



UNIVERSITAT
POLITÈCNICA
DE VALÈNCIA

Arquitecturas Paralelas

Procesado Digital de la Señal en FPGA

Máster Universitario en Ingeniería de Sistemas Electrónicos

Objetivos

- Diseñar e implementar filtros FIR sobre FPGAs de Xilinx y Altera, usando **arquitecturas paralelas** aprovechando los recursos hardware que ofrecen estos dispositivos

Contenido

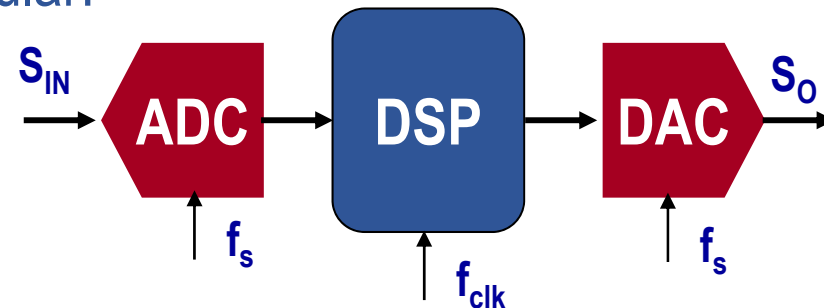
- Introducción a las arquitecturas HW
- Arquitecturas paralelas
- Cuantificación de los filtros FIR
- Segmentación y Retiming
- Procesado entrelazado
- Paralelización de algoritmos



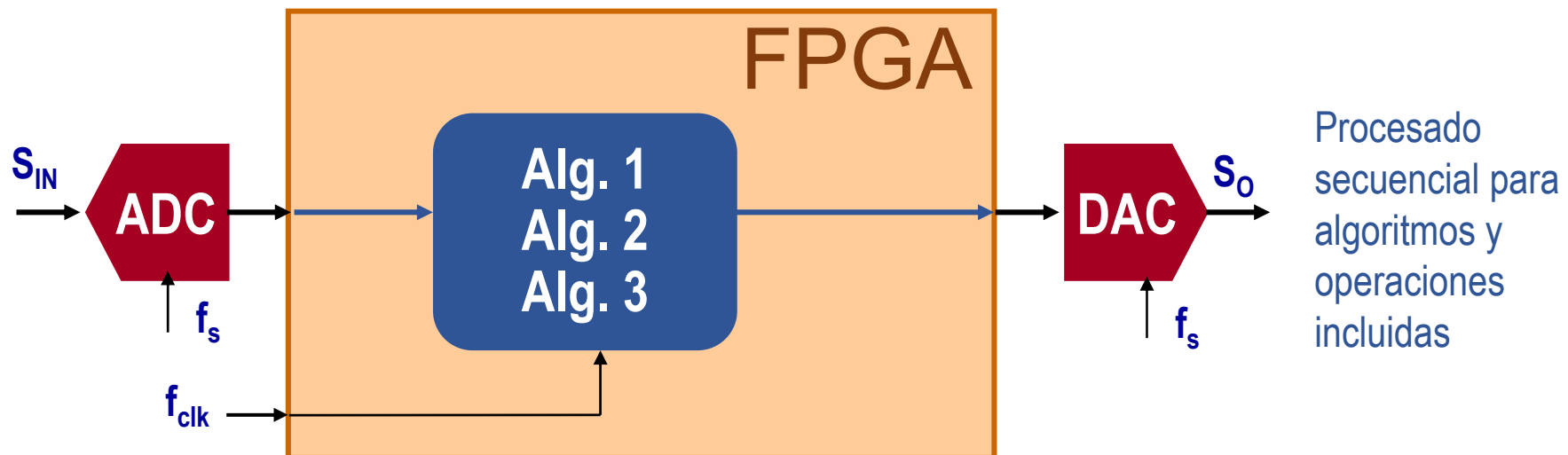
Arquitecturas HW

Ej: Un sistema DSP tiene que calcular:

- Alg.1 (N1 mults)
- Alg.2 (N2 mults)
- Alg.3 (N3 mults)



Caso 1: $f_{clk}/f_s > (N1+N2+N3)$



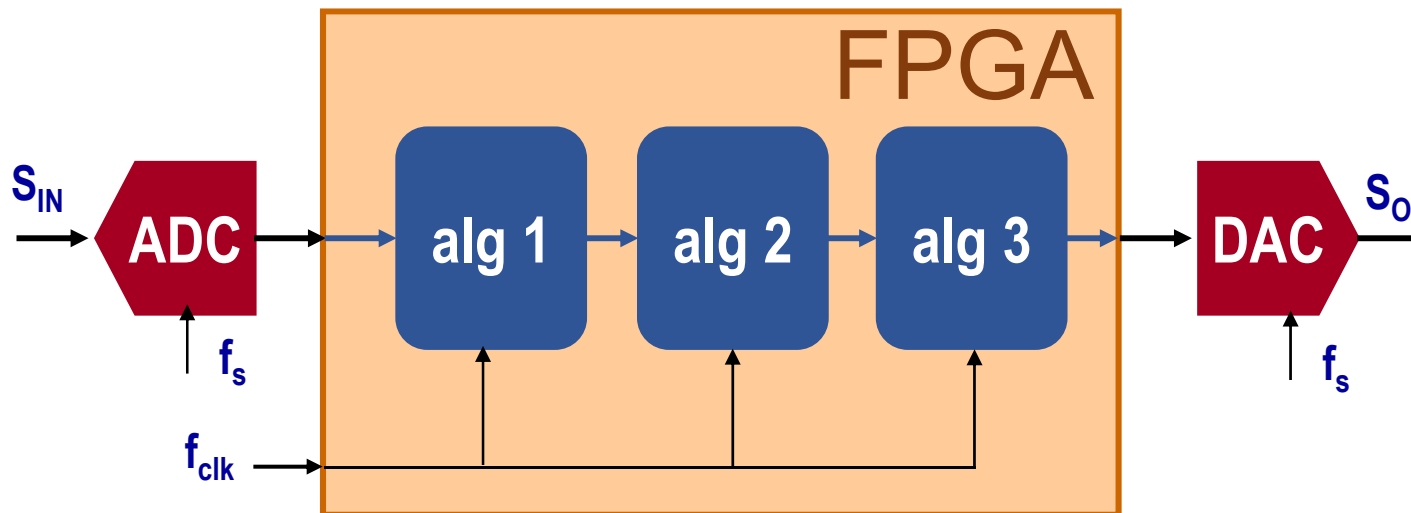
Arquitecturas HW

Caso 2: $(N1, N2, N3) < f_{clk}/f_s < (N1+N2+N3)$

- Se implementa por separado cada algoritmo
- Segmentación a nivel de bloque
- Arquitectura secuencial para cada algoritmo

Caso 3: $f_{clk}/f_s = 1$

- Se implementa por separado cada algoritmo
- Segmentación a nivel de bloque
- Segmentación interna de los algoritmos para conseguir la f_{clk}

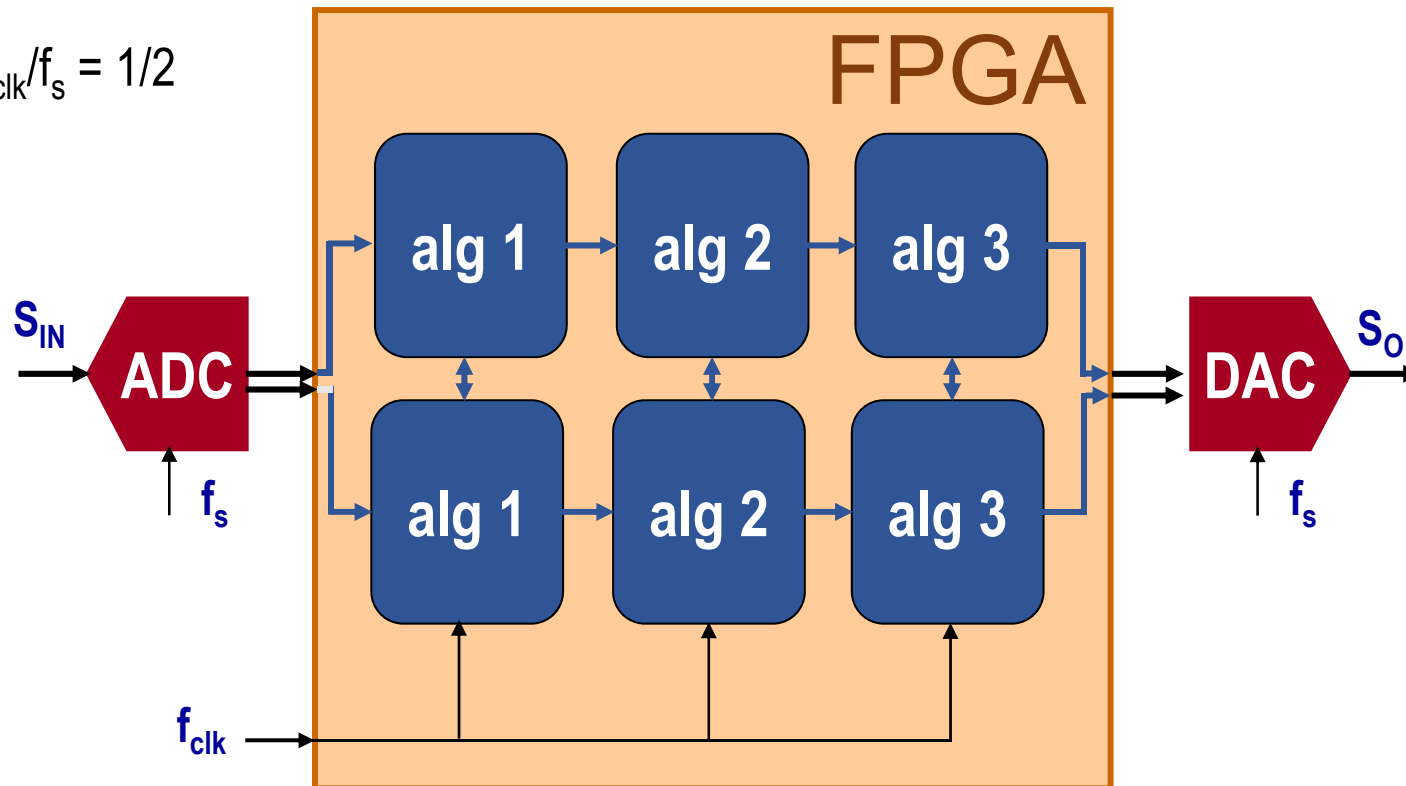


Arquitecturas HW

Caso 4: $f_{\text{clk}}/f_s < 1$

- Se implementa por separado cada algoritmo
- Segmentación a nivel de bloque
- Paralelizamos cada algoritmo y segmentamos para alcanzar f_{clk}

Ej: $f_{\text{clk}}/f_s = 1/2$



Del algoritmo a la arquitectura

$$y(n) = \sum_{k=0}^{M-1} h_k x(n-k)$$

ALGORITMO
 $y(0) = 0$
 for $k = 1$ to M do
 $y(k) = y(k-1) + h(k) * x(k)$
end
 $y = y(n)$

DSP
 Ex: $M=100$
 $\frac{1 \text{ GHz}}{100 \text{ ciclos}} = 10 \text{ MSPS}$

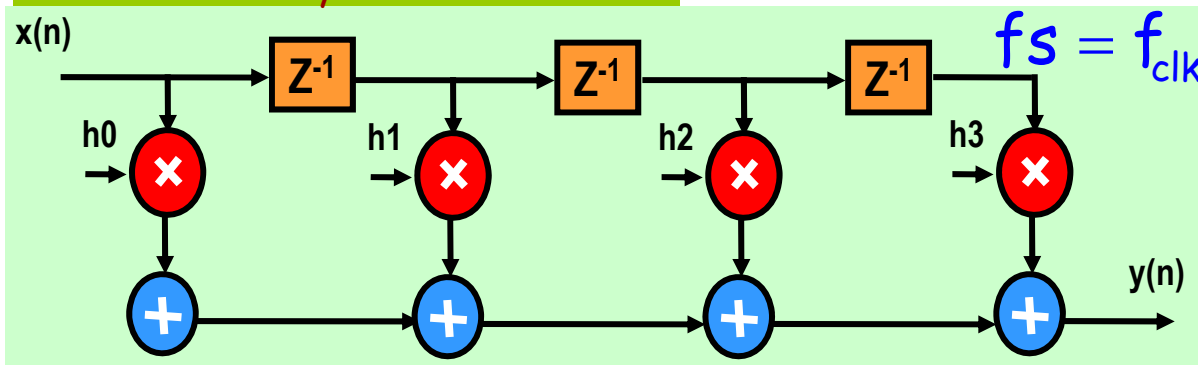
= FPGA

FPGA

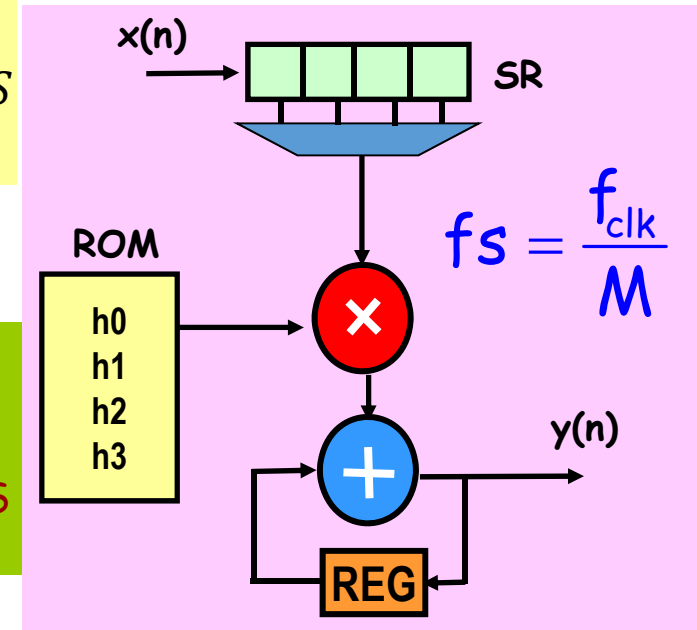
$\frac{400 \text{ MHz}}{100 \text{ cycles}} = 4 \text{ MSPS}$

