Procesamiento Digital de Señales

Unidad 4: Hardware y Software para procesamiento digital de señales

Objetivos

En esta unidad se presenta:

- Conceptos básicos de los procesadores digitales de señal, tales como arquitecturas de procesadores y unidades de hardware.

- Efectos de recursos finitos de hardware en operaciones en formatos de datos en punto fijo y flotante

- Consideraciones para implementaciones en tiempo real

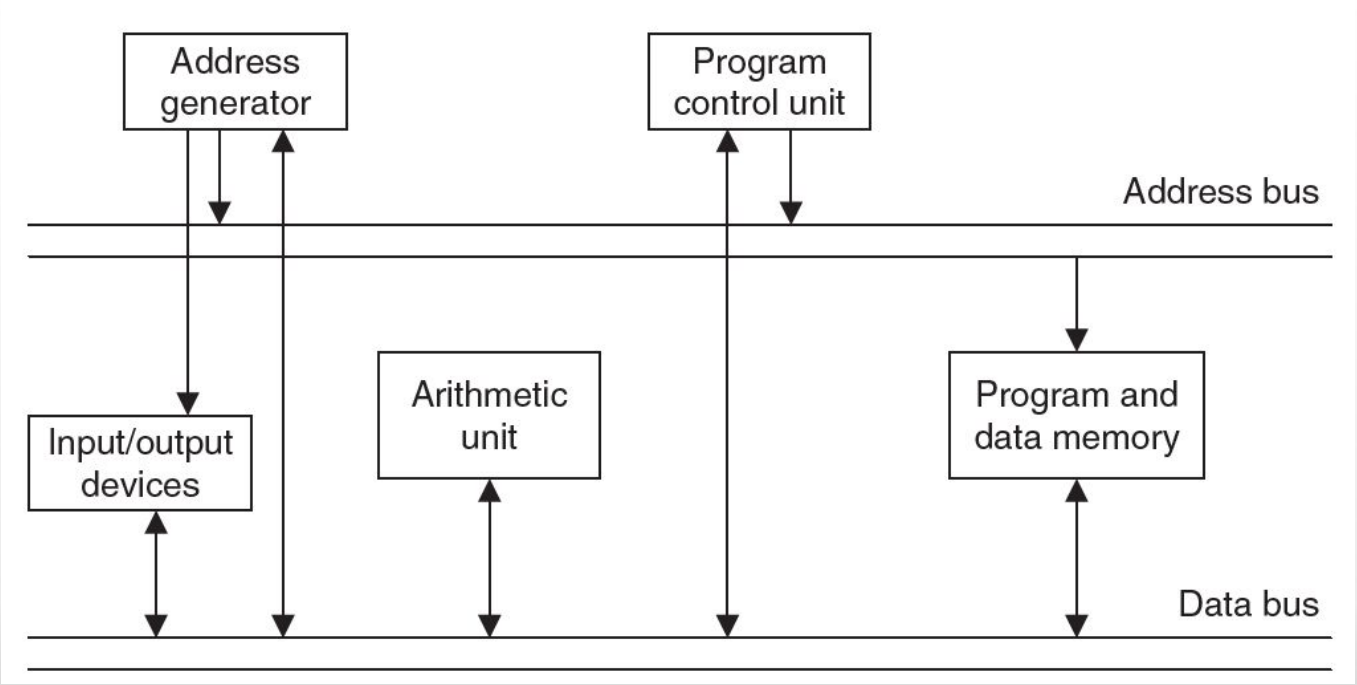
Procesamiento en tiempo real vs. Procesamiento en sistemas embebidos

- En algunos textos son utilizados como términos intercambiables, pero no son lo mismo. Ambos priorizan un procesamiento comparable con la tasa de adquisición de datos, sin embargo, los sistemas embebidos además tienen un condicionamiento en el consumo y el tamaño de los sistemas.

Digital Signal Processor Architecture

- Los DSP tienen características especiales con el objetivo de realizar operaciones matemáticas como FFT, filtrado, convolución y correlación
- Apuntan a un procesamiento en tiempo real basado en muestras o bloques
- Arquitectura de hardware para acelerar el procesamiento (como Harvard,)

