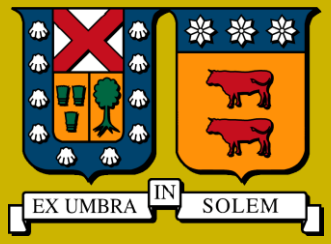
Introducción al procesamiento digital de señales en tiempo real

ELO 314 – Laboratorio de Procesamiento Digital de Señales
Primer semestre - 2018



Gonzalo Carrasco, Ph.D , Matías Zañartu, Ph.D.
Departamento de Electrónica
Universidad Técnica Federico Santa María

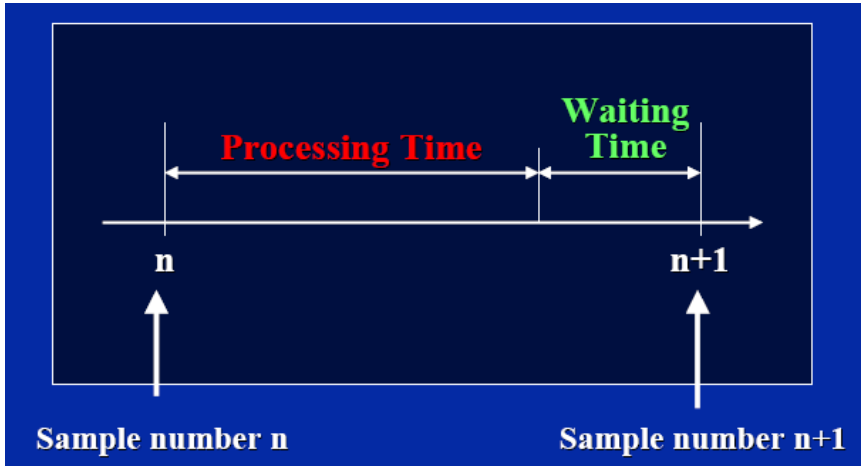
¿Qué es DSP?

- **Procesador digital de señales**
  - Dispositivo electrónico que permite procesar digitalmente señales en tiempo real
  - Microprocesador especializado

- **Procesamiento digital de señales**
  - Es la matemática, los algoritmos y las técnicas usadas para manipular las señales
  - Análisis y procesamiento de señales que han sido originadas en el mundo real, muestradasy representadas como dígitos (variables numéricas)

DSP en tiempo real

- **DSP en tiempo real**
  - El procesamiento se realiza entre muestras (o bloques de muestras) y mantiene la dinámica requerida entre la salida y entrada
  - Cuando esto no es posible, el procesamiento se realiza con un retardo que puede ser significativo o simplemente se almacenan los datos y procesan después (*Offline*, *batch processing*)



Waiting Time ≥ 0

Aplicaciones de DSP

¿Dónde?	¿Para Qué?
DSP Propósito General	Filtrado Digital – Convolución – Correlación, Transformadas - Filtros Adaptivos - Generación de Formas de Onda
Gráficas / Imágenes	Rotación en 3D - Visión de Robots - Transmisión y Compresión de Imágenes Patrones de Reconocimiento - Animación y Mapeo Digital
Instrumentación	Análisis de Espectros - Generación de Funciones - Análisis Transientes - Filtrado Digital
Voz	Mail de Voz - Reconocimiento de Voz - Verificación de Voz
Control	Discos - Servos – Robots - Impresoras Láser - Control Motores
Militar	Comunicaciones Seguras - Procesamiento de Radares - Procesamiento de Sonar - Procesamiento de Imágenes - Navegación - Misiles - Radio Frecuencia
Telecomunicaciones	Cancelación de Eco - Líneas Repetitivas - Multiplexión de Canales - Modems – Ecualización - Encripción de Datos - Fax – Celulares - Video Conferencias
Autos	Control de Motores - Análisis de Vibraciones - Posicionamiento Global Navegación - Comandos de Voz - Radio Digital - Teléfonos Celulares
Industrial	Robótica - Control Numérico - Accesos de Seguridad Monitoreo de Líneas de Energía
Médicas	Monitoreo de Pacientes - Equipamiento de Ultrasonido - Herramientas de Diagnóstico - Monitores Fetales

DSP vs Microprocesador tradicional

- **Algunas diferencias notables:**
  - Arquitectura Harvard en lugar de von Neumann: Memorias independientes para datos y programa
  - Optimización por *hardware* para ejecución de operaciones:
    - Multiplicación por hardware en un ciclo de reloj
    - Varias ALU
    - Accesos a memoria
    - Otra
  - Diversas formas de direccionamiento a memoria
  - En la mayoría de los casos trabajan con punto flotante (suma y multiplicación) en un ciclo de máquina

DSP

Ventajas:

Precisión.

