

Implementación de Filtros Digitales

- ❑ Realización de Filtros Digitales
- ❑ Efectos de Cuantización
- ❑ Implementación Hardware de Filtros Digitales
- ❑ Digital Signal Processors (DSP)

Realización de Filtros Digitales

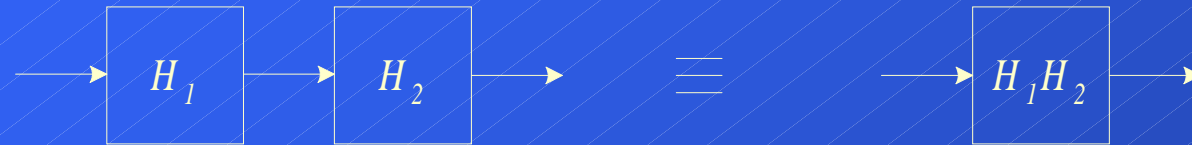
- Hay dos formas de realizar filtros digitales: Software o Hardware. En los dos casos deberemos hacer un diagrama con las operaciones a realizar. En software se habla de un diagrama de flujo, mientras que en hardware es un diagrama de bloques, que especifica los elementos del circuito y sus interconexiones.
- Una correcta elección del diagrama de bloques puede optimizar significativamente las prestaciones de la realización (tiempo de computación, memoria necesaria, minimizar los efectos de cuantización, etc).

Realización de Filtros Digitales

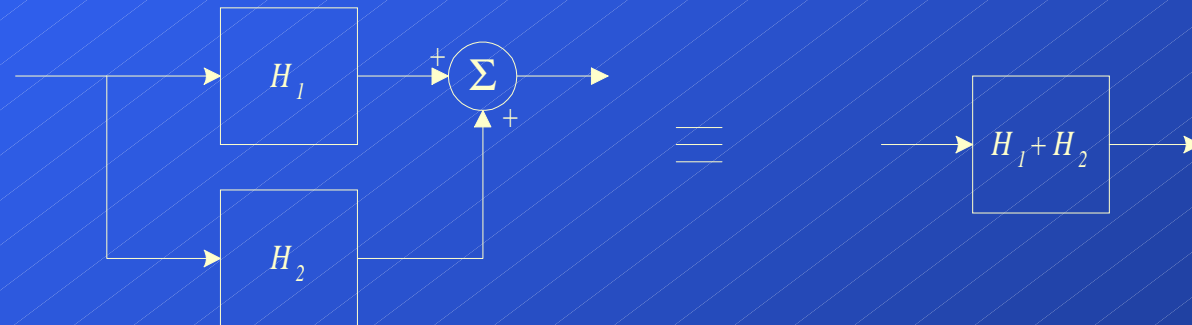
- Veamos algunas propiedades de los diagramas de bloques
 - ◆ Conexiones en cascada: La función de Transferencia global de una conexión en cascada es el producto de las funciones de Transferencia individuales.
 - ◆ Conexiones en paralelo: La función de Transferencia global de una conexión en paralelo es la suma de las funciones de Transferencia individuales.
 - ◆ Conexión en realimentación: La salida se realimenta en la entrada directamente o a través de otros subsistema. La función de Transferencia global viene dada por la relación (ver figura):

$$H_T(z) = \frac{G(z)}{1 + G(z)H(z)}$$

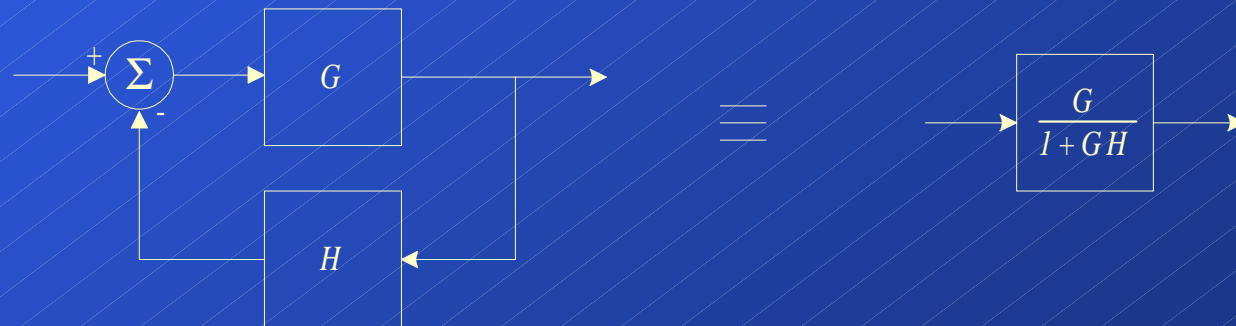
Realización de Filtros Digitales



Conexión de dos sistemas en Cascada



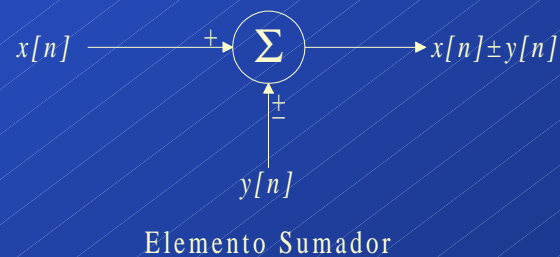
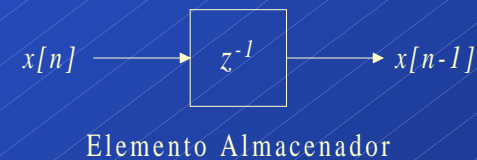
Conexión en paralelo de dos sistemas



Un sistema sencillo con realimentación

Realización de Filtros Digitales

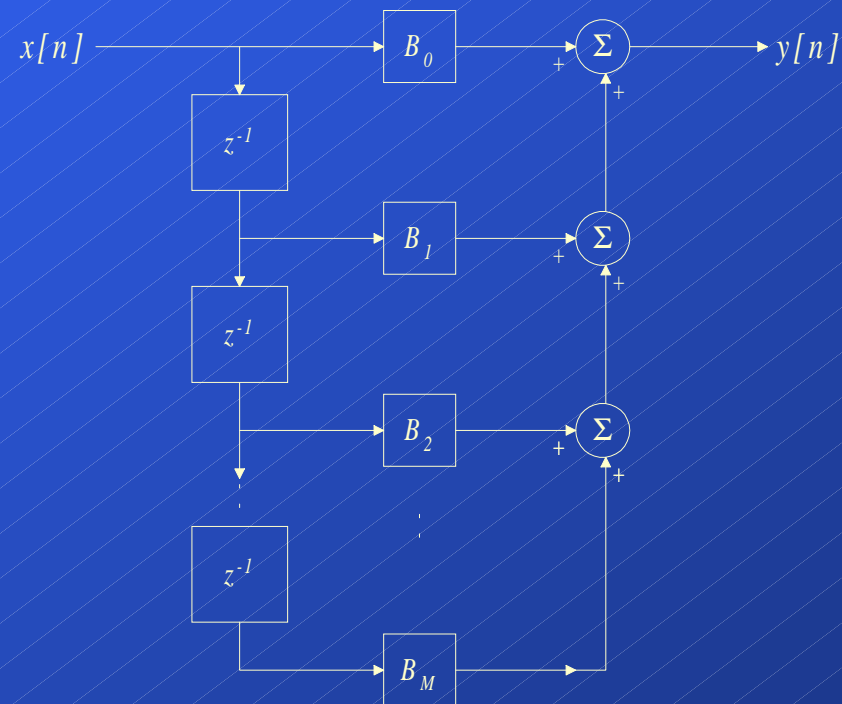
- ❑ Los filtros digitales pueden realizarse usando elementos correspondientes a las operaciones de multiplicación, adición y almacenaje de datos.
- ❑ El almacenaje de un dato significa retrasar su uso una cantidad normalmente igual al periodo de muestreo. Este retraso se representa mediante z^{-1} (retraso de una unidad), z^{-2} (dos unidades, etc).



Realización de Filtros Digitales

- Filtros FIR (MA): Son filtros no recursivos cuya función de Transferencia $H_{MA}(z)$ y su correspondiente ecuación diferencia $y[n]$ son de la forma,

$$H_{MA}(z) = B_0 + B_1 z^{-1} + \dots + B_M z^{-M} \quad y[n] = B_0 x[n] + B_1 x[n-1] + \dots + B_M x[n-M]$$
 Este filtro puede realizarse utilizando el diagrama de la figura

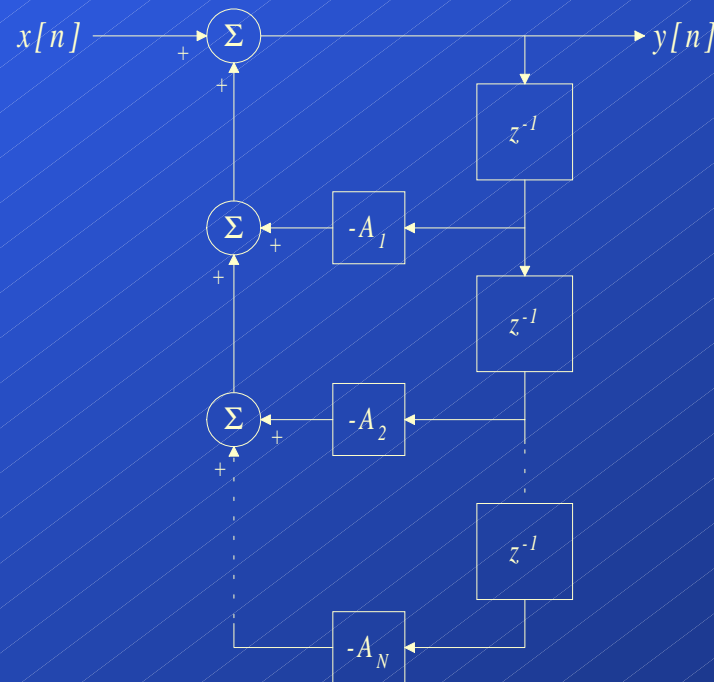


Realización de Filtros Digitales

- Filtros Autoregresivos: Son filtros recursivos cuya función de Transferencia $H_{AR}(z)$ y su correspondiente ecuación diferencia $y[n]$ son de la forma,

$$H_{AR}(z) = \frac{1}{1 + A_1 z^{-1} + \dots + A_N z^{-N}} \quad y[n] = -A_1 y[n-1] - \dots - A_N y[n-N] + x[n]$$

Este filtro puede realizarse utilizando el diagrama de la figura,



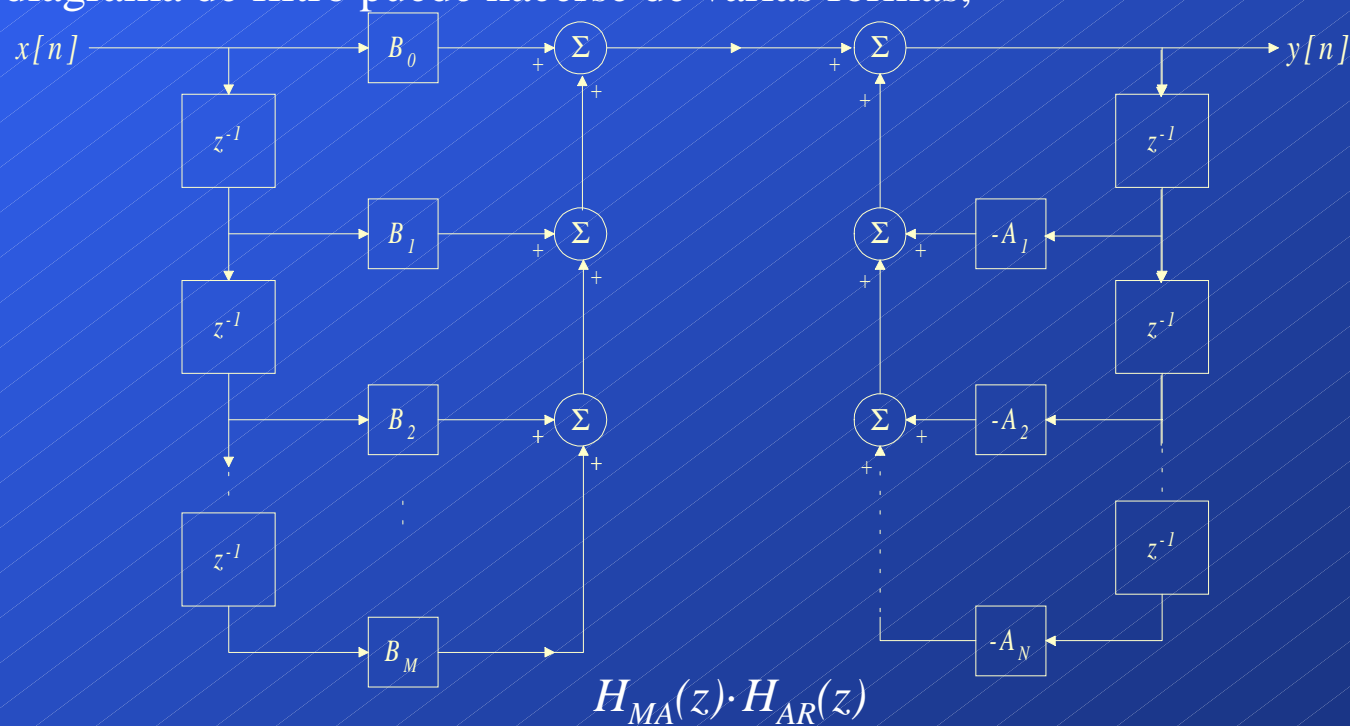
Realización de Filtros Digitales

- Filtros ARMA: Son la combinación de los dos anteriores. Su función de Transferencia y ecuación diferencia son,

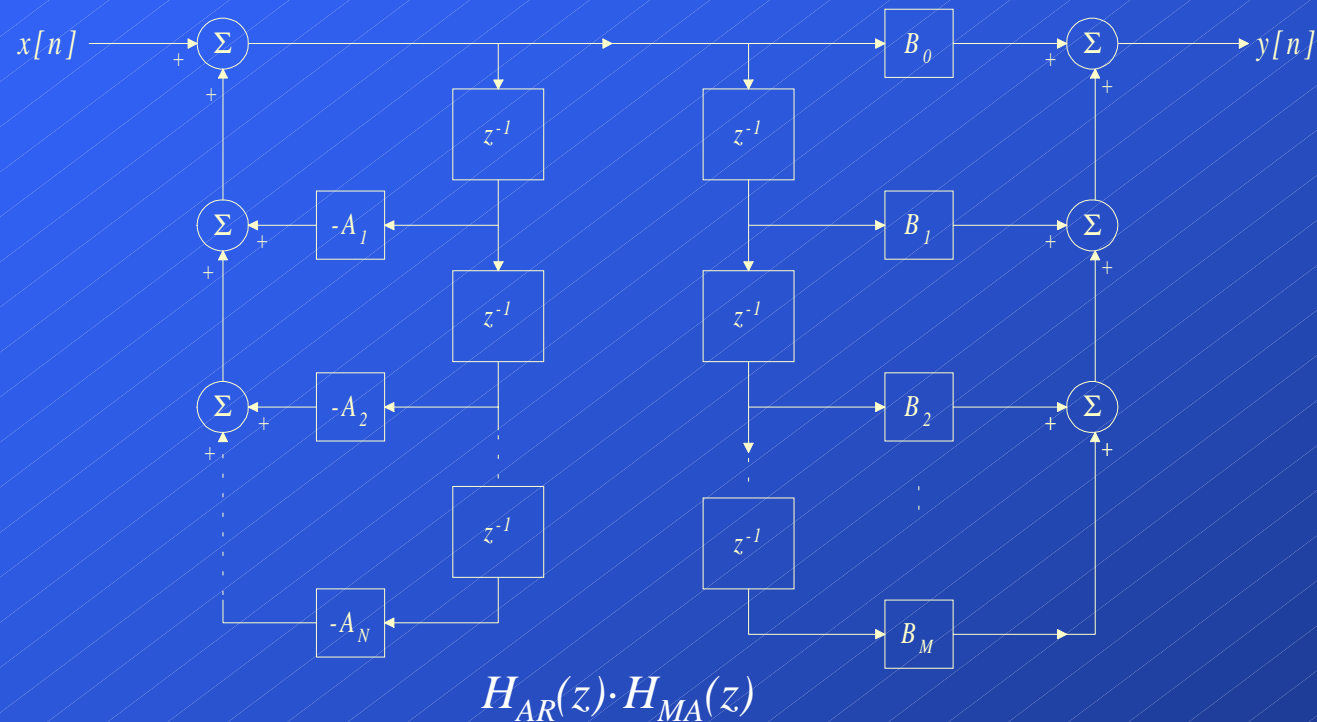
$$H(z) = \frac{B_0 + B_1 z^{-1} + \dots + B_M z^{-M}}{1 + A_1 z^{-1} + \dots + A_N z^{-N}} = H_{AR}(z) H_{MA}(z)$$

$$y[n] = -A_1 y[n-1] - \dots - A_N y[n-N] + B_0 x[n] + \dots + B_M x[n-M]$$

El diagrama de filtro puede hacerse de varias formas,

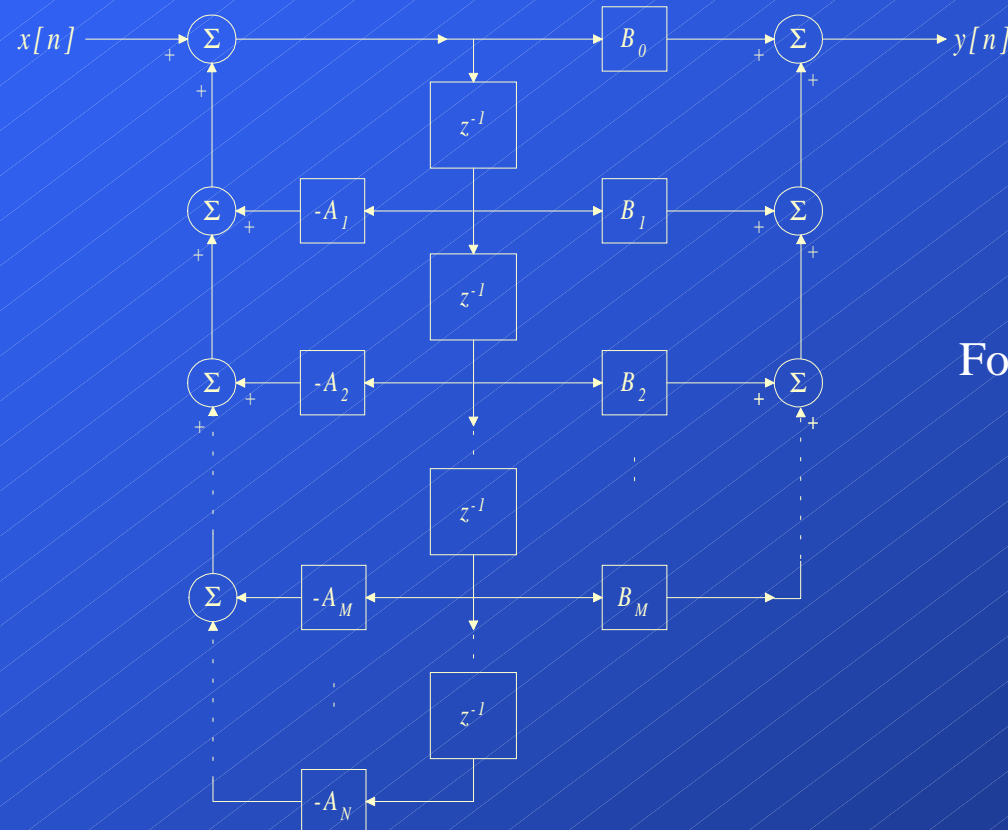


Realización de Filtros Digitales



- ❑ Esta dos formas son lógicamente idénticas. Se denominan forma directa I. Requieren el uso de $(N+M)$ elementos de memoria, $(N+M)$ sumadores y $(N+M+1)$ multiplicadores.
- ❑ Esta última forma sugiere la eliminación M elementos de memoria, ya que están repetidos. El diagrama resultante se denomina forma directa II.