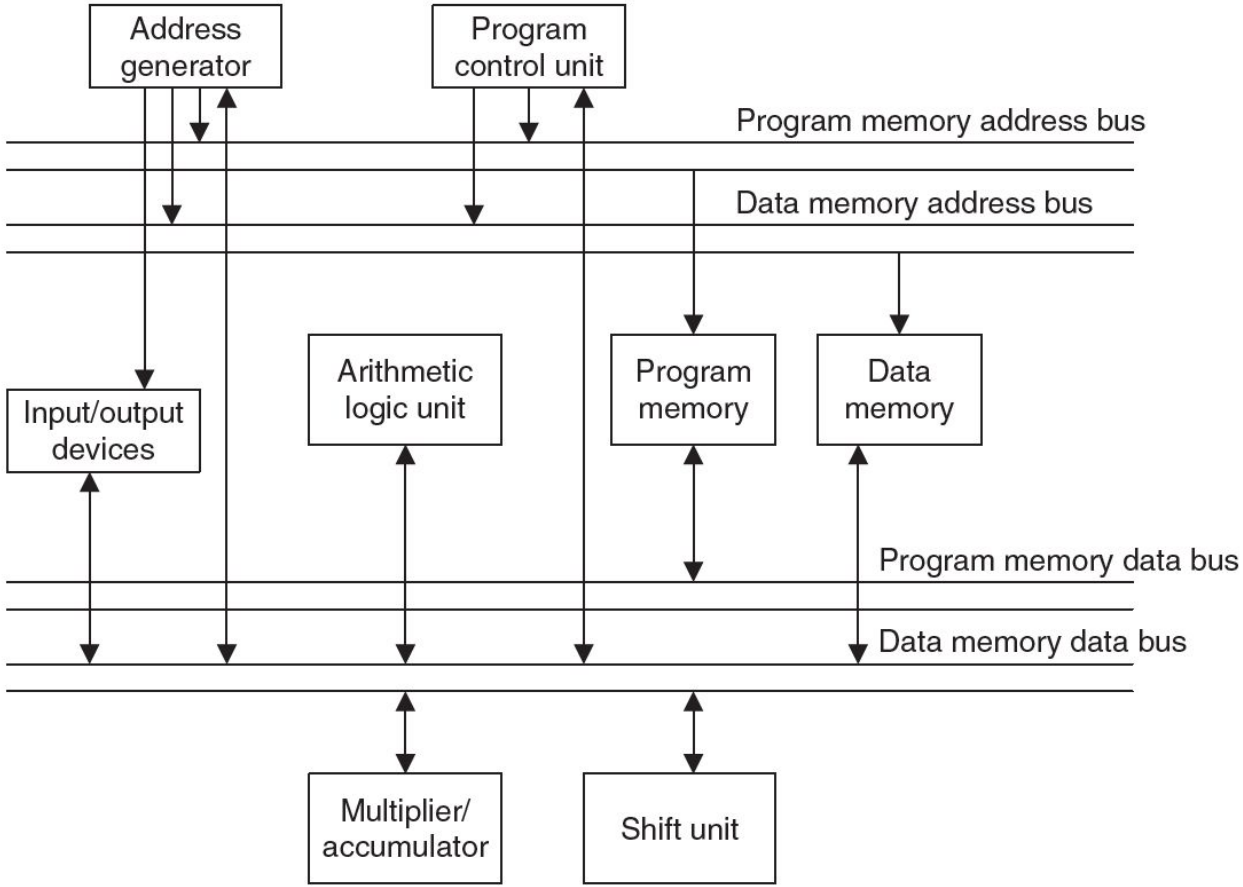
Arquitectura de procesadores Von Neumann



Digital Signal Processor Architecture

- La arquitectura Von Neumann procede de forma serial, en términos de ciclos de ejecución de instrucciones
- Utiliza una única y compartida memoria para el programa y los datos.
- Así, aumentando la velocidad del bus, la memoria, y las unidades de procesamiento, el aumento en el desempeño general no es significativo.
- La arquitectura Harvard es una de las utilizadas para acelerar la velocidad de ejecución en el procesamiento digital de señales. Esta tiene dos espacios de memoria separados. Uno es dedicado al código del programa y otro se emplea con los datos.

Arquitectura Harvard

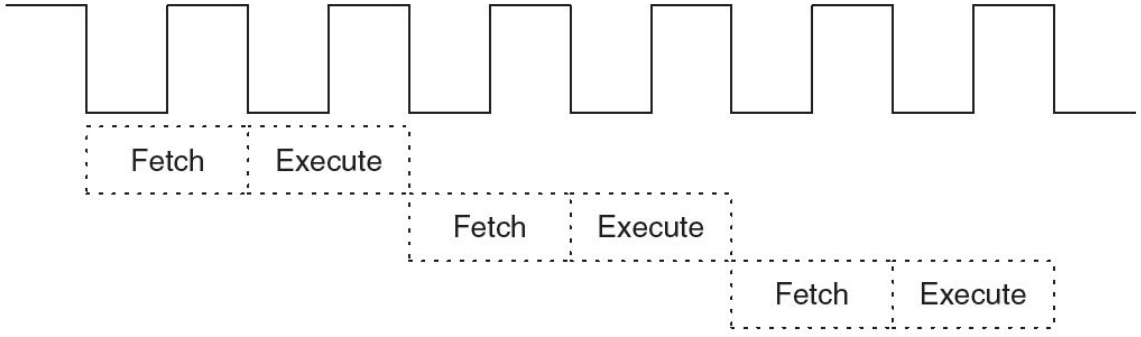


- Son necesarios dos buses de direcciones y dos buses de datos

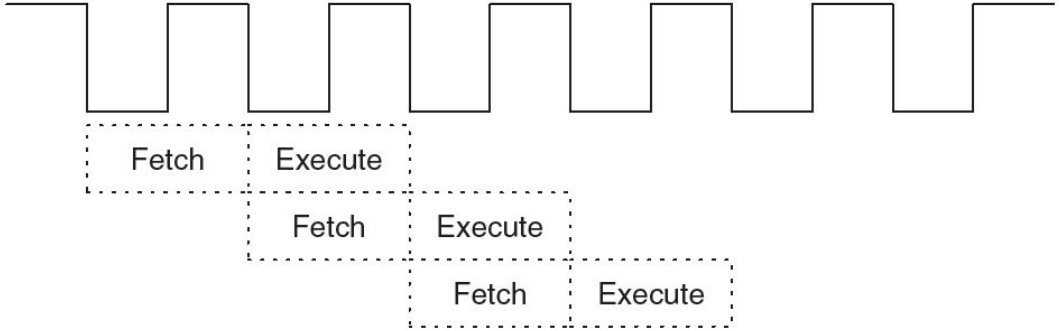
Arquitectura de Procesadores Digitales de Siñal

- Así, el procesador Harvard puede alimentar las instrucciones del programa en paralelo con los datos, al mismo tiempo.
- También, hay una unidad adicional llamada multiplicador y acumulador (*multiplier and accumulator* - MAC)
- Puede ser incluida una unidad de desplazamiento para implementaciones en punto fijo.