Estabilidad.

Acumulación controlada del ruido.

Costo del hardware independiente de la complejidad.

Reprogramable.

Tarea a ser desarrollada por el DSP puede ser simulado.

Fácil de depurar, en un tiempo pequeño.

Menor costo y mayor fiabilidad.

Facilidad de transmisión y almacenamiento.

Desventajas:

Ancho de Banda limitado por la razón de muestreo.

Rango dinámico limitado.

Cuantización del ruido.

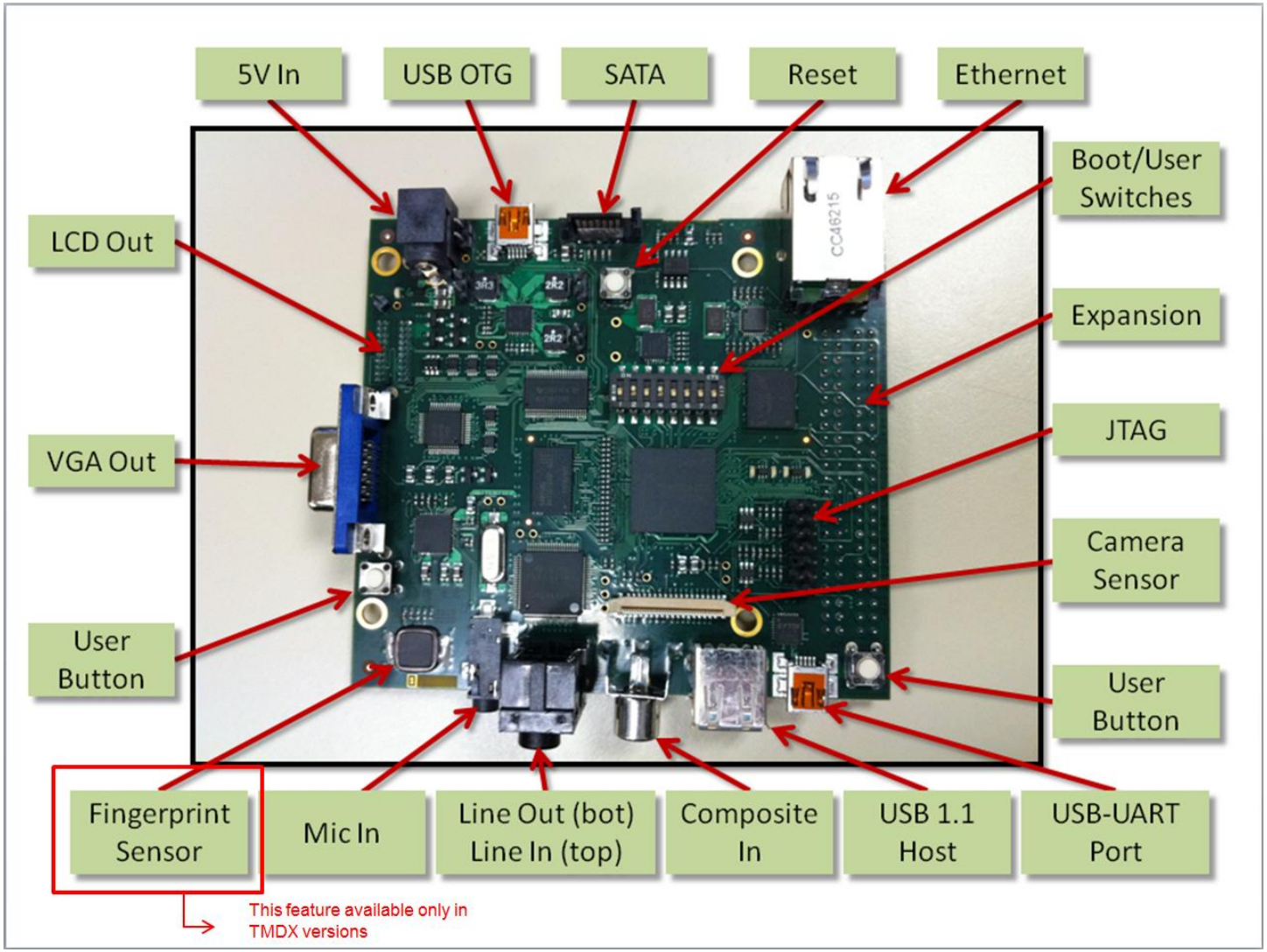
Errores de redondeo.

No es posible una exacta reconstrucción de la señal analógica original a partir de muestras cuantificadas.