Unfolded FIR \equiv Paralelo

FPGA $\frac{100 \text{ MHz}}{1 \text{ cycle}} = 100 \text{ MSPS}$



**Folded FIR
 \equiv Secuencial**



**Area-
 Throughput**

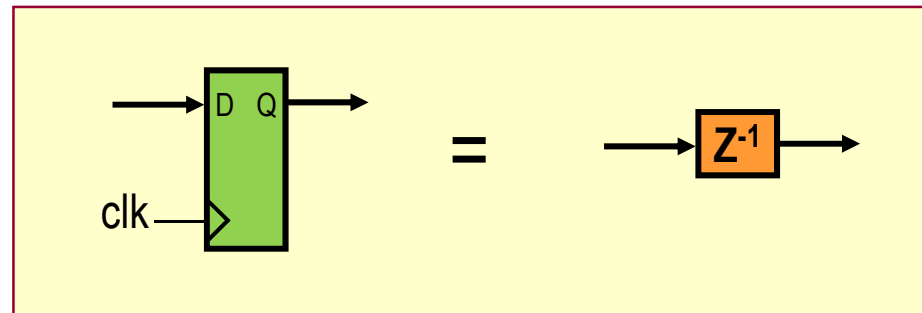
Arquitecturas Paralelas

⇒ Frecuencia de muestreo (f_s) = Frecuencia de reloj (f_{clk})

⇒ La arquitectura necesita tantos recursos como operaciones tiene el algoritmo

⇒ Se obtiene un resultado cada ciclo de reloj

⇒ Un registro equivale a un retardo de una muestra temporal



Arquitecturas paralelas

Modelo Matlab de un Filtro FIR

Filtro FIR de M-etapas

$$y(n) = \sum_{k=0}^{M-1} h_k x(n - k) \quad \leftarrow \text{1 clock cycle}$$

Modelo Matlab :

$$y(i) = x(i) * h(1) + x(i-1) * h(2) + \dots + x(i-(M-1)) * h(M-1);$$

En cada ciclo de reloj:

- Se multiplican M datos por M coeficientes
- Los resultados de las multiplicaciones se suman

Requisitos Hardware :

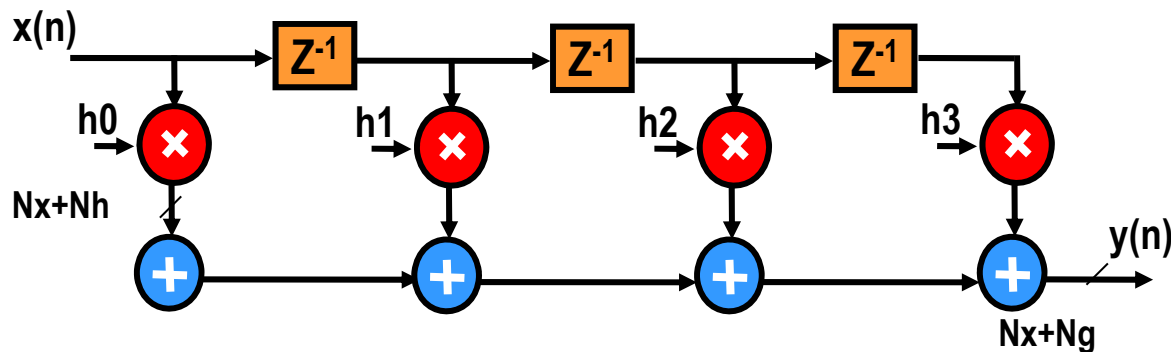
- M multiplicadores
- M-1 sumadores
- M-1 registros para los datos de entrada
- M registros para los coeficientes

Estructuras para implementar filtros FIR

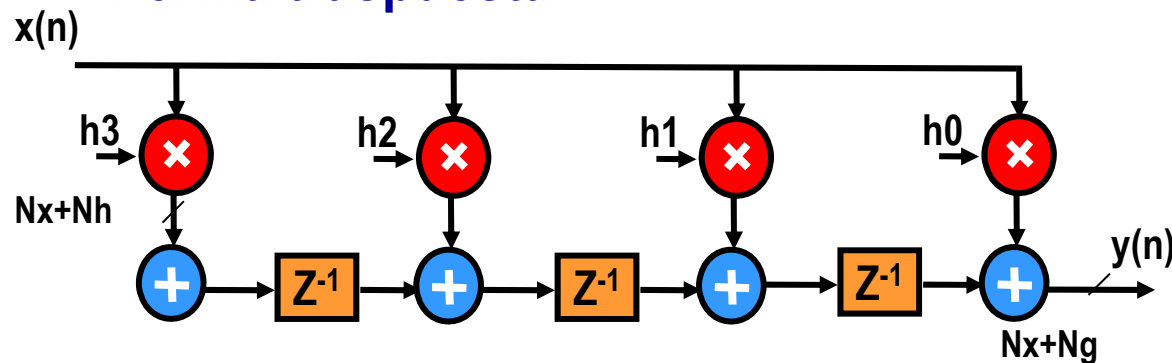
$$y(n) = \sum_{k=0}^3 h_k x(n-k)$$

¿Alguna diferencia?

Forma directa



Forma traspuesta



$X = X_e 2^{-x}$ (N_x bits)
 $H = H_e 2^{-h}$ (N_h bits)
 $P_j = X_i \cdot H_j$ ($N_x + N_g$ bits)
 $N_g = \log_2(G_e)$, $G_e = G \cdot 2^h$

$t_c = t_{\text{mult}} + (M-1)t_{\text{add}}$
 N_x -bit registers

$t_c = t_{\text{mult}} + t_{\text{add}}$
 $(N_x + N_g)$ -bit regist.
 Broadcast (\uparrow fan-out)

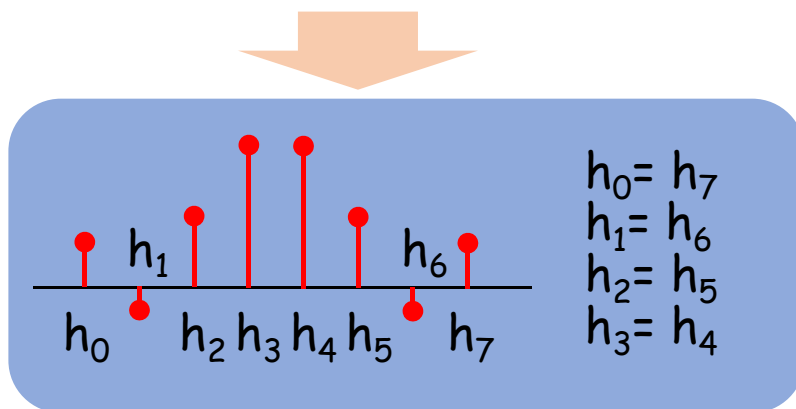
Aprovechando las propiedades de los algoritmos

Los filtros FIR de fase lineal poseen **coeficientes simétricos**

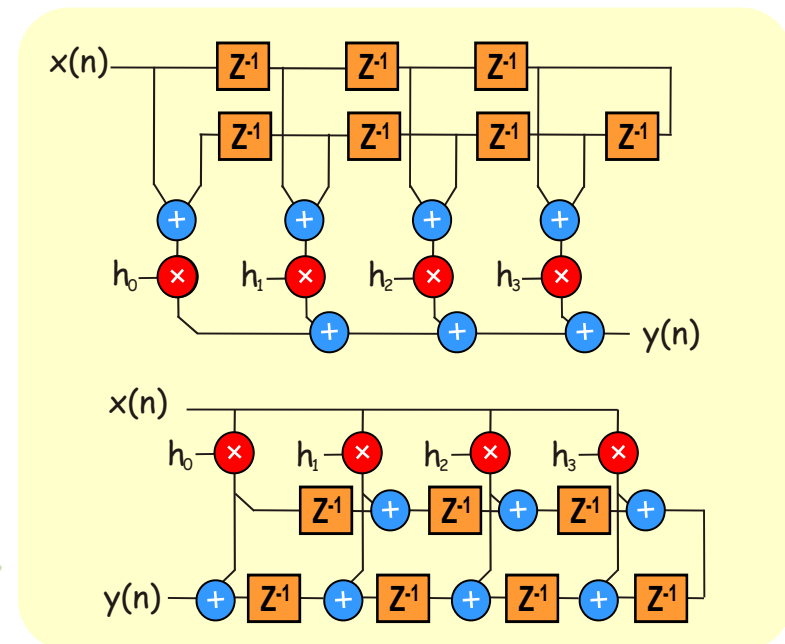
Se puede aprovechar para reducir el número de multiplicadores

FIR simétrico de N-etapas (Ej. N=8)

$$y(n) = h_0x(n) + h_1x(n-1) + h_2x(n-2) + h_3x(n-3) + h_4x(n-4) + h_5x(n-5) + h_6x(n-6) + h_7x(n-7)$$



$$y(n) = h_0[x(n) + x(n-7)] + h_1[x(n-1) + x(n-6)] + h_2[x(n-2) + x(n-5)] + h_3[x(n-3) + x(n-4)]$$



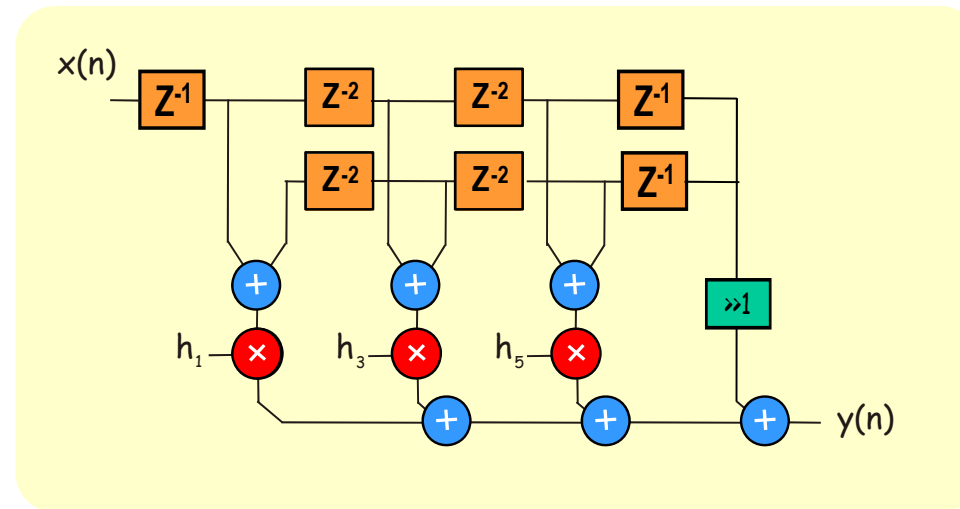
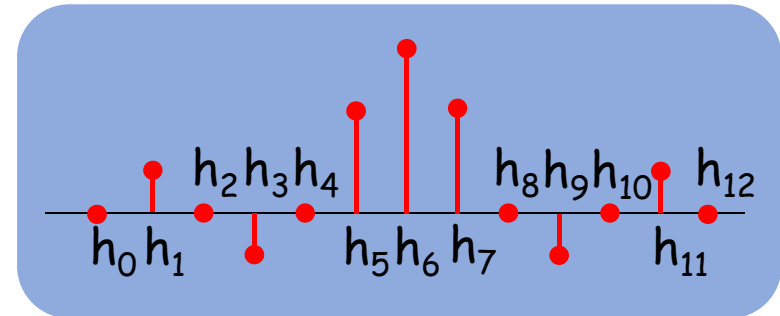
Si el filtro es antisimétrico se cambiarán algunos sumadores por restadores

Aprovechando las propiedades de los algoritmos

Algunos filtros poseen coeficientes nulos
 → no es necesario implementar esos mult.

$$\begin{array}{lll} h_1 = h_{11} & h_3 = h_9 & h_5 = h_7 \\ h_6 = 0.5 & & \\ h_0 = h_{12} = 0 & h_2 = h_{10} = 0 & h_4 = h_8 = 0 \end{array}$$

Filtro MediaBanda



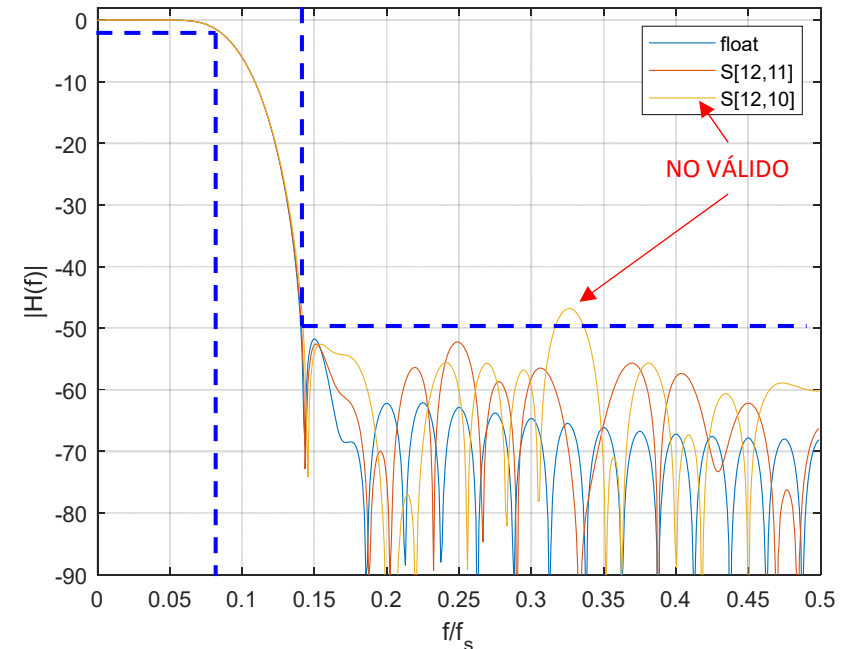
Precisión finita en filtros FIR

Coeficientes:

