los polos complejos conjugados están suficientemente alejados entre sí, lo que disminuye la sensibilidad.

Incluso en el caso de una sección de un filtro de dos polos, la estructura utilizada juega un papel crucial en los errores causados por la cuantificación de los coeficientes. Específicamente, consideremos un filtro de dos polos con una función de sistema:

$$H(z) = \frac{1}{1 - (2r\cos\theta)z^{-1} + r^2 z^{-2}} \tag{19}$$

- Polos en $z = re^{\pm j\theta}$
- Coeficientes, $a_1 = 2r\cos\theta$ y $a_2 = -r^2$.
- Con precisión infinita, es posible obtener un número infinito de posiciones de polos
- Con precisión finita (es decir, con coeficientes cuantificados a_1 y a_2), las posiciones de los polos son limitadas
  - Usando b bits para representar los módulos de a_1 y a_2, existen como máximo $(2^b - 1)^2$ posiciones posibles para los polos en cada cuadrante, excluyendo el caso $a_1 = 0$ y $a_2 = 0$.

Por ejemplo, si $b = 4$, hay 15 valores distintos de cero para a_1 y 15 valores para r^2, resultando en 169 posiciones posibles para los polos. Sin embargo, esta situación es desfavorable para filtros con polos agrupados cerca de $\theta = 0$ y $\theta = \pi$.

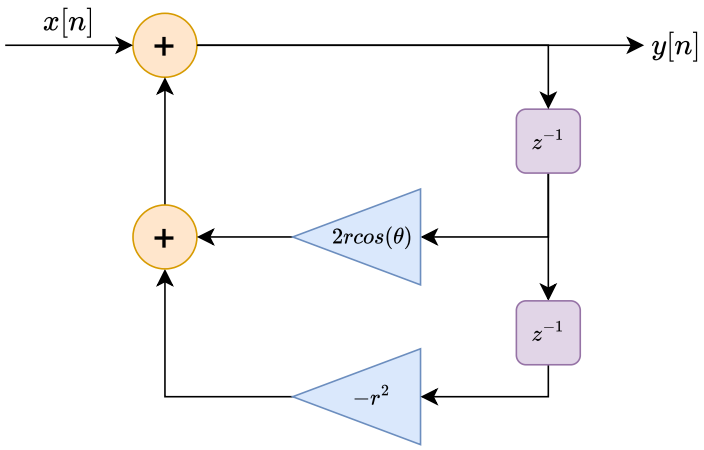

Figure 18: Realización del filtro IIR de dos polos

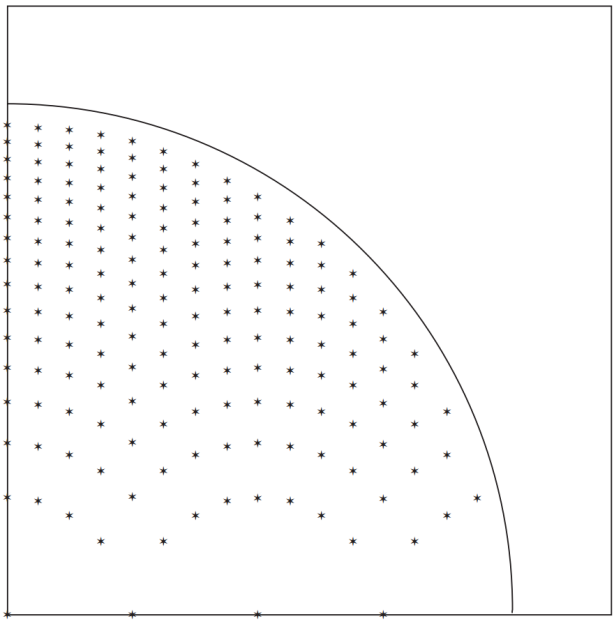

Figure 19: Posibles posiciones de los polos para la implementación del filtro IIR de 2 polos.

Una realización alternativa del filtro de polos es la estructura acoplada:

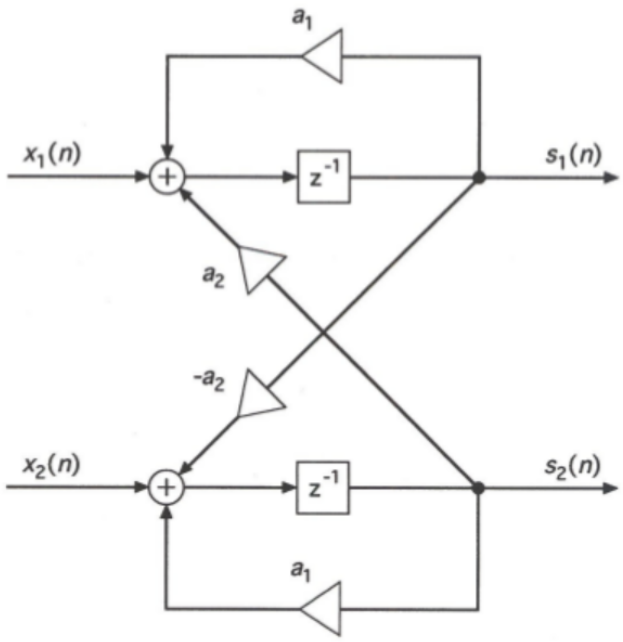

Figure 20: Estructura Acoplada

Tomando $a_1 = r \cdot cos(\theta)$ y $a_2 = r \cdot sin(\theta)$ los polos de la transferencia coinciden con el sistema de 2do orden y como a_1 y a_2 son parte real e imaginaria de los polos, da lugar a una rejilla de cuantización.

$$y_1(n) = x(n) + r\cos\theta\, y_1(n-1) - r\sin\theta\, y(n-1) \tag{20}$$

$$y(n) = r\sin\theta\, y_1(n-1) + r\cos\theta\, y(n-1) \tag{21}$$

Transformando estas ecuaciones en el dominio z, se puede demostrar que:

$$\frac{Y(z)}{X(z)} = H(z) = \frac{(r\sin\theta)z^{-1}}{1 - (2r\cos\theta)z^{-1} + r^2 z^{-2}} \tag{22}$$

En la realización acoplada, los coeficientes $a_1 = r \sin\theta$ y $a_2 = r \cos\theta$ son lineales en r, resultando en posiciones de polos igualmente espaciadas en una cuadrícula rectangular. En este caso hay 198 posibles posiciones para los polos pero tenemos un incremento del cálculo (4 multiplicaciones vs 2)

En resumen, existen múltiples formas de realizar una sección de filtro de segundo orden, cada una con distintas posibilidades para las posiciones de los polos con coeficientes cuantificados.
La selección de la estructura ideal depende de la distribución deseada de los polos y los cálculos necesarios para implementarla.

