# 6.- REALIZACIÓN DE SISTEMAS EN TIEMPO DISCRETO.

## *6.1.- INTRODUCCIÓN.*

Los sistemas digitales que hemos analizado responden a una función de transferencia dada por:

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

Dado que la suma de convolución nos permite calcular la salida de un sistema LTI , éste sería un procedimiento adecuado para calcular la respuesta de un sistema ante una determinada entrada. Esta implementación es válida para sistemas FIR, ya que la suma de convolución es finita, pero no es utilizable para sistemas IIR, en los que se utiliza una implementación recursiva.

IMPLEMENTACIÓN FIR. $y(n) = \sum_{k=0}^{M} h(k)x(n-k)$ (1)

IMPLEMENTACIÓN IIR: $y(n) = \sum_{k=0}^{M} b_k \cdot x(n-k) - \sum_{k=1}^{N} a_k \cdot y(n-k)$ (2)

Observamos que, en ambos casos, se trata de una suma de productos entre los coeficientes del filtro y señales de entrada y salida retardadas.
La implementación puede ser hardware o software, según la aplicación. En ambos casos las señales y los coeficientes se representan con precisión finita.

La implementación DIRECTA de un filtro (sin manipular la función de transferencia, tal como indican las ecuaciones 1 y 2), basada en la suma de convolución o la ecuación en diferencias puede no proporcionar resultados satisfactorios debido trabajar con aritmética finita, es por tanto útil analizar implementaciones alternativas que mejoren dichos resultados. Las representaciones alternativas realizan una reorganización del diagrama de bloques.

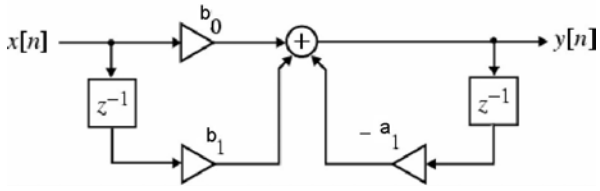
Los parámetros que van a determinar las prestaciones de una estructura son los siguientes:

- Efectos de precisión finita.
- Complejidad computacional. (Número de operaciones suma y producto necesarias)
- Requisitos de memoria. (Relacionado con el número de retardos presentes)
- Posibilidad de procesado en paralelo.

La representación de un sistema mediante bloques básicos conectados entre sí se denomina REALIZACIÓN O ESTRUCTURA. Ésta proporciona la relación entre la entrada y salida y algunas variables intermedias necesarias para su implementación.

Cuando se trabaja con precisión infinita, todas las estructuras proporcionan idénticos resultados, sin embargo estos cambian cuando la representación es finita.

Por ejemplo, dado el siguiente diagrama de bloques:



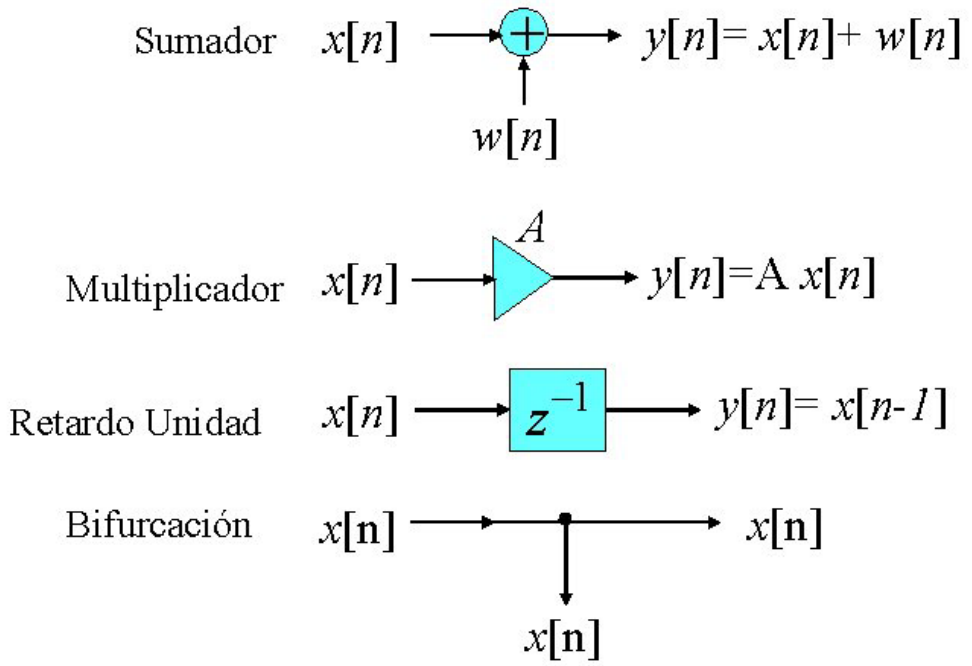

Extraído de: Digital Signal Processing. A computer-based approach. S. K, Mitra

La ecuación en diferencias es: $y(n) = -a_1 y(n-1) + b_0 x(n) + b_1 x(n-1)$, es decir podemos calcular $y(n)$ para $n \geq -1$ conociéndo la condición inicial y(-1) y la entrada x(n) para $n \geq -1$. El algoritmo de computación es el siguiente:

$$y[0] = -a_1 y[-1] + b_0 x[0] + b_1 x[-1]$$
$$y[1] = -a_1 y[0] + b_0 x[1] + b_1 x[0]$$
$$y[2] = -a_1 y[1] + b_0 x[2] + b_1 x[1]$$
$$\ldots$$