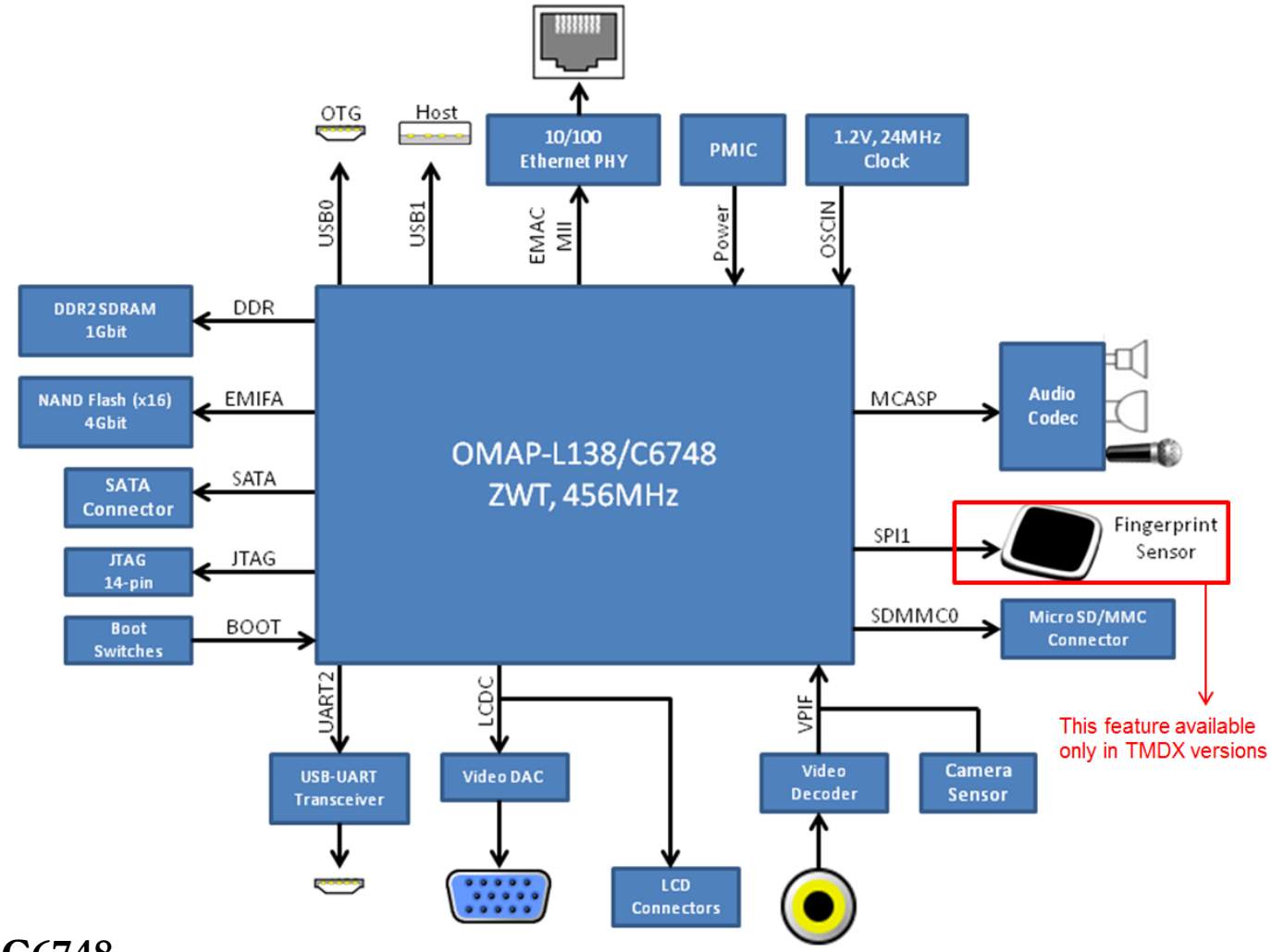
LCDK TMS320C6748

- **Especificaciones generales:**
  - CPU de la familia C674x
    - Punto Fijo y Punto flotante 32 bits,
    - VLIW con 8 bloques (posibilita paralelismo)
    - Puede operar hasta 456MHz
  - DDR2 con 128MB a 150MHz
  - Nand-Flash de 16 bits con 128MB
  - Slot para memoria SD
  - Códec de audio estéreo TLV320AIC3106 (AIC31) de hasta 32 bits y 96 kHz.
    - Tres conectores de 3.5mm en PCB: Lin In, Line Out, Mic In
  - Varias interfaces de comunicación
    - Ethernet, USB, USB OTG, SATA, VGA, Video Compuesto, puerto UART-USB
  - GPIOs para usuario: 2 pulsadores, 4 LEDs, 4 dipswitch
  - Interfaz estándar IEEE JTAG, requiere JTAG externo para debbuging via USB.
  - Puerto universal de alimentación de +5V

