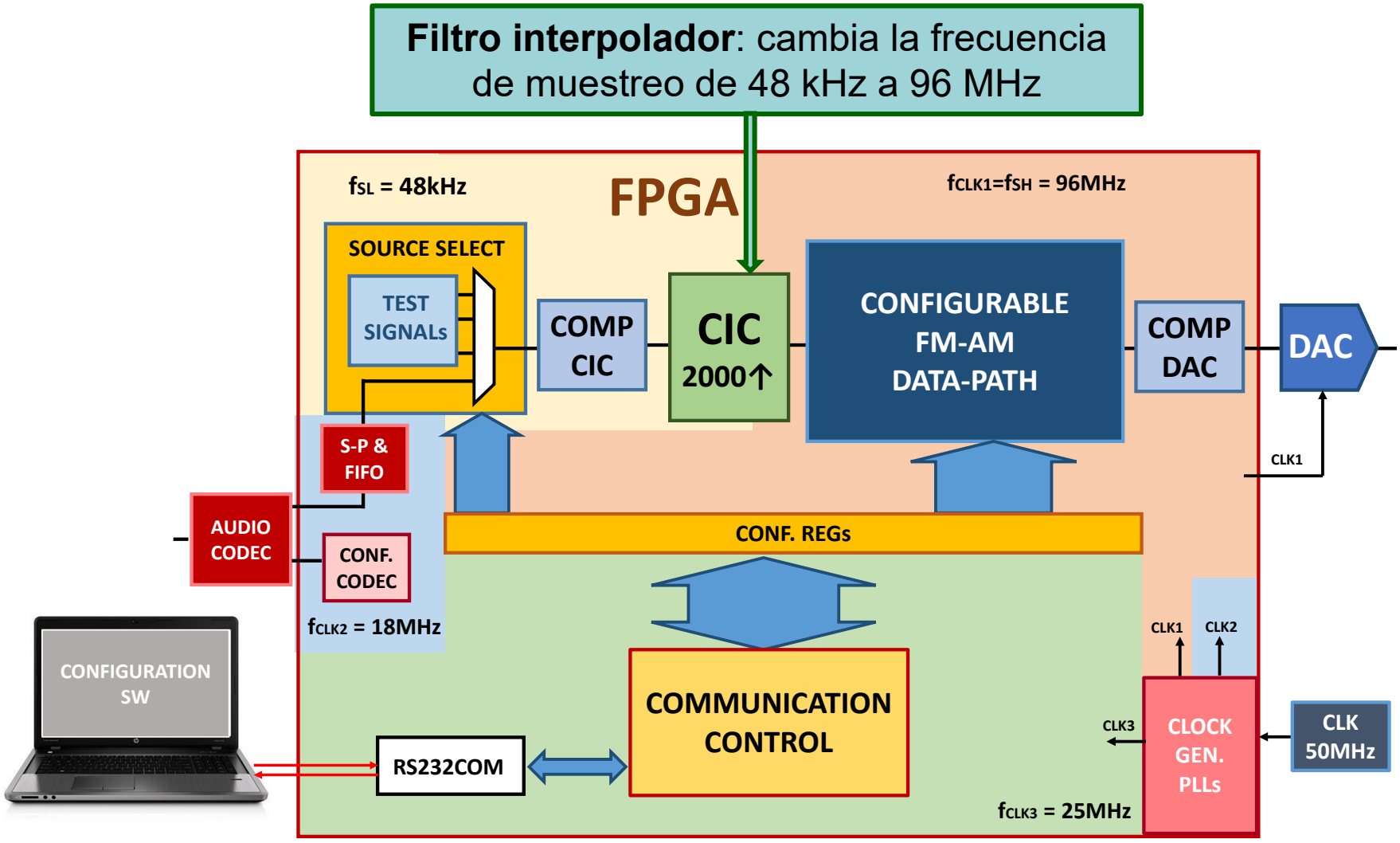


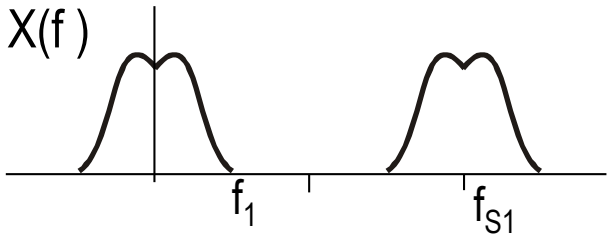
E1: Sintetizador de frecuencias (DDS)
E2: Ruta de datos AM/FM configurable
E3: Filtro interpolador CIC
E4: Filtros compensadores CIC y DAC
E5: Comunicación con PC, control
E6: Completar sistema

Procesado Multitasa:

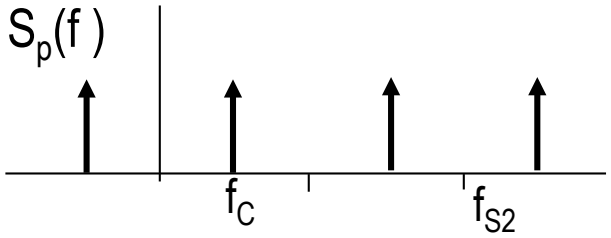
$$s_{IF}(t) = x(t) \cdot \cos(2\pi f_c t)$$

Ej: Modulación digital

$x(m) \rightarrow f_{S1} > 2f_1$



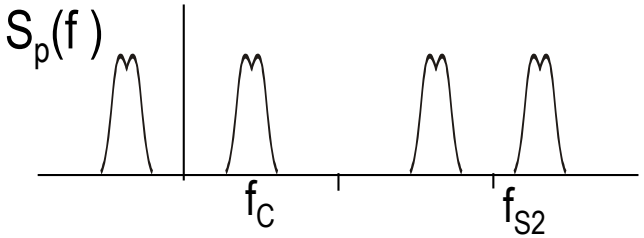
$s_p(n) = \cos(\omega_c n) \rightarrow f_{S2} > 2f_C$

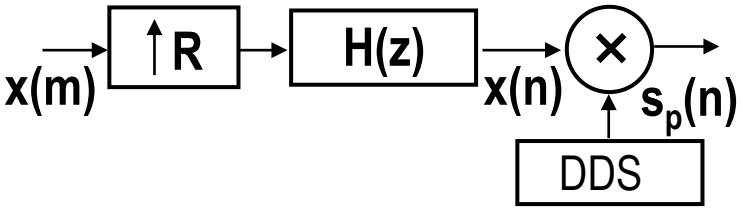


$f_{S2} >> f_{S1}$

$x(n) \rightarrow f_{S2}$

$s_{IF}(n) = x(n) \cdot \cos(\omega_c n)$

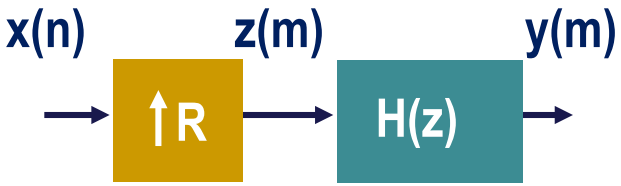




$s_{IF}(n) = x(n) \cdot \cos(\omega_c n)$

Arquitecturas FIR Multitasa

Interpolación

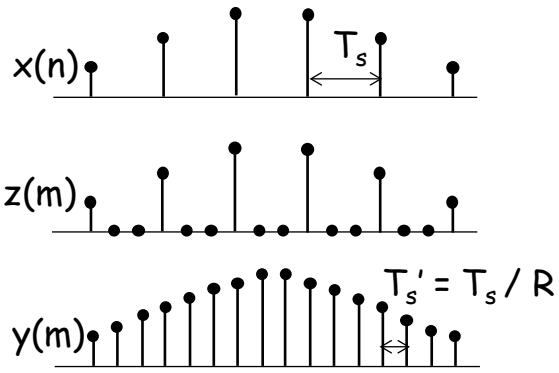


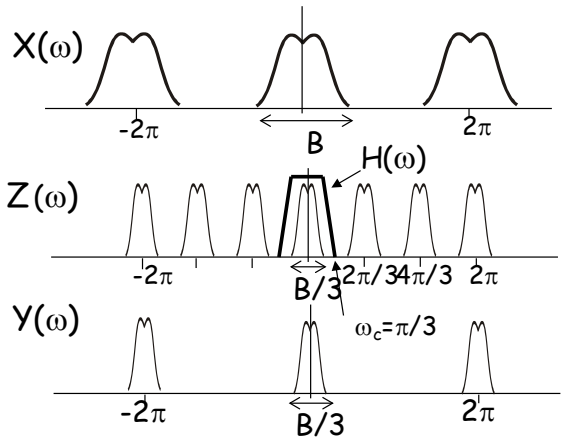
Ganancia=R;

$\omega_c=\pi/R$

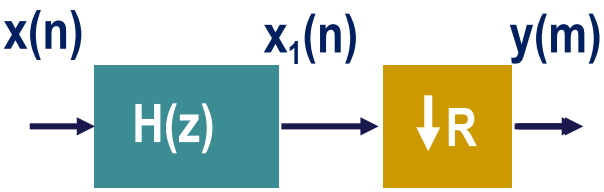
$$z(m) = \begin{cases} x(m/R) & \text{si } m=0, \pm R, \pm 2R,... \\ 0 & \text{otros} \end{cases}$$