### *6.2.- Diagramas de bloques.*

La respresentación de diagramas de bloques se realiza mediante los siguientes bloques básicos:




La representación en diagrama de bloques permite:

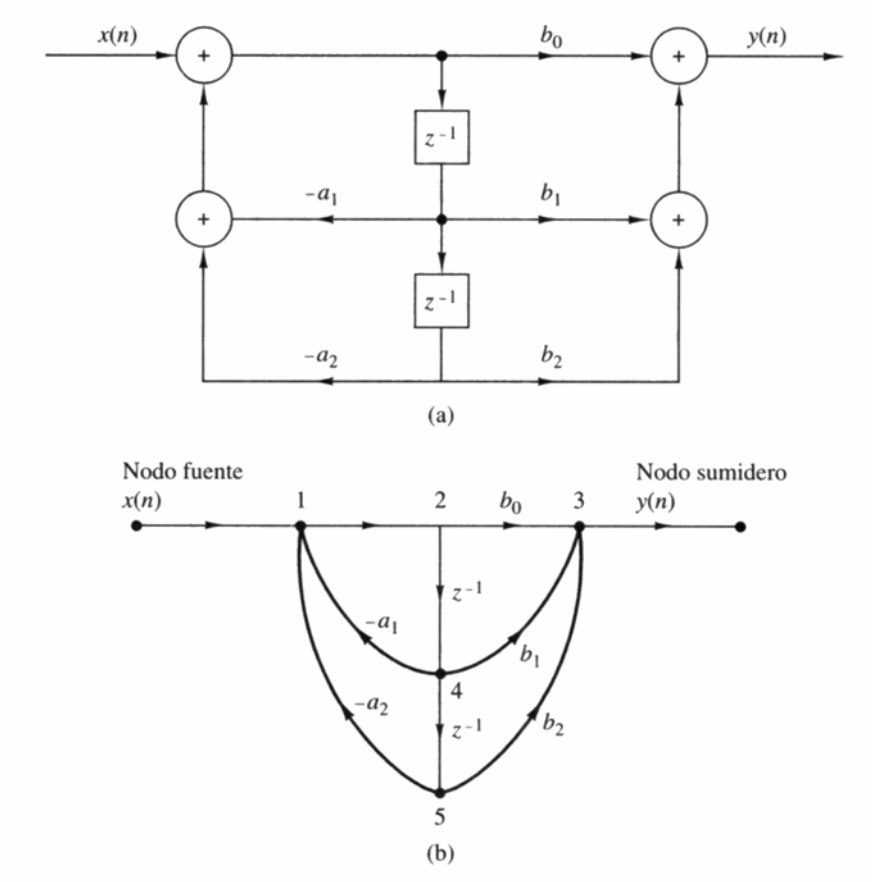
1. Escribir el algoritmo de computación.
2. Determinar la relación entre la entrada y la salida.
3. Manipular dicho diagrama para obtener otros equivalentes (igual relación entrada-salida), con distinto algoritmo de computación.
4. Determinar los requerimientos *hardware* (Memoria, carga computacional)

Representación de diagramas de bloques mediante grafos.

Una representación alternativa a los diagramas de bloques son los GRAFOS. Un gráfo está formado por un conjunto de **nodos** conectados por **ramas orientadas**, y caracterizadas por una **transmitancia de rama**. La salida de un nodo es igual a la suma de las señales a su entrada. La siguiente figura muestra un sistema representado mediante diagrama de bloques y con su grafo correspondiente. Observamos dos nodos especiales:

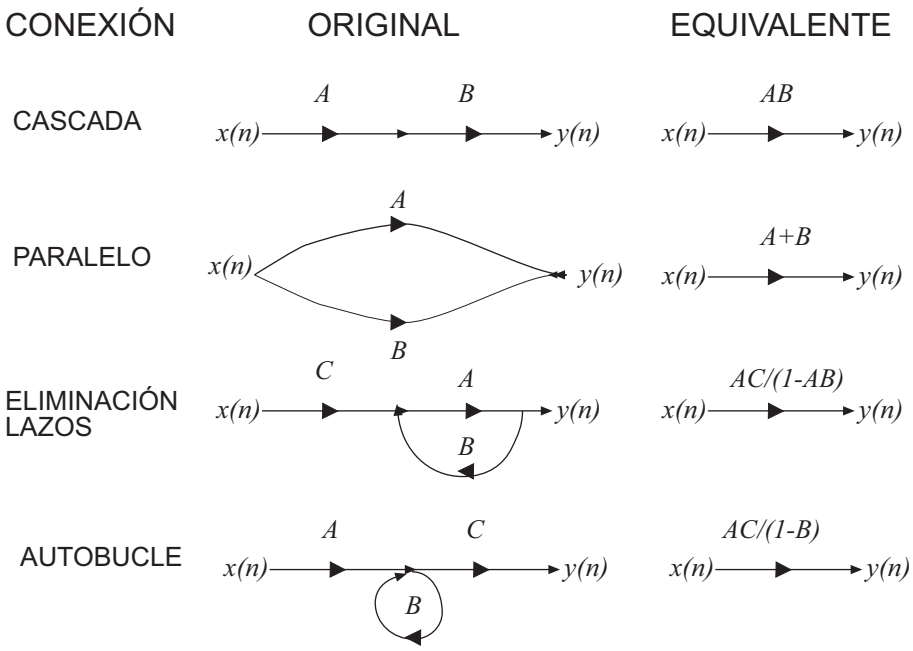
**Nodo fuente**: es aquel que no tiene ramas de entrada.