Pipelining operation



Execution cycle based on the Von Neumann architecture.



Execution cycle based on the Harvard architecture.

Digital Signal Processor Architecture: Micro Signal Architecture (MSA)

- Otra arquitectura utilizada en procesadores digitales de señal para aplicaciones embebidas es MSA
- Fue desarrollada en conjunto por Intel y Analog Devices Inc. para cumplir con las demandas computacionales y restricciones de potencia de aplicaciones de telecomunicación, video y audio de ese momento.
- Incorpora funcionalidades de un DSP y un microcontrolador en un solo core

Digital Signal Processors and Manufacturers

- Los procesadores digitales de señal se clasifican en DSP de propósito general y DSP especiales
- Un DSP general se diseña y optimiza para aplicaciones como filtrado digital, correlación, convolución y FFT.
- Un DSP especial tiene, además de las características para las aplicaciones previas, características optimizadas para aplicaciones específicas, como procesamiento de audio, compresión, cancelación de eco, filtrado adaptavido, etc.
- En DSP de propósito general, los mayores fabricantes son: Texas Instruments (TI), Analog Devices y NXP semiconductors

Formatos de Punto Fijo y de Punto Flotante

- Un DSP de punto fijo representa los datos en el formato entero complemento a 2 y los manipula utilizando aritmética de enteros. Representa un rango dinámico muy estrecho (puede ocurrir overflow).
- Un DSP de punto flotante representa los números usando una parte fraccionaria (mantisa) y un exponente. El codificado se vuelve más sencillo, pero este contiene más unidades de hardware, es más caro y más lento en términos de ciclos de instrucciones (es usualmente una opción en prototipos o desarrollos de prueba de conceptos)

Formatos de Punto Fijo

- En punto fijo, la representación más utilizada es en complemento a 2:

Decimal Number	2's Complement
3	011
2	010
1	001
0	000
−1	111
−2	110
−3	101
−4	100

Rango dinámico de -4 a 3

Formatos de Punto Fijo: Operaciones

Ejemplo 1: $2 \times (-1)$

```
    010
  × 001
  -----
    010
   000
+ 000
  -----
  00010
```

and 2's complement of 00010 = 11110. Removing two extended sign bits gives 110.
The answer is 110 (−2), which is within the system.

Formatos de Punto Fijo: Operaciones

Ejemplo 2: $2 \times (-3)$

$$
\begin{array}{r}
010 \\
\times\ 011 \\
\hline
010 \\
\ 010 \\
+\ 000 \\
\hline
00110
\end{array}
$$

and 2's complement of 00110 = 11010. Removing two extended sign bits achieves 010.

Formatos de Punto Fijo: Operaciones

Ejemplo 2: $2 \times (-3)$

$$
\begin{array}{r}
010 \\
\times\ 011 \\
\hline
010 \\
010 \\
+\ 000 \\
\hline
00110
\end{array}
$$

and 2's complement of 00110 = 11010. Removing two extended sign bits achieves 010.

Fuera del rango dinámico

Formatos de Punto Fijo: Fraccionario

Una forma de evitar el *overflow* en la multiplicación es realizar una normalización de números y trabajar con fracciones:

Decimal Number	Decimal Fraction	2's Complement
3	3/4	0.11
2	2/4	0.10
1	1/4	0.01
0	0	0.00
−1	−1/4	1.11
−2	−2/4	1.10
−3	−3/4	1.01
−4	−4/4 = −1	1.00

Formatos de Punto Fijo: Fraccionario

Al multiplicar números iguales o menores a 1, no se produce el overflow.

Ejemplo: Multiplicar ¾ por -1/2.

Se consideran los números sin signo:

$$
\begin{array}{r}
0.1\,0 \\
\times\ 0.1\,1 \\
\hline
0\,1\,0 \\
0\,1\,0 \\
+\ 0\,0\,0 \\
\hline
0.0\,1\,1\,0
\end{array}
$$

Luego se incorpora el signo:

$$1.1010 = (-1) \times (0.0110)_2 = -\left(0 \times (2)^{-1} + 1 \times (2)^{-2} + 1 \times (2)^{-3} + 0 \times (2)^{-4}\right) = -\frac{3}{8}$$

Finalmente, se puede truncar a tres bits, obteniendo un resultado aproximado:

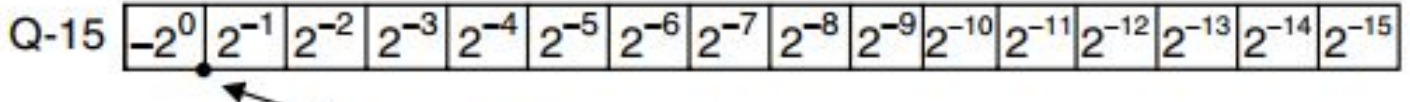
$$1.10 = (-1) \times (0.10)_2 = -\left(1 \times (2)^{-1} + 0 \times (2)^{-2}\right) = -\frac{1}{2}.$$

Formatos de Punto Fijo: Fraccionario

Sin embargo, no se puede prevenir el overflow por adición:

$$\begin{array}{r} 0.11 \\ +\ 0.01 \\ \hline 1.00 \end{array}$$

Este formato es el llamado Q-N con signo
Ejemplo:



Q-15 (fixed-point) format.