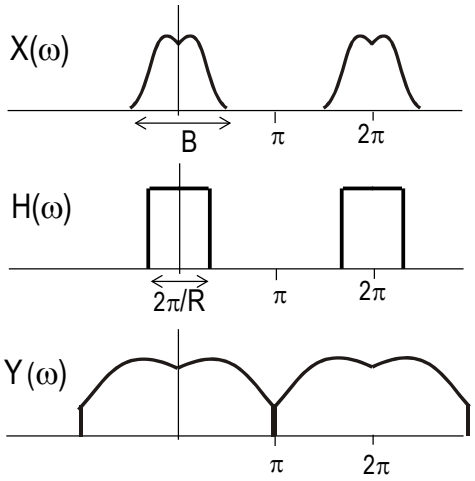
Ej. R=3





Diezmado

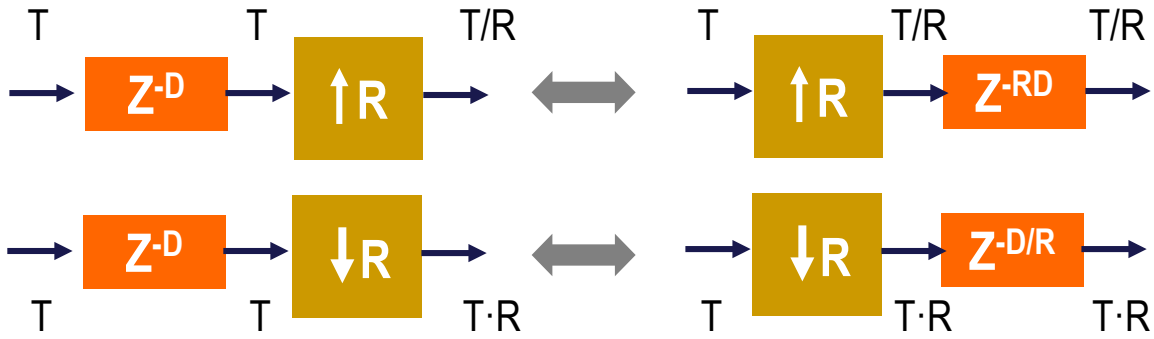




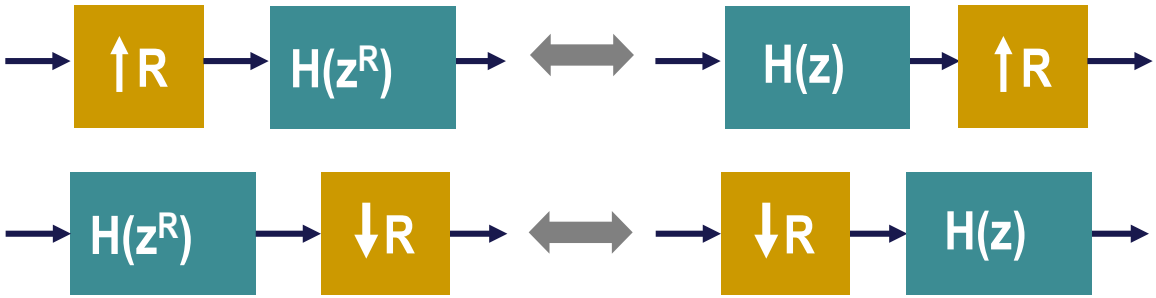
Arquitecturas FIR Multitasa

Identidades de Noble

Se utilizan para pasar los retardos de la entrada a la salida de un diezmador/interpolador



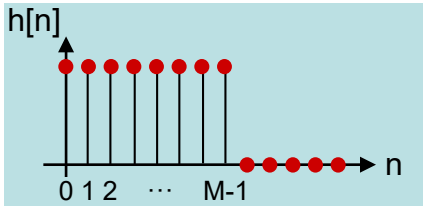
Son también útiles para transformar los filtros



Esta transformación nos lleva a usar filtros más sencillos

Filtro CIC (*Cascade Integrator-Comb*)

- Útil para filtros multitasa con factores grandes de interpolación o diezmado
- No utiliza multiplicadores $\Rightarrow$ coeficientes valen 1