**Nodo sumidero**: es aquel que no tiene ramas de salida.




Extraído de: Tratamiento Digital de Señales. J.G. Proakis

Algunas propiedades de interés:




Para que un diagrama de bloques sea realizable físicamente no debe tener lazos (bucles) sin retardos. Si esto ocurre podemos sustituir dicho lazo sin retardo por su sistema equivalente aplicando las propiedades anteriores.

Estructuras canónicas y no canónicas.

Una estructura se dice que es CANÓNICA si el número de retardos presentes en el diagrama de bloques coincide con el orden del sistema, en otro caso se dice que es NO canónica. Un ejemplo de estructura canónica es el diagrama de bloques anterior.

Estructuras equivalentes.

Dos diagramas de bloques son EQUIVALENTES si representan a la misma función de transferencia.

Si bien la función de transferencia es la misma, las estructuras equivalentes difieren en:

- Carga computacional
- Robustez ante cálculos con registros de longitud finita.
- Sensibilidad frente a la cuantificación de los coeficientes.

Obtención de estructuras equivalentes: la TRASPOSICIÓN.

Un sencillo procedimiento para la obtención de estructuras equivalentes es la trasposición. Se basa en un teorema de la teoría de Grafos que nos dice lo siguiente:

*Si en un grafo se intercambian la entrada y la salida, y la dirección de todas las ramas, el grafo obtenido es equivalente al original.*

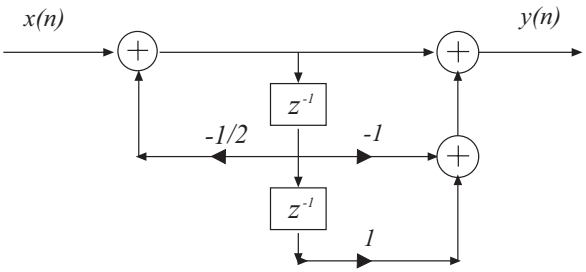
Como consecuencia del cambio de sentido de todas las ramas los nodos distribuidores (bifurcación) pasarán a ser nodos suma y viceversa.

A lo largo de este capítulo veremos otros procedimientos para la obtención de estructuras equivalentes.

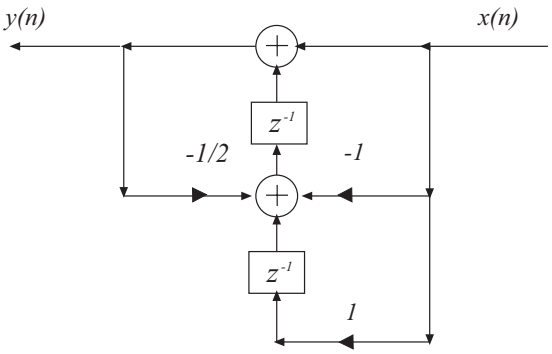
TRASPOSICIÓN DE UNA ESTRUCTURA

ENTRADA <---> SALIDA
NODO FUENTE <---> NODO SUMIDERO
NODO SUMADOR <---> NODO DISTRIBUIDOR

*FORMA ORIGINAL*




*FORMA TRASPUESTA*

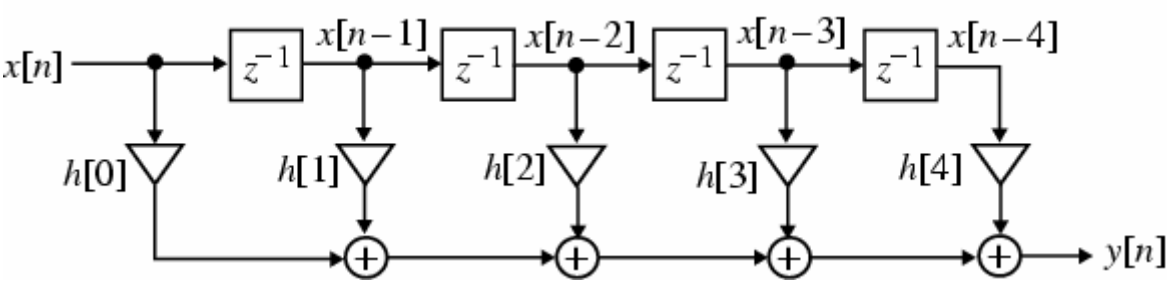


## *6.3.- Estructuras Básicas para sistemas FIR.*

### 6.3.1.- Forma directa I

Se dice que una estructura es DIRECTA cuando los coeficientes de los multiplicadores coinciden con los de la ecuación en diferencias.