Formatos de Punto Fijo: Fraccionario

Ejemplo: Convertir a Q-15 el siguiente número decimal: 0,560123

Number	Product	Carry
0.560123×2	1.120246	1 (MSB)
0.120246×2	0.240492	0
0.240492×2	0.480984	0
0.480984×2	0.961968	0
0.961968×2	1.923936	1
0.923936×2	1.847872	1
0.847872×2	1.695744	1
0.695744×2	1.391488	1
0.391488×2	0.782976	0
0.782976×2	1.565952	1
0.565952×2	1.131904	1
0.131904×2	0.263808	0
0.263808×2	0.527616	0
0.527616×2	1.055232	1
0.055232×2	0.110464	0 (LSB)

MSB, most-significant bit; LSB, least-significant bit.

Formatos de Punto Fijo: Fraccionario Q-N

Ejemplo: Convertir a Q-15 el siguiente número decimal: 0,560123

Number	Product	Carry
0.560123×2	1.120246	1 (MSB)
0.120246×2	0.240492	0
0.240492×2	0.480984	0
0.480984×2	0.961968	0
0.961968×2	1.923936	1
0.923936×2	1.847872	1
0.847872×2	1.695744	1
0.695744×2	1.391488	1
0.391488×2	0.782976	0
0.782976×2	1.565952	1
0.565952×2	1.131904	1
0.131904×2	0.263808	0
0.263808×2	0.527616	0
0.527616×2	1.055232	1
0.055232×2	0.110464	0 (LSB)

MSB, most-significant bit; LSB, least-significant bit.

$$0.100011110110010$$

el truncamiento produce un error inferior al tamaño de intervalo:

$$2^{-15} = 0.000030517$$

Formatos de Punto Fijo: Fraccionario Q-N

Ejemplo: Convertir a Q-15 el siguiente número decimal: -0,160123

Tomando el valor positivo:

$$0.160123 = 0.001010001111110.$$

Haciendo el complemento a 2:

$$-0.160123 = 1.110101110000010.$$

Formatos de Punto Fijo: Fraccionario Q-N

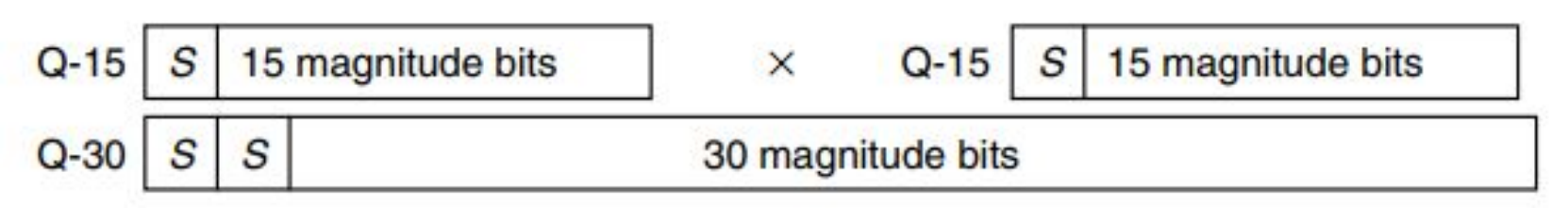
Ejemplo: Convertir a decimal el siguiente número: 1.110101110000010

$$-(2^{-3} + 2^{-5} + 2^{-9} + 2^{-10} + 2^{-11} + 2^{-12} + 2^{-13} + 2^{-14}) = -0.160095.$$

Formatos de Punto Fijo: Fraccionario Q-N

El formato de representación numérica Q es una mejor forma de representación que el entero complemento a 2. Pero necesitamos estar atentos a los siguientes problemas:

- El truncamiento puede producir un error (siempre menor al intervalo 2^-N)
- La suma puede causar overflow (se evidencia con cambios de signos)
- Multiplicar dos números en Q-15 lleva a un número en Q-30



- También puede darse un underflow

Formatos de Punto Fijo: Fraccionario Q-N.M

Un formato que permite acomodar números con diferente rango dinámico es el Q-N.M (combina una parte entera con una parte decimal)

Format (*N.M*)	Largest Positive Value (0x7FFF)	Least Negative Value (0x8000)	Precision (0x0001)
(1.15)	0.999969482421875	−1	0.00003051757813
(2.14)	1.99993896484375	−2	0.00006103515625
(3.13)	3.9998779296875	−4	0.00012207031250
(4.12)	7.999755859375	−8	0.00024414062500
(5.11)	15.99951171875	−16	0.00048828125000
(6.10)	31.9990234375	−32	0.00097656250000
(7.9)	63.998046875	−64	0.00195312500000
(8.8)	127.99609375	−128	0.00390625000000
(9.7)	255.9921875	−256	0.00781250000000
(10.6)	511.984375	−512	0.01562500000000
(11.5)	1,023.96875	−1,024	0.03125000000000
(12.4)	2,047.9375	−2,048	0.06250000000000
(13.3)	4,095.875	−4,096	0.12500000000000
(14.2)	8,191.75	−8,192	0.25000000000000
(15.1)	16,383.5	−16,384	0.50000000000000
(16.0)	32,767	−32,768	1.00000000000000