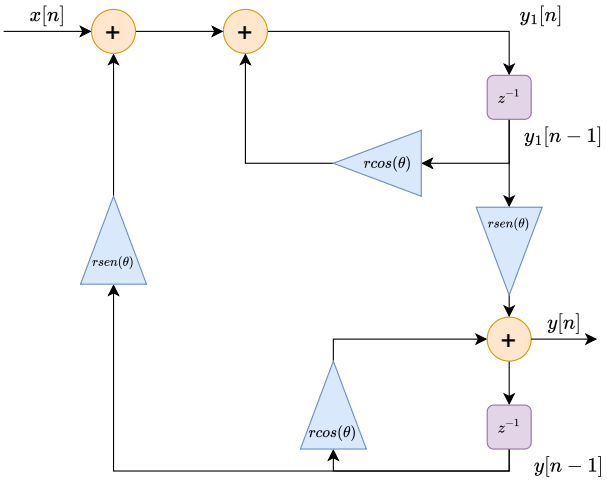

Figure 21: Realización acoplada de un filtro IIR de dos polos

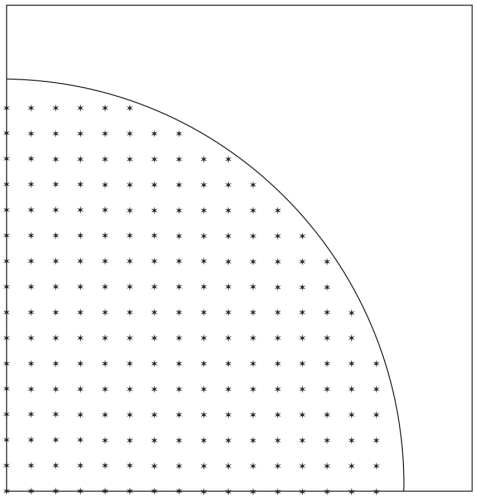

Figure 22: Posibles posiciones de los polos para el filtro acoplado de dos polos.

3.2 Escalado de los coeficientes para prevenir desbordamiento

Utilizando aritmética complemento a 2, la suma de dos números del mismo signo puede provocar un valor que exceda el máximo representable, dando lugar a un resultado de distinto signo.

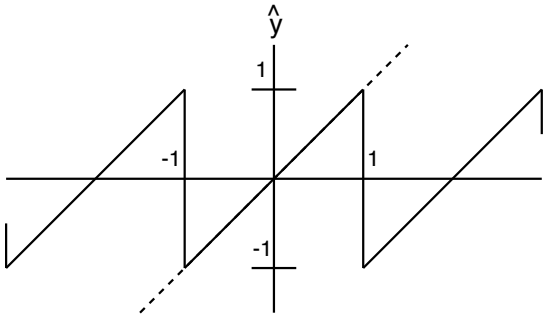

Figure 23: Suma Complemento a 2.

En un filtro recursivo, el desbordamiento producido en una suma, puede volver a producirse debido a la realimentación. Una manera de prevenir el desbordamiento es escalar las entradas de los sumadores por un factor, de manera tal de mantener su salida dentro de los límites de la representación. Al realizar un escalado de la señal de entrada por un factor $S(S < 1)$, de acuerdo con la expresión $SNR_{AD} = (10\log(\sigma_x^2) + 10.8 + 6.02b)db$, la varianza de la señal escalada será $S^2\sigma_x^2$ y dado que $S < 1$ se produciá una disminución de SNR. Sea $f(k)$ la respuesta al impulso del filtro $F(z)$, siendo esta última la función transferencia desde la entrada hasta la salida del nodo sumador considerado.

$$\begin{aligned} L_1 : S &= \sum_{k=0}^{\infty} |f(k)| \\ L_2 : S &= \left[\sum_{k=0}^{\infty} f(k)^2\right]^{\frac{1}{2}} \\ L_\infty : S &= max|F(\omega)| \end{aligned} \tag{23}$$

$$L_2 < L_\infty < L_1 \tag{24}$$

La condición más restrictiva es L_1, asegura que no va a producirse desbordamiento a la salida pero provoca la mayor disminución de SNR.
L_∞: $F(\omega)$ es la respuesta en frecuencia desde la entrada hasta la salida del sumador. Valor pico de la respuesta en frecuencia. Asegura que no va a producirse desbordamiento cunado la entrada es una sinusoide pura.

3.2.1 Escalado formas de orden 2

En la siguiente figura se muestran los bloques de segundo orden con la forma directa I (a) y II (b). Se escala multiplicando la entrada por el factor $\frac{1}{s_1}$ y la salida del filtro se multiplica por s_1 para que la función transferencia total no tenga cambios.

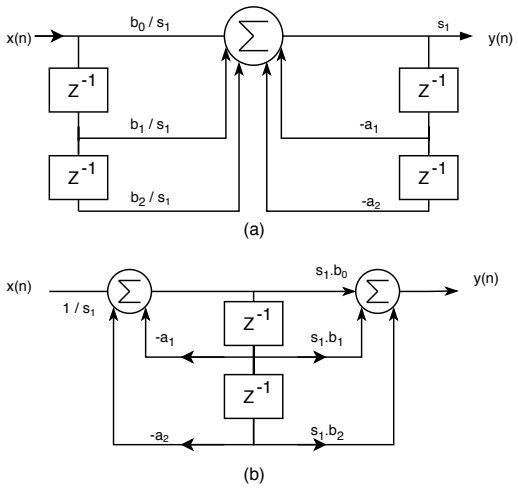

Figure 24: Escalado en estructuras de orden 2.

3.3 Prevención desbordamiento en la salida del filtro

Si utilizamos los factores de escalado determinados por las normas L_2 y $L_{\inf}$ no se evita completamente la posibilidad de desbordamiento (solo se evita completamente con L_1). Si se utiliza aritmética en complemento a 2, el desbordamiento hace que la salida cambie bruscamente entre los niveles máximos y mínimo. Para evitar esto, se utiliza aritmética saturada que en caso de producirse un desbordamiento, mantiene el nivel de la señal de salida en su punto máximo o mínimo, dependiendo del tipo de desbordamiento.

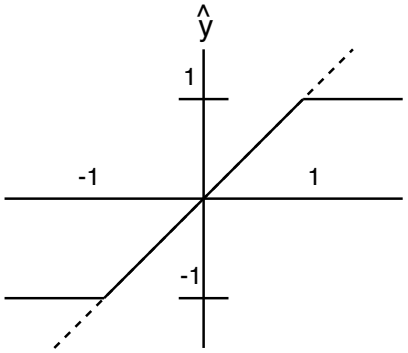

Figure 25: Aritmética saturada.

3.4 Ciclos Límite

- Un filtro digital es un sistema no lineal, debido a la cuantización de las operaciones.
- En los sistemas recursivos, las no linealidades debido a las operaciones aritméticas de precisión finita a menudo producen oscilaciones periódicas en la salida, incluso si la secuencia de entrada es cero o un valor constante distinto de cero.

Consideremos el sistema de un sólo polo descrito por la ecuación en diferencias lineal:

$$y[n] = ay[n-1] + x[n] \tag{25}$$

Mientras que el sistema real, se describe por la ecuación en diferencias **no lineal**:

$$v[n] = Q[ay[n-1]] + x[n] \tag{26}$$

- Entrada $x[n] = \beta\delta[n]$, donde $\beta = 15/16$, que tiene representación 0.1111

Ejemplo: Suponemos que el sistema real se implementa con punto fijo de 4bits para el módulo más un bit de signo. En la siguiente tabla enumeramos la respuesta del sistema real para cuatro posiciones diferentes del polo $z = a$