- Afecta al valor de los ceros
 - ✓ Modifica la respuesta en frecuencia
- Reducción de precisión válida mientras se cumpla la máscara de filtrado
- Utilizar “round” → mayor precisión
- Verificación HDL: Golden Model con coeficientes con precisión finita

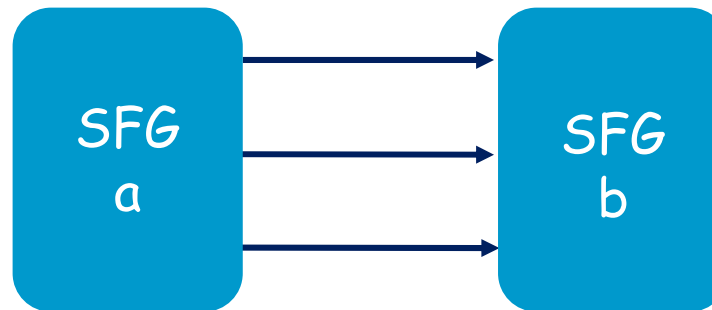
Ruta de datos:

- Introduce ruido de cuantificación
- Usualmente, recorte de precisión solo a la salida del filtro
 - ✓ Los bloques DSP de los FPGA: mult + sumador de ancho grande
 - Xilinx Virtex DSP48: 25x18bits + 48 acc
 - Altera Cyclone V Variable DSP: 18x18bits + 44 acc
- Se suele usar “floor” para no incrementar recursos
 - ✓ Algunos bloques DSP posibilitan la implementación de redondeo final

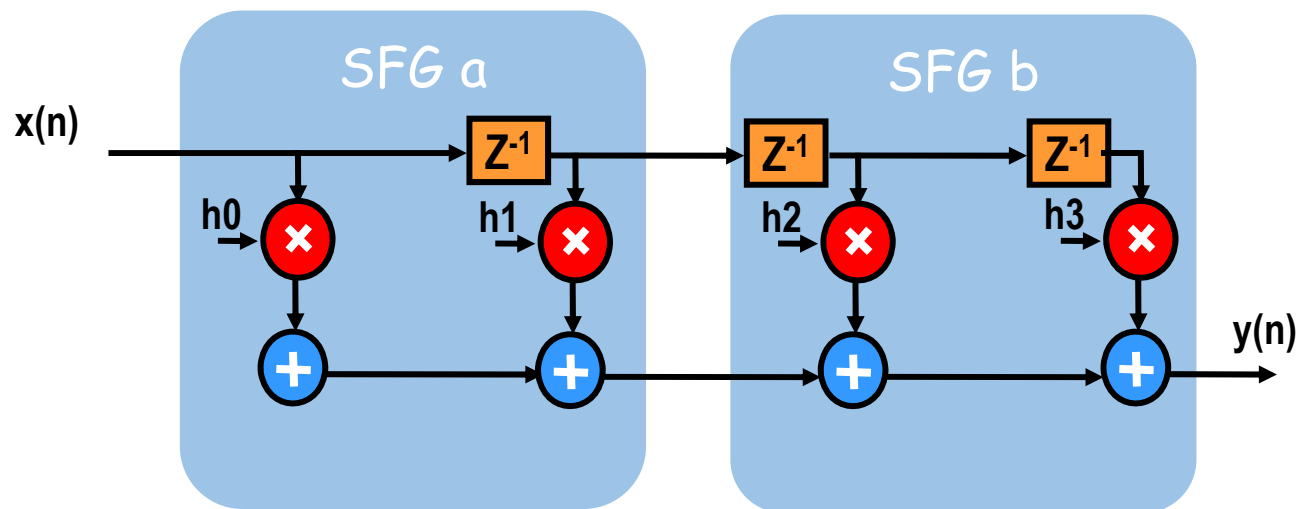


Segmentando los flujos de datos

Grafo de flujo de datos

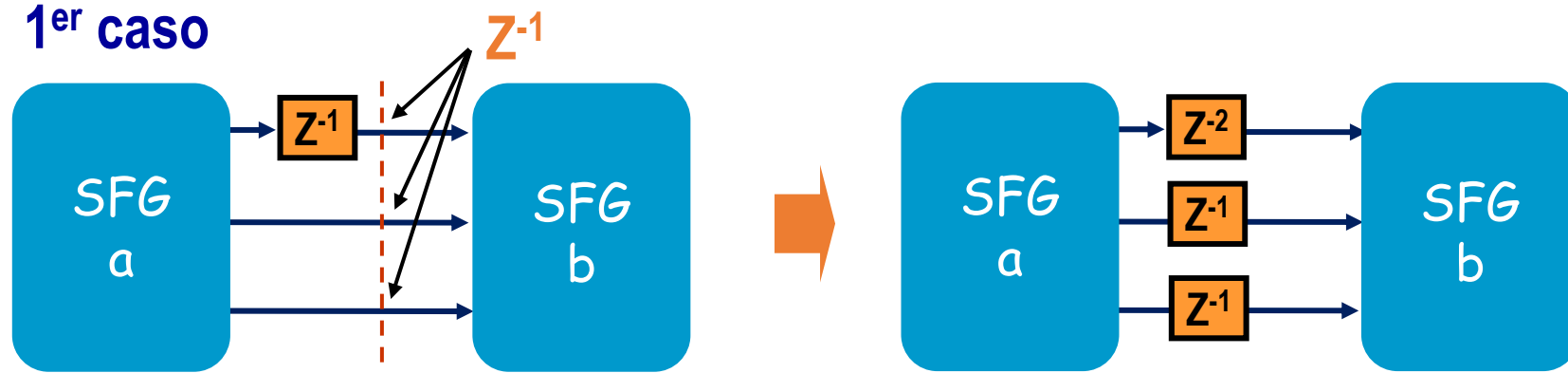


Ejemplo:

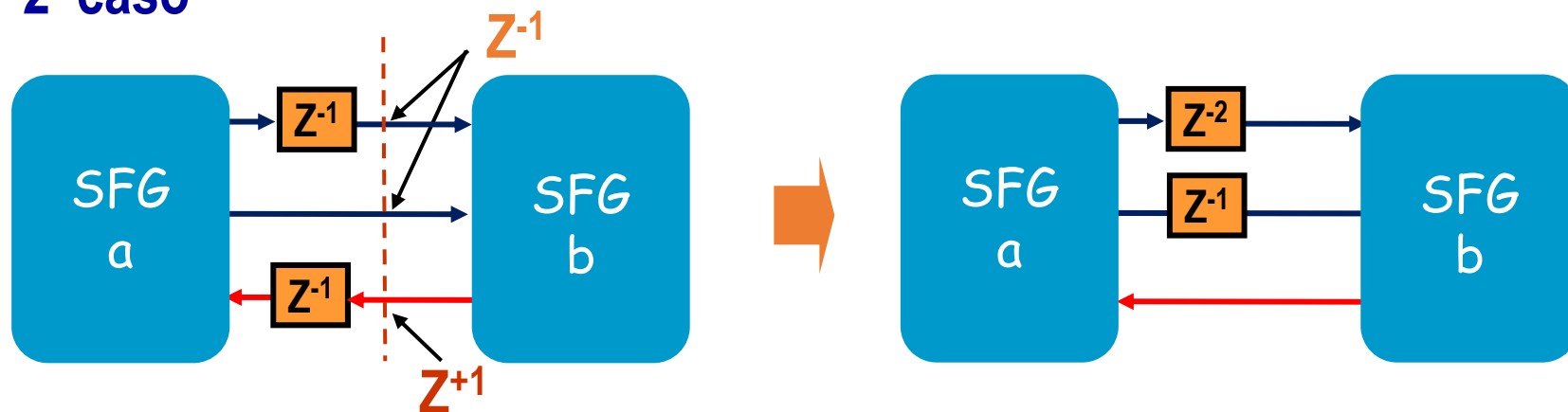


Segmentando los flujos de datos

1^{er} caso



2^o caso



Retiming

Cambiar la ubicación de registros sin modificar el algoritmo

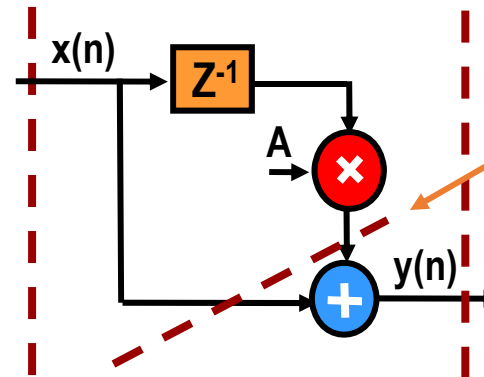
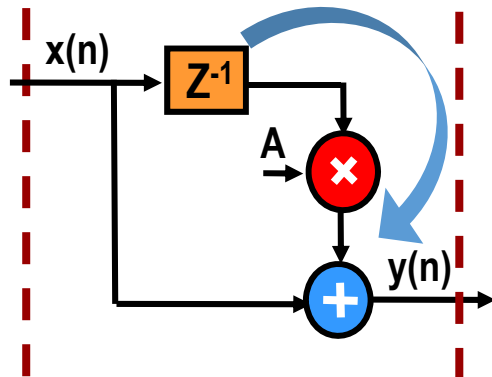
⇒ Transformar la arquitectura → adaptarla a bloques DSP

⇒ Reducir el camino crítico

Ej. Reducción del camino crítico:
 $y(n) = x(n) + A \cdot x(n-1)$

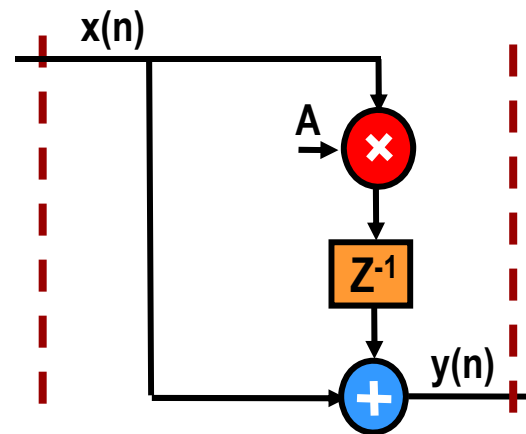
$$T_p = t_{\text{mult}} + t_{\text{sum}}$$

Retiming



Pipelining

- 2 registros
- añade 1 ciclo de latencia



$$T_{p1} = t_{\text{mult}}$$

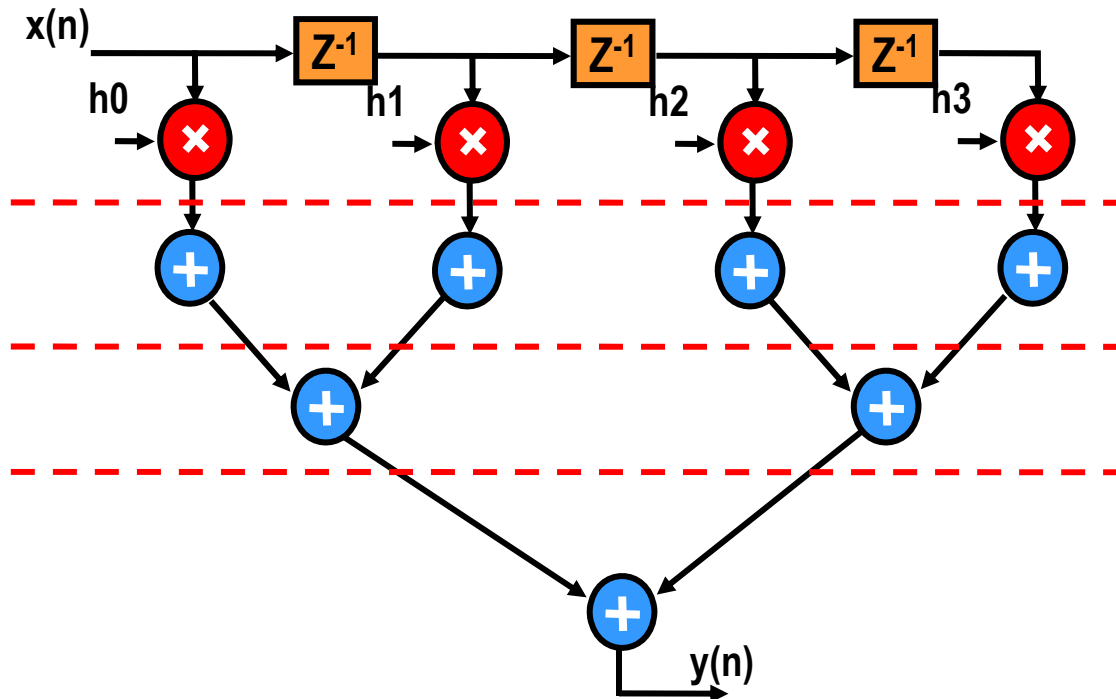
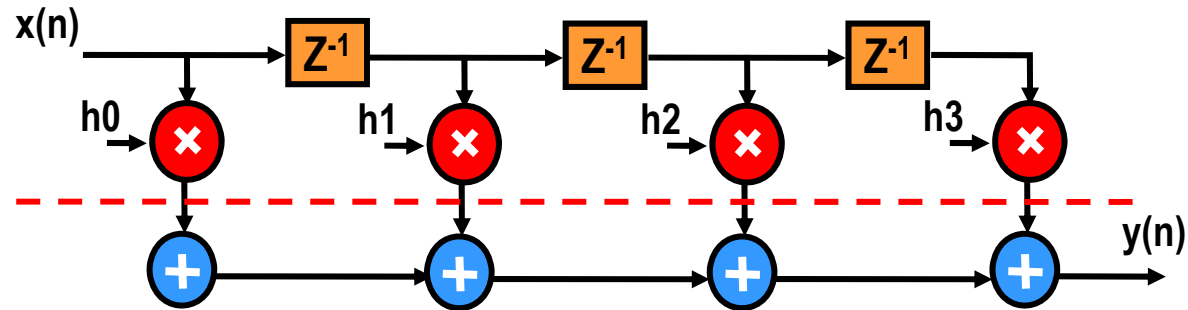
$$T_{p2} = t_{\text{sum}}$$

$$T = \max\{T_{p1}, T_{p2}\} = T_{p1}$$

¡No aumenta la latencia!