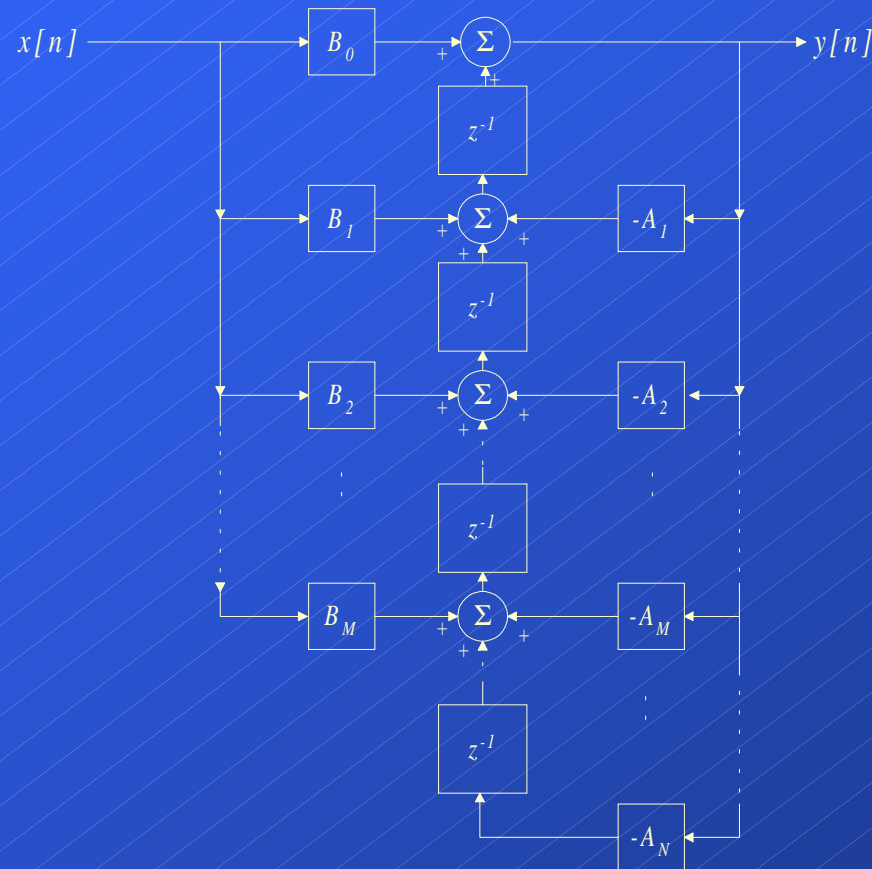
Realización de Filtros Digitales



Forma directa II

- De la forma directa II pasamos a la forma transpuesta o canónica. Consiste en sustituir los nodos por sumas, las sumas por nodos, invertir el sentido de las flechas y finalmente intercambiar los coeficientes y $x[n]$ e $y[n]$.

Realización de Filtros Digitales



Forma Transpuesta o
Canónica

- Esta forma da lugar a una realización con N elementos de memoria, $(N+M+1)$ multiplicadores y N sumadores.

Realización de Filtros Digitales

- Las funciones de Transferencia pueden representarse mediante sumas de fracciones parciales (en paralelo), o mediante producto de factores (en cascada).

- Supongamos un función de Transferencia $H(z)$, expresada como suma de fracciones. Los pares de polos complejos conjugados deben ser combinados en términos de 2° orden,

$$H(z) = b_0 + \sum_{i=1}^L \frac{b_{1i}z^{-1} + b_{0i}}{a_{2i}z^{-2} + a_{1i}z^{-1} + 1} \quad L = \text{int}\left(\frac{N+1}{2}\right)$$

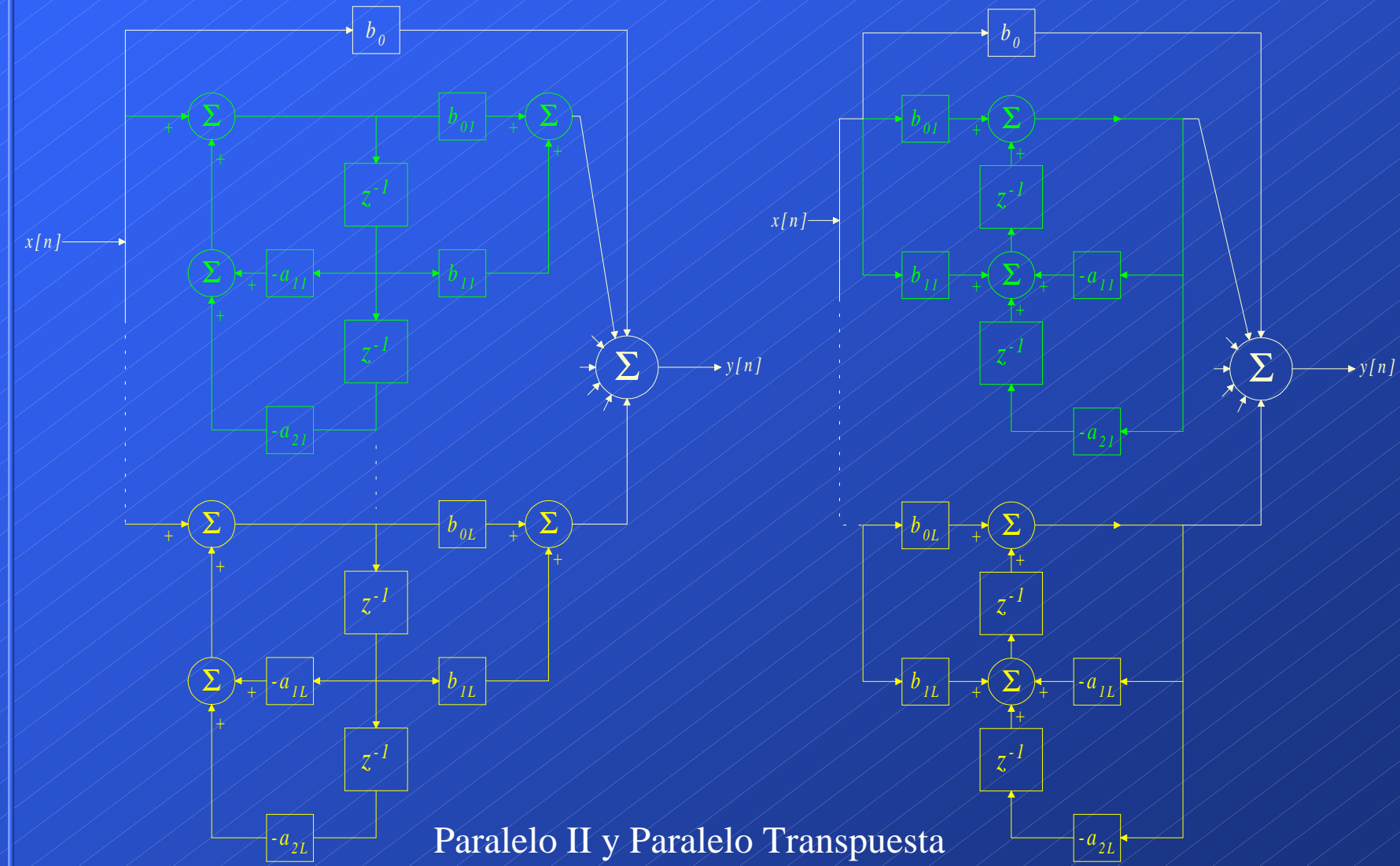
- Esta función de Transferencia puede representarse a través de la forma directa II o de la forma Transpuesta, dando lugar a las formas Paralela II y Paralela Transpuesta (Figura T13).

- La función de Transferencia puede expresarse también como producto de términos,

$$H(z) = \beta_0 \prod_{i=0}^L H_i(z) = \beta_0 \prod_{i=0}^L \frac{\beta_{2i}z^{-2} + \beta_{1i}z^{-1} + 1}{\alpha_{2i}z^{-2} + \alpha_{1i}z^{-1} + 1}$$

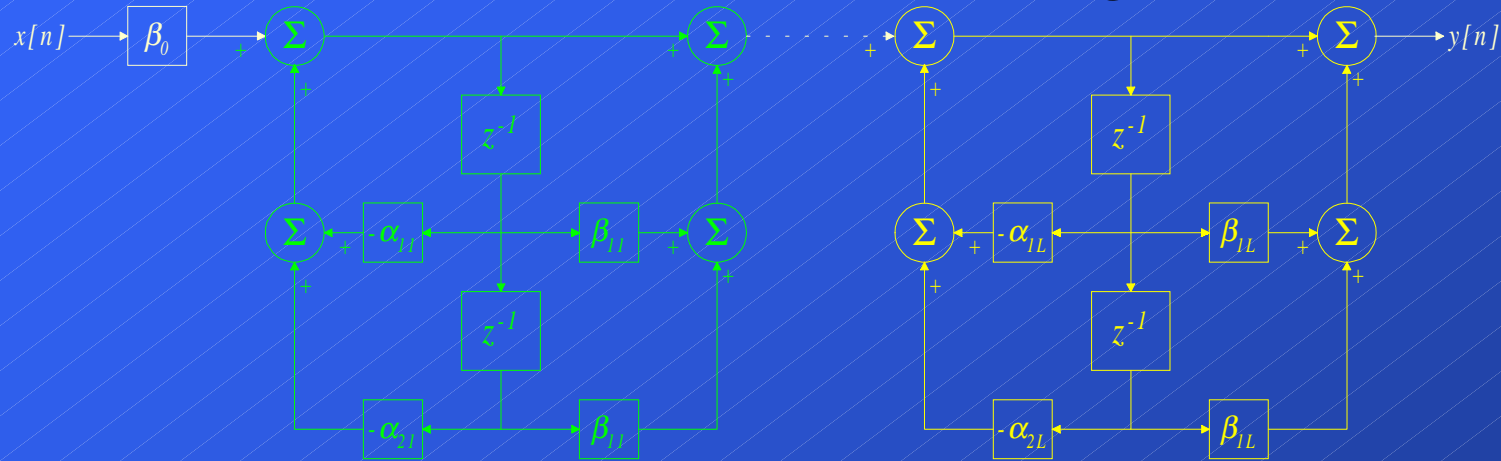
lo que da lugar a las formas en cascada, Cascada II y Cascada Transpuesta (Figura T14).

Realización de Filtros Digitales

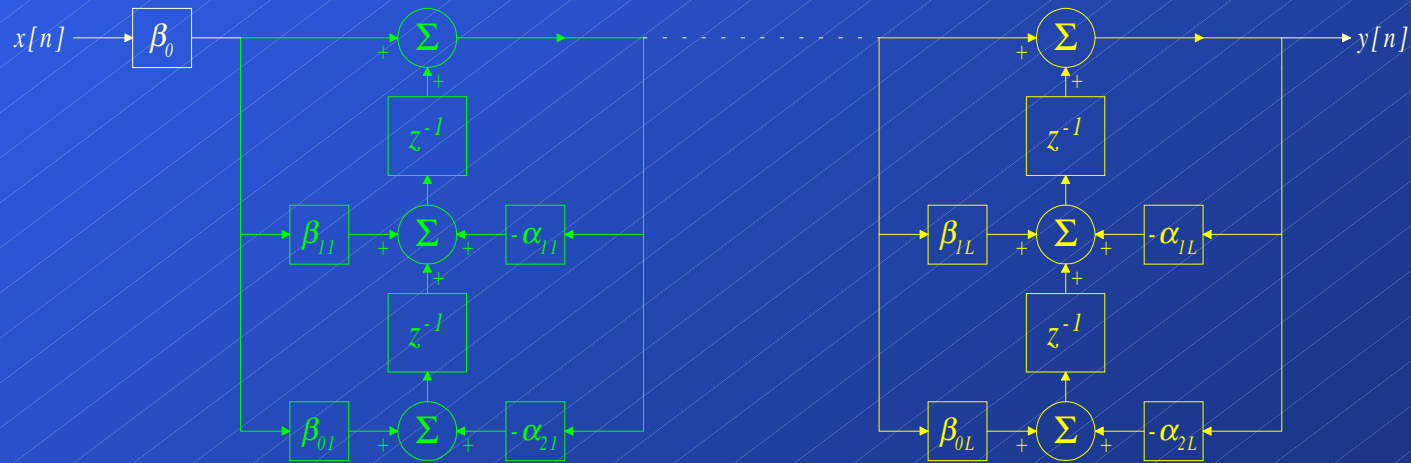


Paralelo II y Paralelo Transpuesta

Realización de Filtros Digitales



Forma en Cascada II



Forma en Cascada Transpuesta

Efectos de Cuantización

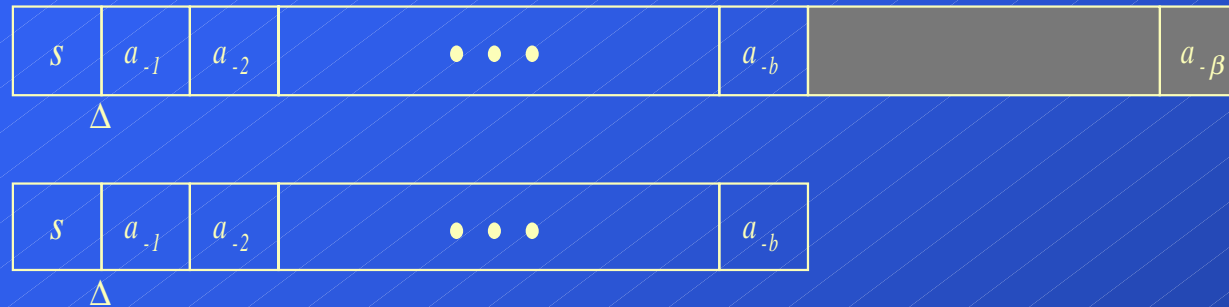
- ❑ Cada una de estas formas tienen sus ventajas e inconvenientes en el momento de realizar el filtro. Uno de los problemas más importantes que debe tener en cuenta una realización son los efectos de cuantización.
- ❑ Los efectos de cuantización se producen al tener obligatoriamente que truncar (o cuantizar) los coeficientes del filtro y las señales de entrada y salida. Esta cuantización puede dar lugar a que las características del filtro realizado difieran de las especificaciones del filtro diseñado.
- ❑ Los efectos de cuantización deben ser tenidos muy en cuenta cuando el diseño se realiza en microprocesadores con aritmética de punto fijo (por ejemplo, DSPs). En caso de utilizar micros de 32 bits con aritmética en punto flotante, los efectos de cuantización pueden ser despreciados.
- ❑ Dividiremos los efectos de cuantización en dos partes: los debidos a la cuantización de las señales (de entrada $x[n]$ o de salida $y[n]$, que incluyen los errores de redondeo o truncamiento en las operaciones aritméticas) y los debidos a la cuantización de los coeficientes.

Efectos de Cuantización

- Cuantización de señales : El efecto de cuantizar la señal puede estudiarse como el efecto de añadir un error o una señal de ruido $e[n]$ a la salida ideal del filtro digital. Este ruido se considera como el efecto conjunto de varios errores producidos en el procesamiento:
- Error de cuantización en el convertidor A/D a la entrada del filtro.
 - ◆ Errores de redondeo o truncamiento en las operaciones (multiplicaciones, sumas).
 - ◆ Error en la cuantización de la salida en el convertidor D/A (menos bits en la salida que en las operaciones).

Errores de Truncamiento: Cuando se implementa un filtro en hardware (DSPs o ASICs) suele ser habitual trabajar en un punto fijo ya que es considerablemente más barato en términos de área de silicio y complejidad en el diseño. Por ejemplo, las variables del filtro (entradas, salidas y coeficientes) pueden estar cuantizadas en 16 bits. Al hacer una multiplicación necesitaremos 32 bits, que es posteriormente truncado de nuevo a 16 bits. Este tipo de error puede ser analizado mejor desde un punto de vista estadístico. Supongamos que truncamos un número de $(\beta+1)$ bits a $(b+1)$ bits, tal y como indica la figura.