Dada la función de transferencia de un sistema FIR $H(z)=\sum_{k=0}^{M} h(k)\cdot z^{-k}$, su ecuación en diferencias viene dada por $y(n)=\sum_{k=0}^{M} h(k)\cdot x(n-k)$. Si representamos esta ecuación mediante diagrama de bloques obtenemos la estructura siguiente, para M=4:




Extraído de: Digital Signal Processing. A computer-based approach. S. K, Mitra

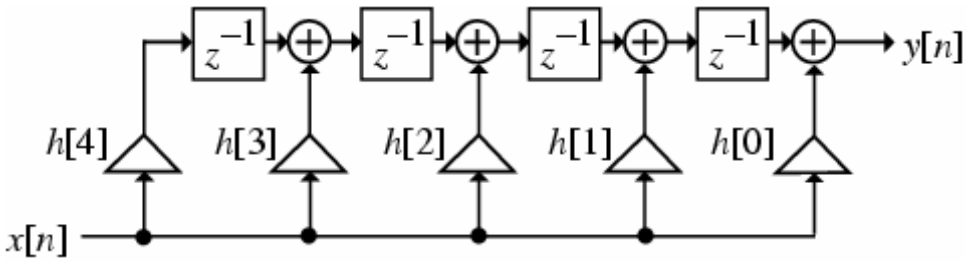
El sistema tiene M+1 coeficientes luego la carga computacional es:

- Productos: M+1
- Sumas: M

La estructura es canónica.

La forma directa I para los filtros FIR también se denomina LINEA DE RETARDO o FORMA TRANSVERSAL.

La forma traspuesta de esta estructura es:



Extraído de: Digital Signal Processing. A computer-based approach. S. K, Mitra

Tanto la forma original como la traspuesta son canónicas.

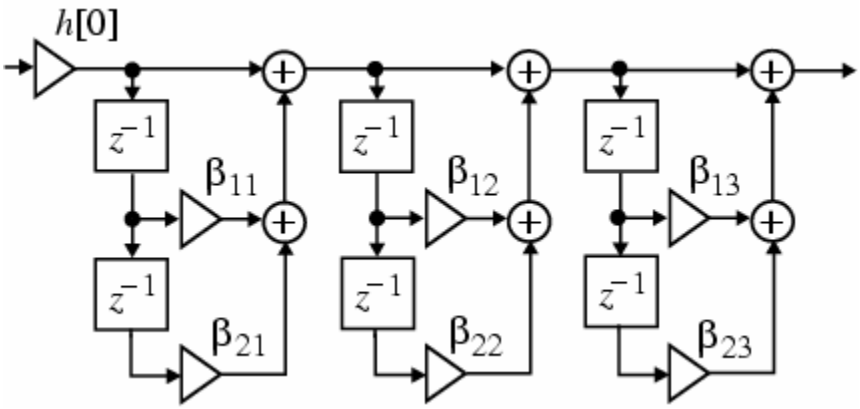
### 6.3.2.- Formas en Cascada.

Dada la función de transferencia de un filtro FIR, puede ser descompuesta en secciones de segundo orden de acuerdo con la expresión:

$H(z) = h[0]\prod_{k=1}^{K}(1+\beta_{1k}z^{-1}+\beta_{2k}z^{-2})$ siendo $K = \frac{M}{2}$ para M par y $K = \frac{M+1}{2}$ para M impar.

En este caso, en la última de las secciones $\beta_{2k} = 0$.

Por ejemplo para M=6, una estructura en cascada formada por bloques representados mediante la forma directa I vendría dada por:



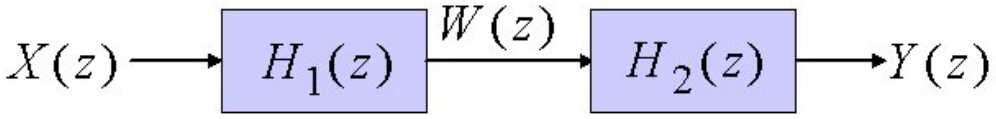

Extraído de: Digital Signal Processing. A computer-based approach. S. K, Mitra

## *6.4.- Estructuras Básicas para sistemas IIR.*

A partir de la ecuación en diferencias de un filtro IIR lo primero que observamos es que ésta está definida de forma recursiva ( $y(n) = \sum_{k=0}^{M} b_k \cdot x(n-k) - \sum_{k=1}^{N} a_k \cdot y(n-k)$ ). Dicha recursividad se reflejará en el diagrama de bloques como una realimentación de la salida del sistema hacia elementos previos.

### 6.4.1.- Formas directas

Dada la función de transferencia de un filtro IIR, consideremos expresado como una descomposición en cascada de un sistema todo ceros y otro todo polos.

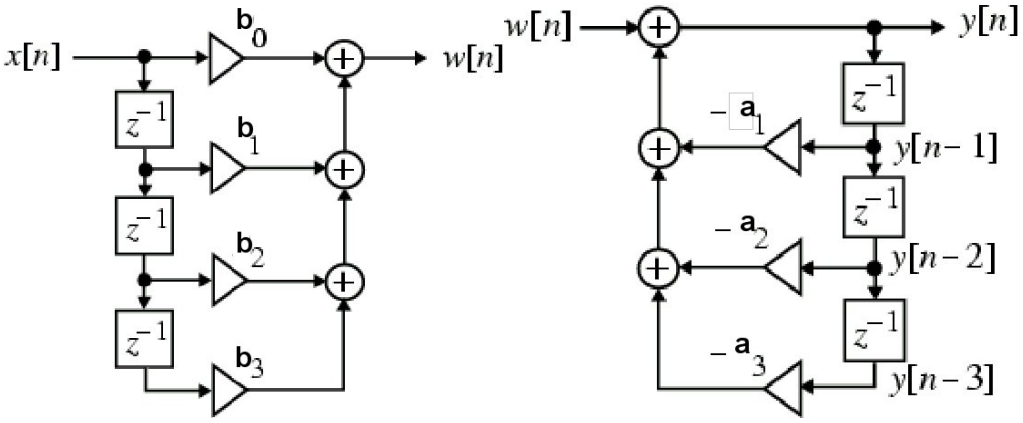



$$H_1(z) = \frac{W(z)}{X(z)} = P(z) = \sum_{k=0}^{M} b_k \cdot z^{-k}$$

$$H_2(z) = \frac{Y(z)}{W(z)} = \frac{1}{D(z)} = \frac{1}{1 + \sum_{k=1}^{M} a_k \cdot z^{-k}}$$