LCDK TMS320C6748



LCDK TMS320C6748



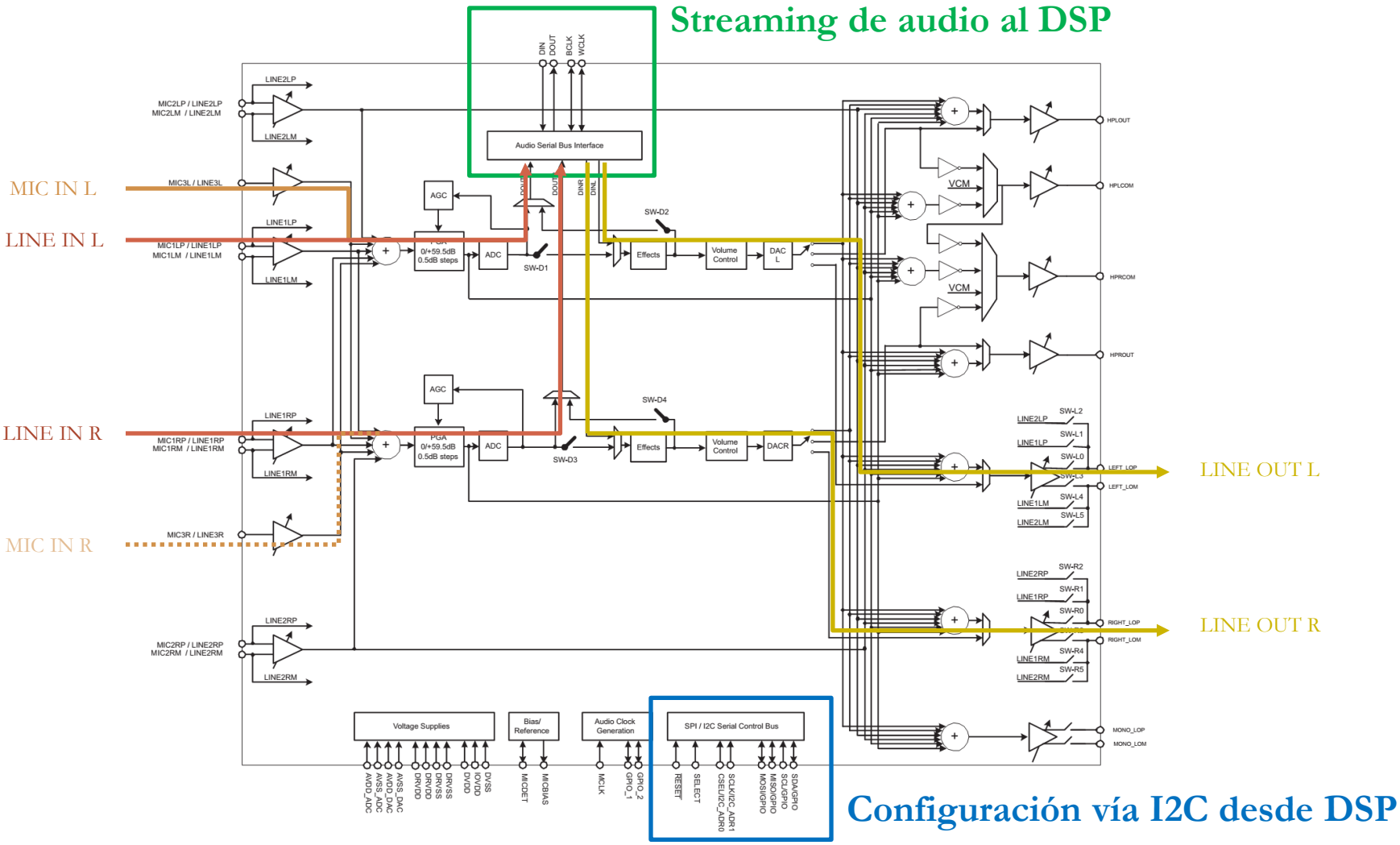
*** DSP es el C6748**

LCDK TMS320C6748 – AIC31

TLV320AIC3106

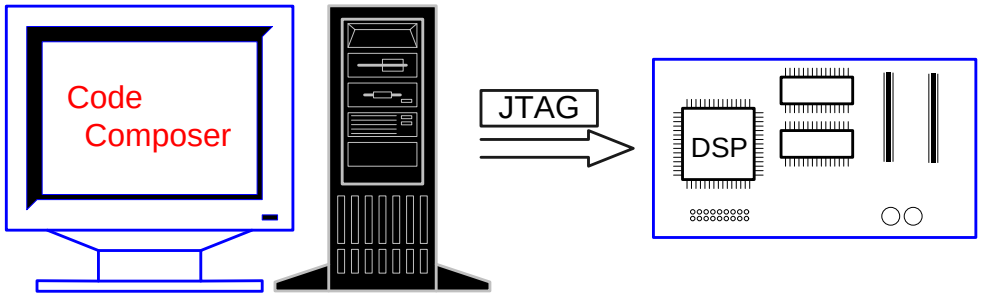
- Resolución: 16, 20, 24, o 32 bits
- Sampling rate: 96kHz, 48kHz, 44.1kHz, 24kHz, 22.05kHz, 12kHz, 11.025kHz, entre otros.
- Configuración de registros AIC31 vía I2C del DSP
- Streaming de entrada/salida de audio vía McASP del DSP
- Full scale de entrada/salida de audio: 0.707Vrms (2Vpp)

LCDK TMS320C6748 – AIC31



Code Composer Studio

- **CCS: Code Composer Studio**
  - IDE (Integrated Develepment Environment) de DSPs Texas Instruments.
  - Interface gráfica que permite interactuar con las CGT (Code Generation Tools)
  - En modo Edit, permite:
    - Editar de código fuente
    - Building (invocar a assembler, compilador, linker, archiver, entre otros)