Efectos de Cuantización



- El error producido al truncar x es: $\varepsilon_t = Q(x) - x$. El error es cero si todos los bits rechazados son cero, y será máximo si todos los bits rechazados son 1. El error máximo es por tanto,

$$\sum_{i=b+1}^{\beta} a_{-i} 2^{-i} = 2^{-b} + 2^{-\beta}$$

El error de truncamiento será siempre negativo para números en complemento de 2, y su valor es $-(2^{-b} - 2^{-\beta}) \leq \varepsilon_t \leq 0$. Si suponemos que $\beta \gg b$,

$$-2^{-b} \leq \varepsilon_t \leq 0$$

A la hora de analizar los errores producidos por el truncamiento se recurre al análisis estadístico. Suponiendo una distribución uniforme de los errores en el rango $(-2^{-b}, 0)$, la media del error es $-2^{-(b+1)}$ y su varianza es $2^{-b}/12$. Estos valores son ciertos en caso de utilizar complemento de 2, lo cual es bastante habitual.

Efectos de Cuantización

- El análisis se hace sumando una señal de ruido a la señal sin truncar. Esa señal de error tiene la media y varianzas calculadas previamente.



Los tipos de realizaciones estudiadas (en paralelo y cascada) tienen un impacto parecido en los errores de truncamiento. Utilizando formas en cascada podemos mejorar la varianza del ruido de truncamiento puede disminuirse emparejando polos y ceros de acuerdo a ciertos criterios y modificando el orden al cual se realizan las operaciones en cascada.

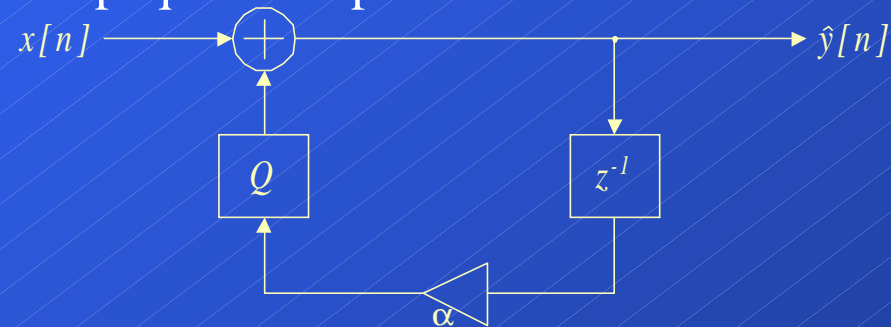
Una consecuencia de las operaciones aritméticas es el overflow, es decir cuando el resultado de una operación rebasa el máximo número admitido por una cierta representación digital. En tal caso la señal debe mantenerse en ese nivel máximo, lo que produce fuertes distorsiones en las señales.

Efectos de Cuantización

- Una ventaja de la aritmética de complemento de 2 es la mostrada en el siguiente ejemplo. Se quiere hacer la suma $0.6875_{10} + 0.8125_{10} - 0.5625_{10}$ con un código digital de 5 bits en complemento de 2. La suma de los dos primeros operando da overflow. Sin embargo, si eliminamos el bit de signo y continuamos sumando el resultado será correcto:
- $01011 + 01101 = 11000$ Eliminamos el bit de signo porque hay overflow 01000
 $01000 + 10111 = 01111$ Lo que equivale a 0.9375 , el resultado correcto.
- Una forma de evitar el overflow multiplicar las operaciones por un factor que evite el overflow. Este factor debe ser lógicamente menor que 1, lo que empeora relación señal ruido del filtro. Vimos en el capítulo 5 que la relación señal-ruido en una señal cuantizada es, $SNR_Q = 10 \log P_s + 10.8 - 20 \log D + 6b$
- P_s es la potencia de la señal de entrada, D es el fondo de escala y b es el número de bits. P_s es proporcional a la varianza de la señal, σ_x^2 . Si multiplicamos la señal por un factor A , la potencia de la señal será $A^2 \sigma_x^2$. Sustituyendo en la ecuación vemos que si $A > 1$, mejora la SNR_Q , pero se corre el peligro de producir overflow. Por el contrario, un valor de $A < 1$, evita el overflow pero empeora la SNR_Q .

Efectos de Cuantización

- ❑ Oscilaciones de ciclo límite : Este tipo de inestabilidades se da en los filtros IIR, debido a la realimentación de la salida. Las oscilaciones de ciclo límite se denominan a menudo granulares, ya que dan lugar a oscilaciones de pequeña amplitud.



En el ejemplo del filtro de 1^{er} orden de la figura. La ecuación diferencial es

$$\hat{y}[n] = Q(\alpha \cdot \hat{y}[n-1]) + x[n]$$

Sin perder generalidad utilizamos una representación digital de 5 bits con signo y haremos redondeo en las operaciones en vez de truncamiento.

Veamos la respuesta del filtro a una entrada impulso $x[0]=0.1101$.

Efectos de Cuantización

	$\alpha=0.1011, y[-1] = 0$		$\alpha=1.1011, y[-1] = 0$	
n	$\alpha y[n-1]$	$y[n]$	$\alpha y[n-1]$	$y[n]$
0	0	0.1101	0	0.1101
1	0.10001111	0.1001	1.1000111	1.1001
2	0.01100011	0.0110	0.0110001	0.0110
3	0.01000010	0.0100	1.0100001	1.0100
4	0.00101100	0.0011	0.0010110	0.0011
5	0.00100001	0.0010	1.0010000	1.0010
6	0.00010110	0.0001	0.0001011	0.0001
7	0.00001011	0.0001	1.0000101	1.0001
8	0.00001011	0.0001	0.0000101	0.0001

Efectos de Cuantización

- Para los dos valores de α se observa que cuando $\alpha > 0$, la salida tiende a un valor constante distinto de cero, mientras que si $\alpha < 0$, la salida oscila entre dos valores. Esto se debe a que el sistema tiene un polo efectivo en el círculo unidad. En este caso el sistema tiene un polo en $z=1$ para $\alpha > 0$, y en $z=-1$ para $\alpha < 0$. Esto implica

$$Q(\alpha \cdot \hat{y}[n-1]) = \begin{cases} \hat{y}[n-1], & \alpha > 0 \\ -\hat{y}[n-1], & \alpha < 0 \end{cases}$$

El error de cuantización debido al redondeo es

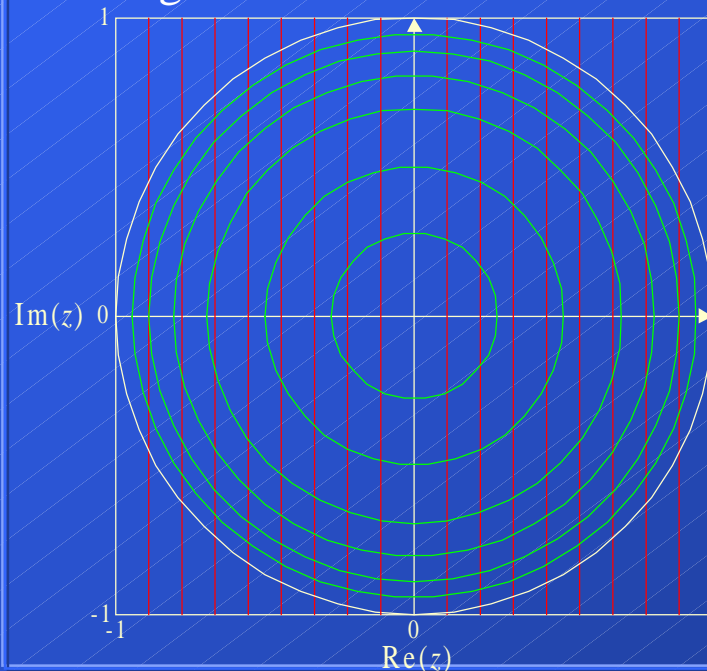
$$|Q(\alpha \cdot \hat{y}[n-1]) - \alpha \cdot \hat{y}[n-1]| \leq \frac{\delta}{2}, \text{ y sustituyendo la anterior expresión,}$$

$$|\hat{y}[n-1]| \leq \frac{\delta}{2(1-|\alpha|)}.$$

Con esta fórmula tenemos delimitado el rango de variación de los ciclos límite. Por tanto, utilizar un mayor número de bits de cuantización disminuye la amplitud del ciclo límite. Esto no es siempre posible, por lo que existen técnicas de eliminación de los ciclos límites basadas en realizaciones en espacio de estado.

Efectos de Cuantización

- Cuantización de coeficientes : Trataremos de investigar el impacto de la cuantización de los coeficientes del filtro en la función de Transferencia del mismo.
- Supongamos una función de Transferencia $H(z)=1/(1+a_1 \cdot z^{-1}+a_2 \cdot z^{-2})$, cuyos polos complejos son p y p^* cumpliéndose que $a_1 = -2 \cdot \text{Re}(p)$ y $a_2 = |p|^2$. Por tanto, cuantizar a_1 significa cuantizar la parte real del polo, mientras que cuantizar a_2 significa cuantizar el radio del polo. Esto se muestra en la figura.



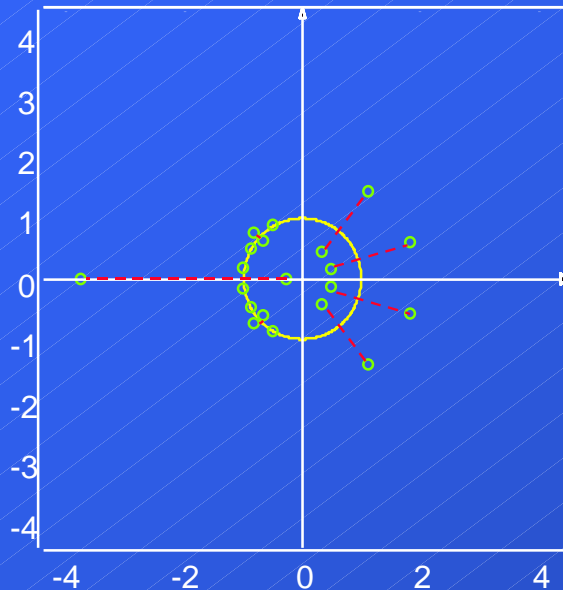
El polo estará definido por la intersección de las líneas verticales y los círculos. Se pueden sacar dos conclusiones:

- ◆ En la vecindad de $z = \pm 1$, los posibles polos están más separados entre sí. Se dice que estos polos son muy sensibles a la cuantización.
- ◆ Si muestreamos a una frecuencia mucho mayor que la señalada por el teorema del muestreo, los polos son empujados hacia $z = 1$. Por tanto, aumentar la frecuencia de muestreo hace a los polos más sensibles a la cuantización.

Efectos de Cuantización

- ◆ Para sistemas de mayor orden es previsible que la sensibilidad de los coeficientes vaya a peor, por lo que realizar filtros que contengan polos (filtros IIR) de forma directa (formas directa I o II) no es aconsejable. Los coeficientes en estas formas no nos dicen nada acerca de la situación de sus polos y mucho menos acerca de las consecuencias de su cuantización.
- ◆ La solución es, lógicamente, utilizar las formas Paralelo o Cascada para tener un mayor control sobre la situación de los polos al cuantizar.
- ◆ Estudiamos ahora el efecto de la cuantización sobre los ceros. En el caso de filtros FIR (compuesto exclusivamente por ceros), sabemos que se caracterizan por ser de fase lineal. Esto es debido a que los coeficientes son simétricos (o asimétricos). Por tanto cuantizar los coeficientes no va a variar la linealidad de fase del filtro.
- ◆ Lo que sí variará es la magnitud de la respuesta. Se puede demostrar que los ceros de un filtro FIR o bien están sobre la circunferencia de radio 1 , o están en parejas con radios recíprocos (ver figura T24). Por tanto, en los filtros FIR lo normal es utilizar la forma directa (I o II). También se podría utilizar la forma en cascada pero se utiliza menos.

Efectos de Cuantización



- ◆ El hecho de que los polos en un filtro digital estén sobre el círculo unidad, hace que los coeficientes del numerador sean $+1$ ó -1 , por lo que en las formas en cascada se pueden ahorrar desde un 25 a un 50% en multiplicaciones comparado con un diseño en paralelo.
- ◆ Los polos en un filtro IIR son más problemáticos. Aquí no tenemos la seguridad de que los coeficientes son simétricos, por lo que habrá que tener las mismas consideraciones que las explicadas en el caso de los polos. Aquí además hay que tener en cuenta el caso habitual en que varios polos estén en $z=\pm 1$ (ver transformaciones bilineales), por lo que su cuantización no tendría efectos graves. Lo más normal es utilizar la forma en cascada, aunque se puede utilizar la forma en paralelo siempre que no haya unas especificaciones demasiado exigentes.

Efectos de Cuantización

- Análisis de cuantización de coeficientes en los filtros FIR.

La función de transferencia de un filtro FIR de orden $M-1$ es : $H(z) = \sum_{n=0}^{M-1} h[n] \cdot z^{-n}$
 La cuantización de los coeficientes resulta en una nueva función de transferencia,

$$\hat{H}(z) = \sum_{n=0}^{M-1} \hat{h}[n] \cdot z^{-n} = \sum_{n=0}^{M-1} (h[n] + e[n]) \cdot z^{-n} = H(z) + E(z)$$

Calculamos el límite superior de la función de transferencia en el círculo unidad:

$$|E(e^{j\omega})| = \left| \sum_{n=0}^{M-1} e[n] \cdot e^{-j\omega n} \right| \leq \sum_{n=0}^{M-1} |e[n]| \cdot |e^{-j\omega n}| \leq \sum_{n=0}^{M-1} |e[n]|$$

En el peor caso en que todos los errores de redondeo sean máximos, el error del filtro está acotado por

$$|E(e^{j\omega})| \leq \frac{M\delta}{2}$$

Este límite es bastante pesimista. Un límite más realista se obtiene si suponemos que la señal $e[n]$ es una variable aleatoria con varianza σ_e^2 . La varianza de $E(e^{j\omega})$ se puede demostrar que es ($\sigma_e^2 = \delta^2/12$),

$$\sigma_E^2(\omega) = \sigma_e^2 \cdot \left(M - 1 + \frac{\sin M\omega}{\sin \omega} \right) = \frac{\delta^2}{12} \left(M - 1 + \frac{\sin M\omega}{\sin \omega} \right) \rightarrow \sigma_E(\omega) \leq \delta \sqrt{\frac{2M-1}{12}}$$

Efectos de Cuantización

- ◆ Haremos ahora un análisis de sensibilidad de los polos y ceros de la función de Transferencia $H(z)$ a pequeñas variaciones en los coeficientes β_i y α_j del filtro diseñado.

$$H(z) = \frac{\beta_m z^m + \beta_{m-1} z^{m-1} + \dots + \beta_1 z + \beta_0}{\alpha_n z^n + \alpha_{n-1} z^{n-1} + \dots + \alpha_1 z + \alpha_0}$$

- ◆ Los polos vienen dados por las raíces del denominador y los polos por las raíces del numerador:

$$N(z) = \beta_m z^m + \beta_{m-1} z^{m-1} + \dots + \beta_1 z + \beta_0 = 0$$

$$D(z) = \alpha_n z^n + \alpha_{n-1} z^{n-1} + \alpha_1 z + \alpha_0 = 0$$

- ◆ Desarrollaremos la sensibilidad de las raíces a la cuantización de los coeficientes. El método es aplicable también a los ceros. Los polos de $D(z)$ son λ_j , $j=1,2,\dots,n$, pero si uno de los coeficientes α_k varía en una cantidad $\delta\alpha_k$, habrá un cambio $\delta\lambda_j$ en las raíces del polinomio.

$$D(z, \alpha_k) = \alpha_n z^n + \alpha_{n-1} z^{n-1} + \dots + \alpha_k z^k + \dots + \alpha_1 z + \alpha_0 = 0$$

Efectos de Cuantización

- ◆ Desarrollamos esta última expresión en una serie de Taylor alrededor de los valores nominales λ_{j0} y α_{k0} :

$$D(z, \alpha_k) = D(\lambda_k, \alpha_{k0}) + \left. \frac{\partial D(z, \alpha_k)}{\partial z} \right|_{\substack{z=\lambda_k \\ \alpha_k=\alpha_{k0}}} \delta\lambda_k + \left. \frac{\partial D(z, \alpha_k)}{\partial \alpha_k} \right|_{\substack{z=\lambda_k \\ \alpha_k=\alpha_{k0}}} \delta\alpha_k + \dots$$

- ◆ Particularizando para $z = \lambda_{j0}$,

$$0 = 0 + \left. \frac{\partial D(z, \alpha_k)}{\partial z} \right|_{\substack{z=\lambda_{j0} \\ \alpha_k=\alpha_{k0}}} \delta\lambda_j + \left. \frac{\partial D(z, \alpha_k)}{\partial \alpha_k} \right|_{\substack{z=\lambda_{j0} \\ \alpha_k=\alpha_{k0}}} \delta\alpha_k$$

- ◆ Determinamos la variación de la raíz λ_j debido a un cambio en α_k .

$$\delta\lambda_j = - \left. \frac{\frac{\partial D(z, \alpha_k)}{\partial \alpha_k}}{\frac{\partial D(z, \alpha_k)}{\partial z}} \right|_{\substack{z=\lambda_{j0} \\ \alpha_k=\alpha_{k0}}} \delta\alpha_k = - \frac{\lambda_{j0}^k}{\prod_{i=1, i \neq k}^n (\lambda_{j0} - \lambda_i)} \delta\alpha_k$$

Efectos de Cuantización

- ◆ De la ecuación anterior se puede sacar como conclusión
 - ◆ El coeficiente más sensitivo será siempre α_0 (β_0).
 - ◆ La sensibilidad aumenta cuando las raíces se encuentran muy agrupadas, incrementándose fuertemente si el orden del sistema n es grande.