Las siguientes figuras muestran la implementación de $H_1(z)$ y $H_2(z)$ para N=M=3,

**Forma directa I**




Extraído de: Digital Signal Processing. A computer-based approach. S. K, Mitra

Las ecuaciones en diferencias son:

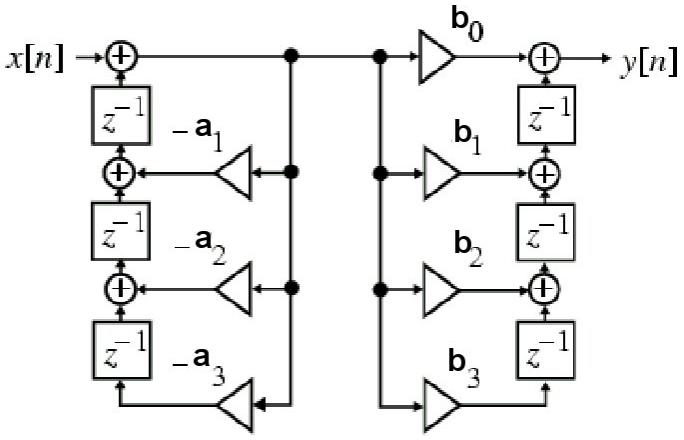
$$w[n] = b_0 x[n] + b_1 x[n-1] + b_2 x[n-2] + b_3 x[n-3]$$

$$y[n] = w[n] - a_1 y[n-1] - a_2 y[n-2] - a_3 y[n-3]$$

La conexión en cascada de ambos diagramas, como se muestra en la figura anterior se denomina FORMA DIRECTA I. Esta estructura no es canónica ya que el sistema es de

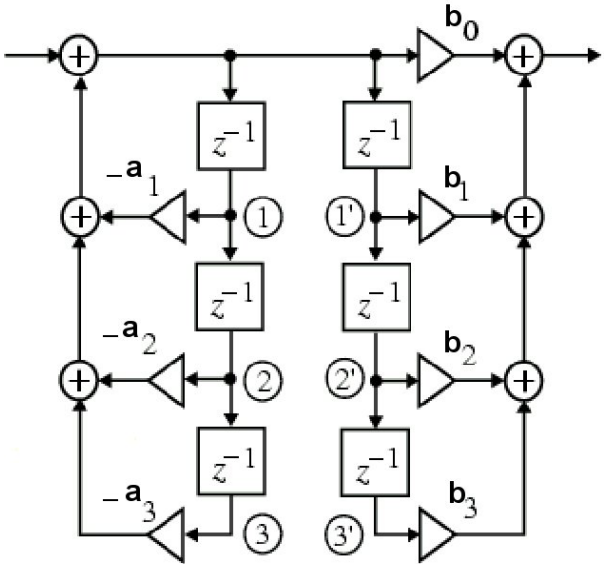
orden=max(N,M)=3 y el número de retardos es 6. La versión traspuesta de la forma directa I se muestra en la siguiente figura.

**Forma directa I. Traspuesta**




Extraído de: Digital Signal Processing. A computer-based approach. S. K, Mitra

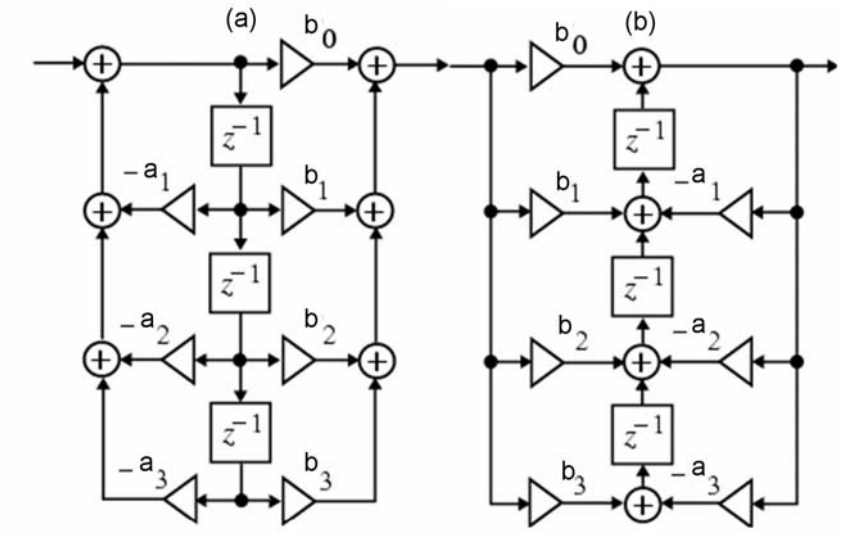
Por otra parte, teniendo en cuenta que la conexión en cascada es conmutativa podemos intercambiar la conexión dando lugar a una estructura equivalente que tiene la particularidad que podemos eliminar una de las ramas centrales ya que en ambas tenemos las mismas señales en los nodos 1--> 1' 2--> 2' y 3--> 3' tal como se indica en la figura.