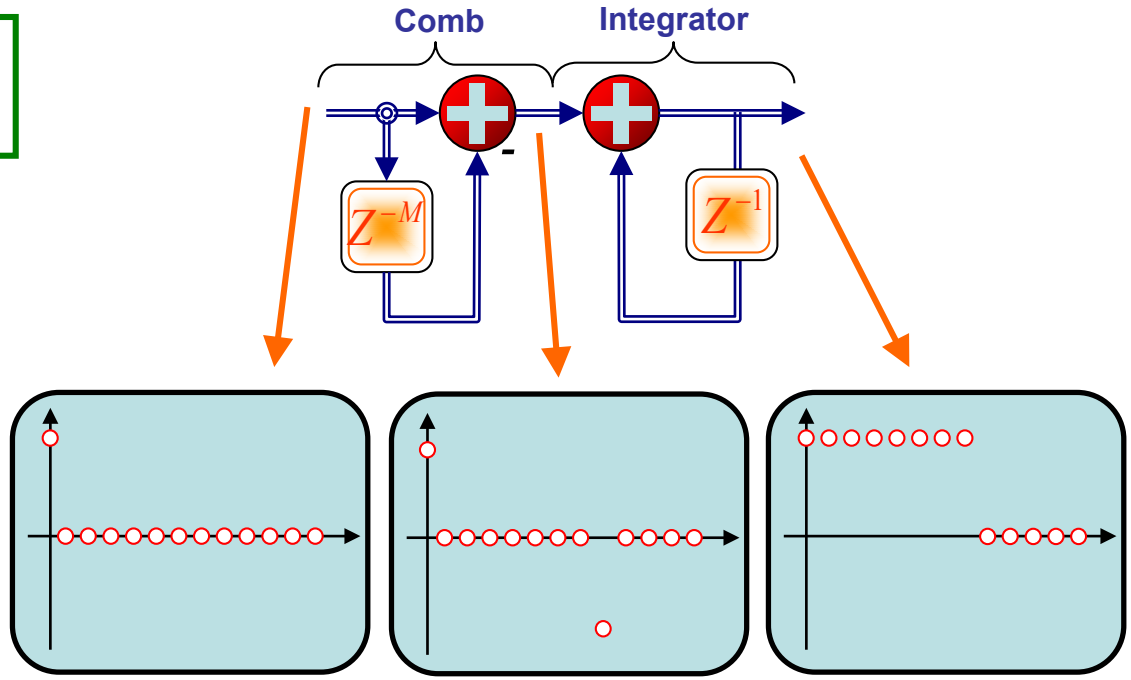
$$H_c(z) = 1 - z^{-M}$$

$$H_i(z) = \frac{1}{1 - z^{-1}}$$

$$H(z) = 1 + z^{-1} + \cdots + z^{-(M-1)}$$

$$H(z) = \frac{1 - z^{-M}}{1 - z^{-1}}$$

Orden del filtro peine = M



Filtro CIC

La respuesta en frecuencia pobre

$\Rightarrow$ Es necesario conectar varias etapas en cascada para mejorarla

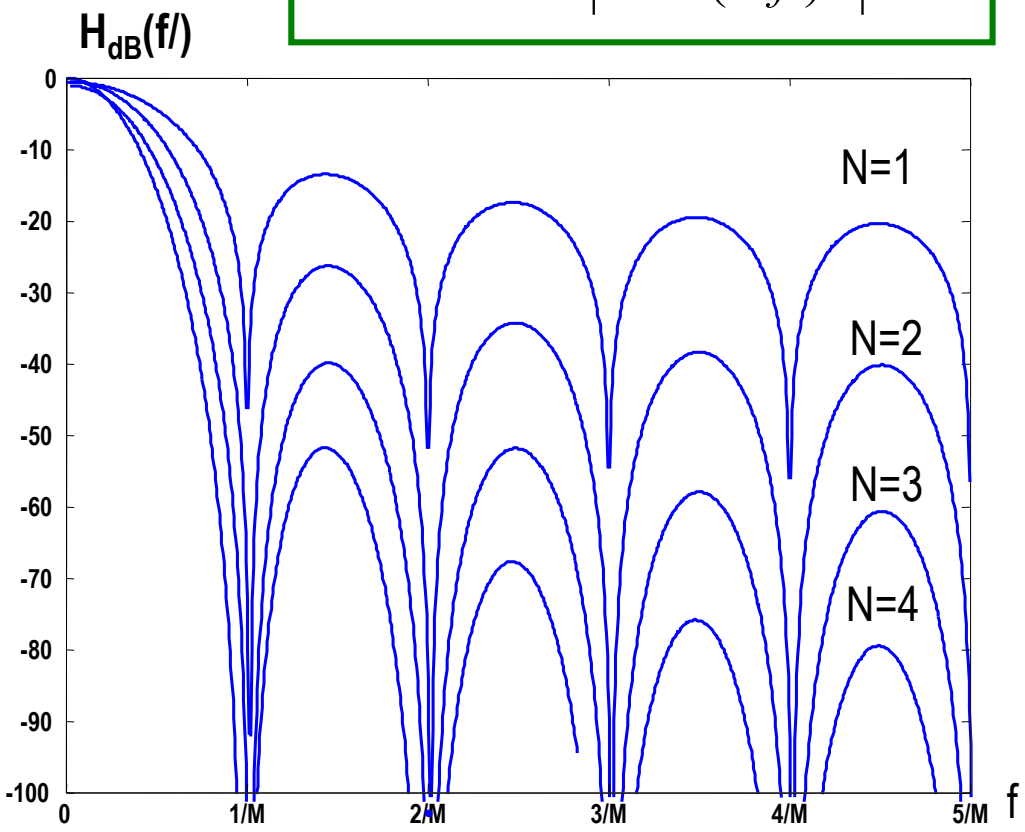
$$H(z) = \left(\frac{1 - z^{-M}}{1 - z^{-1}} \right)^N$$

$$\left| H(f) \right| = \left| \frac{\text{sen}(\pi M f)}{\text{sen}(\pi f)} \right|^N$$