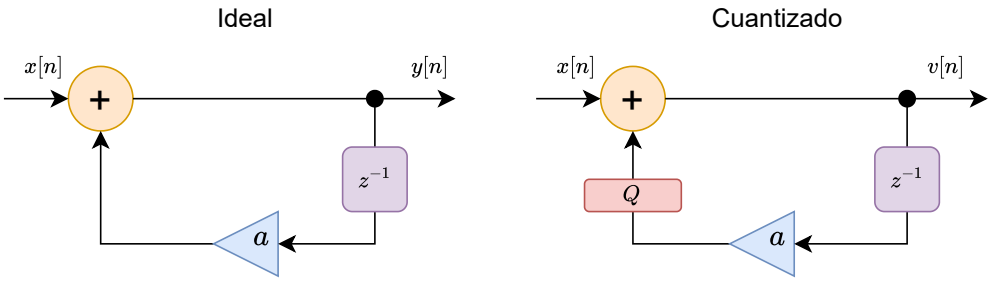

Figure 26: Limit cycle en sistema de 1er orden.

n	$a = 0.1000 = \frac{1}{2}$		$a = 1.1000 = -\frac{1}{2}$		$a = 0.1100 = \frac{3}{4}$		$a = 1.1100 = -\frac{3}{4}$	
0	0.1111	$\left(\frac{15}{16}\right)$	0.1111	$\left(\frac{15}{16}\right)$	0.1011	$\left(\frac{11}{16}\right)$	0.1011	$\left(\frac{11}{16}\right)$
1	0.1000	$\left(\frac{8}{16}\right)$	1.1000	$\left(-\frac{8}{16}\right)$	0.1000	$\left(\frac{8}{16}\right)$	1.1000	$\left(-\frac{8}{16}\right)$
2	0.0100	$\left(\frac{4}{16}\right)$	0.0100	$\left(\frac{4}{16}\right)$	0.0110	$\left(\frac{6}{16}\right)$	0.0110	$\left(\frac{6}{16}\right)$
3	0.0010	$\left(\frac{2}{16}\right)$	1.0010	$\left(-\frac{2}{16}\right)$	0.0101	$\left(\frac{5}{16}\right)$	1.0101	$\left(-\frac{5}{16}\right)$
4	0.0001	$\left(\frac{1}{16}\right)$	0.0001	$\left(\frac{1}{16}\right)$	0.0100	$\left(\frac{4}{16}\right)$	0.0100	$\left(\frac{4}{16}\right)$
5	0.0001	$\left(\frac{1}{16}\right)$	1.0001	$\left(-\frac{1}{16}\right)$	0.0011	$\left(\frac{3}{16}\right)$	1.0011	$\left(-\frac{3}{16}\right)$
6	0.0001	$\left(\frac{1}{16}\right)$	0.0001	$\left(\frac{1}{16}\right)$	0.0010	$\left(\frac{2}{16}\right)$	0.0010	$\left(\frac{2}{16}\right)$
7	0.0001	$\left(\frac{1}{16}\right)$	1.0001	$\left(-\frac{1}{16}\right)$	0.0010	$\left(\frac{2}{16}\right)$	1.0010	$\left(-\frac{2}{16}\right)$
8	0.0001	$\left(\frac{1}{16}\right)$	0.0001	$\left(\frac{1}{16}\right)$	0.0010	$\left(\frac{2}{16}\right)$	0.0010	$\left(\frac{2}{16}\right)$

Figure 27: Ciclos limite para el filtro pasa bajo de un solo polo

4 IIR Rápidos

El rendimiento de los filtros FIR se puede mejorar, esencialmente sin costo, mediante el uso de *pipeline*. Sin embargo, el uso de *pipeline* en los filtros IIR es más sofisticado y no es gratuito. Ya que, introducir registros de pipeline, especialmente en el feedback path, muy probablemente cambiará las ubicaciones de los polos y, por lo tanto, la función de transferencia del filtro IIR. Sin embargo, se han reportado estrategias en la literatura que no cambian la función de transferencia y permiten un mayor rendimiento. Un resumne de los métodos reportados para mejorar el rendimiento de los filtros IIR son:

- Look-ahead interleaving en el dominio del tiempo [1]
- Clustered look-ahead pole/zero assignment [2, 3]
- Scattered look-ahead pole/zero assignment [1, 4]
- Diseño de filtros de decimación IIR [5]
- Procesamiento en paralelo [6]
- Implementación RNS

Los primeros cinco métodos se basan en la arquitectura del filtro o en técnicas de flujo de señal, y el último se basa en la aritmética computacional.

4.1 Look-ahead interleaving en el dominio del tiempo

Considere la ecuación diferencial de un sistema IIR de primer orden:

$$y[n+1] = ay[n] + bx[n]. \tag{27}$$

La salida del sistema de primer orden, $y[n+1]$, se puede calcular usando la técnica de *look-ahead* (anticipación) al sustituir $y[n+1]$ en la ecuación diferencial para $y[n+2]$. Es decir:

$$y[n+2] = ay[n+1] + bx[n+1] = a^2y[n] + abx[n] + bx[n+1]. \tag{28}$$

Este concepto se puede generalizar aplicando la transformación de *look-ahead* para $(S-1)$ pasos, resultando en:

$$y[n+S] = a^S y[n] + \sum_{k=0}^{S-1} a^k bx[n+S-1-k]. \tag{29}$$

Se puede ver que el término (η) define un filtro FIR con coeficientes $\{b, ab, a^2b, \ldots, a^{S-1}b\}$, que se puede implementar utilizando las técnicas de pipeline presentadas en la materia (ejemplo: multiplicador y árboles de sumadores con pipeline). La parte recursiva de la ecuación ahora también se puede implementar con un multiplicador pipeline de S etapas para el coeficiente a^S.

4.1.1 Lossy Integrator con Look-Ahead

Consideramos nuevamente el *lossy integrator*, pero ahora utilizando la téecnica de look-ahead. En la siguiente figura podemos observar la implementación propuesta, que es una combinación de una parte no recursiva (es decir, filtro FIR para x) y una parte recursiva con un retraso de 2 y un coeficiente de 9/16.

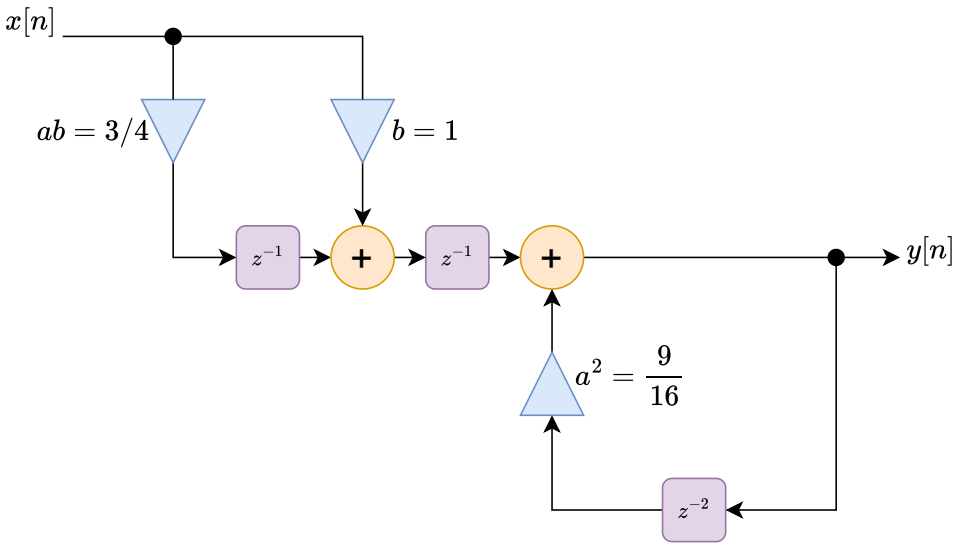

Figure 28: Lossy Integrator utilizando aritmética look-ahead

$$y[n+2] = \frac{3}{4}y[n+1] + x[n+1] = \frac{3}{4}\left(\frac{3}{4}y[n] + x[n]\right) + x[n+1] \tag{30}$$