Extraído de: Digital Signal Processing. A computer-based approach. S. K, Mitra

En la siguiente figura mostramos la estructura resultante, llamada FORMA DIRECTA II (a), y su versión traspuesta (b).

Ambas estructuras son CANÓNICAS. El número de productos es el mismo en ambas, y si consideramos sumas de 2 sumandos, el número de sumas también coincide (Téngase en cuenta que en la rama central hay 2 sumas de 3 sumando cada una.)




Extraído de: Digital Signal Processing. A computer-based approach. S. K, Mitra

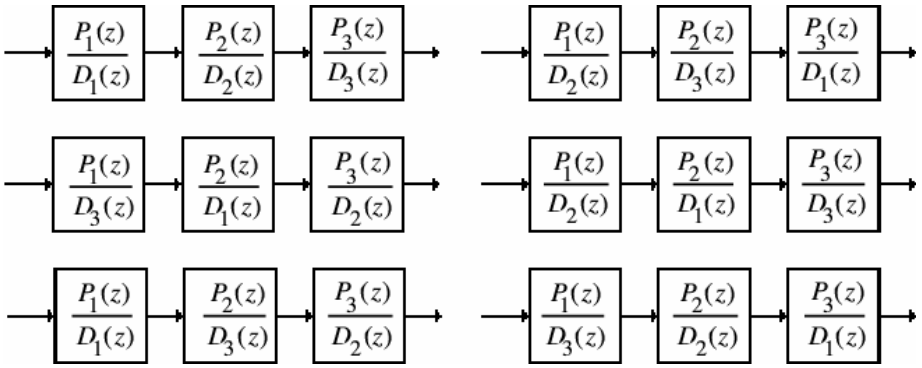
### 6.4.2.- Descomposiciones en cascada IIR.

Expresando el numerador y denominador de la función de transferencia como un producto de polinomios de menor orden (en general de orden 1 para raíces reales, y orden 2 para raíces complejas) un filtro digital puede expresarse como una cascada de secciones de menor orden.

Por ejemplo consideremos un sistema factorizado de la forma:

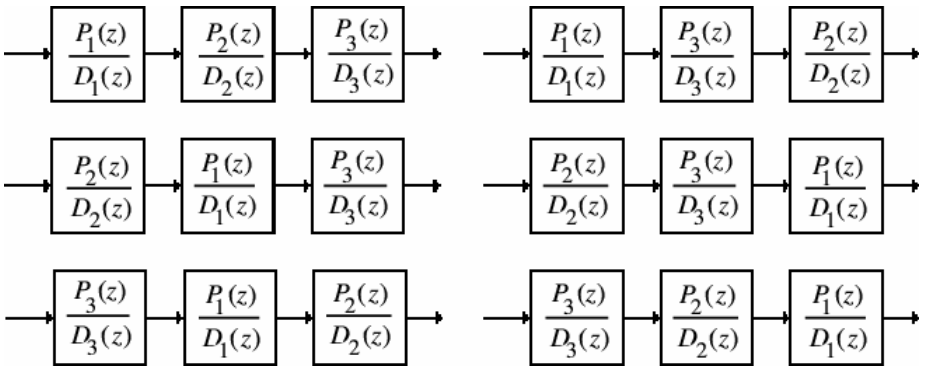
$$H(z)=\frac{P(z)}{D(z)}=\frac{P_1(z)P_2(z)P_3(z)}{D_1(z)D_2(z)D_3(z)}$$

A partir de esta descomposición tenemos, por una parte, 6 formas distintas de agrupar los ceros y polos tal como se indica:




Extraído de: Digital Signal Processing. A computer-based approach. S. K, Mitra

y además para cada agrupación tenemos 6 formas distintas de ordenar las secciones:




Extraído de: Digital Signal Processing. A computer-based approach. S. K, Mitra

Tenemos un total de 36 combinaciones posibles de conexión en cascada. Cada una de ellas tiene un comportamiento distinto cuando se trabaja con aritmética de coma fija.

En general H(z) es factorizada en etapas de primer y segundo orden de la forma siguiente:

$$H(z) = b_0 \prod_k \left( \frac{1+\beta_{1k} z^{-1} + \beta_{2k} z^{-2}}{1+\alpha_{1k} z^{-1} + \alpha_{2k} z^{-2}} \right) \text{ con } \alpha_{2k} = \beta_{2k} = 0 \text{ para etapas de primer orden.}$$