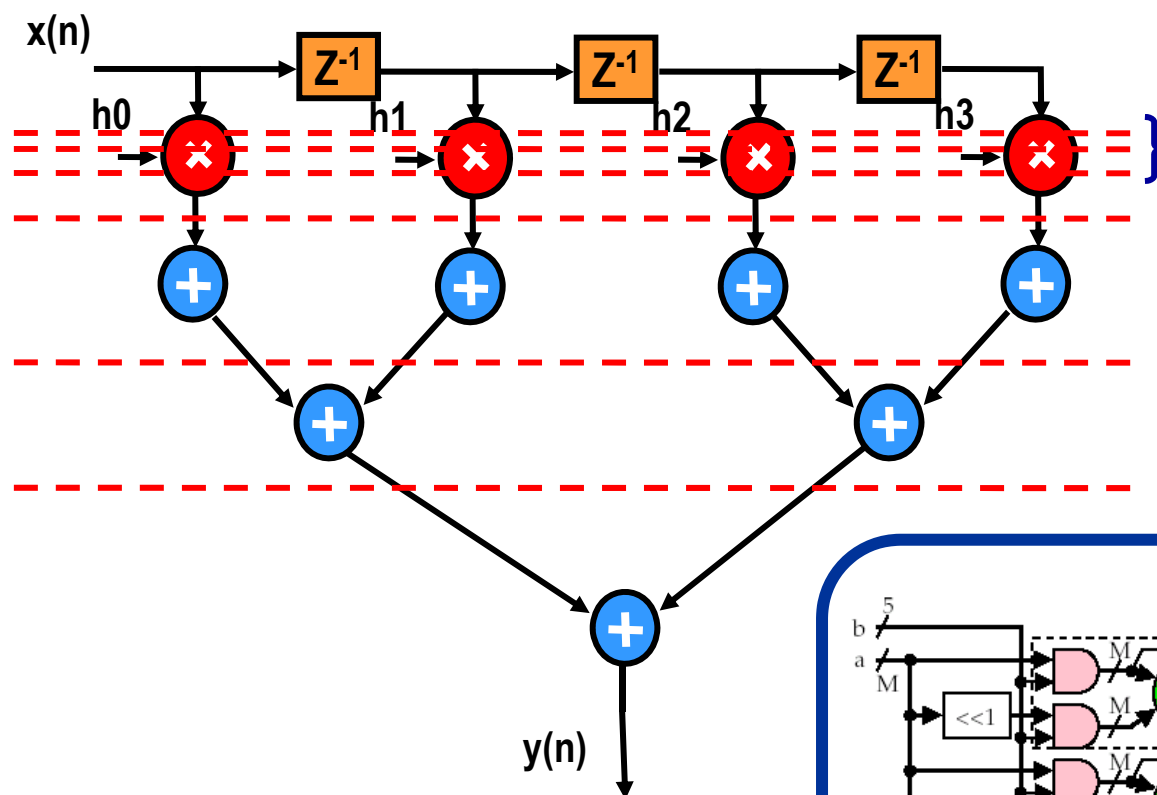
Segmentación en filtros FIR paralelos

Forma Directa

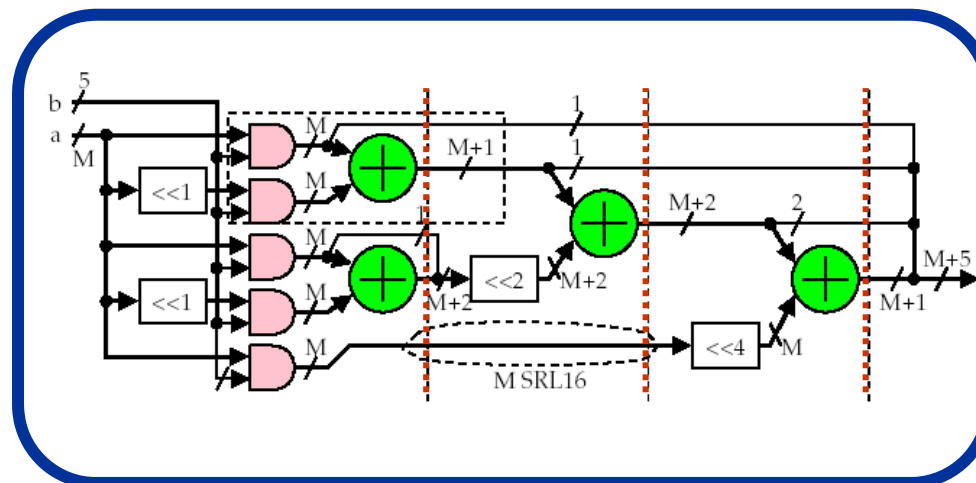


Segmentación en filtros FIR paralelos

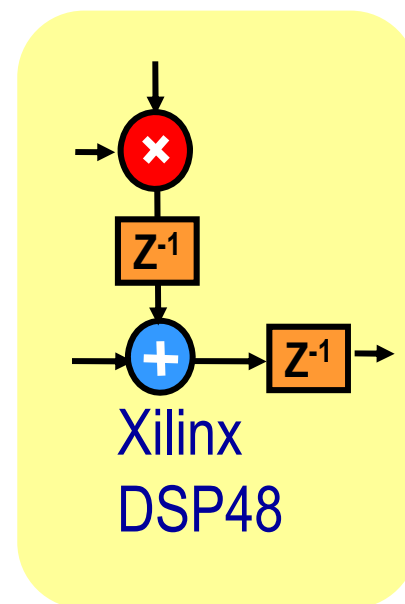
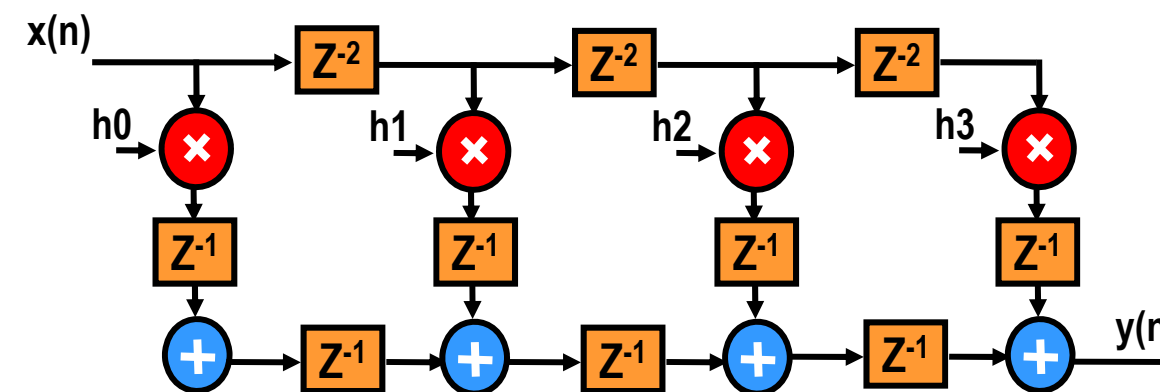
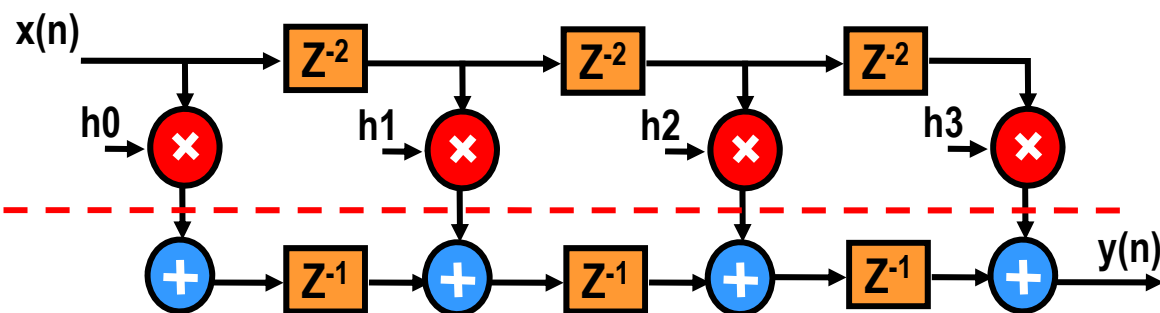
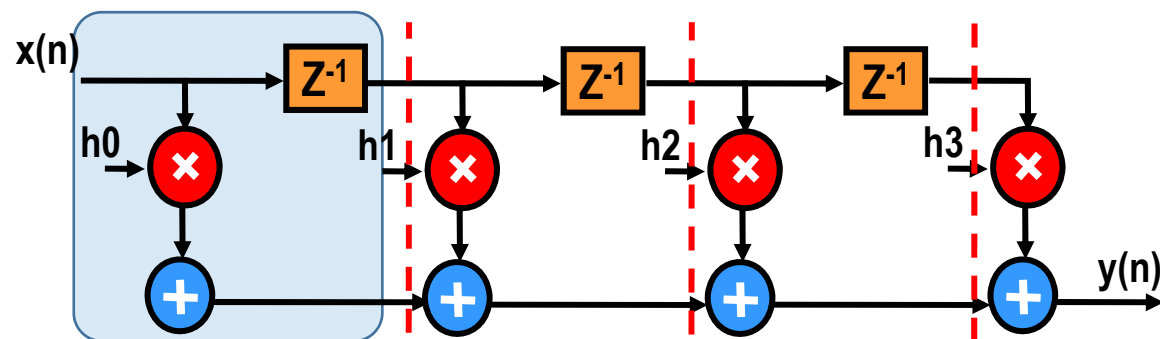
Forma Directa