$$= \frac{9}{16}y[n] + \frac{3}{4}x[n] + x[n+1]. \tag{31}$$

$$y[n] = \frac{9}{16}y[n-2] + \frac{3}{4}x[n-2] + x[n-1]. \tag{32}$$

```
module iir_pipe #(parameter W = 14) (
    input clk, // System clock
    input reset, // Asynchronous reset
    input signed [W:0] x_in, // System input
    output signed [W:0] y_out
);

reg signed [W:0] x, x3, sx;
reg signed [W:0] y, y9;

always @(posedge clk or posedge reset) begin
    if (reset) begin // Asynchronous clear
        x <= 0; x3 <= 0; sx <= 0; y9 <= 0; y <= 0;
    end else begin
        x  <= x_in; // use non-blocking FF assignments
        x3 <= (x >>> 1) + (x >>> 2);
              // i.e. x / 2 + x / 4 = x*3/4
        sx <= x + x3;//Sum of x element i.e. output FIR part
        y9 <= (y >>> 1) + (y >>> 4);
              // i.e. y / 2 + y / 16 = y*9/16
        y  <= sx + y9; // Compute output
    end
end

assign y_out = y ; // Connect register y to output pins
endmodule
```

Listing 2: IIR Look-Ahead

4.1.2 Resumen

- Podemos utilizar pipeline en el forward path para aumentar el throughput
- Incrementa el delay total del sistema
- Se aumenta la velocidad del sistema, pero necesitamos aumentar el budget de area para este bloque
- En el feedback path podemos utilizar un multiplicador pipeline para el coeficiente a^S.

4.2 Clustered and Scattered Look-Ahead Pipelining

Los esquemas de clustered (agrupamiento) y scattered (dispersión) look-ahead (anticipación) en pipelines añaden polos y ceros auto-cancelantes al diseño para facilitar el pipeling de la parte recursiva del filtro. En el método de clustered, se introducen polos/ceros adicionales de tal manera que en el denominador de la función de transferencia los coeficientes para $z^{-1}, z^{-2}, \ldots, z^{-(S-1)}$ se vuelven cero.

Dado el siguiente sistema:

$$H(z) = \frac{1}{(1 - r_1 z^{-1})(1 - r_2 z^{-1})} = \frac{1}{1 - (r_1 + r_2)z^{-1} + r_1 r_2 z^{-2}} \tag{33}$$

Agregamos un par polo/cero:

$$H(z) = \frac{1 + (r_1 + r_2)z^{-1}}{(1 + (r_1 + r_2)z^{-1})(1 - (r_1 + r_2)z^{-1} + r_1 r_2 z^{-2})} \tag{34}$$

$$H(z) = \frac{1 + (r_1 + r_2)z^{-1}}{1 - (r_1 + r_2)z^{-1} + r_1 r_2 z^{-2} + (r_1 + r_2)z^{-1}(r_1 r_2 z^{-2}) - (r_1 + r_2)^2 z^{-2} + (r_1 + r_2) r_1 r_2 z^{-3}} \tag{35}$$

$$H(z) = \frac{1 + (r_1 + r_2)z^{-1}}{1 + (r_1 r_2 - (r_1 + r_2)^2)z^{-2} + (r_1 + r_2) r_1 r_2 z^{-3}} \tag{36}$$

4.2.1 Ejemplo: Método Clustered

Se asume que una función de transferencia de segundo orden tiene un polo en 1/2 y 3/4 y su función de transferencia dada por:

$$F(z) = \frac{1}{1 - 1.25z^{-1} + 0.375z^{-2}} = \frac{1}{(1 - 0.5z^{-1})(1 - 0.75z^{-1})}. \tag{37}$$

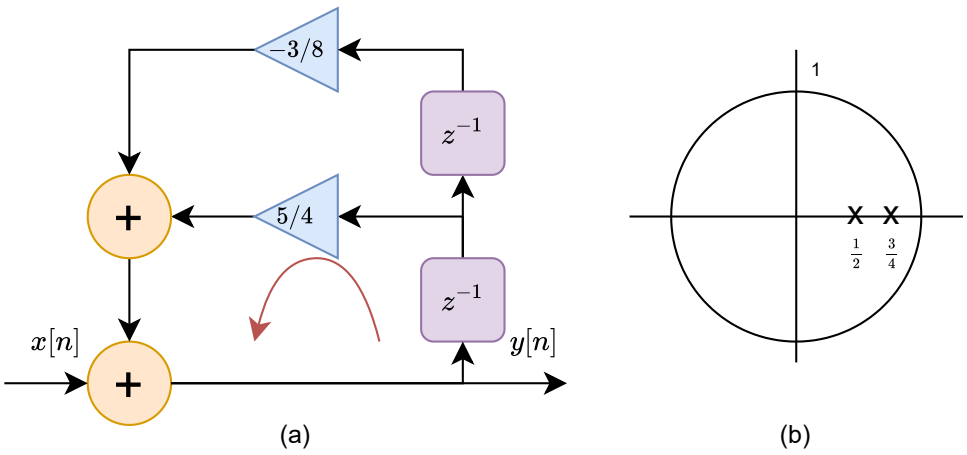

Figure 29: IIR 2do orden con polos en 1/2 y 3/4. (a) Diagrama en bloques. (b) Diagrama polos y ceros

Tenemos 2 loops, uno interior y otro exterior, pero el peor escenario lo tenemos en el loop marcado en la figura donde:

$$T_\infty = \frac{T_M + 2T_A}{Num_{delays}} = \frac{T_M + 2T_A}{1} \tag{38}$$

Por lo tanto para mejorar esa métrica debemos agregar mas delay en ese loop. Añadiendo un polo/cero cancelante en $z = -\frac{5}{4} = -1.25$ resulta en una nueva función de transferencia:

$$F(z)_{m=2} = \frac{1 + \frac{5}{4}z^{-1}}{1 - \frac{19}{16}z^{-2} + \frac{15}{32}z^{-3}}. \tag{39}$$

Si reiteramos el proceso iterando sobre el término $-1.1875z^{-2}$ se obtiene:

$$F(z)_{m=3} = \frac{1 + \frac{5}{4}z^{-1} + \frac{19}{16}z^{-2}}{1 - \frac{65}{64}z^{-3} + \frac{57}{128}z^{-4}}. \tag{40}$$

La parte recursiva del filtro ahora se puede implementar con una etapa de pipeline adicional.

El problema con el clustered es que el par de polos/ceros cancelados puede estar fuera del círculo unitario, como es el caso en el ejemplo anterior (i.e., $z_\infty = -1.25$). Esto **introduce inestabilidad en el diseño si la aniquilación de polos/ceros no es perfecta**. En general, un sistema de segundo orden con polos en r_1, r_2 y con un par cancelante adicional, tiene una ubicación de polo en $-(r_1 + r_2)$, lo cual está fuera del círculo unitario si $|r_1 + r_2| > 1$. Soderstrand et al. [3] han descrito un método de agrupamiento estable, que en general introduce más de un par de polos/ceros cancelantes.