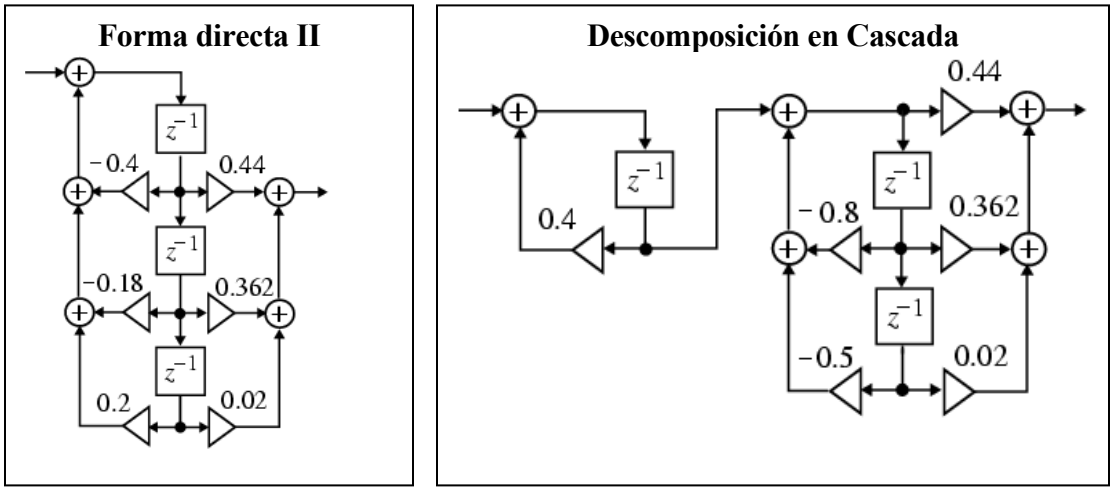
**Ejemplo:**

Representa la forma directa II y una descomposición es cascada para el sistema

$$H(z) = \frac{0.44 z^{-1} + 0.362 z^{-2} + 0.02 z^{-3}}{1 + 0.4 z^{-1} + 0.18 z^{-2} - 0.2 z^{-3}}$$

Si factorizamos la expresión anterior obtenemos: $H(z)=\left(\frac{0.44+0.362z^{-1}+0.02z^{-2}}{1+0.8z^{-1}+0.5z^{-2}}\right)\left(\frac{z^{-1}}{1-0.4z^{-1}}\right)$

Luego los diagramas de bloques obtenidos serán:




Extraído de: Digital Signal Processing. A computer-based approach. S. K, Mitra

Si analizamos las cargas computacionales, cada bloque de segundo orden requiere 4 Sumas y 5 Productos.

Las descomposiciones en cascada presentan un buen comportamiento cuanto se trabaja con registros finitos, además la utilización de bloques repetidos permite reutilizar el código. Cada etapa de segundo orden se suele implementar mediante la forma directa II para disminuir el número de retardos, dando así lugar a formas canónicas.

### 6.4.3.- Descomposicion en paralelo IIR.

Si la función de transferencia de un sistema IIR causal se descompone en fracciones simples y agrupamos en términos de primer y segundo orden, para tener coeficientes reales, llegamos a una expresión del tipo:

$$H(z)=\gamma_0+\sum_k\left(\frac{\gamma_{0k}+\gamma_{1k}z^{-1}}{1+\alpha_{1k}z^{-1}+\alpha_{2k}z^{-2}}\right) \text{ con } \alpha_{2k}=\gamma_{1k}=0 \text{ para polos reales.}$$

Posteriormente cada una de los términos suma de 1er y 2º orden se implementarán con alguna de las estructuras vistas, en general la forma directa II.

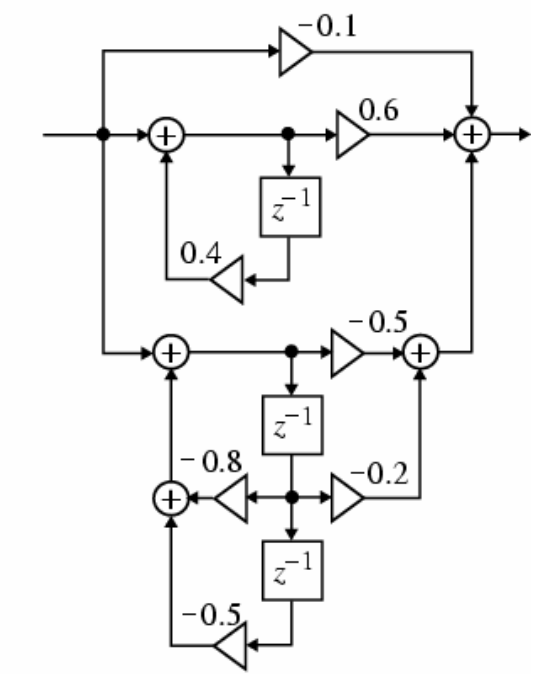
**Ejemplo:**

Realizar una descomposición en paralelo del sistema definido por la función de transferencia siguiente: $H(z)=\dfrac{0.44z^{-1}+0.362z^{-2}+0.02z^{-3}}{1+0.4z^{-1}+0.18z^{-2}-0.2z^{-3}}$

Si realizamos una descomposición en fracciones simples en potencias de $z^{-1}$ obtenemos:

$$H(z)=-0.1+\frac{0.6}{1-0.4z^{-1}}+\frac{-0.5-0.2z^{-1}}{1+0.8z^{-1}+0.5z^{-2}}$$

Cuyo diagrama de bloques, utilizando la forma directa II, es el siguiente:




Extraído de: Digital Signal Processing. A computer-based approach. S. K, Mitra

### 6.4.4.- Funciones de Matlab relacionadas.

Descomposición en paralelo (Fracciones simples): [R,P,K] = *residuez*(B,A)

[B,A] = *residuez*(R,P,K)

| | |
|---|---|
| Descomposición en Cascada: | [SOS,G] = *zp2tf*(Z,P,K) |
| | [SOS,G] = *tf2sos*(B,A) |
| | [Z,P,K] = *sos2zp*(SOS,G) |
| | [B,A] = *sos2tf*(SOS,G) |

**Funciones de filtrado:**

| | |
|---|---|
| Forma Directa II traspuesta: | Y=*filter*(B,A,x); |

**Funciones adicionales:**

| | |
|---|---|
| Raíces de un polinomio: | *roots*(C) |
| Cálculo de un polinomio a partir de sus raíces: | *poly*(V) |

### 6.4.5.- Implementación de la realización de un sistema digital a partir de su estructura.

Sabemos que Matlab tiene funciones para implementar eficientemente las estructuras analizadas a partir de su función de transferencia, sin embargo, podemos implementar cualquier estructura a partir de las ecuaciones en diferencias obtenidas de su diagrama de bloques. Cada diagrama de bloques genera unas ecuaciones en diferencias diferentes con un distinto número de operaciones suma y producto. Veamos un ejemplo y su implementación en Matlab mediante bucles.