Multiplicadores
implementados
en LUTs



Segmentación en filtros FIR paralelos

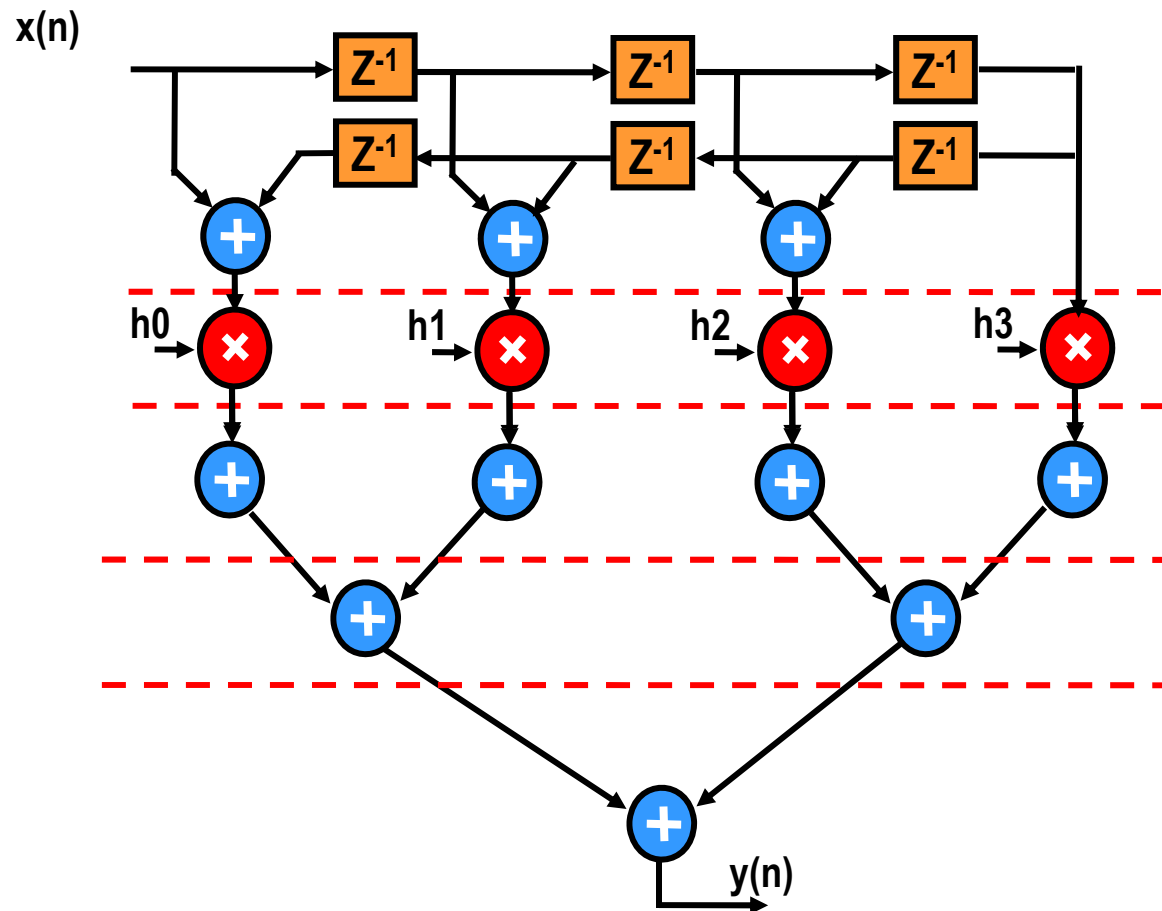


Array sistólico

- Regular
- Modular
- Comunicación local

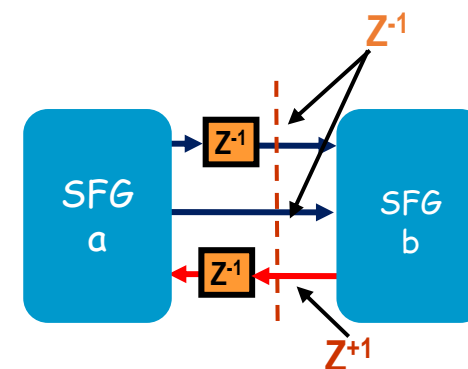
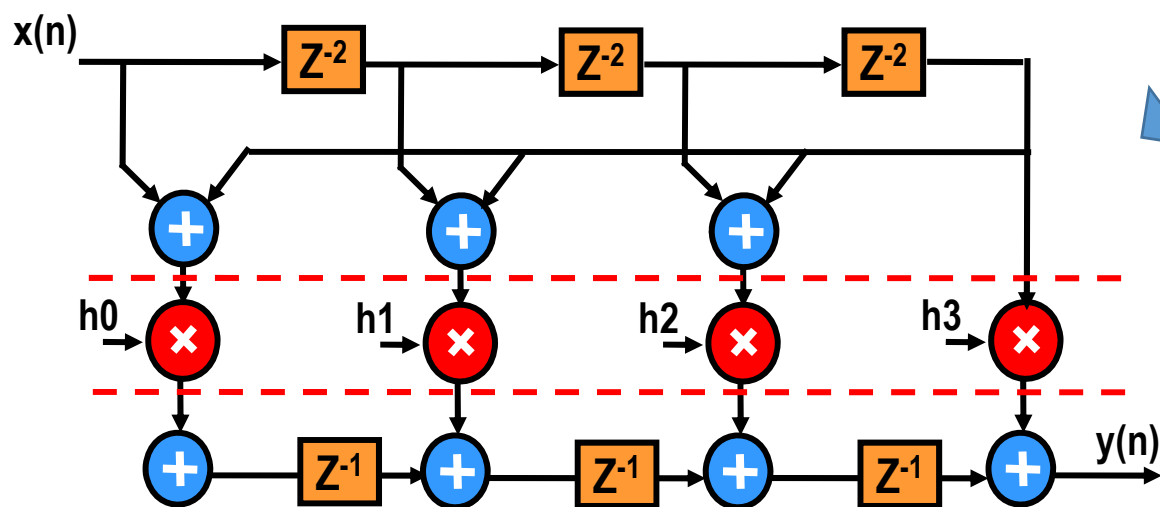
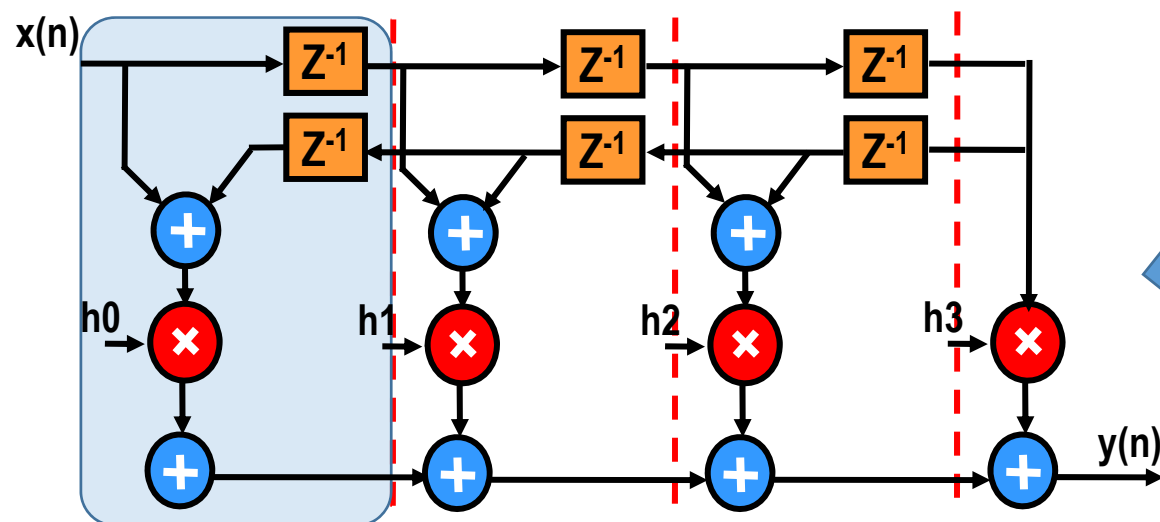
Segmentación en filtros FIR paralelos

FIR Simétrico



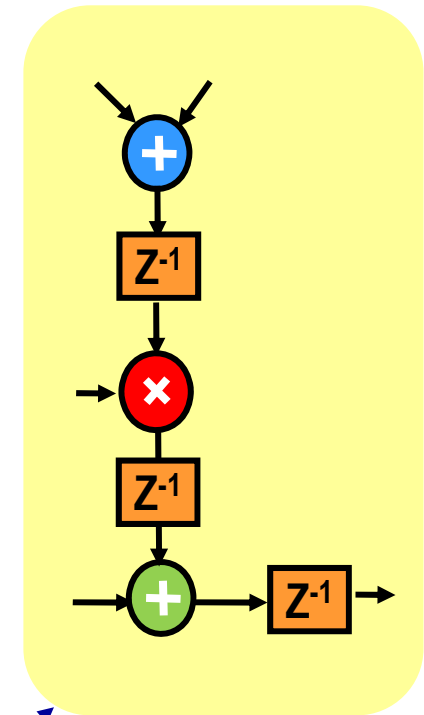
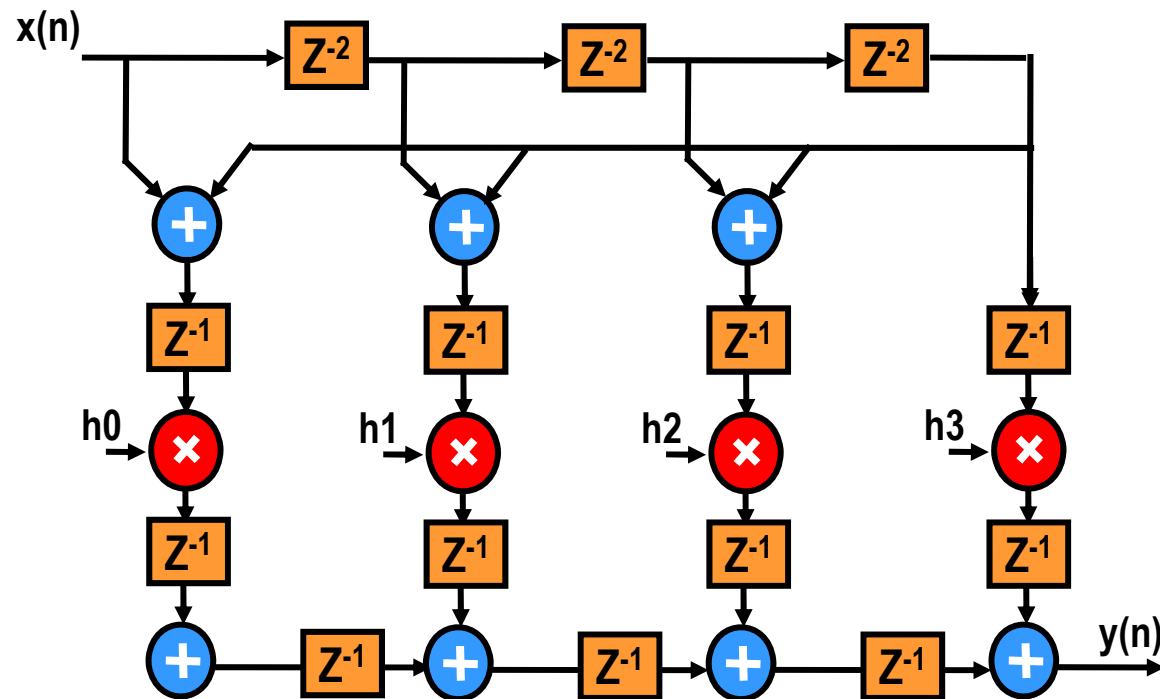
Segmentación en filtros FIR paralelos

FIR Simétrico



Segmentación en filtros FIR paralelos

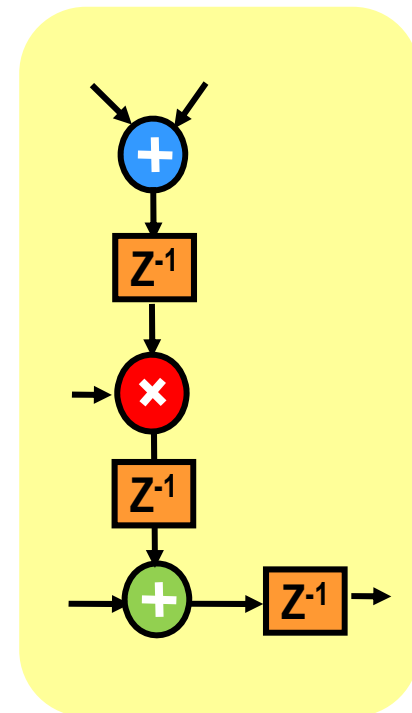
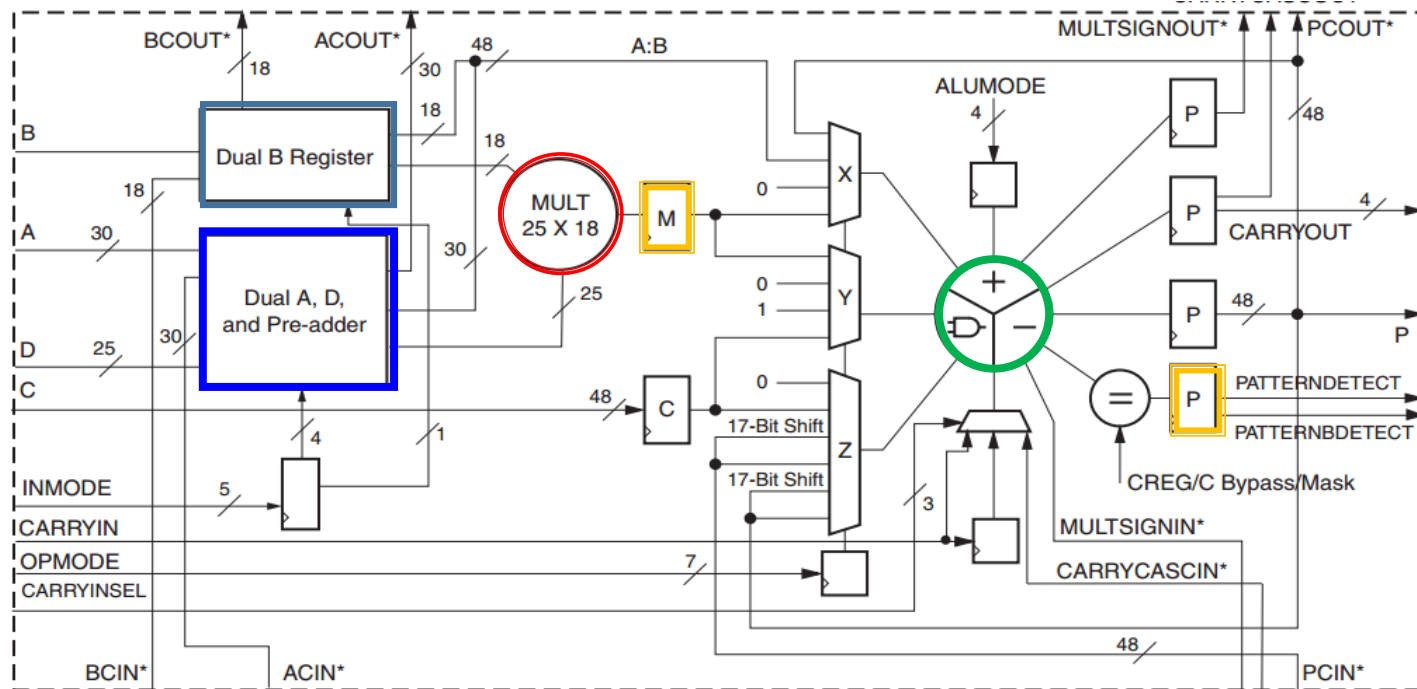
FIR Simétrico