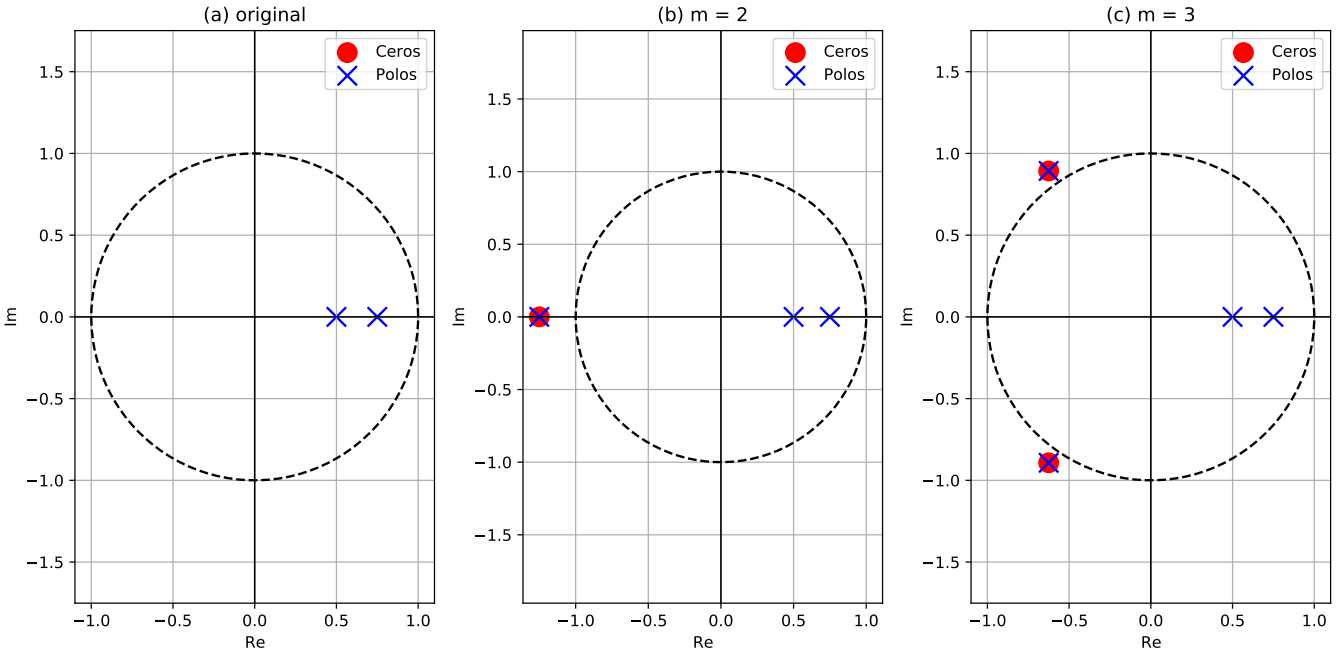

Figure 30: Estabilidad en método clustered

El enfoque de look-ahead scattered no introduce problemas de estabilidad. Introduce $(S-1)$ pares de polos/ceros cancelantes ubicados en $z_k = pe^{j\pi k/S}$, para un filtro original con un polo ubicado en p. El denominador de la función de transferencia tiene, como resultado, solo coeficientes cero asociados con los términos z^0, z^S, z^{-2S}, etc.

4.2.2 Ejemplo: Método scattered look-ahead

Consideramos la implementación de un sistema de segundo orden con polos ubicados en $z_{\infty 1} = 0.5$ y $z_{\infty 2} = 0.75$ con dos etapas de pipeline adicionales. Una función de transferencia de un filtro con polos en 1/2 y 3/4 tiene la función de transferencia:

$$F(z) = \frac{1}{1 - 1.25z^{-1} + 0.375z^{-2}} = \frac{1}{(1 - 0.5z^{-1})(1 - 0.75z^{-1})}. \tag{41}$$

Supongamos que trabajamos la transferencia como si fuera una cascada de filtros de 1 polo y aplicamos clustered a ambos:

$$\begin{aligned} F(z) &= \tfrac{1}{1-\frac{1}{2}z^{-1}} \cdot \tfrac{1}{1-\frac{3}{4}z^{-1}} \\ F(z) &= \tfrac{1+\frac{1}{2}z^{-1}}{1-\frac{1}{4}z^{-2}} \cdot \tfrac{1+\frac{3}{4}z^{-1}}{1-\frac{9}{16}z^{-2}} \\ F(z) &= \tfrac{1+\frac{5}{4}z^{-1}+\frac{3}{8}z^{-2}}{1-\frac{13}{16}z^{-2}+\frac{9}{64}z^{-4}} \end{aligned} \tag{42}$$

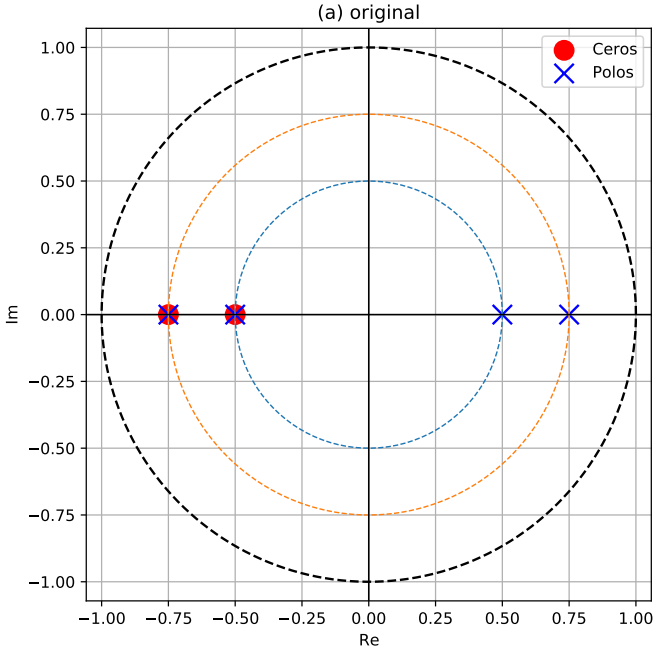

Figure 31: Scattered aplicado a F(z)

En general, un par de polos/ceros en p y p^* resulta en una función de transferencia de:

$$(1 - pz^{-1})(1 - p^*z^{-1}) = 1 - (p + p^*)z^{-1} + rr^*z^{-2}. \tag{43}$$

En particular, con $p = re^{(j2\pi/3)}$, tenemos:

$$(1 - pz^{-1})(1 - p^*z^{-1}) = 1 - 2r\cos(2\pi/3)z^{-1} + r^2z^{-2} = 1 + rz^{-1} + r^2z^{-2}. \tag{44}$$

Scattered Look-Ahead introduce dos etapas de pipeline adicionales añadiendo pares de polos/ceros en $0.5e^{\pm j2\pi/3}$ y $0.75e^{\pm j2\pi/3}$. Añadiendo un polo/cero cancelante en esta ubicación resulta en:

$$\begin{aligned} F(z) &= \tfrac{1}{1-1.25z^{-1}+0.375z^{-2}} \times \tfrac{(1+0.5z^{-1}+0.25z^{-2})(1+0.75z^{-1}+0.5625z^{-2})}{(1+0.5z^{-1}+0.25z^{-2})(1+0.75z^{-1}+0.5625z^{-2})} \\ &= \tfrac{1+1.25z^{-1}+1.1875z^{-2}+0.4687z^{-3}+0.1406z^{-4}}{1-0.5469z^{-3}+0.0527z^{-6}} \\ &= \tfrac{512+640z^{-2}+608z^{-2}+240z^{-3}+72z^{-4}}{512-280z^{-3}+27z^{-6}}, \end{aligned} \tag{45}$$

y la parte recursiva se puede implementar con dos etapas de pipeline adicionales.