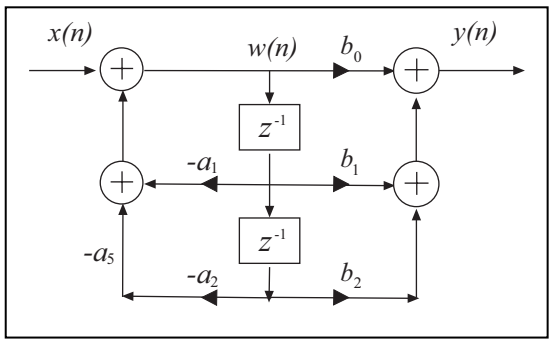
Consideremos un sistema de segundo orden implementado mediante la forma directa II y la forma directa II traspuesta.

Para ejecutar los programas que se indican es necesario dar valores a los coeficientes de los filtros y definir la variable de entrada *x(n)* que se desea filtrar. Por ejemplo

```
x=randn(1,1000);
b0=1; b1=2; b2=0.5;
```

```
a1=0.5;a2=0.75;
```

**Forma directa II.**



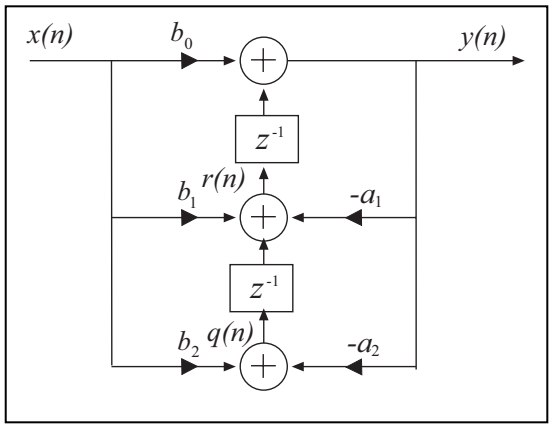

Ecuaciones :

$$y(n) = b_0 \cdot w(n) + b_1 \cdot w(n-1) + b_2 \cdot w(n-2)$$
$$w(n) = x(n) - a_1 \cdot w(n-1) - a_2 \cdot w(n-2)$$

Código Matlab de la implementación:

```
%Implementacion de la forma directa II
%Necesitamos almacenar 3 valores de w(n)
%x(n) sera la sencuencia de entrada que queremos filtrar e y(n) la salida
% Los coeficientes del filtro seran b0,b1,b2, a1, a2
w=zeros(1,3);
%Definimos un vector de tantos elementos como retardos tiene la variable
%intermedia
% w(n) --> w(3)
% w(n-1) --> w(2)
% w(n-2) --> w(1)
%Iteracion
N=length(x);  %Numero de elementos de la secuencia
for n=1:N
%Hemos de tener en cuenta el orden para el calculo de las ecuaciones
w(3)=x(n)-a1*w(2)-a2*w(1);
y(n)=b0*w(3)  +b1*w(2) +b2*w(1);
%Ahora cambiamos los valores de la variable w para la iteracion
%siguiente
w(1)=w(2);
w(2)=w(3);
end
```

**Forma directa II traspuesta**




Ecuaciones :

$$y(n) = x(n) + r(n-1)$$
$$r(n) = b_0 \cdot x(n) - a_1 \cdot y(n) + q(n-1)$$
$$q(n) = b_1 \cdot x(n) - a_2 \cdot y(n)$$

Código Matlab de la implementación:

```
%Implementacion de la forma directa II traspuesta
%Necesitamos almacenar 2 valores de r(n) y q(n)
%x(n) sera la sencuencia de entrada que queremos filtrar e y(n) la salida
% Los coeficientes del filtro seran b0,b1,b2, a1, a2
q=zeros(1,2); r=zeros(1,2);
%Definimos un vector de tantos elementos como retardos tiene la variable
%intermedia
% q(n) --> q(2)
% q(n-1) --> q(1)
% r(n) --> r(2)
% r(n-1) --> r(1)
%Iteracion
N=length(x); %Numero de elementos de la secuencia
for n=1:N
%Hemos de tener en cuenta el orden para el calculo de las ecuaciones
y(n)=x(n) +r(1);
r(2)=b0*x(n)-a1*y(n) +q(1);
q(2)=b1*x(n)-a2*y(n);
q(1)=q(2);  %Movemos valores para la siguiente iteracion
r(1)=r(2);
end
```

Observamos que aunque se trata del mismo filtro de segundo orden, dependiendo de la estructura elegida las ecuaciones para su implementación son distintas. Cuando trabajamos con precisión total (P. Ej. Matlab) los resultados son idénticos con ambos programas; sin embargo, cuando se trabaja en un micro o un DSP de coma fija los resultados son diferentes ya que el número de multiplicaciones y sumas es distinto.