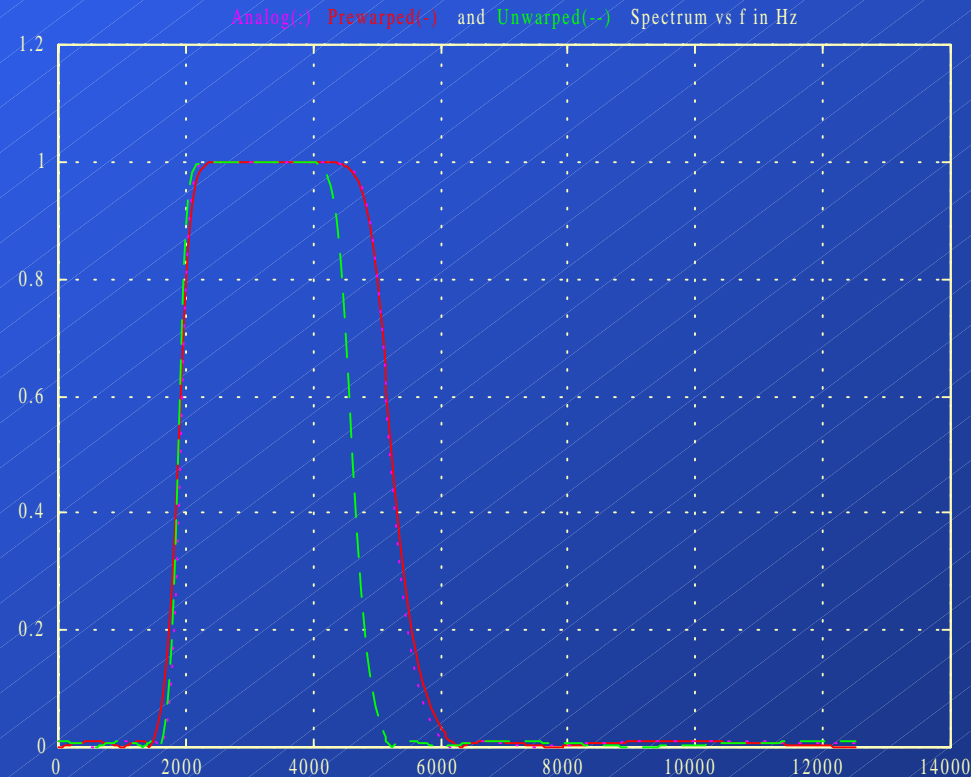
Efectos de Cuantización

□ Ejemplos en MATLAB

Dado el filtro digital IIR con $F_s = 25 \text{ KHz}$, cuyos coeficientes B y A son

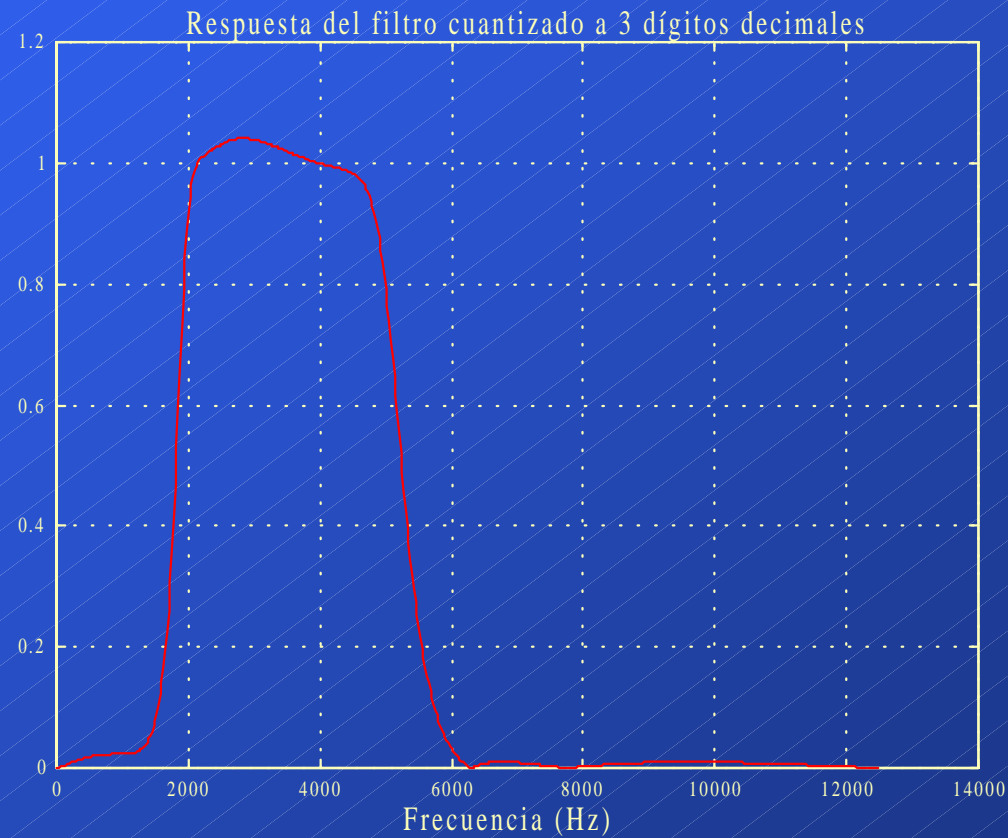
```
>> B = [0.0280 -0.0850 0.1037 -0.0942 0.0782 0 -0.0782 0.0942 -0.1037 0.0850 -
0.0280]
```

```
>> A = [1.0000 -5.0041 12.5162 -20.3913 23.8604 -20.8072 13.6498 -6.6537 2.3175 -
0.5229 0.0594]
```



Efectos de Cuantización

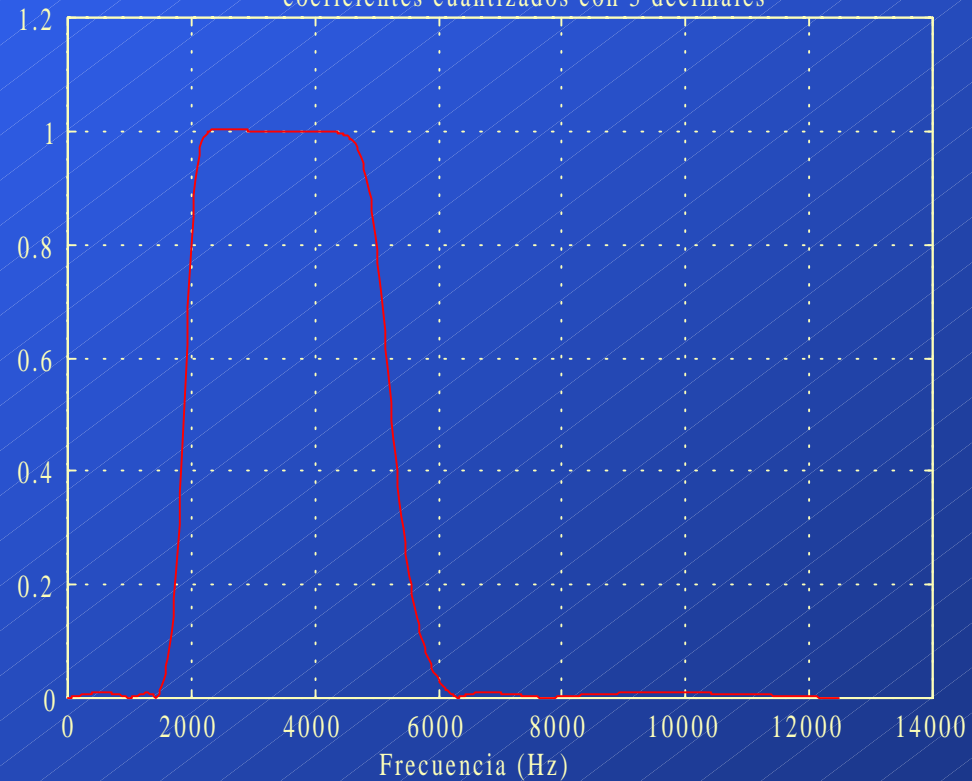
```
>> Bq = round(B*1000)/1000;  
>> Aq = round(A*1000)/1000;  
>> [H F]=freqz(Bq,Aq,500,25000);plot(F,abs(H));grid;
```



Efectos de Cuantización

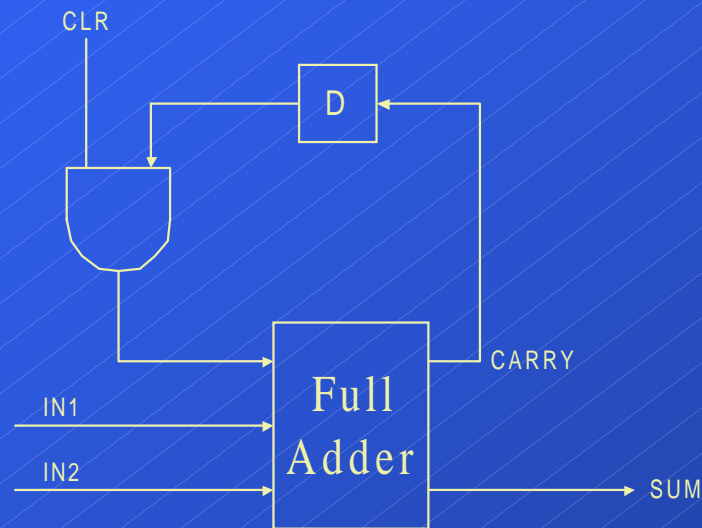
```
>> [Z P K]=tf2zp(B,A);  
>> SOS = zp2sos(Z,P,K,'down');SOSq=round(SOS*1000)/1000;M=size(SOS,1);  
>> Ht=ones(500,1);for i=1:M;Bsosq=SOSq(i,1:3);Asosq=SOSq(i,4:6);[H  
F]=freqz(Bsosq,Asosq,500,25000);Ht=H.*Ht;end;  
>> plot(F,abs(Ht));grid;
```

Realización de filtro con Secciones de 2º Orden y
coeficientes cuantizados con 3 decimales

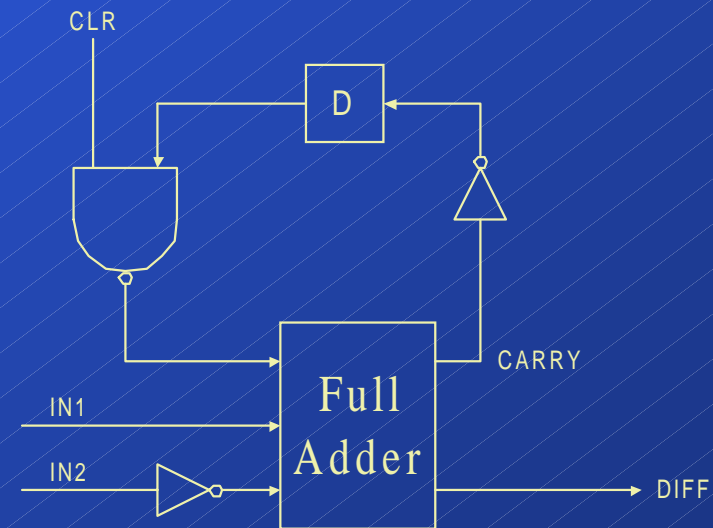


Implementación Hardware

Las estructuras más simples para sumar y restar son las estructuras en serie. El tamaño del circuito es mínimo pero requiere de N ciclos de reloj para producir el resultado (N es el tamaño de los datos en bits). Sin embargo, un circuito tan pequeño (retrasos pequeños) permite utilizar un reloj de muy alta frecuencia, lo cual compensa el hecho de necesitar N ciclos.



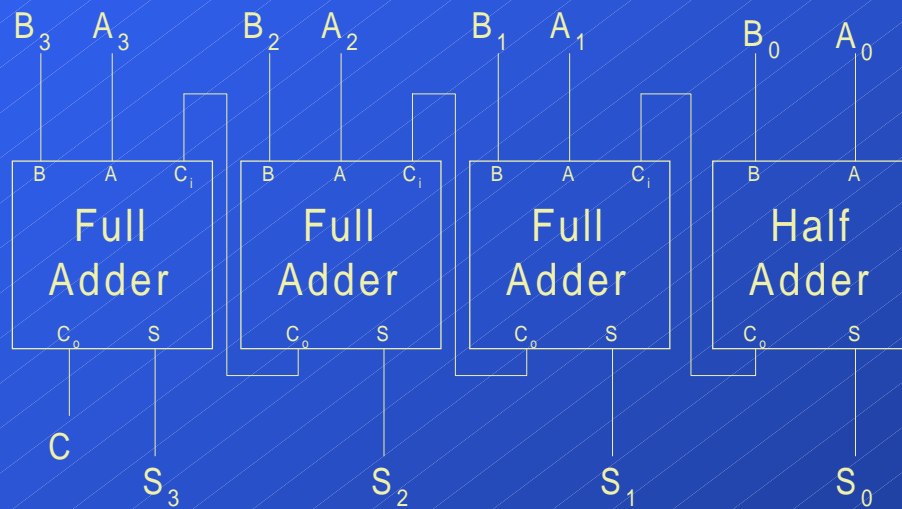
Sumador Serie



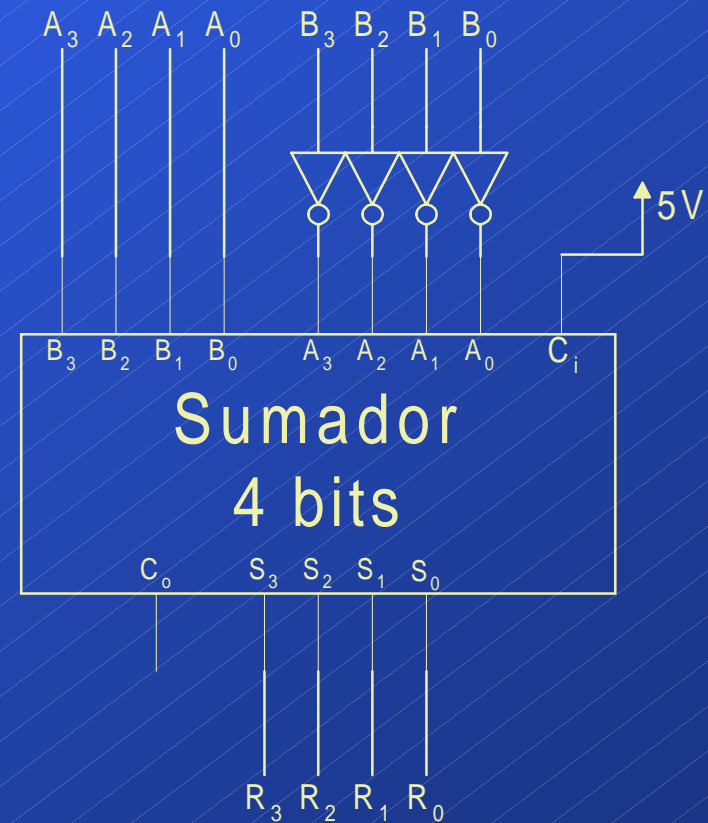
Restador Serie

Implementación Hardware

Los sumadores (restadores) en paralelo más sencillos son los que propagan las llevadas (*carry propagate* o *ripple carry*) de un bit a otro. Como contrapartida producen los retrasos más largos.



Sumador Paralelo (Carry Propagate)



Restador Paralelo (Carry Propagate)

Implementación Hardware

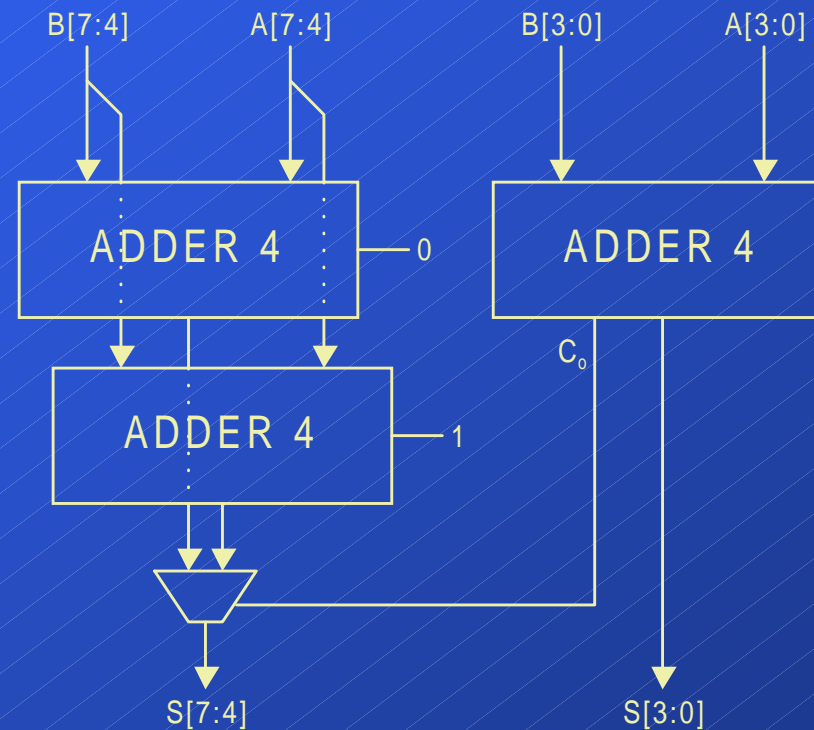
- Existen varias formas de acelerar la realización de una suma de varios bits. La más conocida es el “*carry look-ahead*”. Aquí la llevada es de alguna forma precalculada a partir de los bits de entrada, lo que acelera el cálculo de la suma pero añade más transistores ($\propto N^2$). Esto hace que para valores grandes de N , el *carry look-ahead* no sea eficiente.

$$\begin{aligned}
 C_o &= A \cdot B + C_i \cdot (A + B) \\
 C_g &= A \cdot B; C_p = A + B \quad \longrightarrow \\
 C_o &= C_g + C_i \cdot C_p
 \end{aligned}$$

$$\begin{aligned}
 C_{o0} &= C_{g0} \\
 C_{o1} &= C_{g1} + C_{i1} \cdot C_{p1} = C_{g1} + C_{o0} \cdot C_{p1} = C_{g1} + C_{g0} \cdot C_{p1} \\
 C_{o2} &= C_{g2} + C_{i2} \cdot C_{p2} = C_{g2} + C_{o1} \cdot C_{p2} = \\
 &= C_{g2} + (C_{g1} + C_{g0} \cdot C_{p1}) \cdot C_{p2} \\
 &= C_{g2} + C_{g1} \cdot C_{p2} + C_{g0} \cdot C_{p1} \cdot C_{p2} \\
 C_{o3} &= C_{g3} + C_{i3} \cdot C_{p3} = C_{g3} + C_{o2} \cdot C_{p3} = \\
 &= C_{g3} + [C_{g2} + (C_{g1} + C_{g0} \cdot C_{p1}) \cdot C_{p2}] \cdot C_{p3} = \\
 &= C_{g3} + C_{g2} \cdot C_{p3} + C_{g1} \cdot C_{p2} \cdot C_{p3} + C_{g0} \cdot C_{p1} \cdot C_{p2} \cdot C_{p3}
 \end{aligned}$$

Implementación Hardware

- ❑ *Carry Select Adders*: Los datos se dividen en grupos de bits y se realizan las sumas para cada uno de esos grupos en las dos condiciones posibles: llevada inicial $C=0$ y $C=1$. Una vez obtenido el resultado de C , se elige el resultado correcto mediante un multiplexor.