N : nº de etapas
M: Orden del filtro peine

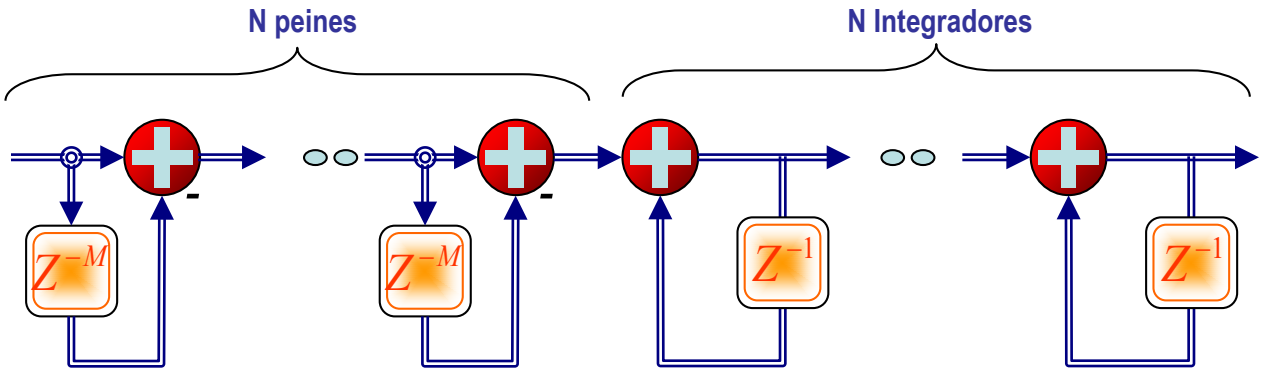
Ganancia = M^N

Ceros en f = k / M k=1,2,...



Filtro CIC de N estados

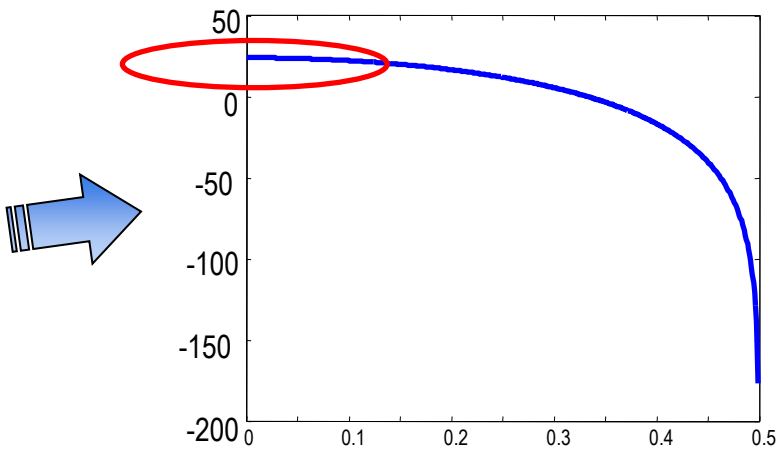
$$H(z) = \left[\frac{1 - z^{-M}}{1 - z^{-1}} \right]^N$$



Matlab

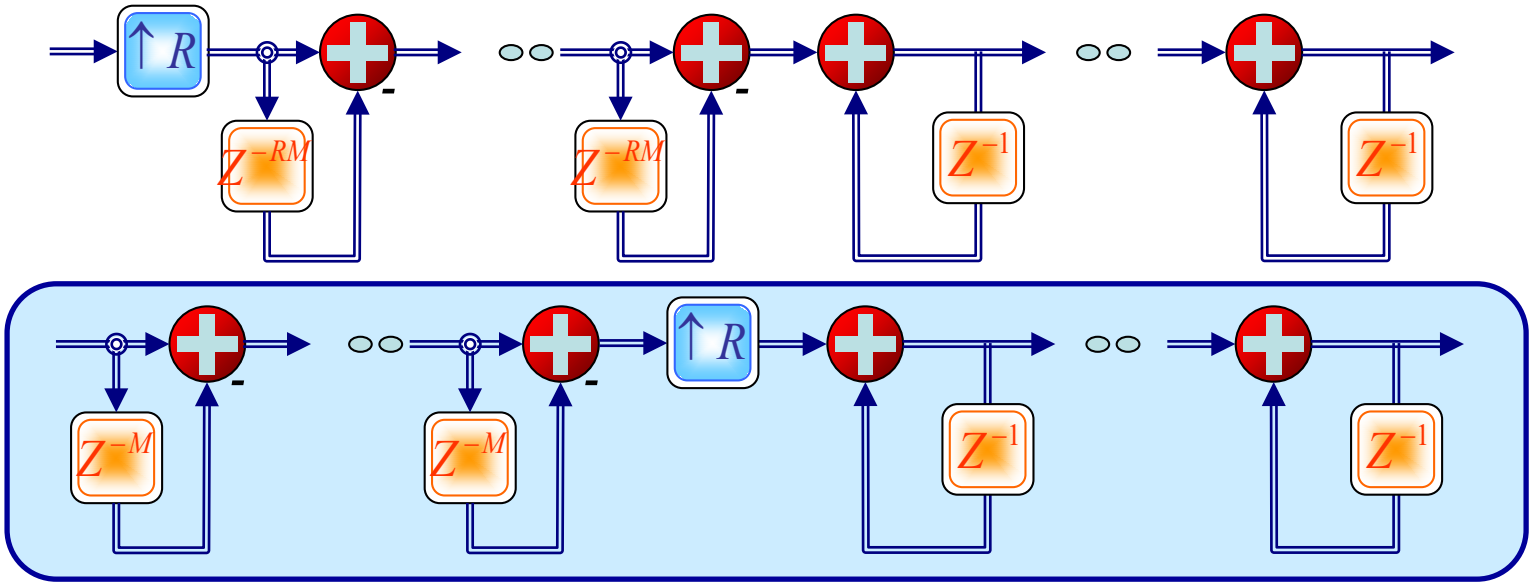
```
N=4;M=2; w=[0:0.05:1];
h1=ones(M,1);
hcic=1;
for i=1:N
        hcic=conv(h1,hcic);
end;
[h,w]=freqz(hcic,1);
plot(w/(2*pi),20*log10(abs(h)));
```