Xilinx
DSP48E

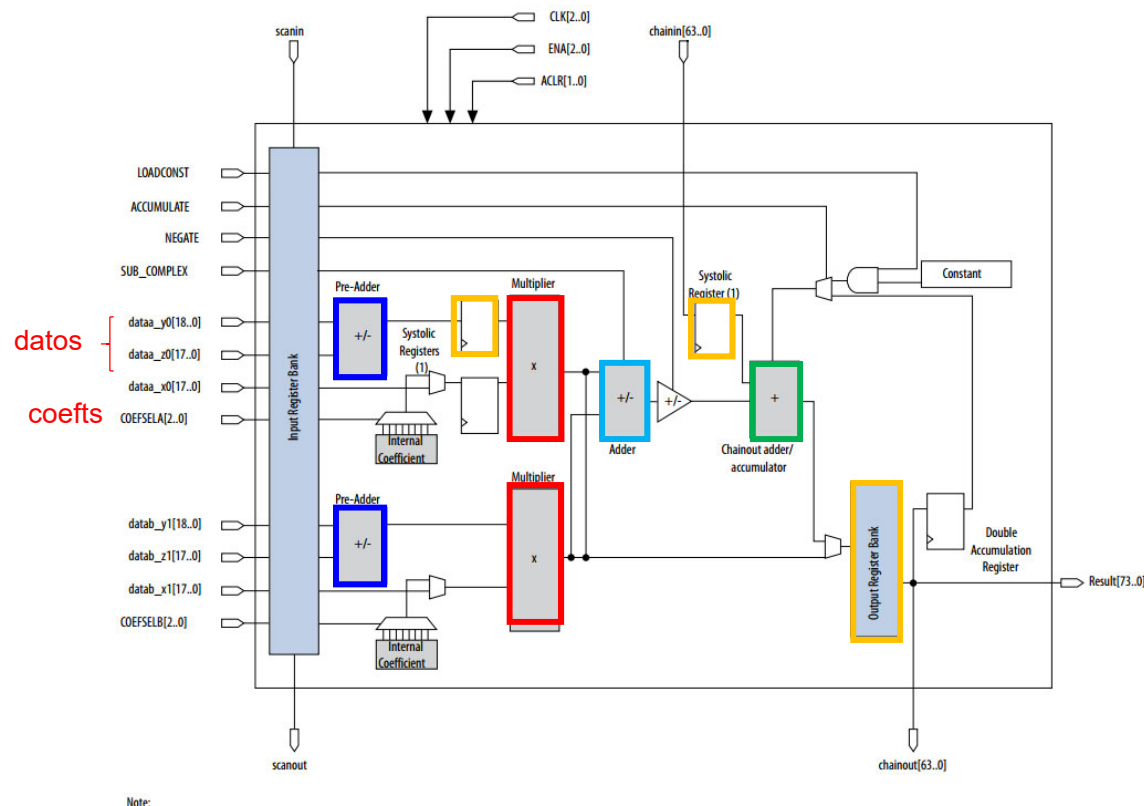
Segmentación en filtros FIR paralelos

Xilinx DSP48E

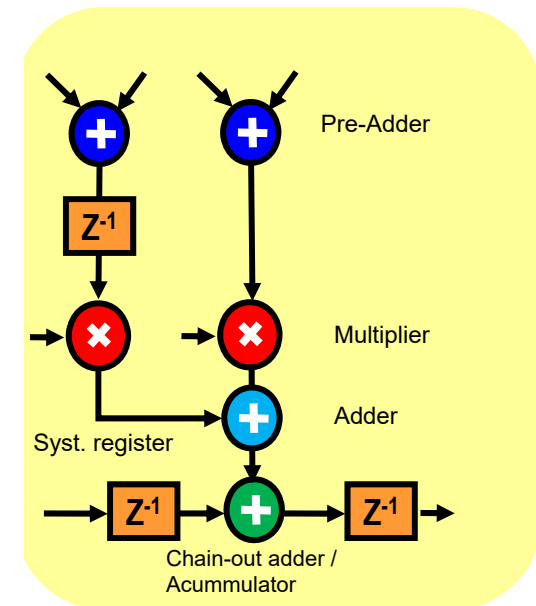


Segmentación en filtros FIR paralelos

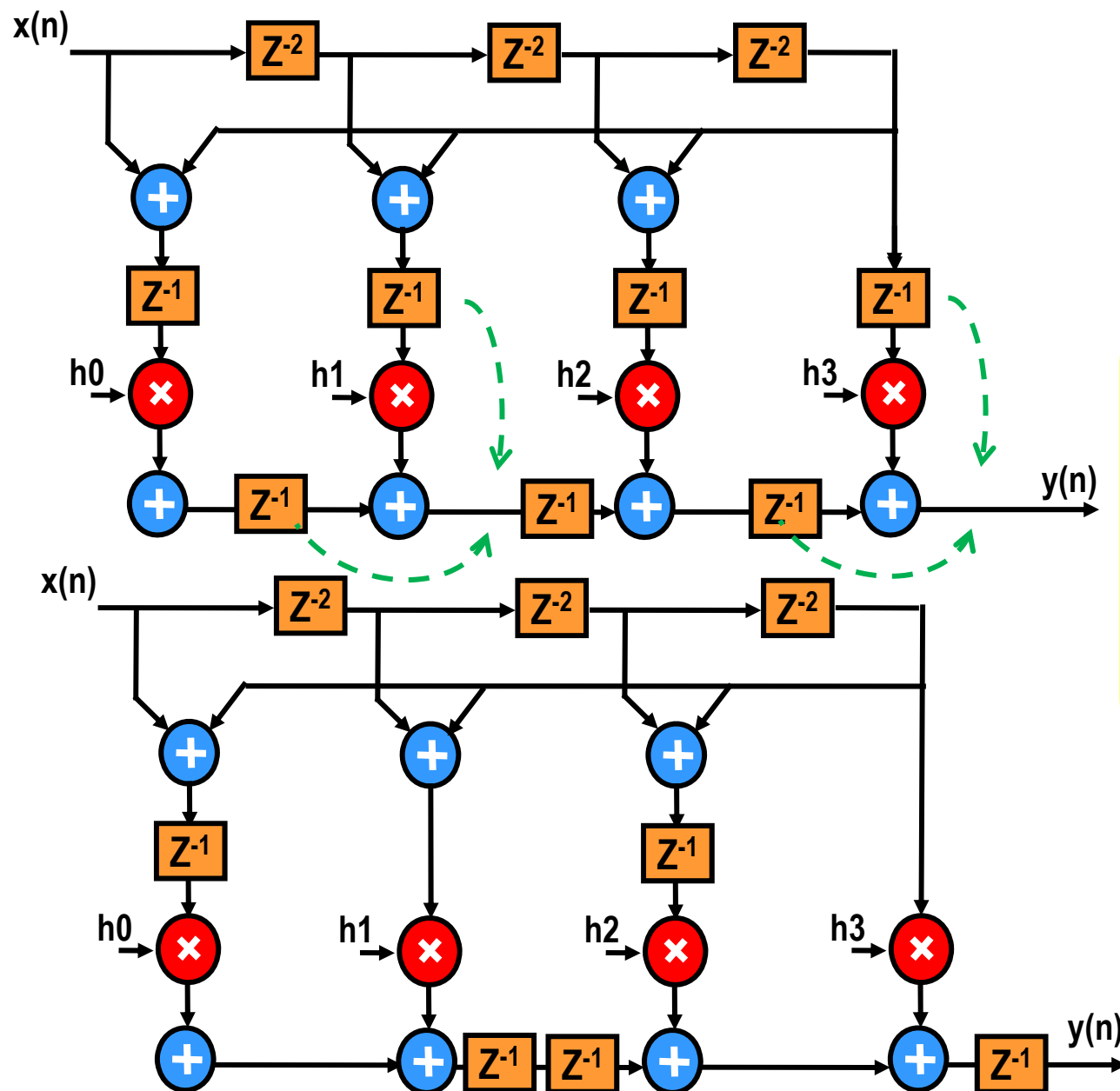
Altera _Cyclone V DSP Block
configurado como Filtro sistólico de 18 bits



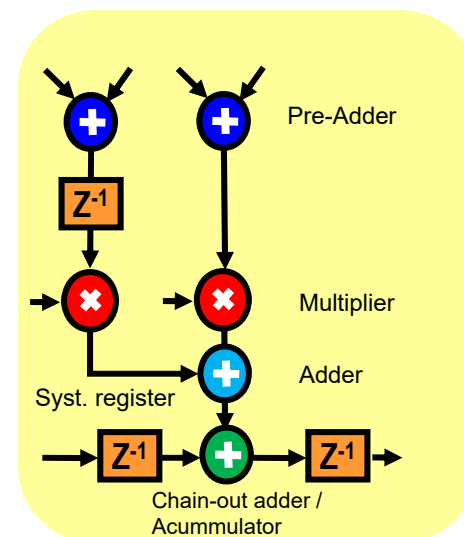
Permite almacenar 8 coeficientes constantes de 18 bits o de 27 bits



Segmentación en filtros FIR paralelos

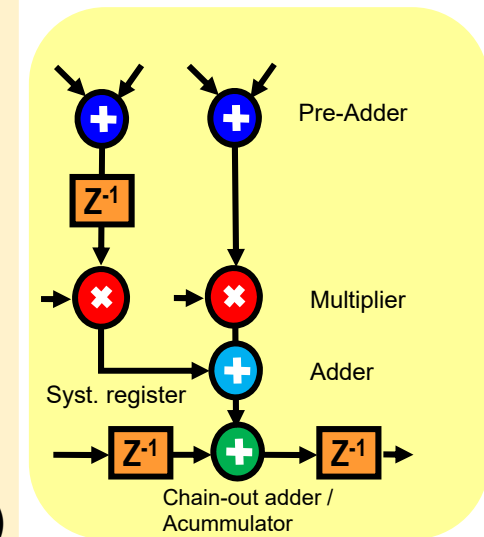
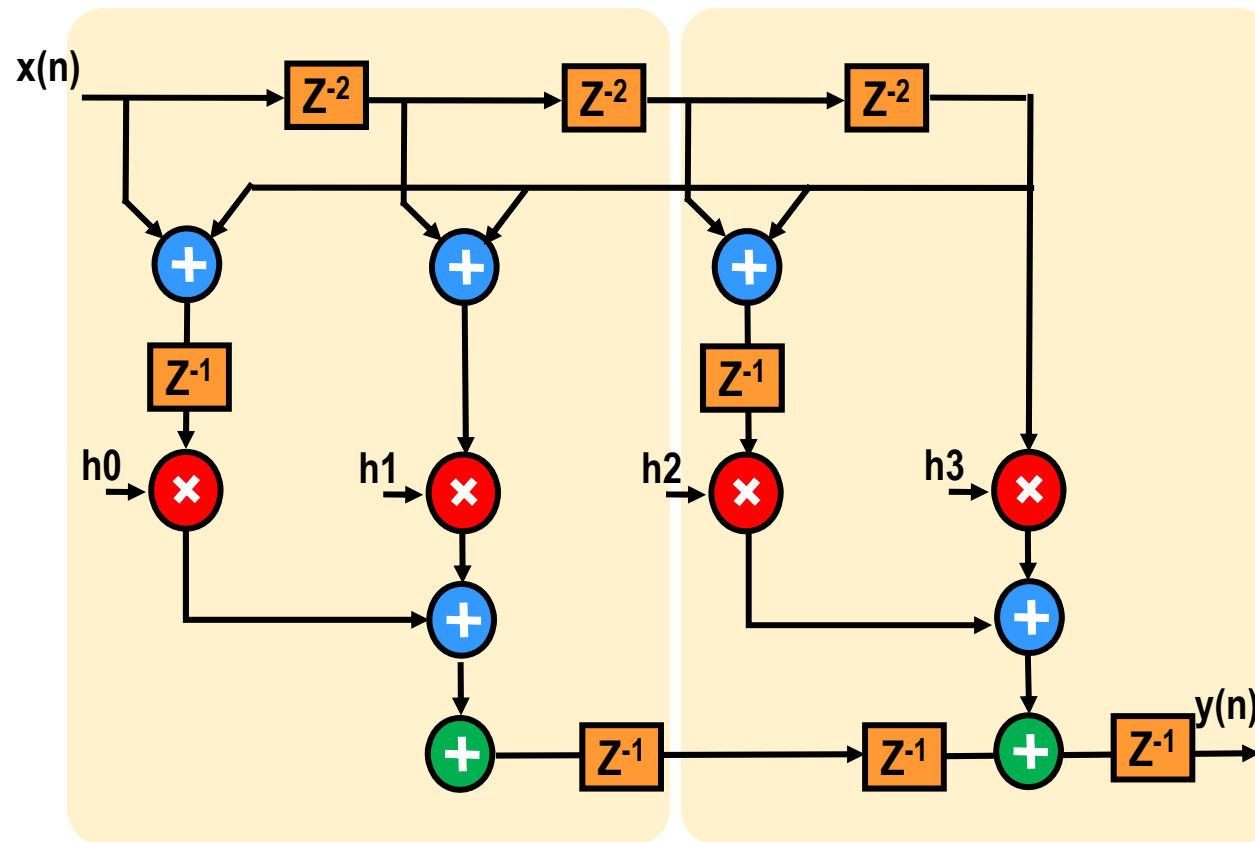