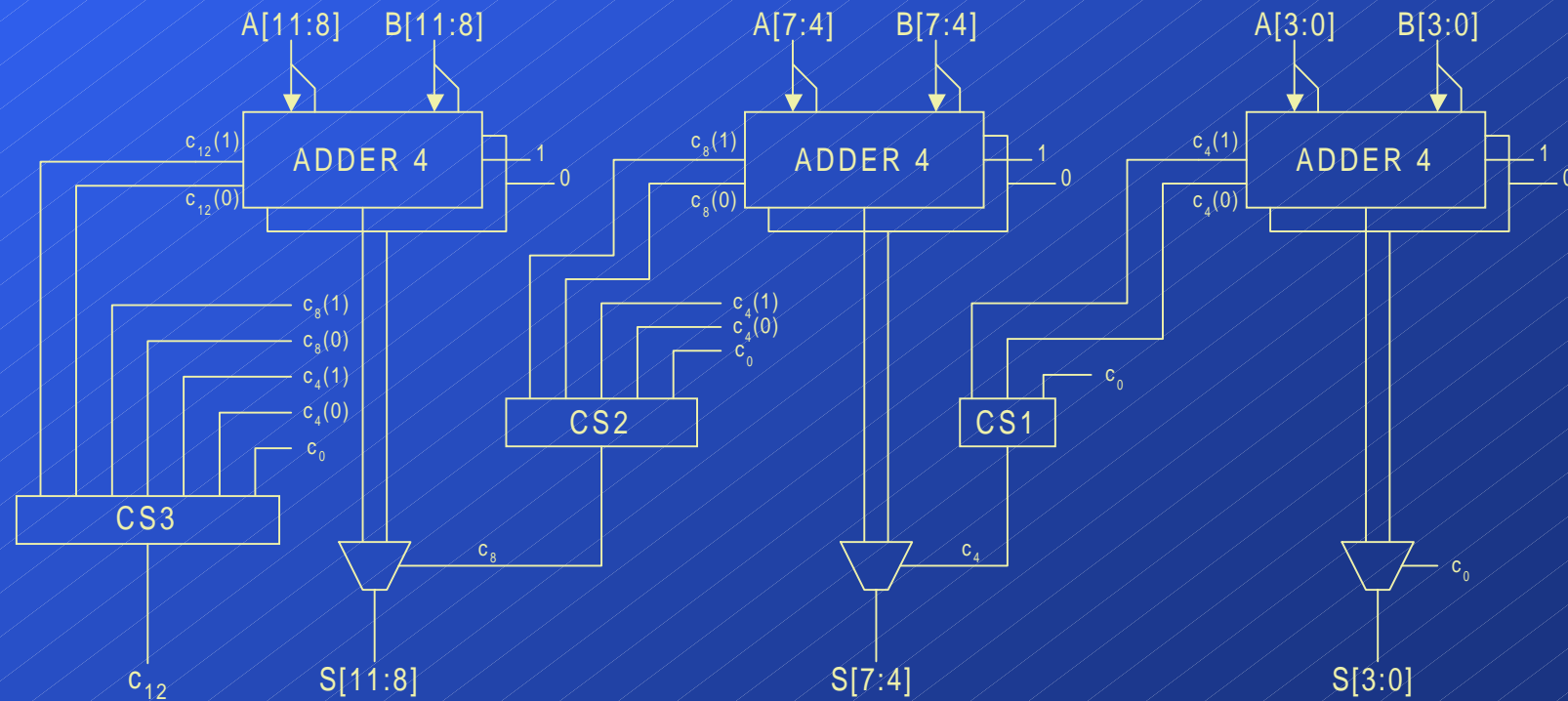


Implementación Hardware

- Una generalización de la estructura anterior se muestra en la siguiente figura. Los bloques CS es un circuito combinatorial que realiza la función

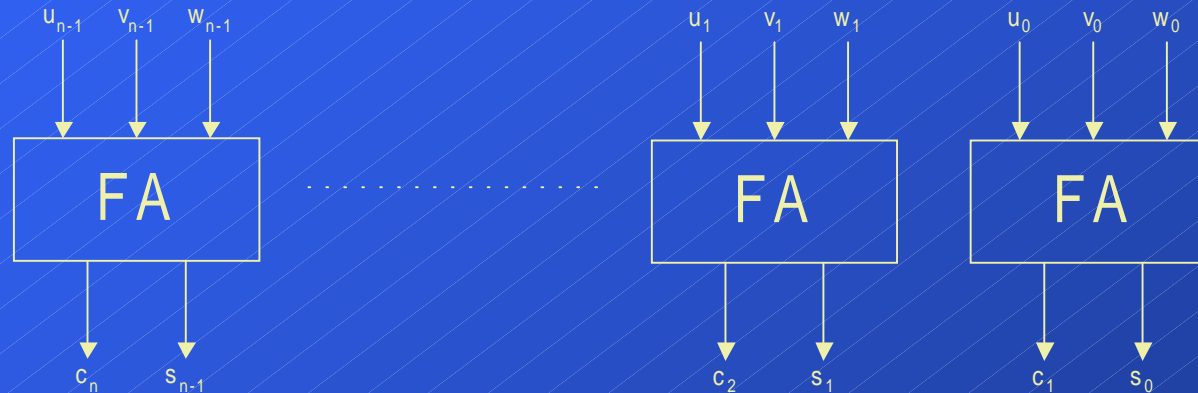
$$c_{(k+1)m} = c_{(k+1)m}(0) + c_{(k+1)m}(1) \cdot c_{km}$$

donde $c_{(k+1)m}(0)$ es la llevada resultante cuando $c_{km}=0$, y $c_{(k+1)m}(1)$ es la llevada resultante cuando $c_{km}=1$.



Implementación Hardware

- *Carry Save Adder (CSA)*: Se utilizan para sumar un gran número de operandos. La estructura básica del CSA está formada por N FAs :

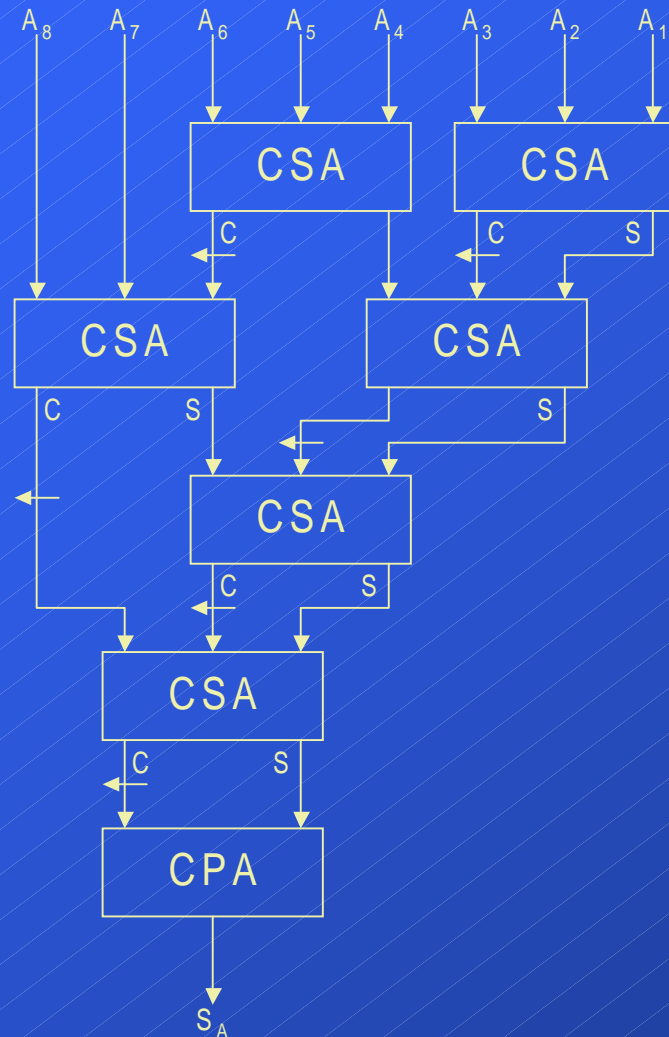


- La suma de tres operandos queda convertida en la suma de dos. Lo único que hay que hacer es multiplicar por 2 el vector C (desplazar un bit a la izda):

$$U+V+W=S+2C$$

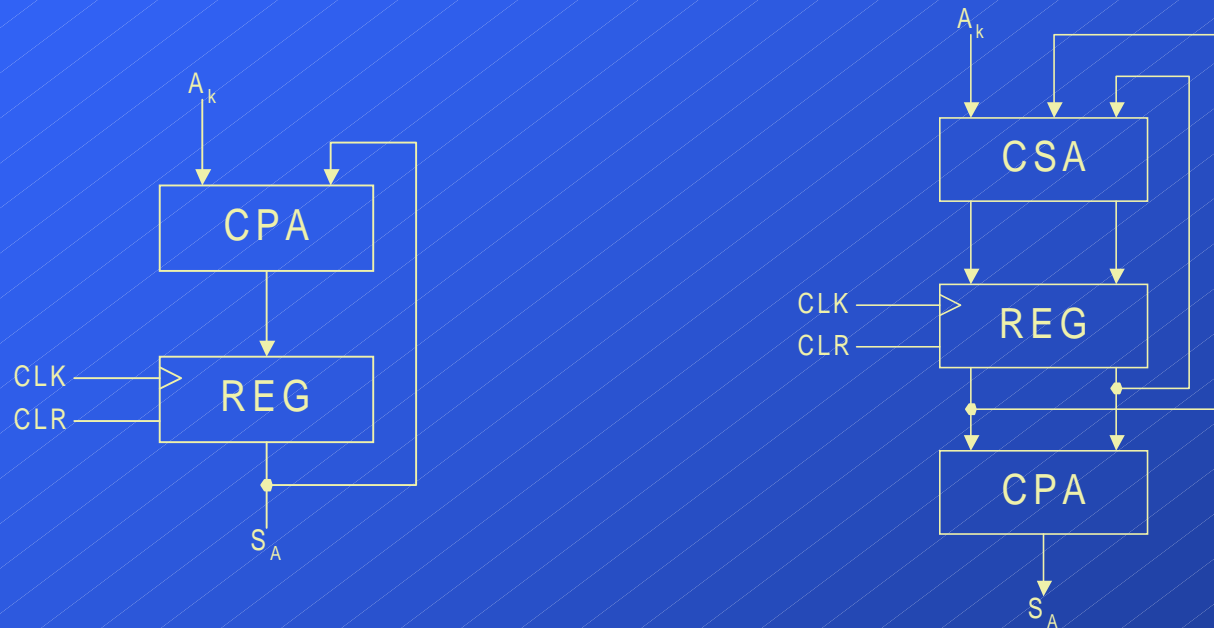
- Cuando hay más de tres operandos se utilizan los CSA hasta que solo queden dos operandos. En este momento se hace una suma normal por cualquiera de los procedimientos descritos anteriormente.

Implementación Hardware



- ❑ Los retrasos son mucho menores ya que en los CSA no hay propagación de la llevada.
- ❑ Al hacer el desplazamiento hacia la izquierda con valores negativos en complemento de 2 debe hacerse la extensión de signo para producir resultados correctos.
- ❑ Otras formas de hacer sumas de varios operandos se muestra en las figuras de la página siguiente.

Implementación Hardware



- La figura de la izquierda muestra el circuito más sencillo formado por un sumador “*carry propagate*” y un registro. El hardware es mínimo pero las prestaciones también (la frecuencia del reloj debe ser pequeña para permitir que el CPA termine la suma). La otra figura muestra un circuito más complejo pero más rápido. Consta de un CSA, un registro de $2N$ bits y un CPA. La frecuencia del reloj podrá ser bastante alta ya que el CSA solo tiene los retrasos debidos a un sumador completo de un bit.

Implementación Hardware

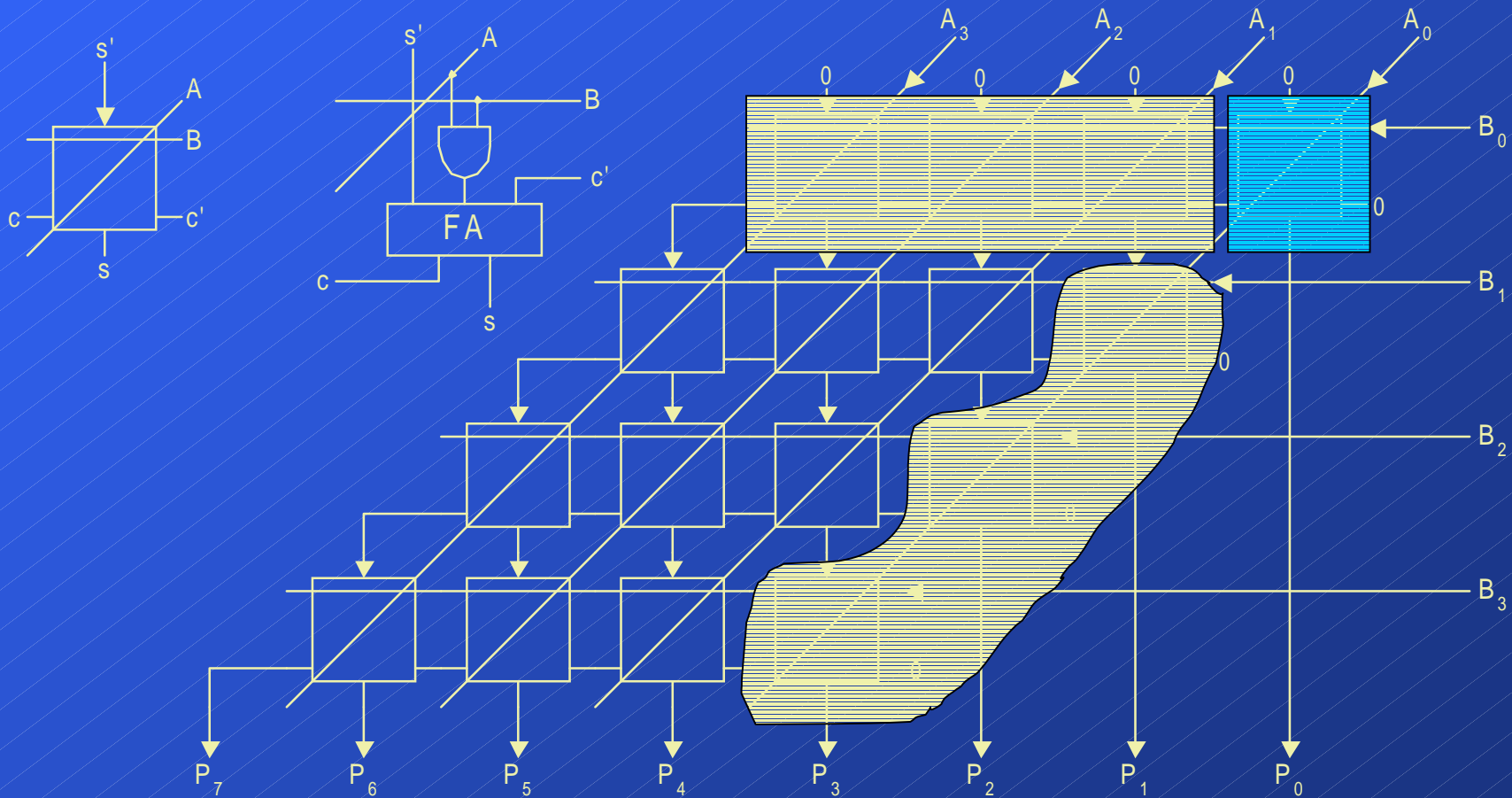
Multiplicadores de números positivos

				A_3	A_2	A_1	A_0
				B_3	B_2	B_1	B_0
				A_3B_0	A_2B_0	A_1B_0	A_0B_0
			A_3B_1	A_2B_1	A_1B_1	A_0B_1	
		A_3B_2	A_2B_2	A_1B_2	A_0B_2		
	A_3B_3	A_2B_3	A_1B_3	A_0B_3			
P_7	P_6	P_5	P_4	p_3	P_2	P_1	P_0

El Multiplicador más sencillo es el llamado “*Ripple Carry Multiplier*”, que se deriva de la figura anterior. La estructura a que da lugar es bastante regular (formada por la repetición de una celda compuesta por una puerta *AND* y un *FA*) pero requiere la propagación de la llevada y de las sumas parciales a través de todo el circuito. Las zonas sombreadas de la figura son las correspondientes a las celdas que no tienen porqué ser *FA*. Para una multiplicación de dos números de N y M bits respectivamente se necesitan NM puertas *AND*, $(N+M-2)$ semisumadores (*HA*) y $(NM-N-M+1)$ *FAs*. El retraso en el cálculo de la multiplicación es aproximadamente 4 veces el retraso de un sumador de N bits.