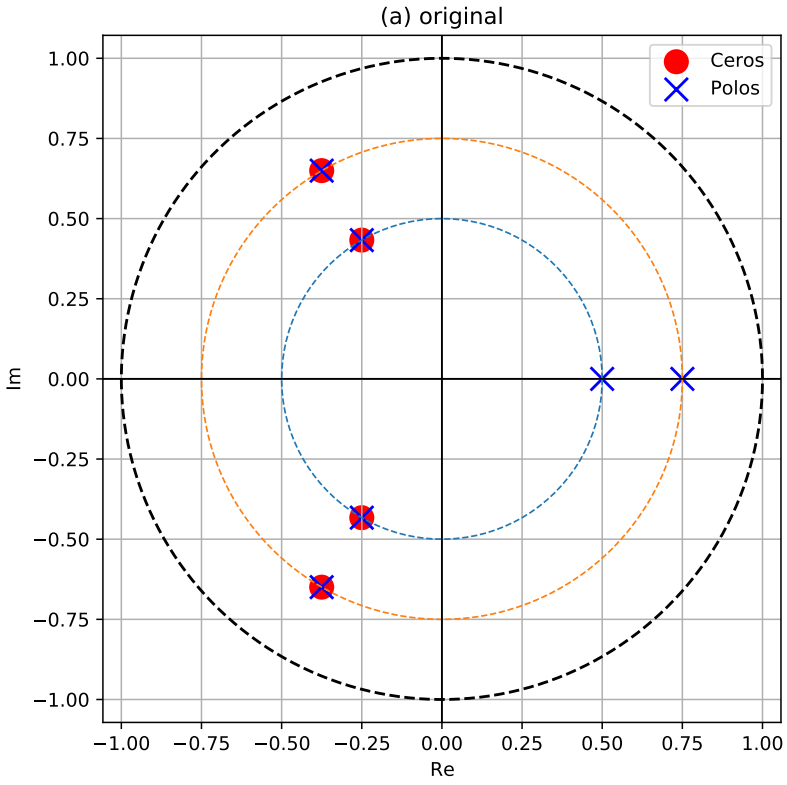

Figure 32: Par polo/zeros adicionales en $p = re^{(j2\pi/3)}$

Es interesante notar que para un sistema IIR de primer orden, los métodos de clustered y scattered look-ahead resultan en el mismo par de polos/ceros cancelantes ubicados en un círculo alrededor del origen con diferencias angulares $2\pi/S$. La parte no recursiva se puede realizar con una *descomposición de potencias de dos* según:

$$(1 + az^{-1})(1 + a^2z^{-2})(1 + a^4z^{-4}) \cdots . \tag{46}$$

La siguiente figura muestra dicha representación de polos/ceros para una sección de primer orden, lo que permite una implementación con cuatro etapas de pipeline en la parte recursiva.

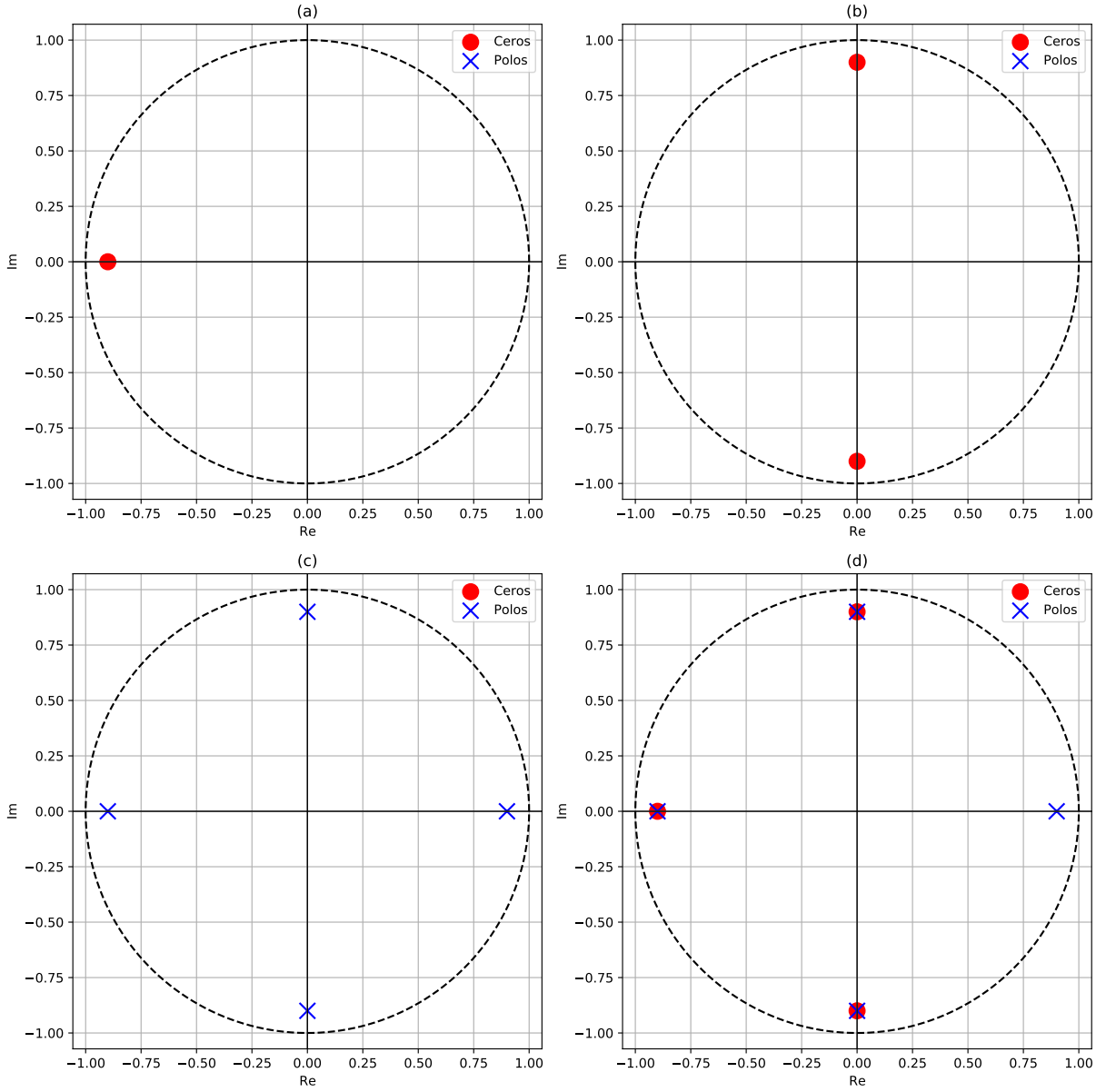

Figure 33: Plot de polos y ceros para un filtro IIR de primer orden utilizando scattered look-ahead. (a) $F_1(z) = (1 + az^{-1})$ (b) $F_2(z) = (1 + a^2z^{-2})$ (c) $F_3(z) = 1/(1 + a^4z^{-4})$ (d) $F(z) = \prod_k F_k(z) = \frac{(1+az^{-1})(1+a^2z^{-2})}{(1-a^4z^{-4})} = \frac{1}{(1-az^{-1})}$

4.2.3 Resumen

- Scattered permite estabilidad, ya que desplazamos los polos sobre el circulo que los contiene.
- Clustered tiene como contra que necesitamos una aniquilación polo/cero prefecta cuando el polo está por fuera del círculo unidad.
- Permiten eliminar términos de menor orden en el feedback path.
- Siempre que aumentemos el grado de pipelining, tendremos mayor circuiteria y complejidad asociada al diseño.