Formato de punto flotante

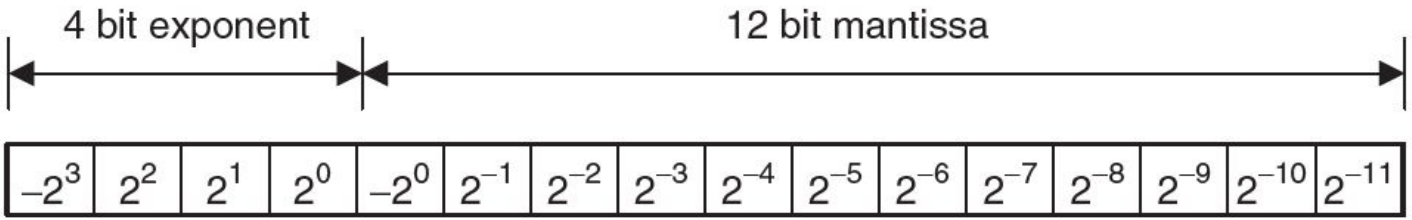
Para mejorar el rango dinámico, se propone el punto flotante
Formato general:

$$x = M \cdot 2^E$$

M: mantisa
E: exponente
Números con signo

Veamos algunos ejemplos…

Formato de Punto Flotante



$$\text{Most negative number} = (1.00000000000)_2 \cdot 2^{0111_2} = (-1) \times 2^7 = -128.0$$

$$\text{Most positive number} = (0.11111111111)_2 \cdot 2^{0111_2} = \left(1 - 2^{-11}\right) \times 2^7 = 127.9375.$$

The smallest positive number is given by

$$\text{Smallest positive number} = (0.00000000001)_2 \cdot 2^{1000_2} = \left(2^{-11}\right) \times 2^{-8} = 2^{-19}.$$

Formato de Punto Flotante

Ejemplo 1: Convertir en formato binario de punto flotante: 0,1601230

En primer lugar, escalar a $0.160123/2^{-2} = 0.640492$

$$0.160123 = 0.640492 \times 2^{-2}$$

Así, el exponente (en complemento a 2) es $-2 = 1110$.

y la mantisa, 0.640492 en Q-11, 010100011111

Resultando finalmente,

$$1110010100011111$$

Formato de Punto Flotante

Ejemplo 2: Convertir en formato binario de punto flotante: -20,430527

En primer lugar, escalar a $-20.430527/2^5 = -0.638454,$

$$-20.430527 = -0.638454 \times 2^5$$

Otras opciones posibles:

$$-20.430527 = -0.319227 \times 2^6$$
$$-20.430527 = -0.1596135 \times 2^7$$

Tomando la primera, el exponente es 0101, y convirtiendo -0.638454 a Q-11, resulta 010100011011, que en complemento a 2 es

1010111100101

El resultado final es: 0101101011100101

Formato de Punto Flotante: Operaciones

La manipulación de números en punto flotante es más complicada:

$$x_1 = M_1 2^{E_1}$$
$$x_2 = M_2 2^{E_2}$$

The floating-point sum is performed as follows:

$$x_1 + x_2 = \begin{cases} \left(M_1 + M_2 \times 2^{-(E_1 - E_2)}\right) \times 2^{E_1}, & \text{if } E_1 \geq E_2 \\ \left(M_1 \times 2^{-(E_2 - E_1)} + M_2\right) \times 2^{E_2} & \text{if } E_1 < E_2 \end{cases}$$

As a multiplication rule, given two properly normalized floating-point numbers:

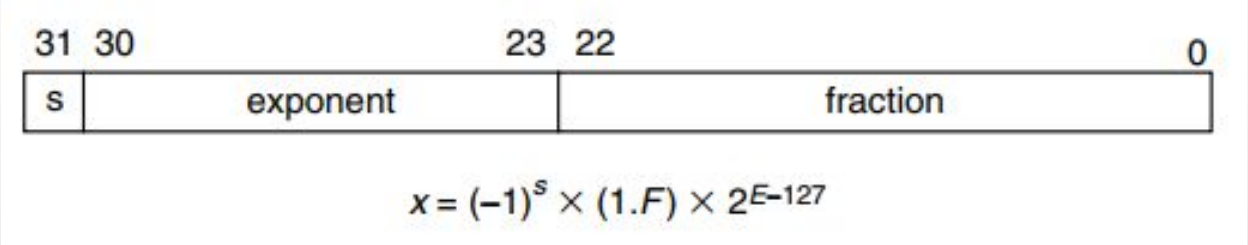
$$x_1 = M_1 2^{E_1}$$
$$x_2 = M_2 2^{E_2},$$

where $0.5 \leq |M_1| < 1$ and $0.5 \leq |M_2| < 1$. Then multiplication can be performed as follows:

$$x_1 \times x_2 = (M_1 \times M_2) \times 2^{E_1 + E_2} = M \times 2^E.$$

Formato de Punto Flotante: Formatos de la IEEE

- Precisión simple (32 bits) y precisión doble (64 bits)
- Simple: 23 de fracción, 8 de exponente, 1 de signo.
- La mantisa está siempre normalizada entre +1 y +2.



Formato de Punto Flotante: Formatos de la IEEE

Ejemplos:.