Implementación Hardware

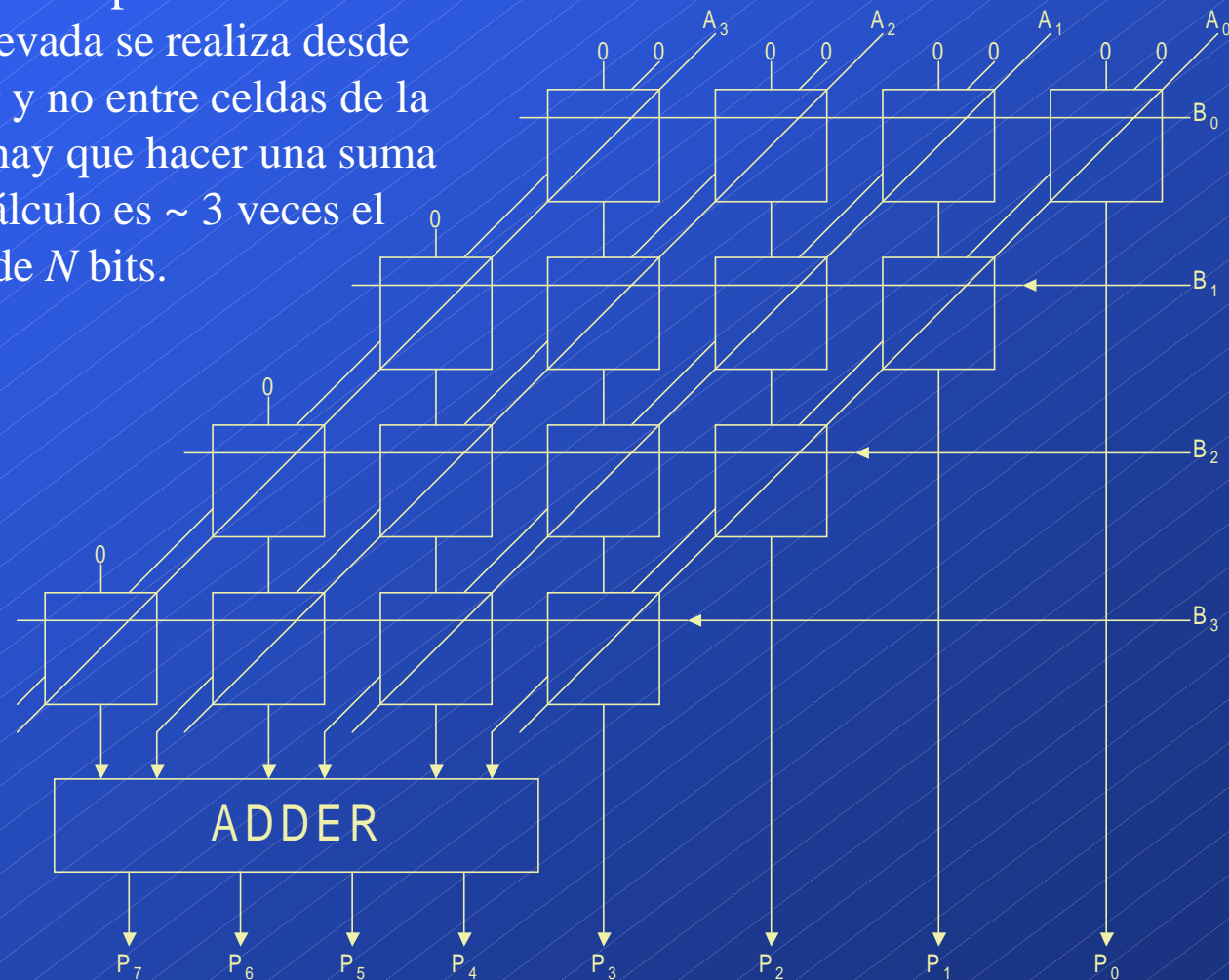
Ripple Carry Multiplier



Implementación Hardware

Carry Save Multipliers

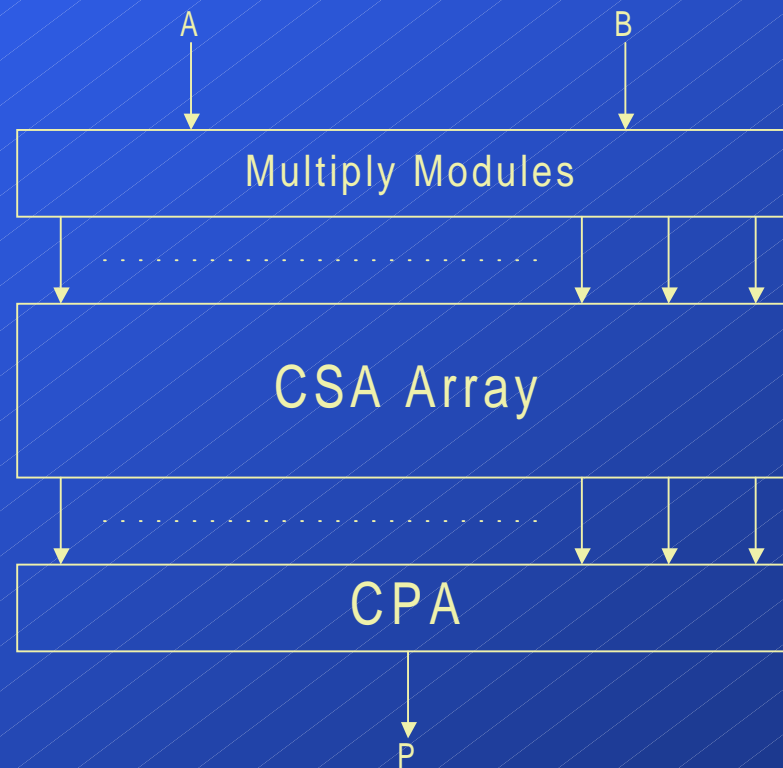
La propagación de la llevada se realiza desde una fila a otra posterior y no entre celdas de la misma fila, por lo que hay que hacer una suma final. El retraso en el cálculo es ~ 3 veces el retraso de un sumador de N bits.



Implementación Hardware

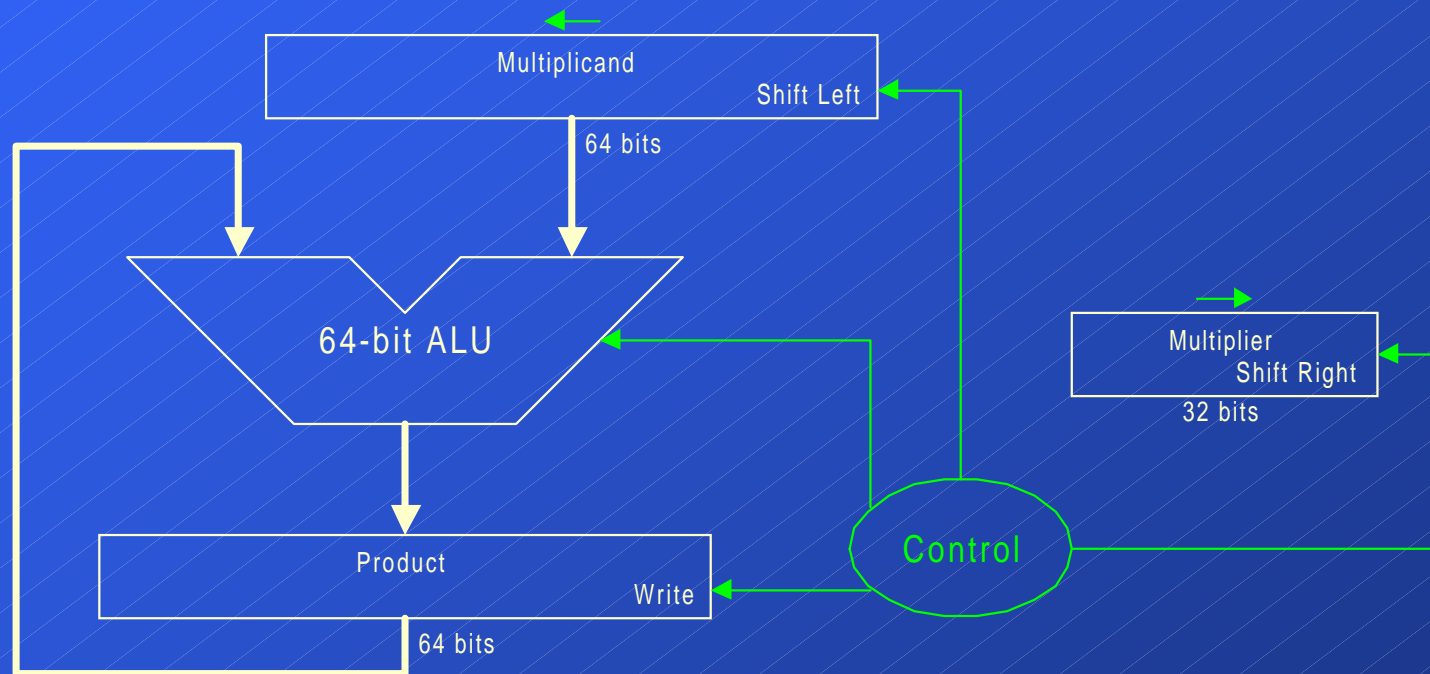
Wallace Tree

Los productos parciales se calculan todos a la vez en la primera etapa. Luego esos productos parciales se suman con una *carry save adder*. Se utilizan para valores grandes de N y el retraso depende del logaritmo de N .



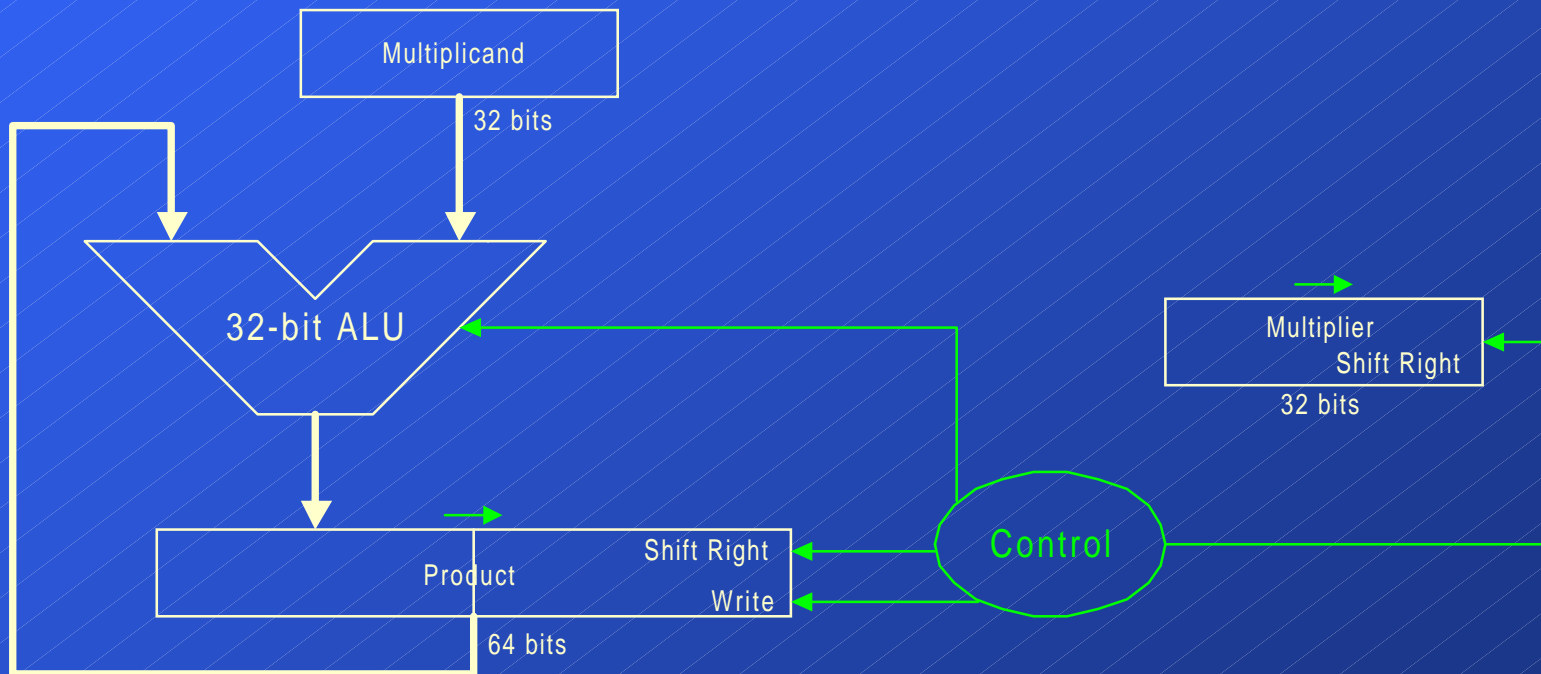
Implementación Hardware

Arquitecturas Standard de Multiplicadores



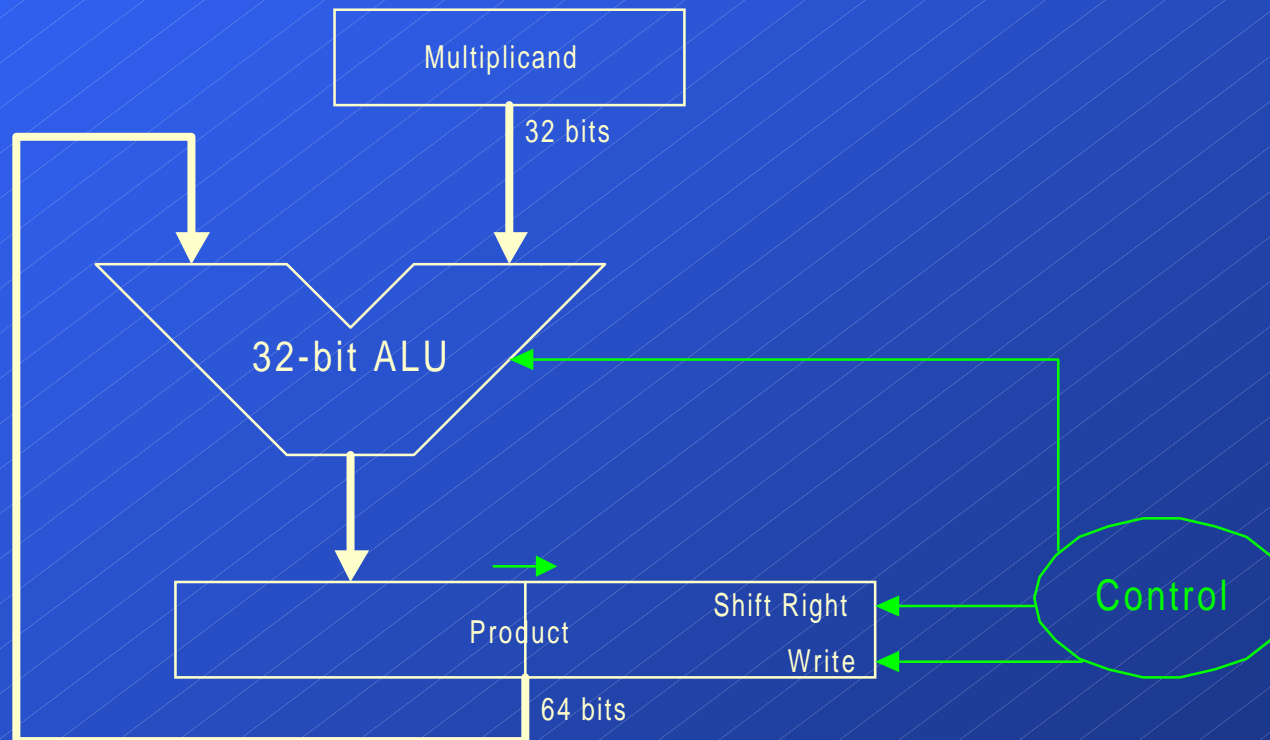
Implementación Hardware

Arquitecturas Standard de Multiplicadores



Implementación Hardware

Arquitecturas Standard de Multiplicadores



Implementación Hardware

Multiplicadores de números con signo

Algoritmo de Booth

El algoritmo de Booth está basado en una representación digital denominada “*Signed Digit*” (*SD*). Para números en base 2, la representación *SD* de un dígito tiene 3 posibles valores (0, 1 y -1), por lo que existe una redundancia (los números se pueden representar de varias formas). Por ejemplo, el valor -9 (10111), se puede representar de las siguientes formas con *SD* de 5 bits:

$$0 \bar{1} 0 0 \bar{1} = -8 -1$$

$$0 \bar{1} 0 \bar{1} 1 = -8 -2 +1$$

$$0 \bar{1} \bar{1} 1 1 = -8 -4 +2 +1$$

$$\bar{1} 0 1 1 1 = -16 +4 +2 +1$$

$$\bar{1} 1 0 0 \bar{1} = -16 +8 -1$$

$$\bar{1} 1 0 \bar{1} 1 = -16 +8 -2 +1$$

$$\bar{1} 1 \bar{1} 1 1 = -16 +8 -4 +2 +1$$

$$A = -a_{n-1}2^{n-1} + \sum_{i=0}^{n-2} a_i 2^i$$

De todas estas representaciones, existe una con el menor número de valores distintos de 0 (forma canónica).