GAIN=M^N=16=> 20*log10(16)=24.1 dB



CIC Interpolador

$$H(z) = \left[\frac{1 - z^{-RM}}{1 - z^{-1}} \right]^N$$

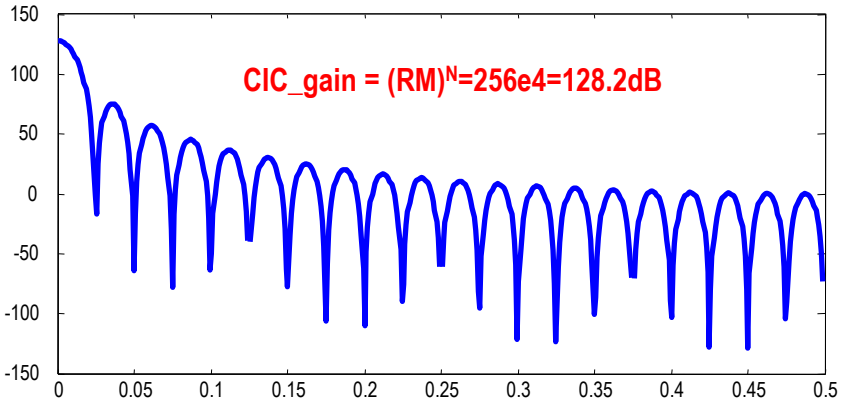


Matlab

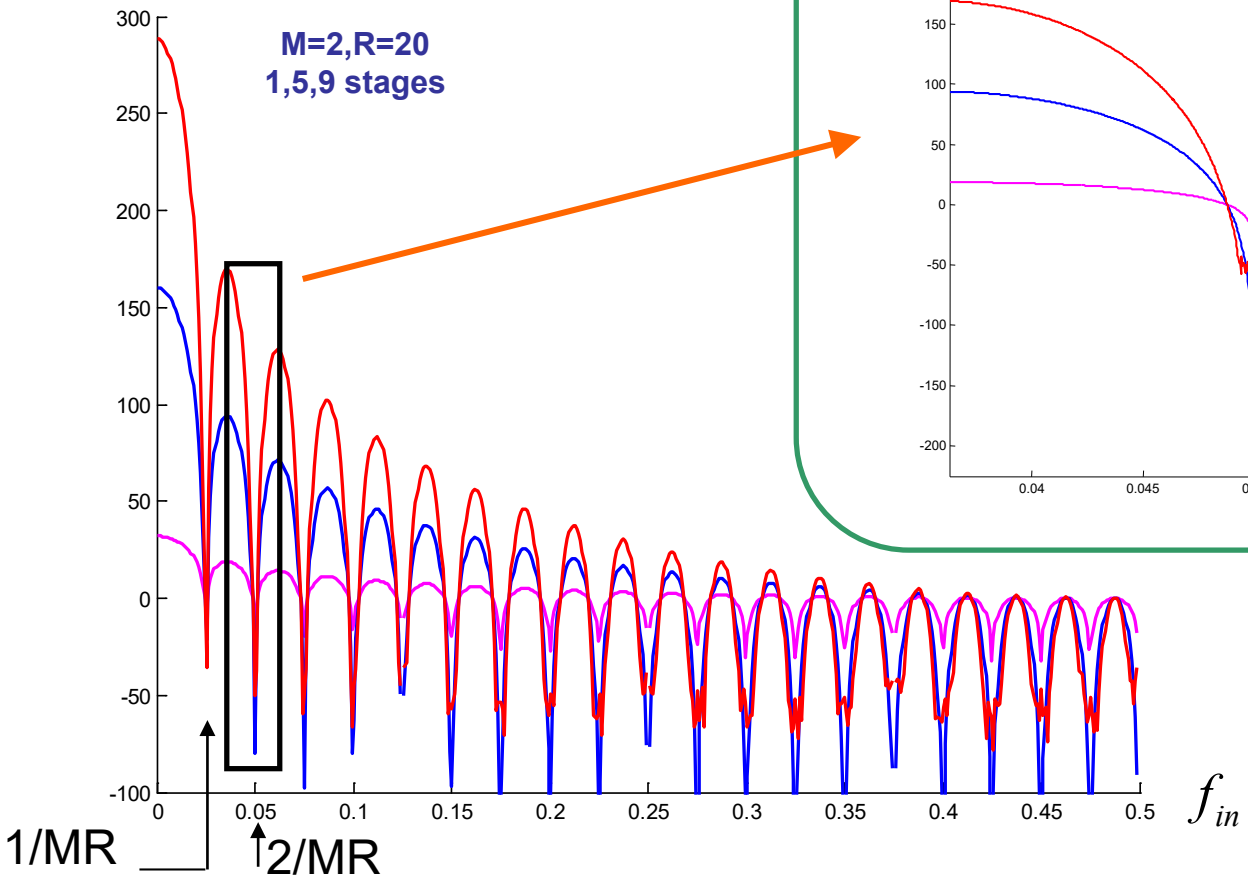
```
N=4;M=2;R=20