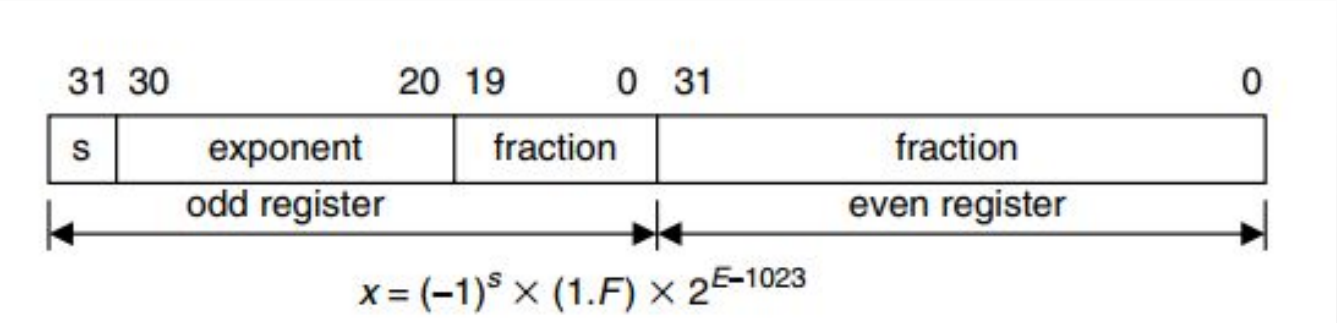
$$0\ 10000000\ 00000000000000000000000 = (-1)^0 \times (1.0_2) \times 2^{128-127} = 2.0$$

$$0\ 10000001\ 10100000000000000000000 = (-1)^0 \times (1.101_2) \times 2^{129-127} = 6.51$$

$$1\ 10000001\ 10100000000000000000000 = (-1)^1 \times (1.101_2) \times 2^{129-127} = -6.5.$$

Formato de Punto Flotante: Formatos de la IEEE

Doble precisión



11 bits de exponente
52 bits de parte fraccionaria

Implementación de FIR en punto fijo

Se prefiere el formato Q para evitar overflow en las multiplicaciones

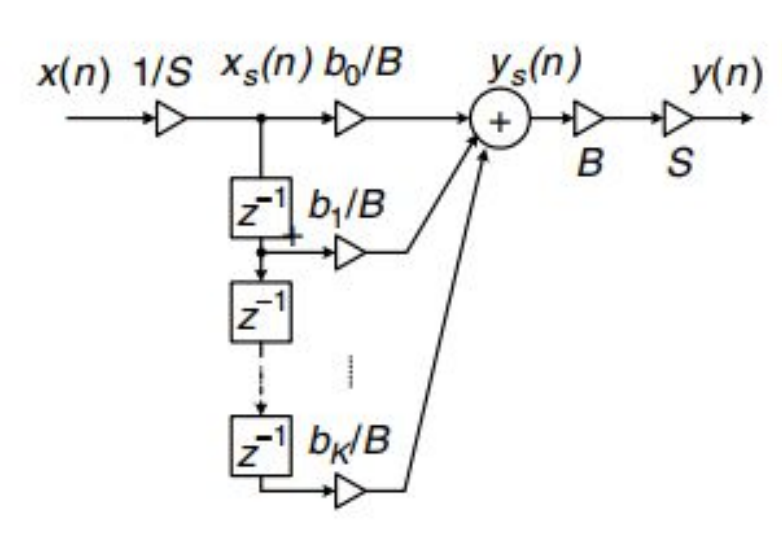
Para evitar el overflow en la suma, se realiza una normalización de la señal de entrada, dada por:

$$S = I_{\max} \cdot \sum_{k=0}^{\infty} |h(k)| = I_{\max} \cdot (|h(0)| + |h(1)| + |h(2)| + \cdots),$$

donde $I_{\max}$ es la máxima señal de entrada posible y $h(k)$ los coeficientes del filtro. Así, se asume el máximo valor alcanzable (esto es perjudicial si S/N es baja)

Implementación de FIR en punto fijo

Además, si alguno de los coeficientes está fuera del rango de Q-*N* (mayor a 1), se realiza una normalización de los coeficientes



Donde B se selecciona como un múltiplo de 2, implicando un desplazamiento en la representación binaria

Implementación de FIR en punto fijo

Ejemplo: Considerando el filtro FIR

$$y(n) = 0.9x(n) + 3x(n-1) - 0.9x(n-2),$$

con una ganancia en la banda de paso igual a 4, asumiendo que el rango de la señal de entrada ocupa solo ¼ del rango completo en la aplicación considerada,

- Desarrollar las ecuaciones de implementación en un sistema DSP en formato Q-15.

Implementación de FIR en punto fijo

Ejemplo: Considerando el filtro FIR

$$y(n) = 0.9x(n) + 3x(n-1) - 0.9x(n-2),$$

Solución:

- El factor de escala de la señal de entrada es

$$S = \frac{1}{4}(|h(0)| + |h(1)| + |h(2)|) = \frac{1}{4}(0.9 + 3 + 0.9) = 1.2$$

Se selecciona $S = 2$, para que sea una potencia de 2.

Además, se hace $B = 4$ para que todos los coeficientes sean menores a 1

Implementación de FIR en punto fijo

Ejemplo: Considerando el filtro FIR

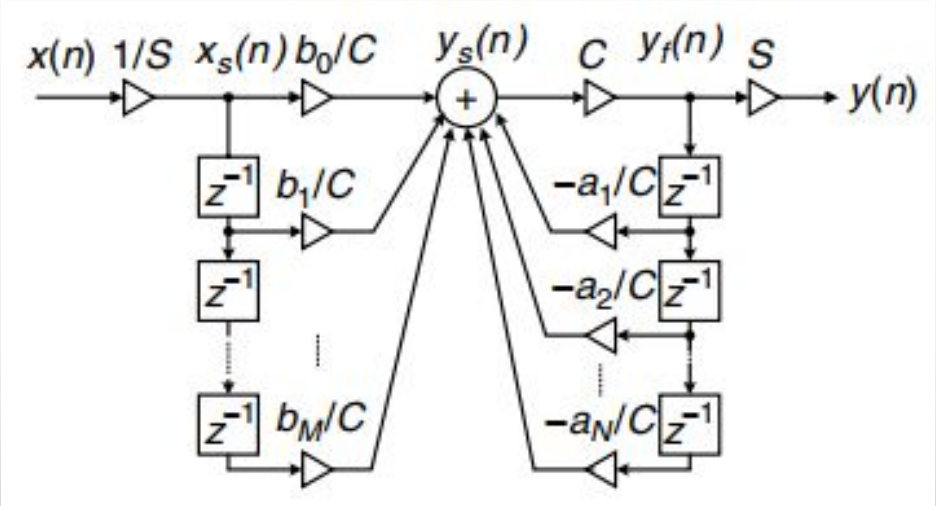
$$y(n) = 0.9x(n) + 3x(n-1) - 0.9x(n-2),$$