5 Procesamiento en Paralelo

- En una implementación de filtro de procesamiento en paralelo [6], se forman N caminos IIR paralelos, cada uno ejecutándose a una tasa de muestreo de entrada de $1/N$.
- Las salidas se combinan utilizando un multiplexor.
- Un multiplexor, en general, será más rápido que un multiplicador y/o sumador, haciendo que el enfoque en paralelo sea más rápido.
- Cada camino P tiene un factor de P más tiempo para calcular su valor.
- La desventaja de esta técnica es su alto costo de implementación en términos de área. Aumentamos el uso de recursos para ir a mayor velocidad.

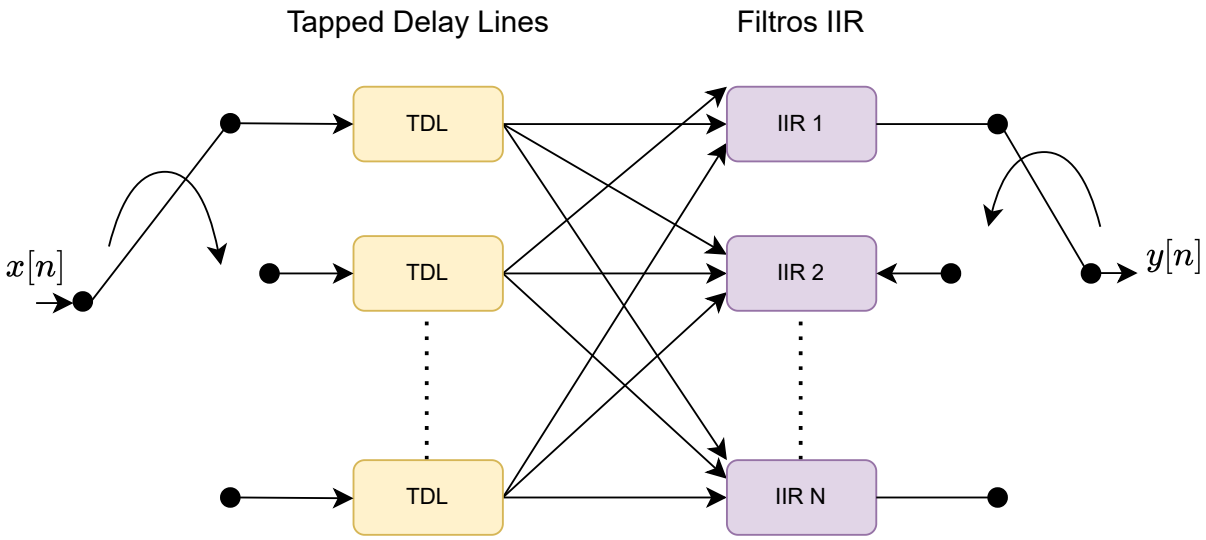

Figure 34: Implementación filtro IIR en paralelo. N canales de entrada procesando en un sample rate de $1/N$

Considerando un lossy integrator de primer orden:

$$y[n] = ay[n-1] + x[n-1] \tag{47}$$

Aplicamos Look-Ahead para obtener $P = 2$:

$$y[n+2] = ay[n+1] + x[n+1] = a^2 y[n] + ax[n] + x[n+1] \tag{48}$$

Separamos las secuencias de salidas entre pares $n = 2k$ e impares $n = 2k-1$, obteniendo:

$$y[n+2] = \begin{cases} y[2k+2] = a^2 y[2k] + ax[2k] + x[2k+1] \\ y[2k+1] = a^2 y[2k-1] + ax[2k-1] + x[2k] \end{cases} \tag{49}$$

donde $n, k \in \mathbb{Z}$. Estas dos ecuaciones son la base para la siguiente implementación RTL de filtro IIR en paralelo.