    - Compila código en C/C++ y permite generar archivo de salida .out que se carga en una de las memorias del DSP.
  - En modo Debug, permite
    - Cargar .out al DSP y controlar el flujo del código.
    - Visualización y modificación de variables y registros en el DSP, de manera "on-line".
  - CCSv7, IDE versión actual y compatible con OS Windows solo posteriores a XP.

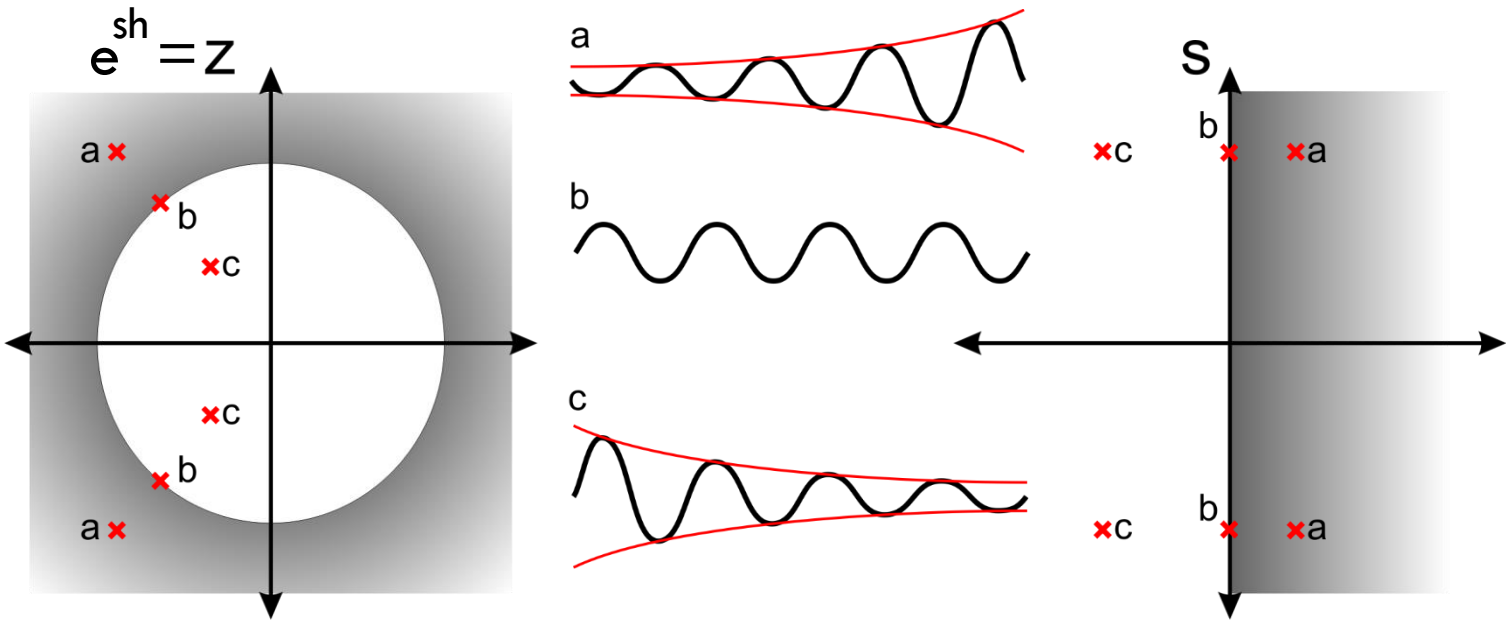


Conceptos básicos de sistemas lineales

- **Señales en tiempo discreto**
  - Muestreo de señales a taza fija (sampling rate)
  - Señales en tiempo discreto suelen representar señales análogas de tiempo continuo
- **Sistemas en tiempo discreto**
  - Respuesta a impulso
  - Convolución
  - Transformada Z
  - Sistemas interconectados
  - Polos y ceros en el plano Z
  - Osciladores digitales