RETIMING



Segmentación en filtros FIR paralelos

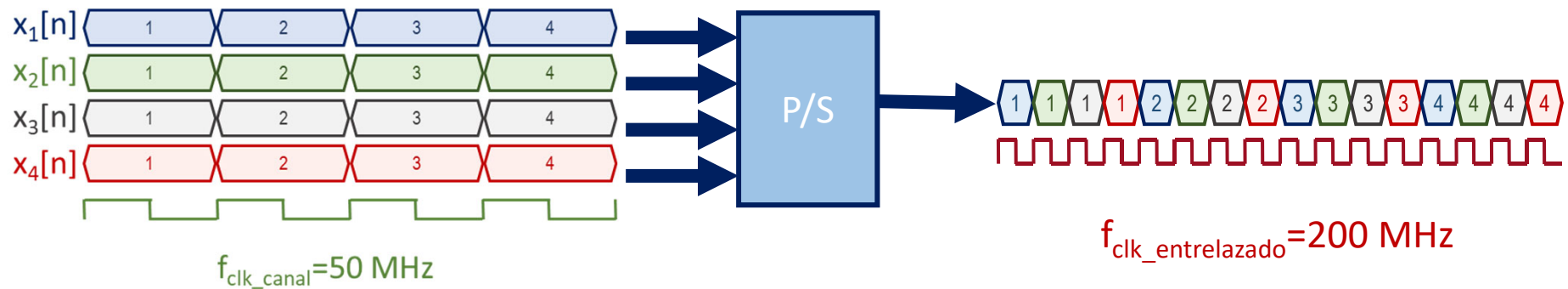
FIR Simétrico



Altera _Cyclone V DSP Block configurado como filtro sistólico de 18 bit

Procesado entrelazado

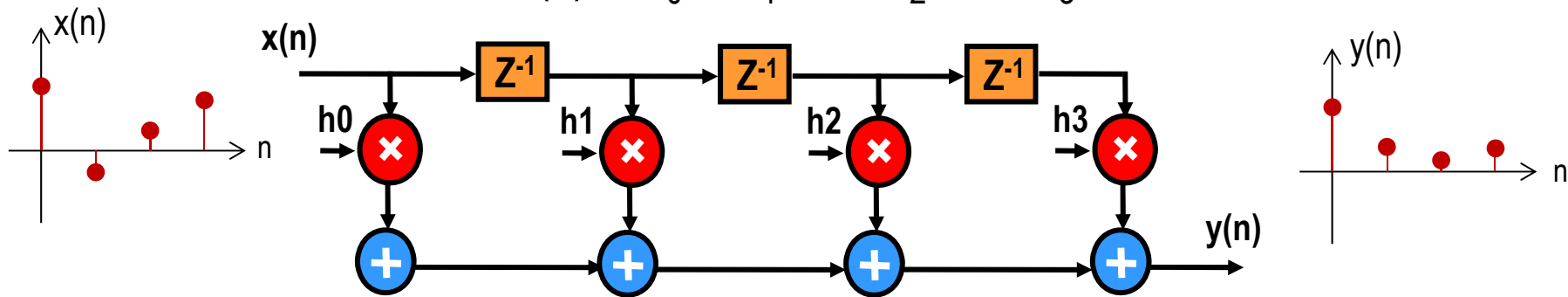
- Aplicaciones en las que se requiere aplicar el mismo procesamiento a múltiples flujos de datos independientes
 - Modulaciones digitales (ramas I y Q)
 - Interfaz de algunos conversores AD y DA



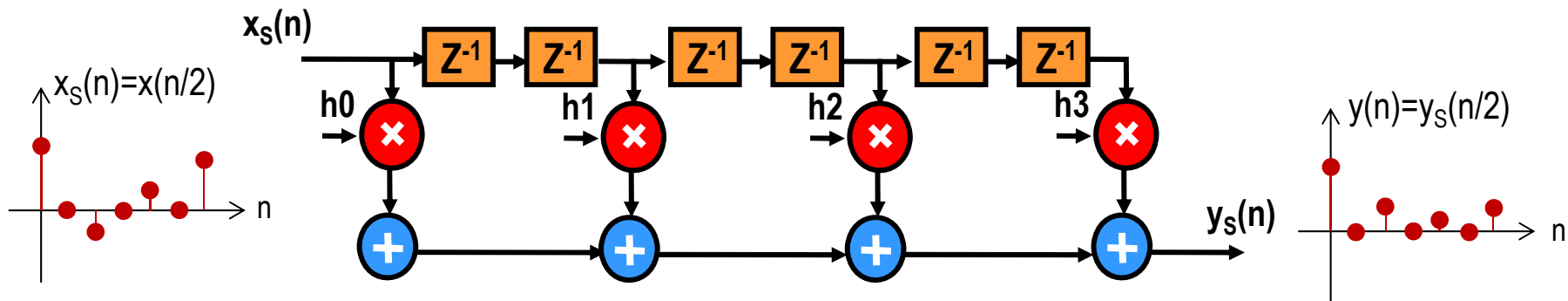
Procesado entrelazado

Escalado temporal

$$H(z) = h_0 + h_1 z^{-1} + h_2 z^{-2} + h_3 z^{-3}$$

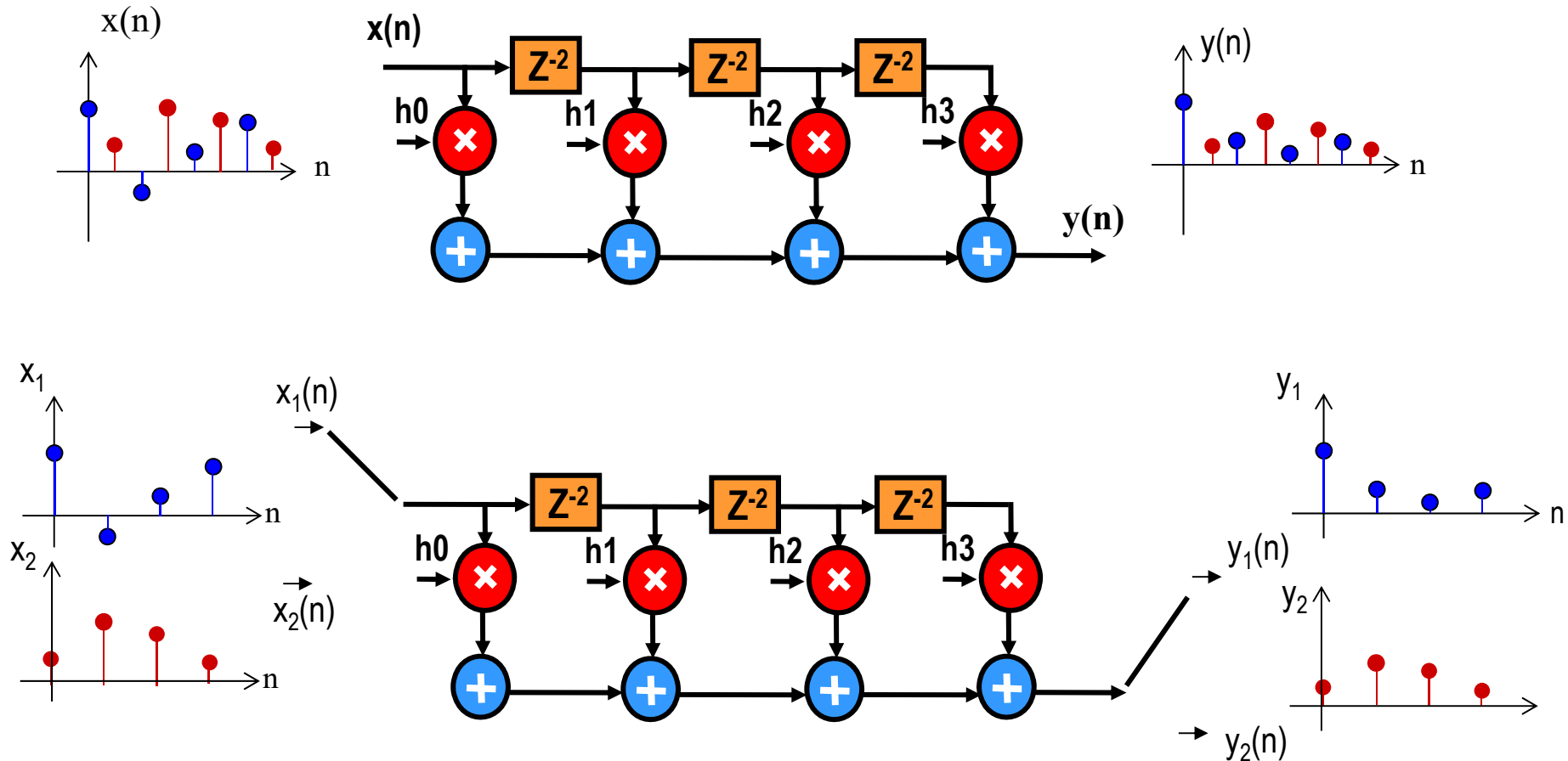


$$H_s(z) = H(z^2) \quad H_s(z) = h_0 + h_1 z^{-2} + h_2 z^{-4} + h_3 z^{-6}$$



Procesado entrelazado

$$H_S(z) = H(z^2)$$



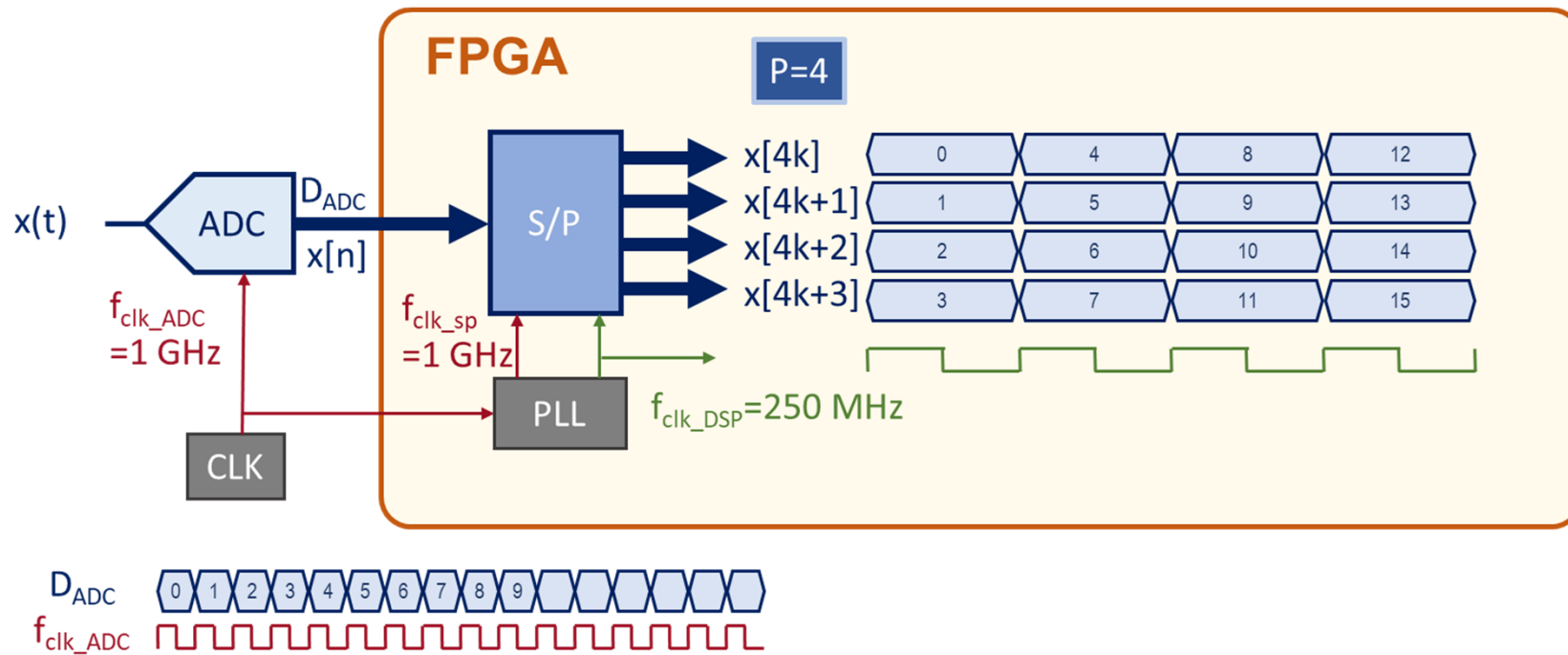
• $H(z^N) \rightarrow N$ flujos de datos se pueden procesar a $f_s = f_{\text{clk}}/N$

Paralelización de algoritmos

- FPGA actuales $\rightarrow f_{clk_max} < 500\text{-}600\text{ MHz}$
 - ✓ en Cyclone IV $f_{clk_max} = 250\text{ MHz}$
- Diseños grandes / FPGAs congestionada \rightarrow difícil mantener una ruta de datos a f_{clk_max}
- Aplicaciones que requieren el procesamiento de señales con BW grande $\rightarrow f_s > f_{clk_max}$
 - Comunicaciones ópticas
 - Procesado digital de señales de RF
 - Microscopía con ultrasonidos
 - ...
- Paralelización \rightarrow el flujo de datos se divide en P flujos que se procesan a una $f_{clk} = f_s / P$
- Bloques IO de FPGAs \rightarrow recursos para gestionar I/O a f_{clk} altas y conversión a N flujos
 - ✓ Conversores P/S y S/P (SERDES: SERializer/DESerializer)
 - ✓ PLLs
 - ✓ Delays

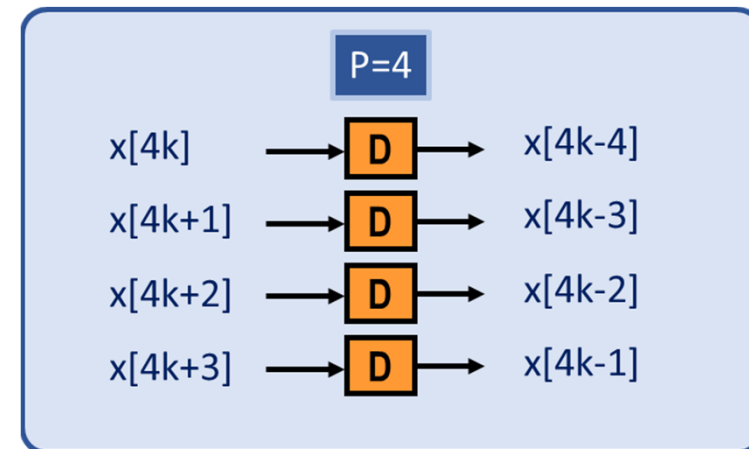
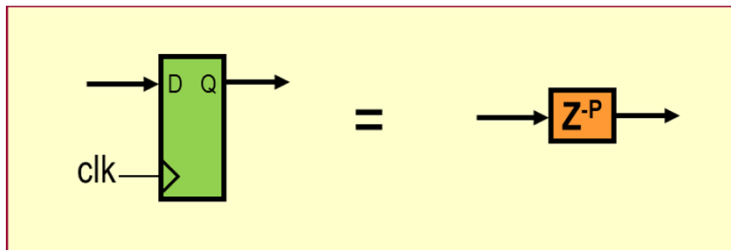
Paralelización de algoritmos