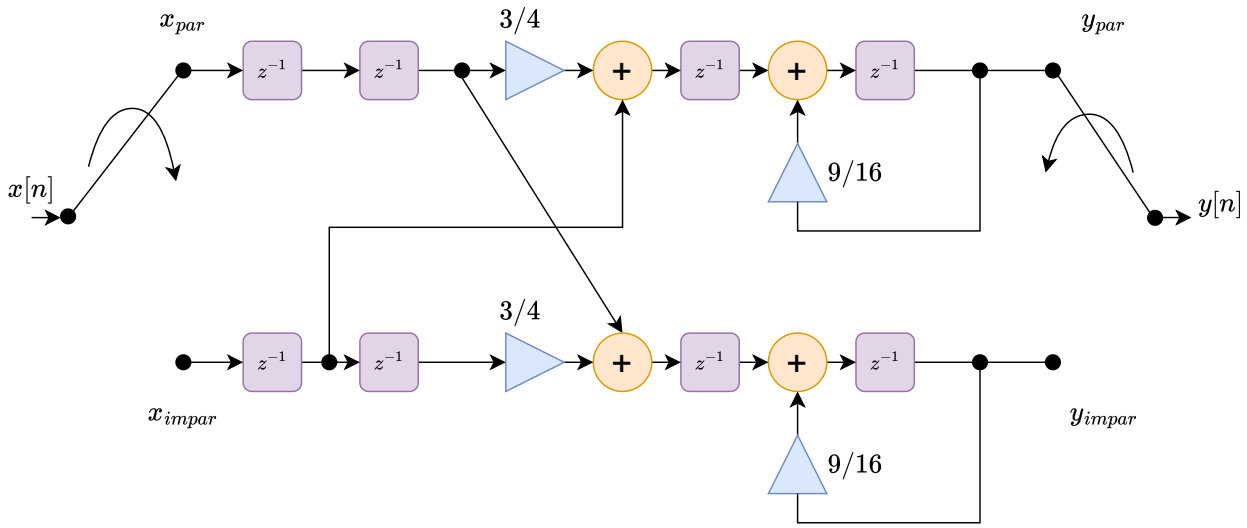

Figure 35: Implementación IIR de 2 ramas en paralelo. Cada rama corre a 1/2 de la tasa de muestreo o sample rate.

```
1
2  module iir_par #(parameter W = 14)(
3      input clk, // System clock
4      input reset, // Asynchronous reset
5      input signed [W:0] x_in, // System input
6      output signed [W:0] x_e, x_o,// Even/odd input x_in
7      output signed [W:0] y_e, y_o,// Even/odd output y_out
8      output clk2, // Clock divided by 2
9      output signed [W:0] y_out
10 );
11
12 reg signed [W:0] x_even, xd_even, x_odd, xd_odd, x_wait;
13 reg signed [W:0] y_even, y_odd, y_wait, y;
14 reg signed [W:0] sum_x_even, sum_x_odd;
15 reg clk_div2;
16 reg [0:0] state;
17
18 always @(posedge clk or posedge reset) begin : clk_divider // by 2 for input clk
19     if (reset) clk_div2 <= 0;
20     else       clk_div2 <= ! clk_div2;
21 end
22
23 always @(posedge clk or posedge reset) begin : Multiplex // even and odd samples;
24     parameter even=0, odd=1; // recombine y at clk rate
25     if (reset) begin // Asynchronous reset
26         state <= even; x_even <= 0; x_odd  <= 0;
27         y     <= 0;    x_wait <= 0; y_wait <= 0;
28     end else begin
29         case (state)
30             even : begin
31                 x_even <= x_in;
32                 x_odd  <= x_wait;
33                 y      <= y_wait;
34                 state  <= odd;
35             end
36             odd : begin
37                 x_wait <= x_in;
38                 y      <= y_odd;
39                 y_wait <= y_even;
40                 state  <= even;
41             end
42         endcase
43     end
44 end
45
46 assign y_out = y;
47 assign clk2  = clk_div2;
```

```
assign x_e   = x_even; // Monitor some extra test signals
assign x_o   = x_odd;
assign y_e   = y_even;
assign y_o   = y_odd;

always @(negedge clk_div2 or posedge reset) begin: Arithmetic
    if (reset) begin
        xd_even <= 0; sum_x_even <= 0; y_even <= 0;
        xd_odd  <= 0; sum_x_odd  <= 0; y_odd  <= 0;
    end else begin
        xd_even    <= x_even;
        sum_x_even <= x_odd      + (xd_even >>> 1) + (xd_even >>> 2);
        // i.e. x_odd + x_even/2 + x_even /4
        y_even     <= sum_x_even + (y_even  >>> 1) + (y_even  >>> 4);
        // i.e. sum_x_even + y_even/2+ y_even /16
        xd_odd     <= x_odd;
        sum_x_odd  <= xd_even    + (xd_odd >>> 1)  + (xd_odd  >>> 4);
        // i.e. x_even + xd_odd/2+ xd_odd /4
        y_odd      <= sum_x_odd  + (y_odd  >>> 1)  + (y_odd   >>> 4);
        // i.e. sum_x_odd + y_odd/2+ y_odd / 16
    end
end
endmodule
```

Listing 3: IIR 2 Paths en Paralelo

5.0.1 Ejercicio: Paralelización mediante fracciones parciales

Encuentre la implementación en paralelo del sistema caracterizado por la función de transferencia:

$$H[Z] = \frac{z(5z-2)}{(z+\frac{1}{2})(z-\frac{1}{3})} \tag{50}$$

Debemos encontrar la fracción parcial de $H[Z]/Z$:

$$\frac{H[Z]}{Z} = \frac{(5z-2)}{(z+\frac{1}{2})(z-\frac{1}{3})} = \frac{A}{(z+\frac{1}{2})} + \frac{B}{(z-\frac{1}{3})} \tag{51}$$

Donde

$$A = \left.\frac{(5z-2)}{(z-\frac{1}{3})}\right|_{z=-\frac{1}{2}} \quad \text{Entonces,} \quad A = \frac{27}{5} \tag{52}$$

y

$$B = \left.\frac{(5z-2)}{(z+\frac{1}{2})}\right|_{z=\frac{1}{3}} \quad \text{Y} \quad B = -\frac{2}{5} \tag{53}$$

Entonces,

$$\begin{aligned} H[Z] &= \left(\tfrac{27}{5}\right)\tfrac{z}{(z+\frac{1}{2})} - \left(\tfrac{2}{5}\right)\tfrac{z}{(z-\frac{1}{3})} \\ H[Z] &= \left(\tfrac{27}{5}\right)\tfrac{1}{(1+\frac{1}{2}z^{-1})} - \left(\tfrac{2}{5}\right)\tfrac{z}{(1-\frac{1}{3}z^{-1})} \end{aligned} \tag{54}$$

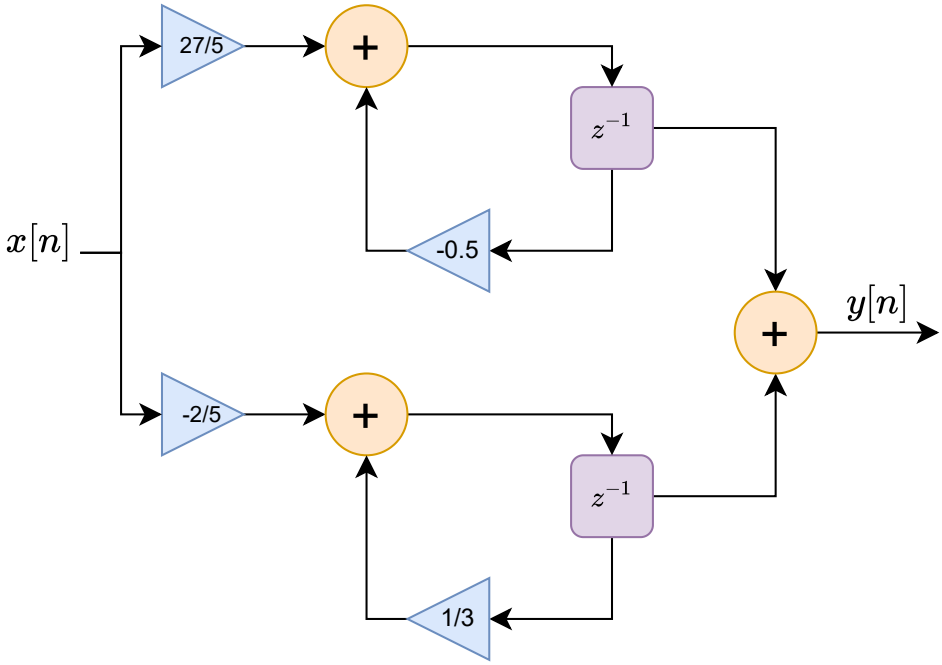

Figure 36: IIR Paralelo mediante fracciones parciales

References

[1] K. Parhi, D. Messerschmidt: "Pipeline Interleaving and Parallelism in Recursive Digital Filters - Part I: Pipelining Using Scattered Look-Ahead and Decomposition," IEEE Transactions on Acoustics, Speech and Signal Processing 37(7), 1099–1117 (1989).

[2] H. Loomis, B. Sinha: "High Speed Recursive Digital Filter Realization," Circuits, Systems, Signal Processing 3(3), 267–294 (1984).

[3] M. Soderstrand, A. de la Serna, H. Loomis: "New Approach to Clustered Look-ahead Pipelined IIR Digital Filters," IEEE Transactions on Circuits and Systems II 42(4), 269–274 (1995).

[4] J. Living, B. Al-Hashimi: "Mixed Arithmetic Architecture: A Solution to the Iteration Bound for Resource Efficient FPGA and CPLD Recursive Digital Filters," in IEEE International Symposium on Circuits and Systems Vol. I (1999), pp. 478–481.

[5] H. Martinez, T. Parks: "A Class of Infinite-Duration Impulse Response Digital Filters for Sampling Rate Reduction," IEEE Transactions on Acoustics, Speech and Signal Processing 26(4), 154–162 (1979).

[6] Parhi, D. Messerschmidt: "Pipeline Interleaving and Parallelism in Recursive Digital Filters - Part II: Pipelined Incremental Block Filtering," IEEE Transactions on Acoustics, Speech and Signal Processing 37(7), 1118–1134 (1989)

[7] L. Jackson: "Roundoff-Noise Analysis for Fixed-point Digital Filters Realized in Cascade or Parallel Form," IEEE Transactions on Audio and Electroacoustics 18(2), 107–123 (1970)

[8] Proakis, J.G. and Manolakis, D.G (1997) Digital Signal Processing Principles, Algoritms, and Applications. Prentice Hall, Inc.