Osciladores Digitales

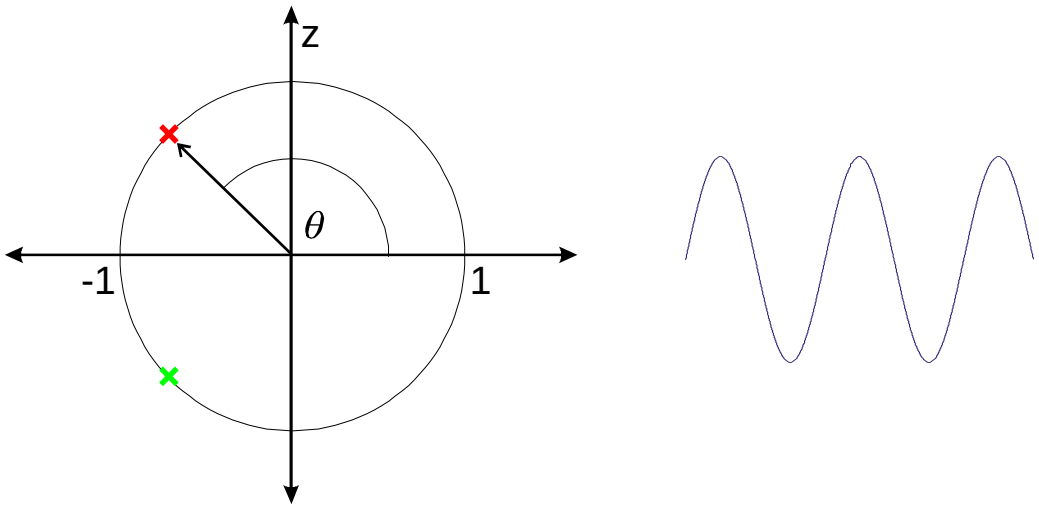
Para obtener una oscilación estacionaria es necesario ubicar los polos de un sistema de segundo orden en z, **en** el círculo unitario (equivalente al eje Re{s}=0 en sistemas continuos).



Osciladores Digitales

∴ para tener una oscilación sostenida, los polos deben ser de la forma:

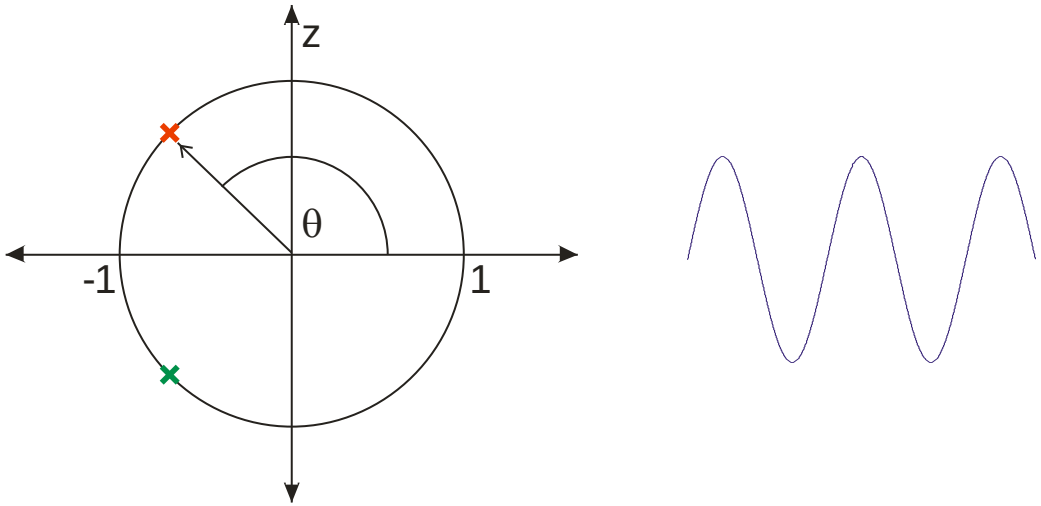
$$z = \cos(\theta) \pm j \sin(\theta)$$



Osciladores Digitales

La frecuencia puede controlarse variando el ángulo θ.