- Ej: ADC 1 Gsps



Paralelización de algoritmos

- ⇒ Frecuencia de muestreo (f_s) > Frecuencia de reloj (f_{clk}) → $f_s = P \cdot f_{clk}$
- ⇒ La arquitectura necesita unas P veces los recursos que se requieren para implementar todas las operaciones del algoritmo
- ⇒ Se obtienen P resultados en cada ciclo de reloj
- ⇒ Un registro equivale a un retardo de P muestras temporales



⇒ Métodos de diseño:

- En el **tiempo** → ↑ complejidad de diseño e implementación, ↓ recursos
- En **z** mediante la **transf. polifásica** → ↓ complejidad, ↑ recursos

Paralelización de algoritmos

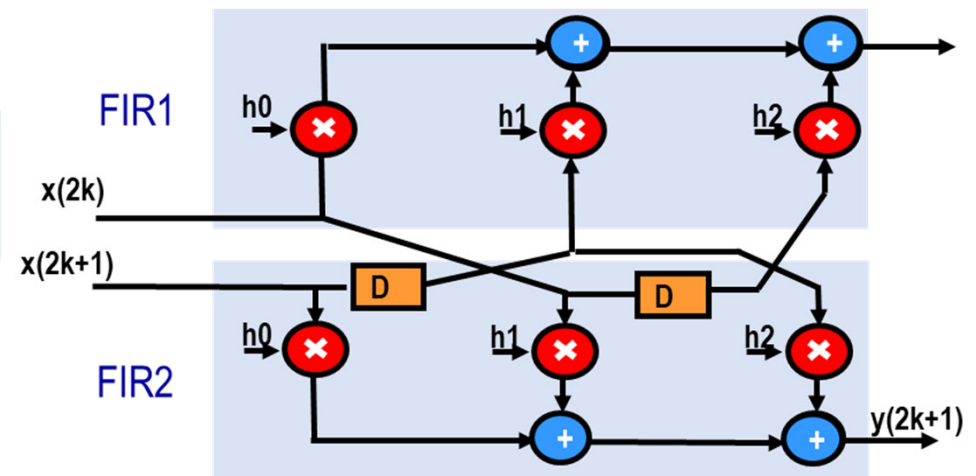
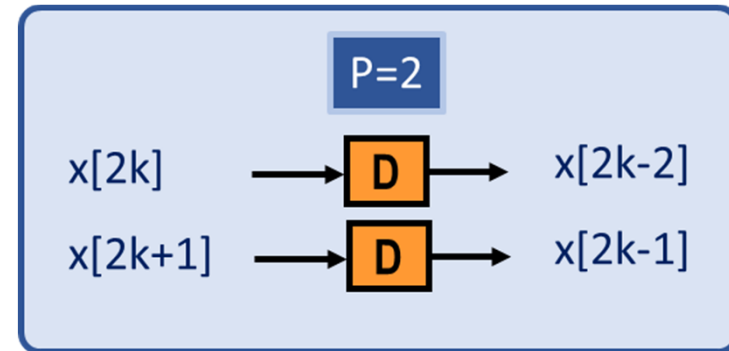
Método de diseño en el tiempo:

Ej. Filtro FIR de 3 etapas paralelizado por $P=2$

$$y(n) = h_0 \cdot x(n) + h_1 \cdot x(n-1) + h_2 \cdot x(n-2)$$

$P=2$

$$y(2k) = h_0 \cdot x(2k) + h_1 \cdot x(2k-1) + h_2 \cdot x(2k-2)$$
$$y(2k+1) = h_0 \cdot x(2k+1) + h_1 \cdot x(2k) + h_2 \cdot x(2k-1)$$

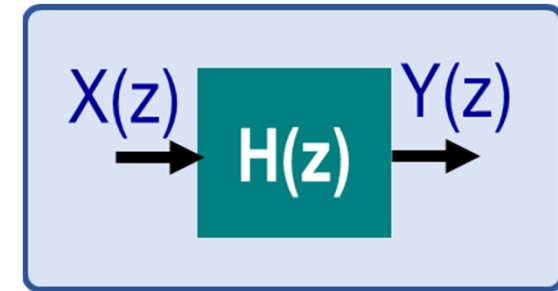


- 1) Evaluación $y(n)$. en $n=P \cdot k, P \cdot k+1, P \cdot k+2, \dots, P \cdot k+P-1$
- 2) Implementación en paralelo cada ec.

Paralelización de algoritmos

Método de diseño basado en la descomposición polifásica:

- 1) Aplicación de la descom. en P fases a $X(z)$, $H(z)$ e $Y(z)$
- 2) Cálculo de $Y(z)=H(z) \cdot X(z)$
- 3) Implementación en paralelo la ec. de cada fase de $Y(z)$



Ej. Filtro FIR de 3 etapas paralelizado por $P=2$

Descomp. polifásica en $P=2$ fases:

$$X(z) = x_0 + x_1 z^{-1} + x_2 z^{-2} + x_3 z^{-3} = (x_0 + x_2 z^{-2}) + z^{-1}(x_1 + x_3 z^{-2}) = X_0(z^2) + z^{-1}X_1(z^2)$$

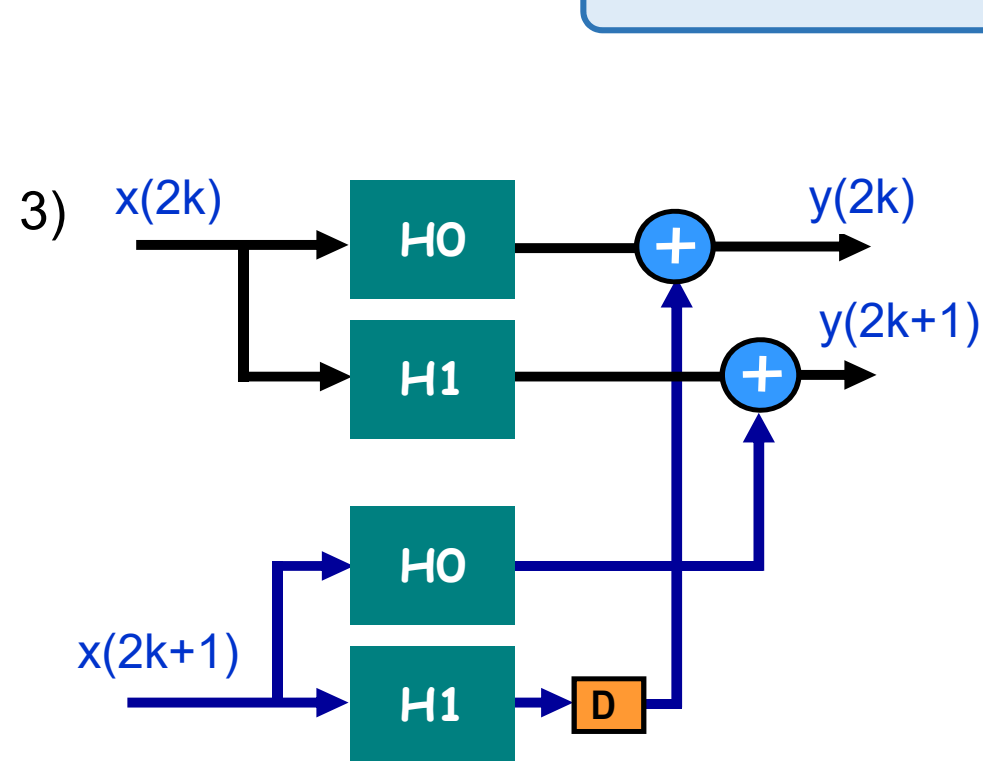
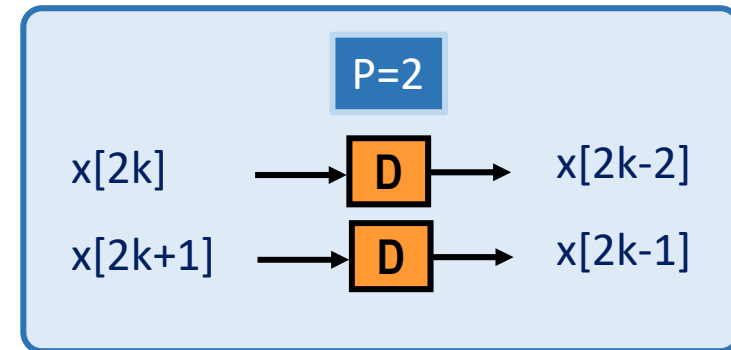
$$1) \begin{cases} X(z) = X_0(z^2) + z^{-1}X_1(z^2) \\ H(z) = H_0(z^2) + z^{-1}H_1(z^2) \\ Y(z) = Y_0(z^2) + z^{-1}Y_1(z^2) \end{cases} \quad 2) \begin{cases} Y(z) = H(z)X(z) = [X_0(z^2) + z^{-1}X_1(z^2)] \cdot [H_0(z^2) + z^{-1}H_1(z^2)] = \\ = X_0(z^2)H_0(z^2) + z^{-1}X_0(z^2)H_1(z^2) + z^{-1}X_1(z^2)H_0(z^2) + z^{-2}X_1(z^2)H_1(z^2) = \\ = Y_0(z^2) + z^{-1}Y_1(z^2) \end{cases}$$

$$\begin{cases} Y_0(z^2) = X_0(z^2)H_0(z^2) + z^{-2}X_1(z^2)H_1(z^2) \\ Y_1(z^2) = X_0(z^2)H_1(z^2) + X_1(z^2)H_0(z^2) \end{cases}$$

Paralelización de algoritmos

Ej. Filtro FIR de 3 etapas paralelizado por $P=2$

$$2) \begin{cases} Y_0(z^2) = X_0(z^2)H_0(z^2) + z^{-2}X_1(z^2)H_1(z^2) \\ Y_1(z^2) = X_0(z^2)H_1(z^2) + X_1(z^2)H_0(z^2) \end{cases}$$

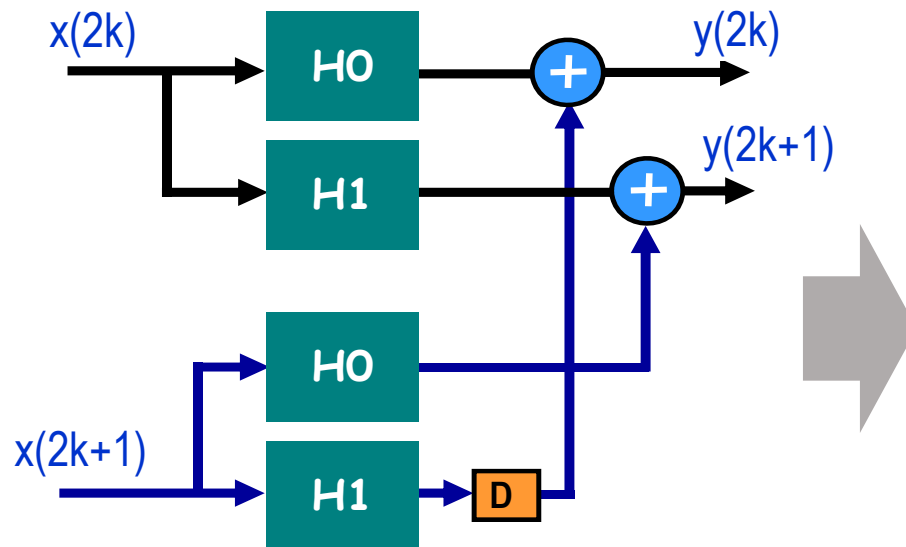


Paralelización de algoritmos

Ej. Filtro FIR de 3 etapas paralelizado por $P=2$

El área de un filtro FIR paralelo puede reducirse explotando las características de la estructura

$$\begin{cases} Y_0 = X_0 H_0 + z^{-2} X_1 H_1 \\ Y_1 = X_0 H_1 + X_1 H_0 \end{cases}$$



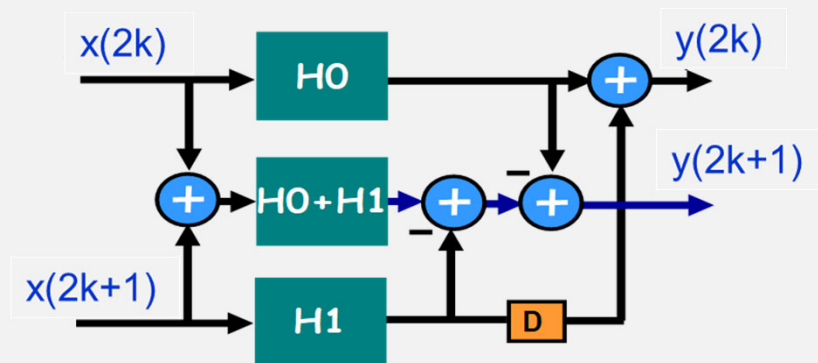
Reducción de la complejidad

$$Y_0 = X_0 H_0 + z^{-2} X_1 H_1$$

$$Y_1 = (X_0 + X_1)(H_0 + H_1) - X_0 H_0 - X_1 H_1$$

Nuevo bloque
coef: $H_0 + H_1$

Bloques
compartidos



Conclusiones

- Estudio, diseño e implementación de Arquitecturas Paralelas
- Técnicas para mejorar las prestaciones de los circuitos y adaptar sus estructuras para implementarlas en distintos dispositivos FPGA
- Uso eficiente de los recursos DSP de las FPGA de Xilinx y Altera
- Técnicas de Paralelización de algoritmos



UNIVERSITAT
POLITÈCNICA
DE VALÈNCIA

Arquitecturas Paralelas

Procesado Digital de la Señal en FPGA

Máster Universitario en Ingeniería de Sistemas Electrónicos