h1=ones(R*M,1);
hcic=1;
for i=1:N
        hcic=conv(h1,hcic);
end;
[h,w]=freqz(hcic,1);
plot(w/(2*pi),20*log10(abs(h)));
```

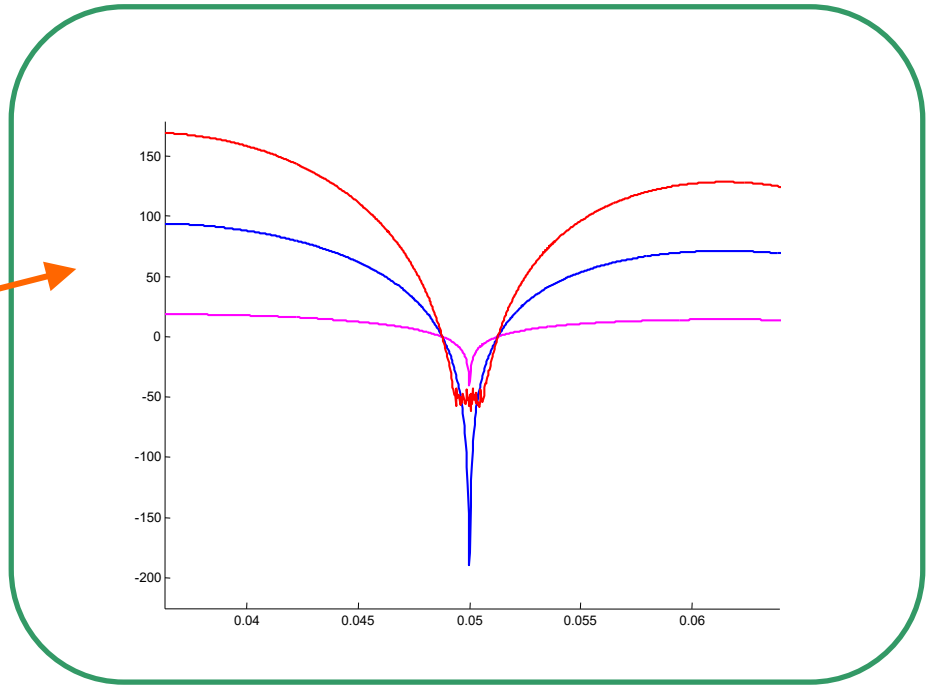
Cada etapa necesita un crecimiento de log2(RM) bits