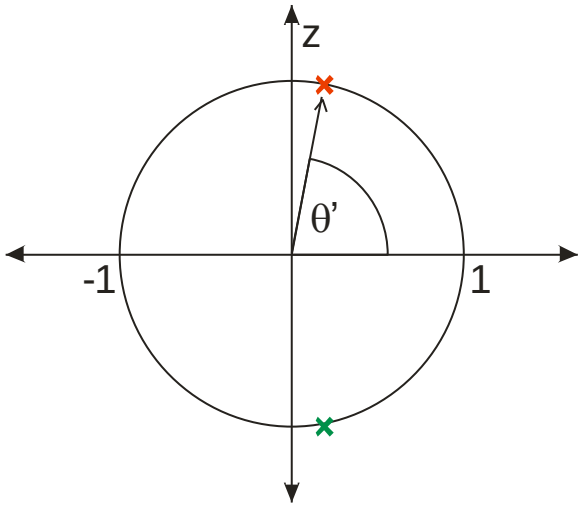
$$z = \cos(\theta) \pm j \sin(\theta)$$

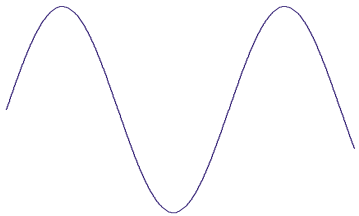


Osciladores Digitales

La frecuencia puede controlarse variando el ángulo θ.

$$z = \cos(\theta') \pm j \sin(\theta')$$





Osciladores Digitales

Por lo tanto, la función de transferencia es la siguiente:

$$
\begin{aligned}
H(z) &= \frac{\overline{b_o}\, z^2}{(z - \cos(\theta) + j\sin(\theta))(z - \cos(\theta) - j\sin(\theta))} \\
&= \frac{\overline{b_o}\, z^2}{(z - \cos(\theta))^2 - j^2 \sin(\theta)^2} \\
&= \frac{b_o\, z^2}{z^2 - 2\cos(\theta)z + \cos(\theta)^2 + \sin(\theta)^2} \\
&= \frac{\overline{b_o}\, z^2}{z^2 - 2\cos(\theta)z + 1}
\end{aligned}
$$

Osciladores Digitales

$\therefore$ la función de transferencia es:

$$H(z) = \frac{b_o}{1 - 2\cos(\theta) z^{-1} + z^{-2}}$$

para mantener constante la amplitud de salida independiente de la frecuencia:

$$b_o = \sin(\theta)$$

Recuerde que para que Y(z)=H(z), el sistema DEBE ser excitado por un ***impulso*** o ***delta de kroenecker***.

$$Y(z) = H(z)\ \delta(z)$$

Filtro Biquad

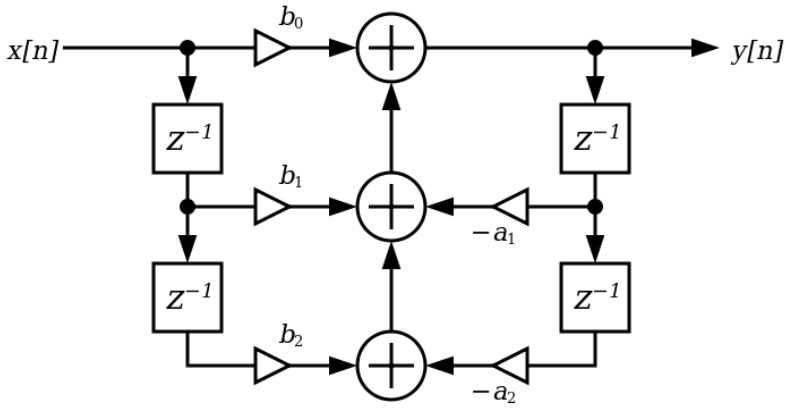
Transformada Z del sistema lineal discreto (normalizado):

$$H(z) = \frac{b_0 + b_1 z^{-1} + b_2 z^{-2}}{1 + a_1 z^{-1} + a_2 z^{-2}}$$

Ecuación de diferencias:

$$y[n] = b_0 x[n] + b_1 x[n-1] + b_2 x[n-2] - a_1 y[n-1] - a_2 y[n-2]$$

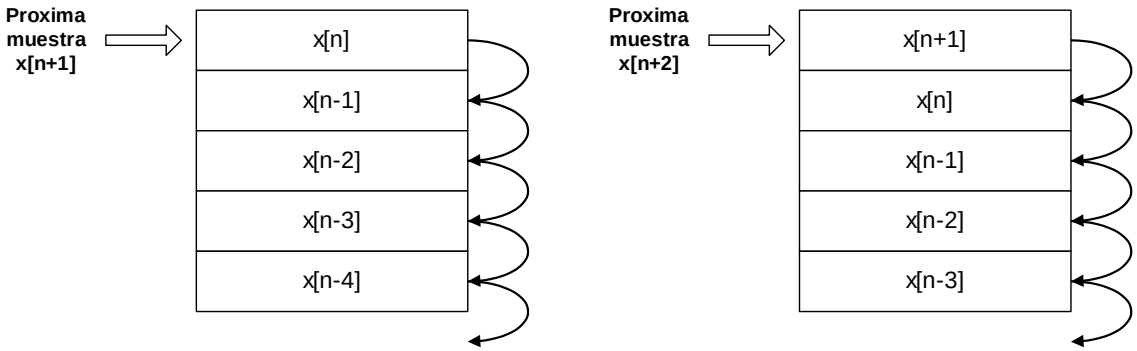
Implementación de filtro biquad (Directa 1):



Se requieren buffers para almacenar las muestras de las señales de entrada y salida

Buffers Lineales vs. Circulares

- **Buffer lineales**
  - Se desplazan todos los datos una posición hacia el final del buffer y se escribe el nuevo dato en la posición inicial
  - Intuitivo pero con alto costo en tiempo de ejecución para mover grandes cantidades de datos.



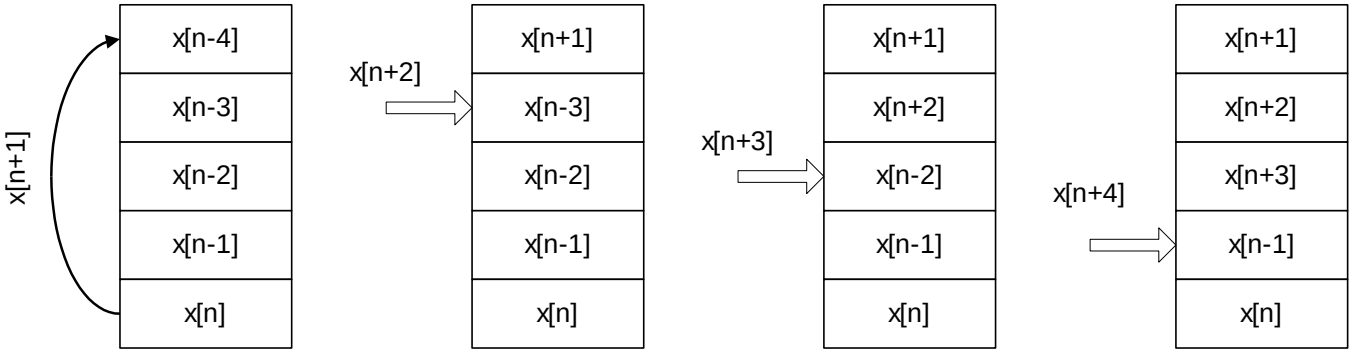
Buffers Lineales vs. Circulares

- **Buffer lineales**
  - Organizados por el índice de tiempo (muestra)
  - Algoritmo de actualización:
    1) Desecha el dato final del arreglo
    2) Mueve todos los datos una posición
    3) Inserta el nuevo dato
  - El dato más nuevo está siempre separado del más antiguo por todo el arreglo

```
for (i=BUFFER_SIZE-1; i>0; i--)
      buffer[i] = buffer[i-1];
buffer[0] = new_data;
```

Buffers Lineales vs. Circulares

- **Buffer circulares**
  - Se escribe el nuevo dato en la posición del más antiguo del arreglo
  - La operación módulo es conveniente para ubicar dicha posición
  - Menos intuitivo pero muy eficiente



Buffers Lineales vs. Circulares

- **Buffer circulares**
  - Organizados por la posición del dato más antiguo
  - Algoritmo de actualización:
    1) Ubica la posición del dato más antiguo
    2) Elimina ese dato e inserta el nuevo en dicha posición
    3) Actualiza la posición del dato más antiguo
  - El dato más nuevo está siempre junto al más nuevo (considerando los extremos del arreglo como unidos también)

```
buffer[i] = new_data;
i = ((i+1)%BUFFER_SIZE);
```

Buffers Lineales vs. Circulares

- **Observaciones**
  - Todo tipo de buffer requiere una inicialización previa a su uso (Condición inicial)
  - Retardos en tiempos de ejecución mayores al período de muestreo pueden traer importantes problemas en el manejo de la señal
  - Buffer lineales no son usados en DSP en tiempo real debido al costo en tiempo de gestión de buffer lineal.