Solución:
Las ecuaciones de implementación resultan:

$$x_s(n) = \frac{x(n)}{2}$$
$$y_s(n) = 0.225x_s(n) + 0.75x_s(n-1) - 0.225x_s(n-2)$$
$$y(n) = 8y_s(n)$$

Implementación de IIR en punto fijo

Así como en el FIR, para el IIR se mantienen todos los coeficientes dentro del rango mediante una normalización (dividiendo por *C* en este caso)



Implementación de IIR en punto fijo

Ejemplo: Para el filtro

$$y(n) = 2x(n) + 0.5y(n-1),$$

y considerando que la máxima señal de entrada es:

$$I_{\max} = 0.010\ldots0_2 = 0.25$$

Determinar las ecuaciones de implementación en un sistema Q-15

Solución:

La función de transferencia del filtro está dada por: $H(z) = \frac{2}{1-0.5z^{-1}} = \frac{2z}{z-0.5}$

Implementación de IIR en punto fijo

Ejemplo: Para el filtro

$$y(n) = 2x(n) + 0.5y(n-1),$$

Solución:
Respuesta al impulso del filtro está dada por:

$$h(n) = 2 \times (0.5)^n u(n)$$

Así, $$S = 0.25 \times \left(2(0.5)^0 + 2(0.5)^1 + 2(0.5)^2 + \cdots\right) = \frac{0.25 \times 2 \times 1}{1 - 0.5} = 1.$$

no es necesario escalar la entrada

Implementación de IIR en punto fijo

Ejemplo: Para el filtro

$$y(n) = 2x(n) + 0.5y(n-1),$$

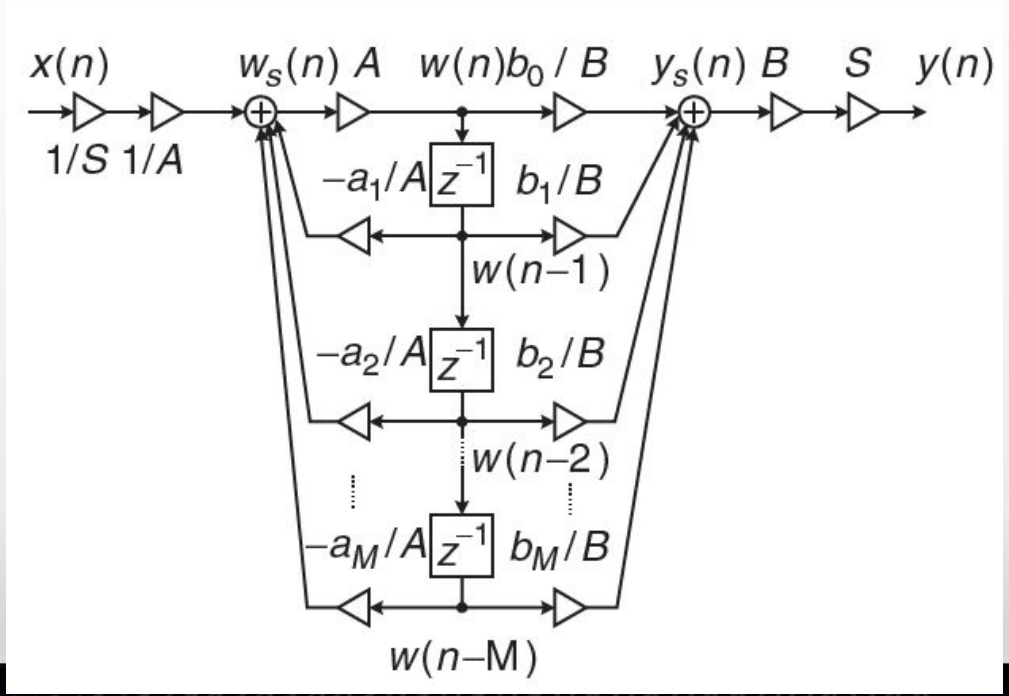
Solución:
Los coeficientes si deben normalizarse

$$\begin{aligned} x_s(n) &= x(n) \\ y_s(n) &= 0.5x_s(n) + 0.125y_f(n-1) \\ y_f(n) &= 4y_s(n) \\ y(n) &= y_f(n). \end{aligned}$$

En este caso, se ha dividido la ecuación a diferencias por 4 y luego se compensa la salida

Implementación de IIR en punto fijo

La implementación en la forma directa II es más complicada, como se ilustra en la figura



Implementación de IIR en punto fijo

- Dos factores de escala, *A* y *B*, son diseñados para escalar los coeficientes del denominador y el numerador respectivamente.
- *S* escala la señal de entrada para no saturar el primer sumador.