Implementación Hardware

- Un número digital se convierte a *SD* en forma canónica (*CSD*) observando cadenas de *1s* en la representación digital normal (complemento de 2).
- Una cadena de *1s* puede convertirse a *CSD* de la siguiente forma:

$$\begin{array}{l} \dots 001111 \dots 1100 \dots \rightarrow \dots 010000 \dots 0\bar{1}00 \dots \\ \quad 111 \dots 1100 \dots \rightarrow 000 \dots 0\bar{1}00 \dots \end{array}$$
- Utilizando esta recodificación, podemos modificar la forma de multiplicar. Supongamos que queremos multiplicar $A (a_{n-1}, \dots, a_0)$ por B , los pasos a realizar son los siguientes:
 - ◆ Si $a_i=0$ y $a_{i-1}=0$, sumar 0 a P
 - ◆ Si $a_i=0$ y $a_{i-1}=1$, sumar B a P
 - ◆ Si $a_i=1$ y $a_{i-1}=0$, resta B de P
 - ◆ Si $a_i=1$ y $a_{i-1}=1$, suma 0 a P

Implementación Hardware

- Ejemplo: Multiplicar -6 (1010) por -5 (1011)

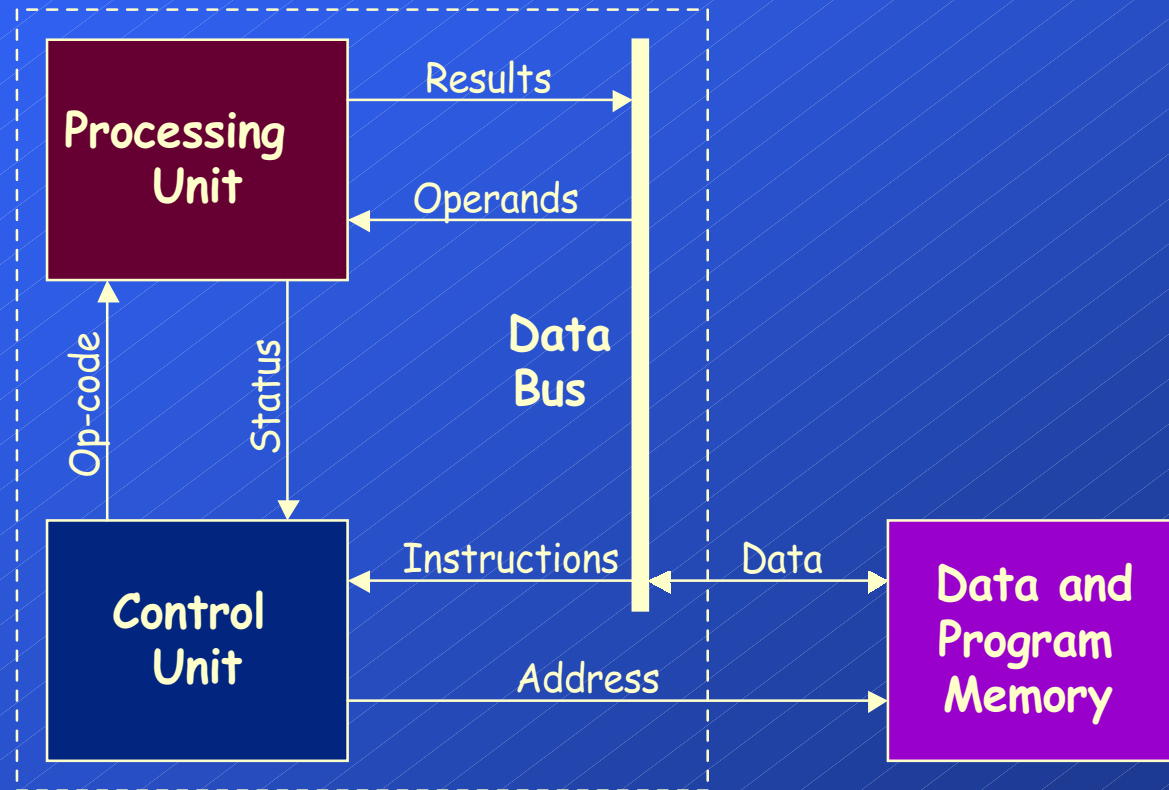
P	A	
0000	1010	
0000	1010	$a_0=a_{-1}=0$, sumar 0
0000	0101	Desplazar un bit a la izquierda
+0101		$a_0=0, a_{-1}=1$, restar B (1011)
<hr/>		
0101	0101	
0010	1010	Desplazar un bit a la izquierda
+1011		$a_0=1, a_{-1}=0$, sumar B (1011)
<hr/>		
1101	1010	
1110	1101	Desplazar un bit a la izquierda (extensión de signo)
+0101		$a_0=0, a_{-1}=1$, restar B (1011)
<hr/>		
0011	1101	
0001	1110	Desplazar un bit a la izquierda

Digital Signal Processors

¿Qué distingue a un DSP de otros micros de propósito general?

- ◆ ALU flexible y rápida: las operaciones estandar (bit a bit, suma, etc), multiplicaciones, multiplicaciones con acumulación y desplazamientos arbitrarios deben ser realizados en un ciclo de reloj.
- ◆ Rango dinámico extendido para multiplicaciones+acumulación, de forma que no ocurra overflow.
- ◆ Carga de operandos en un ciclo de reloj, lo que significa un direccionamiento flexible para varias memorias de datos.
- ◆ Hardware específico que permita el almacenamiento circular.
- ◆ No son necesarias instrucciones adicionales (condiciones, comparaciones) para bucles y saltos.