Interpolator gain=1/R La ganancia total será = $(R*M)^N$ *1/R



Interpolador CIC





Características de la respuesta en frecuencia del CIC

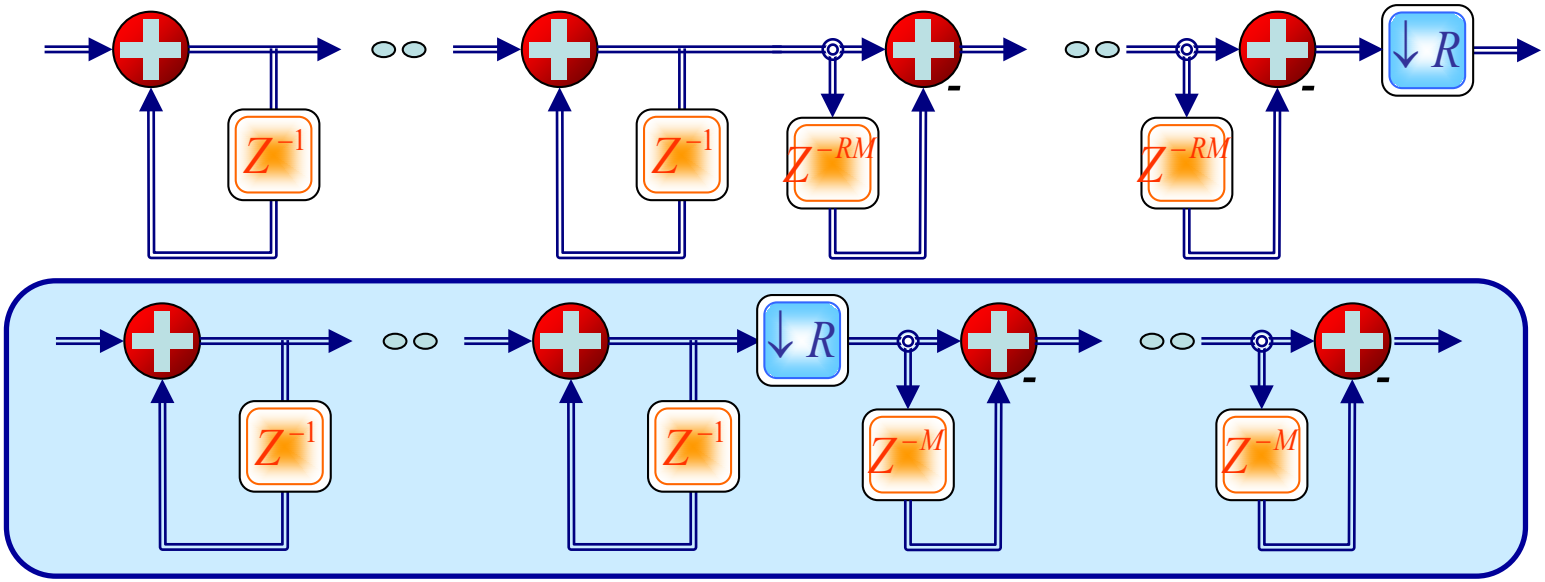
Los ceros se sitúan en múltiplos de 1/(RM)

Respuesta no-ideal en la banda de paso

→ podremos corregirla añadiendo un filtro compensador

CIC Diezmador

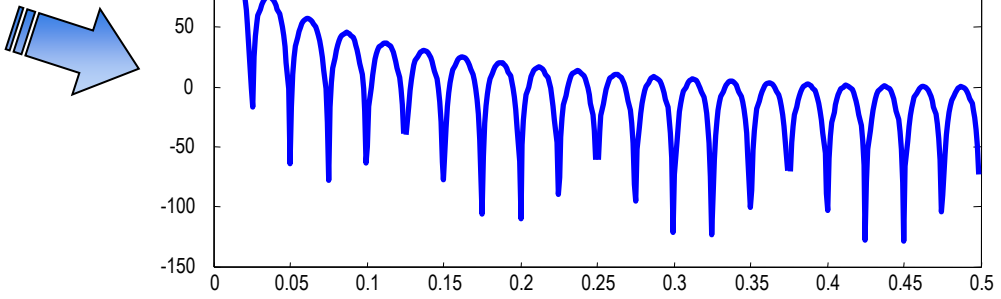
$$H(z) = \left[\frac{1 - z^{-RM}}{1 - z^{-1}} \right]^N$$



Matlab

```
N=4;M=2;R=20
h1=ones(R*M,1);
hcic=1;
for i=1:N
        hcic=conv(h1,hcic);
end;
[h,w]=freqz(hcic,1);
plot(w/(2*pi),20*log10(abs(h)));
```

Cada etapa necesita un crecimiento de log2(RM) bits



Verilog: Módulo parametrizado

Parámetro :

- Constante que local en la definición del módulo

```
module COMB
#(parameter Wd=16)
(
    // ports f(M,N)
);
    // behavior f(M,N)
endmodule
```

```
// instances:

COMB #(.Wd(16)) U1 ( ,  , );

COMP #(.Wd(14)) U2 ( , , );
```

Módulo parametrizado:

- Módulo escrito de tal manera que la anchura de los puertos y su comportamiento cambia con los valores de los parámetros (nos servirá para implementar el mismo circuito con diferentes parámetros)

Verilog: Generate

- ***Generate*** permite la creación de múltiples instancias de un módulo o de un grupo de asignaciones o procesos
- ***genvar*** declara la variable integer que sólo se utilizan en el generate.
- Declaración:

genvar *genvar_name, ... ;*

generate
generate_items
endgenerate