Para definir estos valores, se considera primero las ecuaciones originales:

$$w(n) = x(n) - a_1 w(n-1) - a_2 w(n-2) - \cdots - a_M w(n-M)$$
$$y(n) = b_0 w(n) + b_1 w(n-1) + \cdots + b_M w(n-M).$$

Implementación de IIR en punto fijo

A se define de tal forma que ningún coeficiente del denominador sea mayor a 1.

$$w_s(n) = \frac{1}{A}w(n) = \frac{1}{A}x(n) - \frac{1}{A}a_1 w(n-1) - \frac{1}{A}a_2 w(n-2) - \cdots - \frac{1}{A}a_M w(n-M)$$

$$w(n) = A \times w_s(n).$$

B se define igualmente pero considerando el numerador

$$y_s(n) = \frac{1}{B}y(n) = \frac{1}{B}b_0 w(n) + \frac{1}{B}b_1 w(n-1) + \cdots + \frac{1}{B}b_M w(n-M)$$

$$y(n) = B \times y_s(n)$$

Implementación de IIR en punto fijo

Finalmente,

$$S = I_{\max}(|h(0)| + |h(1)| + |h(2)| + \cdots),$$

Donde se utiliza la respuesta al impulso del filtro cuya función de transferencia es la recíproca del polinomio del denominador, cuyos polos pueden causar *grandes valores en la primera suma*

$$h(n) = Z^{-1}\left(\frac{1}{1 + a_1 z^{-1} + \cdots + az^{-M}}\right)$$

Implementación de IIR en punto fijo

Ejemplo: Dado el siguiente filtro IIR

$$y(n) = 0.75x(n) + 1.49x(n-1) + 0.75x(n-2) - 1.52y(n-1) - 0.64y(n-2),$$

con banda de paso de ganancia unitaria, determinar las ecuaciones de implementación para un formato Q-15.

Solución:

Las ecuaciones de implementación en la forma directa II sin escalamiento son

$$w(n) = x(n) - 1.52w(n-1) - 0.64w(n-2)$$

$$y(n) = 0.75w(n) + 1.49w(n-1) + 0.75w(n-2)$$

Implementación de IIR en punto fijo

Ejemplo: Dado el siguiente filtro IIR

$$y(n) = 0.75x(n) + 1.49x(n-1) + 0.75x(n-2) - 1.52y(n-1) - 0.64y(n-2),$$

Solución:

Para prevenir la saturación del primer sumador, se considera la respuesta al impulso del denominador:

$$A(z) = \frac{1}{1 + 1.52z^{-1} + 0.64z^{-2}}$$

Implementación de IIR en punto fijo

Se puede utilizar el código:

```
import numpy as np
from scipy.signal import lfilter

# Define the filter coefficients
b = [1]
a = [1, 1.52, 0.64]

# Compute the impulse response
n = 10000
impulse = np.zeros(n)
impulse[0] = 1
h = lfilter(b, a, impulse)

# Compute the sum of absolute values of the impulse response
sf = np.sum(np.abs(h))

print("sf =", sf)
```

```
sf = 10.409927309299313
```

Implementación de IIR en punto fijo

Así, $S = 16$, considerando una potencia de 2, además, se selecciona $A = 2$.

Finalmente, la salida del segundo sumador es

$$y_s(n) = \frac{0.75}{B} w(n) + \frac{1.49}{B} w(n-1) + \frac{0.75}{B} w(n-2),$$

Debemos garantizar que todos los coeficientes sean menores a 1, pero también que la suma así lo sea, es decir: $\frac{0.75}{B} + \frac{1.49}{B} + \frac{0.75}{B} < 1$

Lo que queda garantizado con $B = 4$.

Implementación de IIR en punto fijo

Así, las ecuaciones de implementación resultan:

$$
\begin{aligned}
x_s(n) &= x(n)/16 \\
w_s(n) &= 0.5x_s(n) - 0.76w(n-1) - 0.32w(n-2) \\
w(n) &= 2w_s(n) \\
y_s(n) &= 0.1875w(n) + 0.3725w(n-1) + 0.1875w(n-2) \\
y(n) &= (B \times S)y_s(n) = 64y_s(n)
\end{aligned}
$$

Diseño y prototipado rápido de sistemas de DSP

- La disminución del tiempo y costo en el desarrollo de productos es algo muy deseable desde el punto de vista de la competitividad.
- La electrónica en general y los sistemas de DSP en particular han sido foco de numerosos estudios en este sentido, especialmente durante la década de los 90’
- Los avances en esta materia han sido traccionados por el surgimiento de las plataformas de matrices de puertas lógicas programables en campo (*field programmable gate array* - FPGA) y los lenguajes descriptores de hardware (VHDL)

Diseño y prototipado rápido de sistemas de DSP

Se presenta un estudio de caso basado en la estrategia top-down de codiseño de software y hardware

Se inicia con la captura de los requerimientos del sistema en forma de un ejecutable, y a través de sucesivos refinamientos, se logra un diseño detallado de hardware

VHDL y el proceso de diseño

VHDL es utilizado en el proceso de proyecto y diseño por los siguientes motivos:

1) Es un standard de la IEEE con continuas actualizaciones y mejoras
2) Tiene la habilidad de describir sistemas y circuitos con múltiples niveles de abstracción
3) Es apropiado para realizar síntesis y simulación
4) Es posible documentar el sistema a lo largo del proceso de diseño en la forma de ejecutable

Prototipo virtual

El objetivo es lograr un prototipo virtual, definido como un ejecutable con requerimientos de un sistema embebido y la descripción de sus respectivos estímulos, en múltiples niveles de abstracción

Con el prototipo virtual se logra una especificación, simulación y verificación conjunta de software y hardware