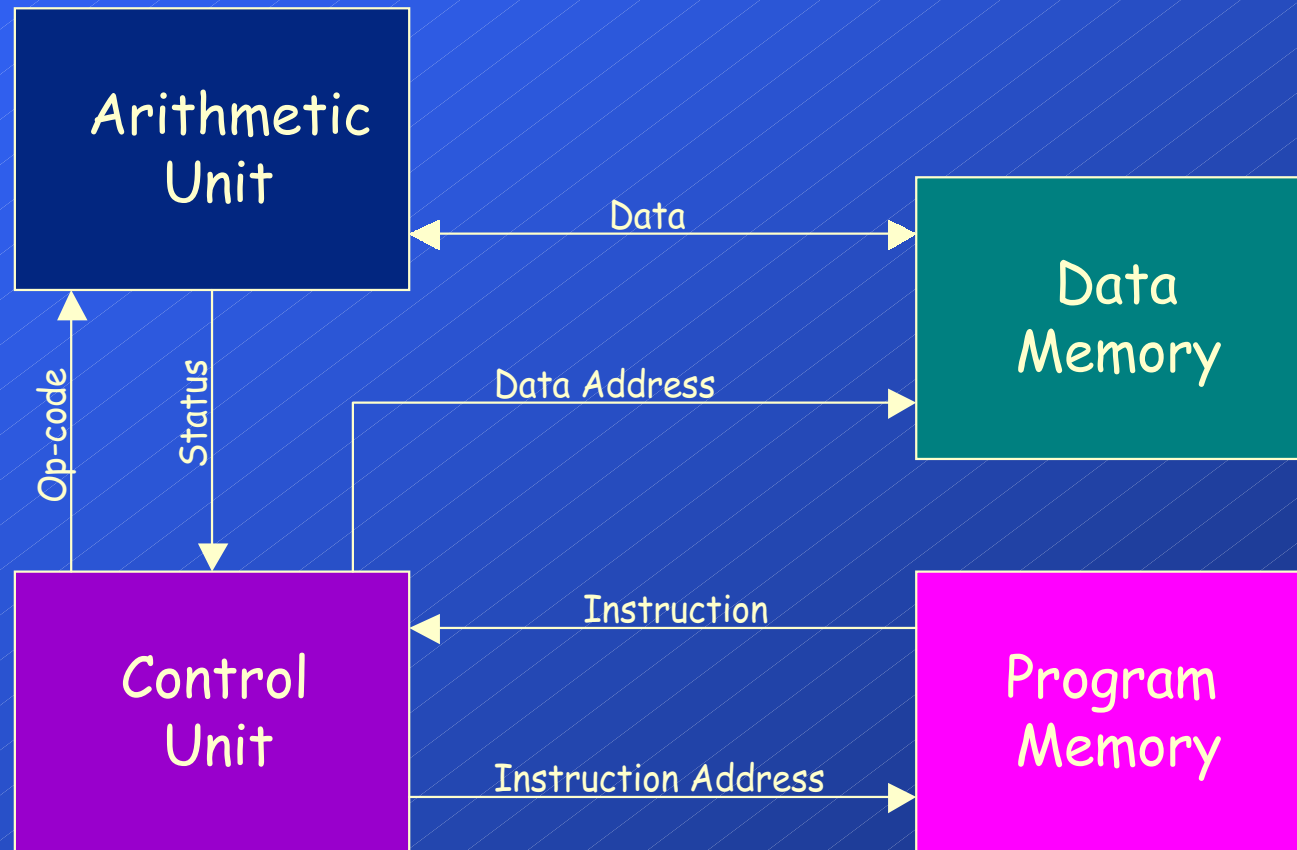
Digital Signal Processors



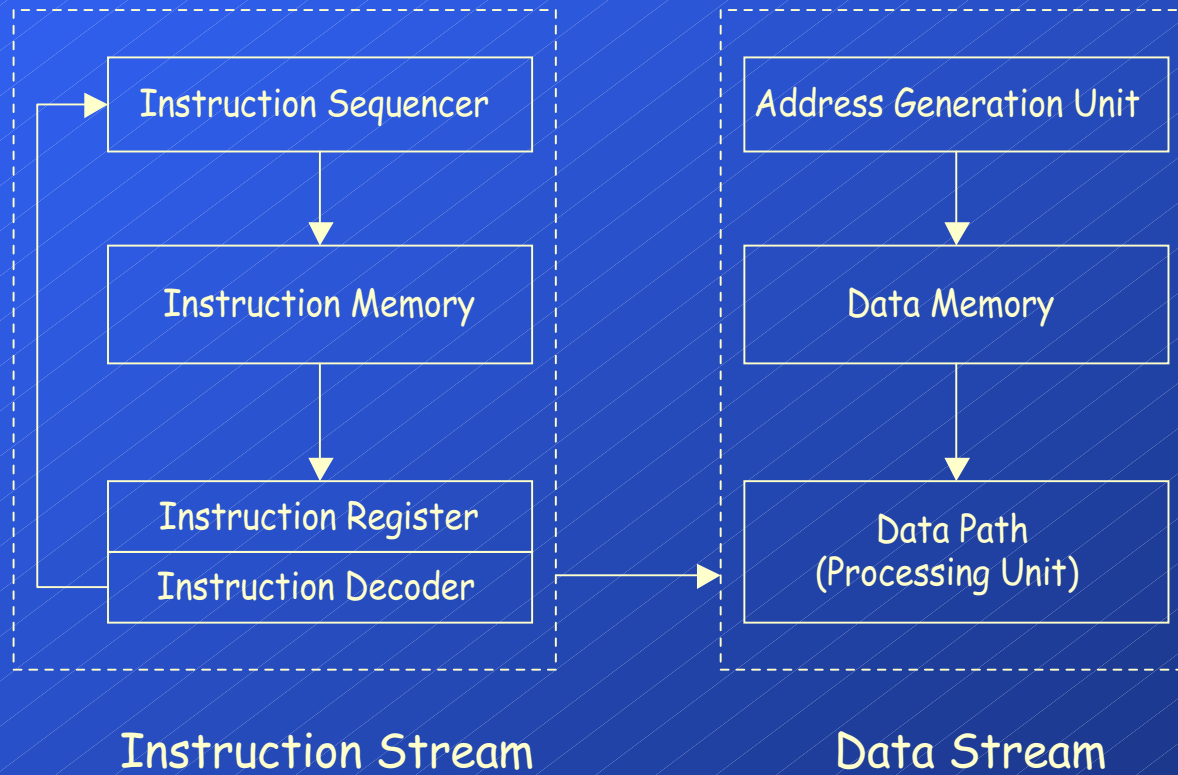
Arquitectura de Von Neumann

Digital Signal Processors

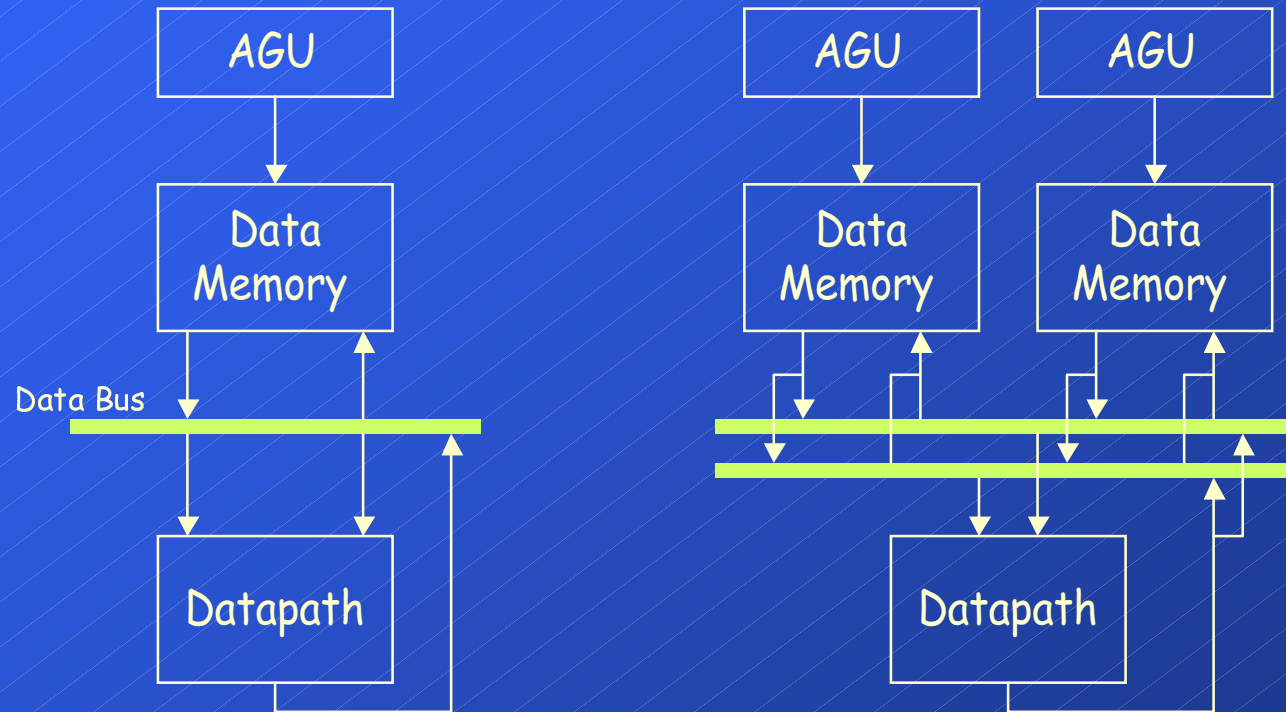
Arquitectura Harvard



Digital Signal Processors

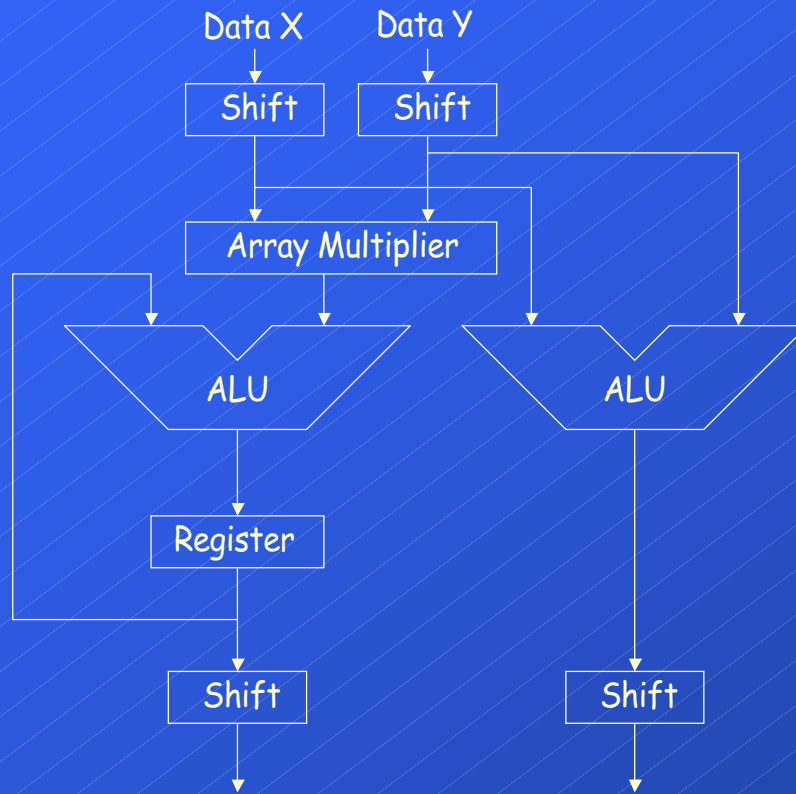


Digital Signal Processors

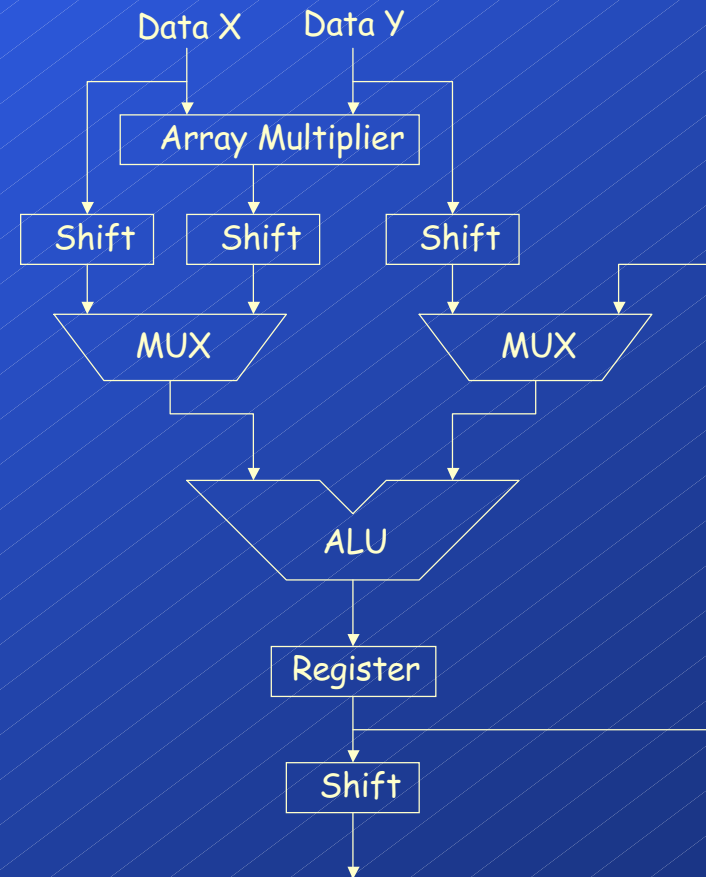


Procesamiento con uno y dos buses

Digital Signal Processors

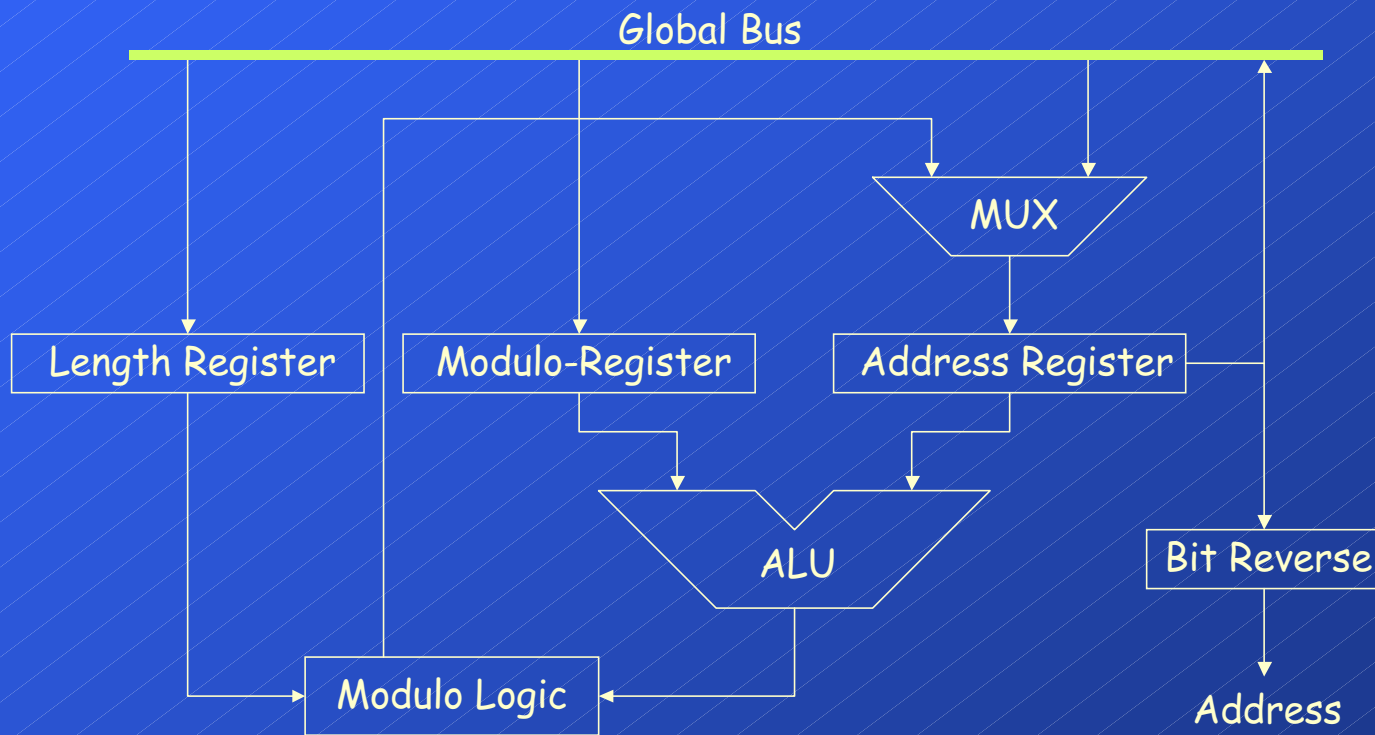


Datapath con MAC (Multiply and ACcumulate) y ALU



Datapath basado en una combinación de Multiplicador y ALU

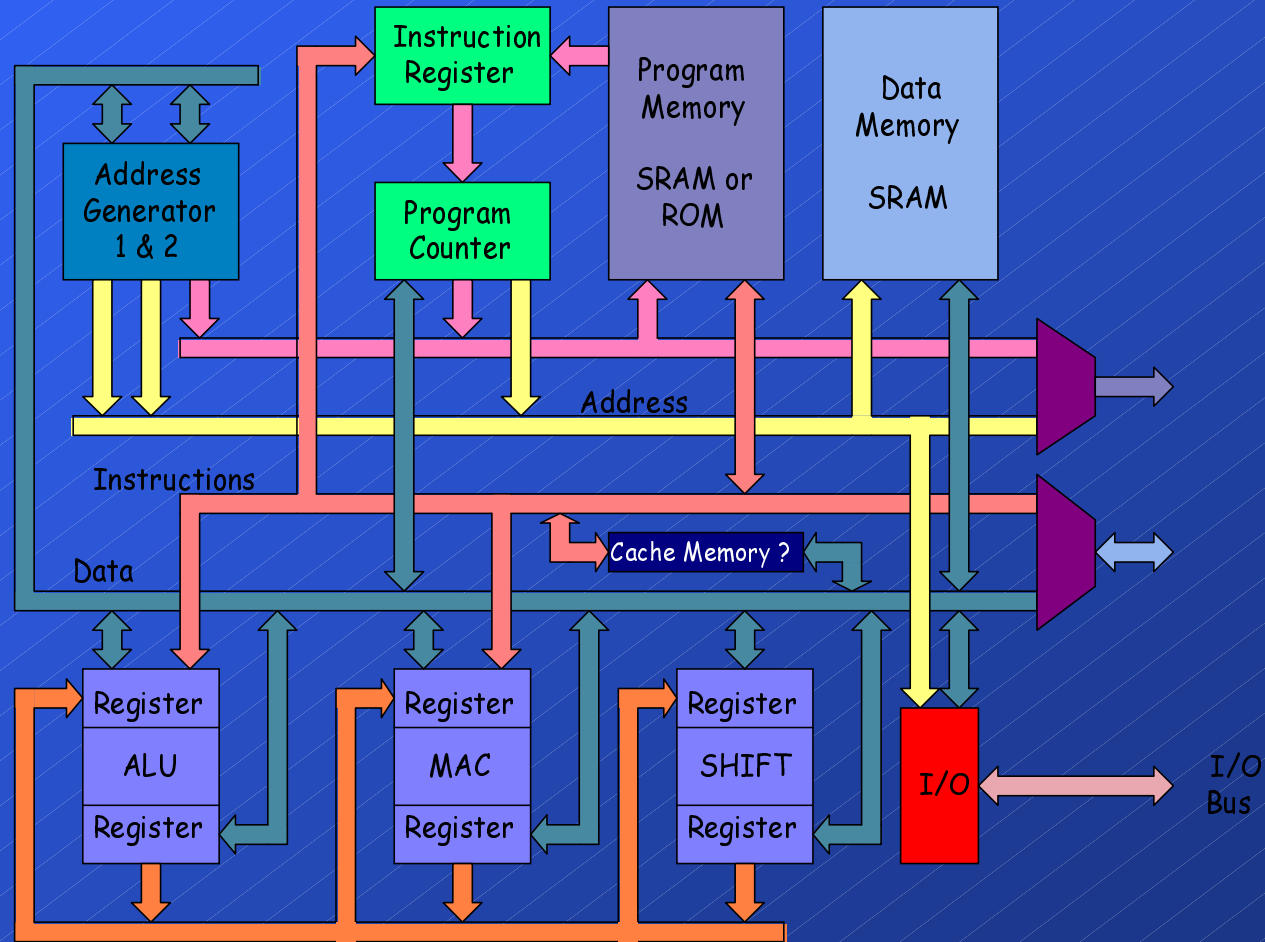
Digital Signal Processors



Generación de direcciones en un DSP

Digital Signal Processors

Arquitectura del ADSP2101



Digital Signal Processors

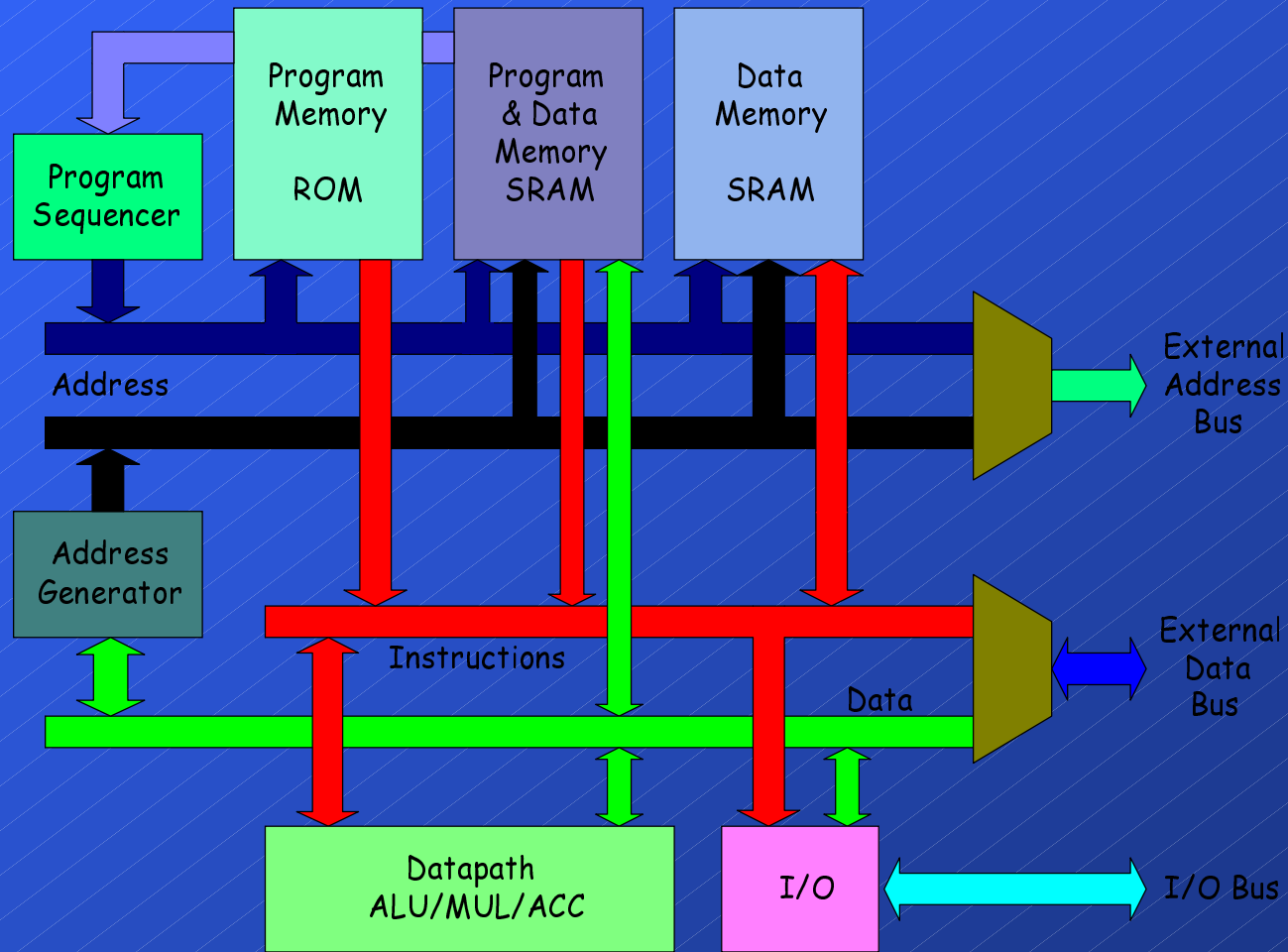
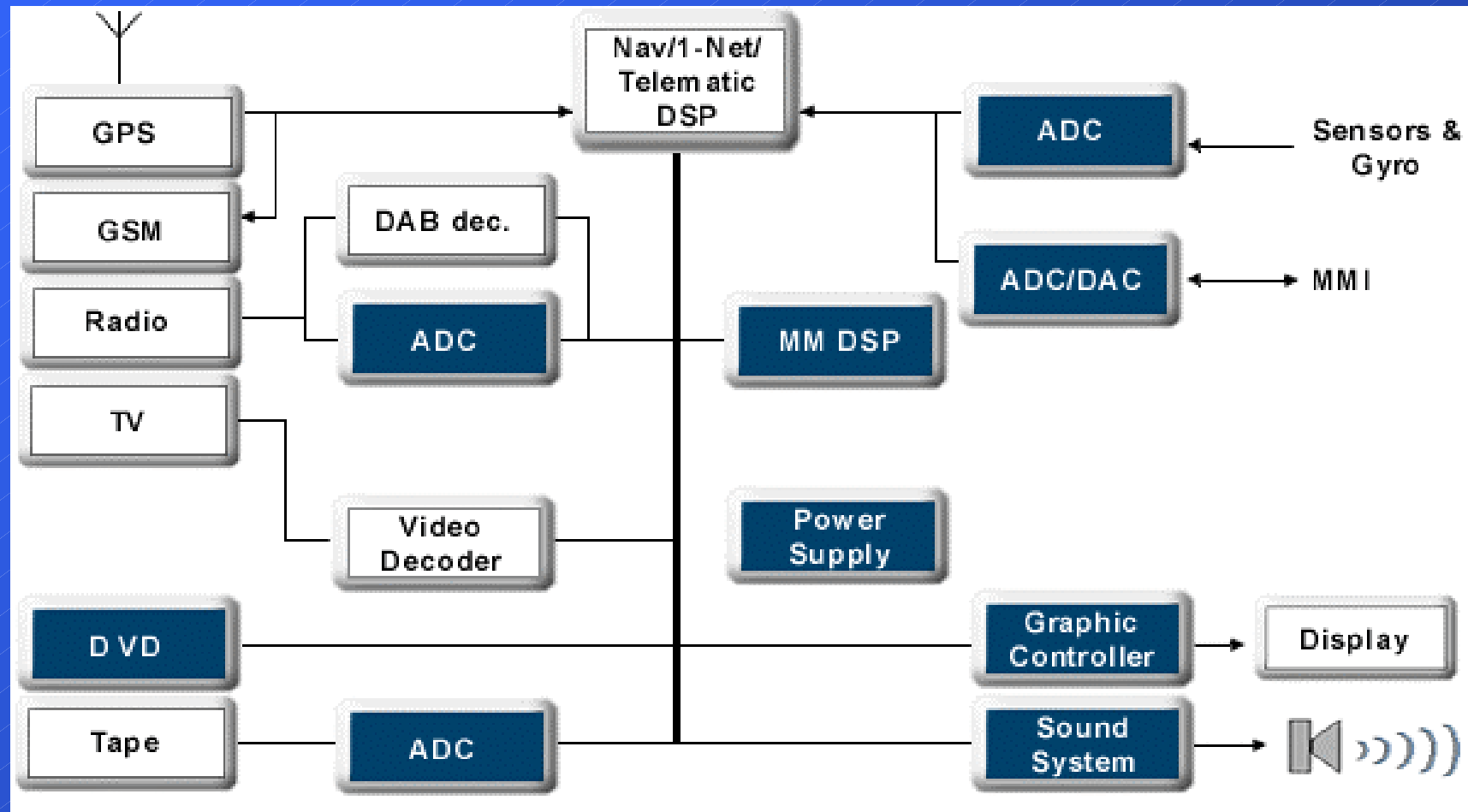
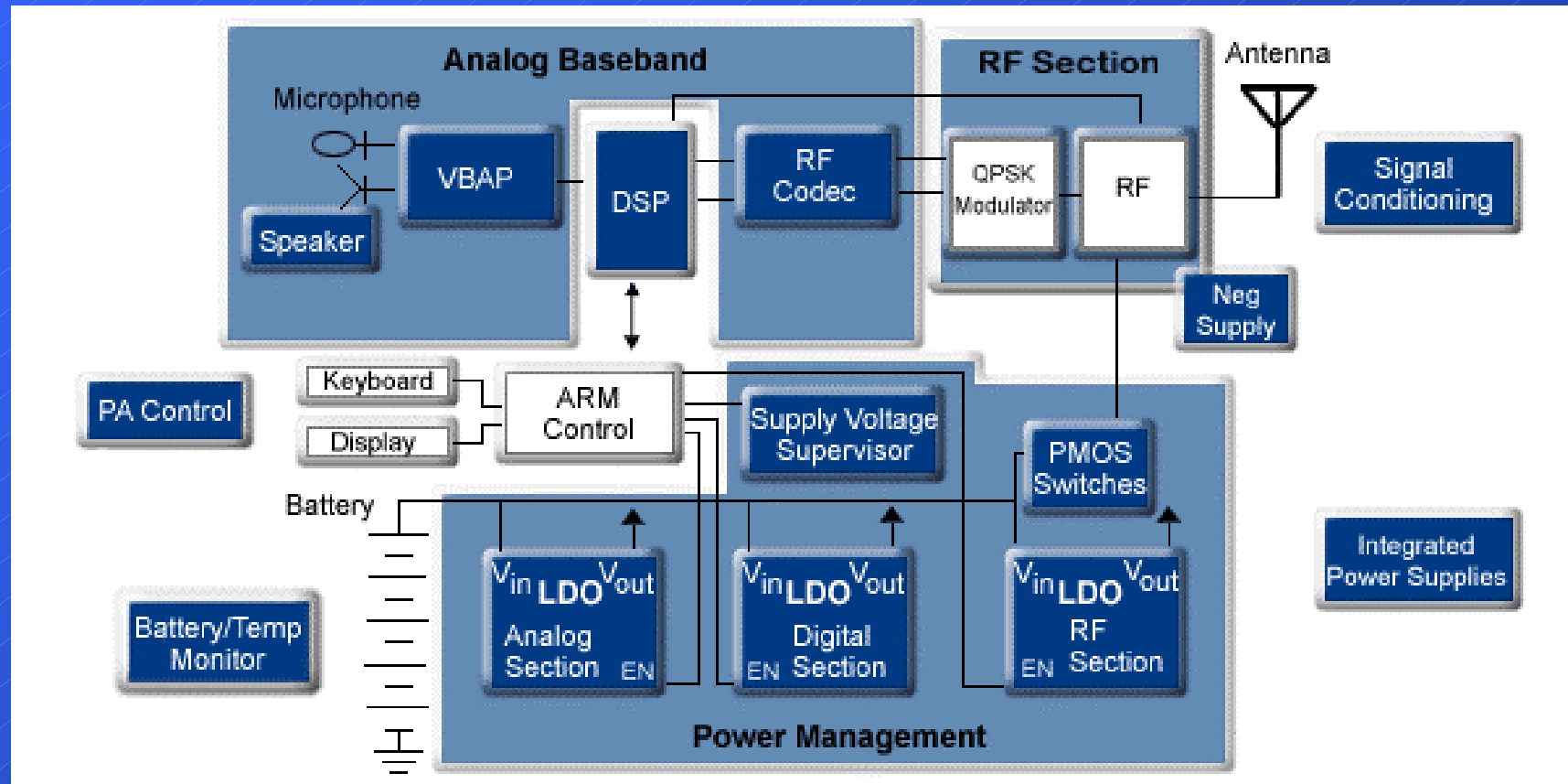


Diagrama de Bloques del TMS 320C25

Digital Signal Processors



Digital Signal Processors



Digital Signal Processors

