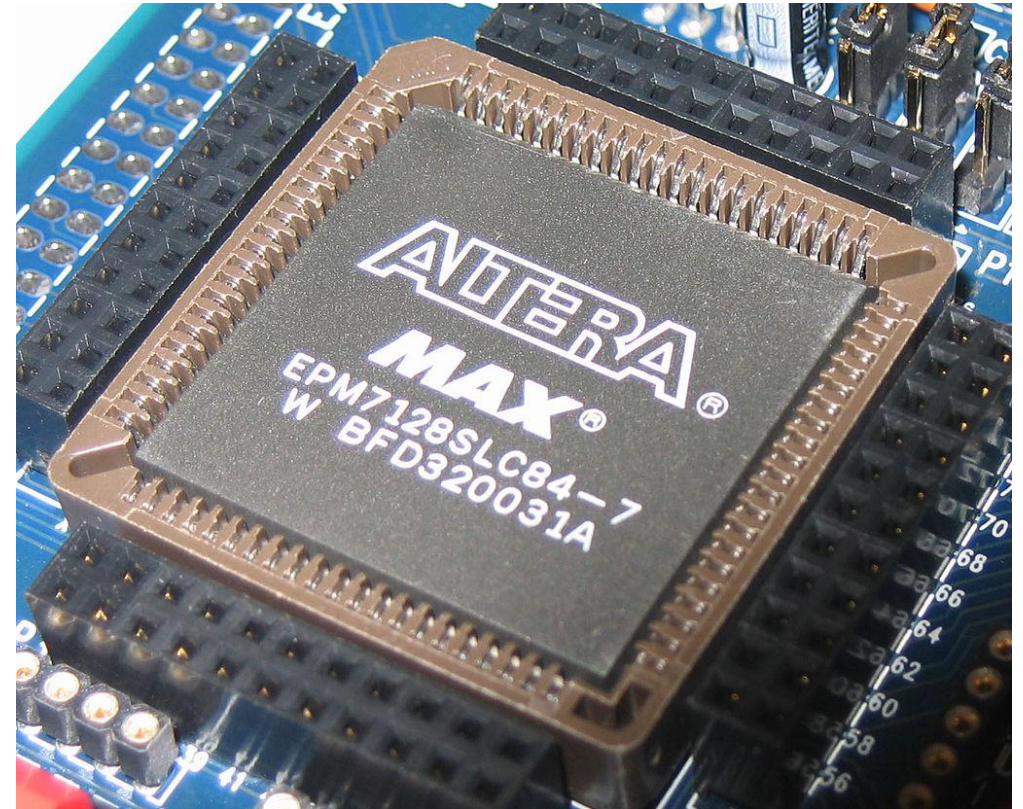


# *Sistemas embebidos II*

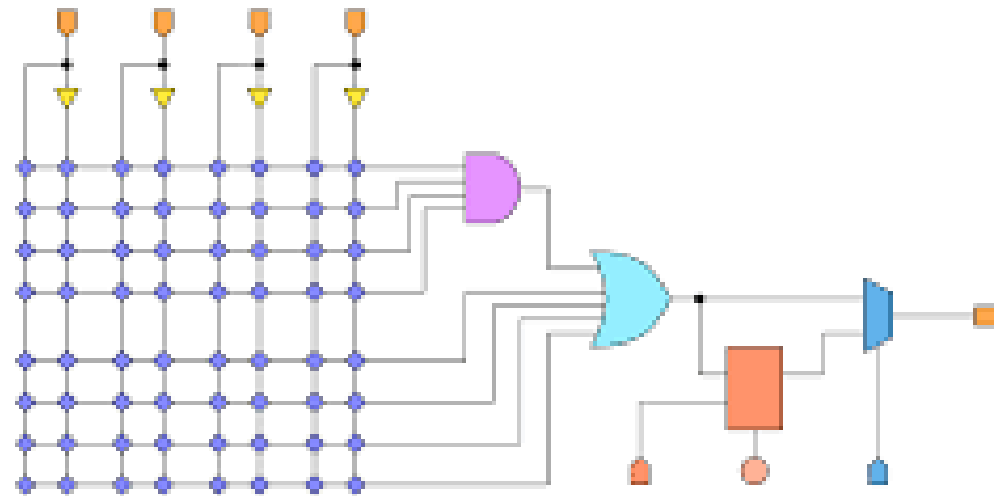
## *Sesión 3*

# Arquitectura de las FPGA's

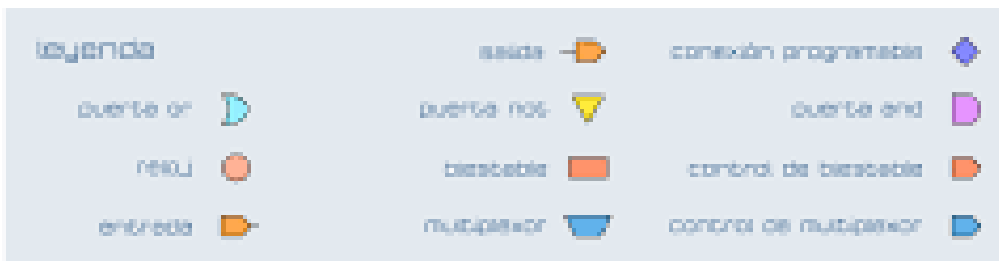
Las FPGAs son dispositivos altamente flexibles en el desarrollo y aplicación de tecnología, caracterizados por una arquitectura compleja que debe ser cuidadosamente considerada al trabajar con ellos. Estas FPGAs están compuestas por Bloques Lógicos Programables (CLB), conexiones internas entre estos bloques conocidas como el "Fabric" o tejido, que actúan como rutas para la transferencia de datos. Además, incluyen bloques dedicados a la comunicación externa de la FPGA, conocidos como bloques de entrada y salida.



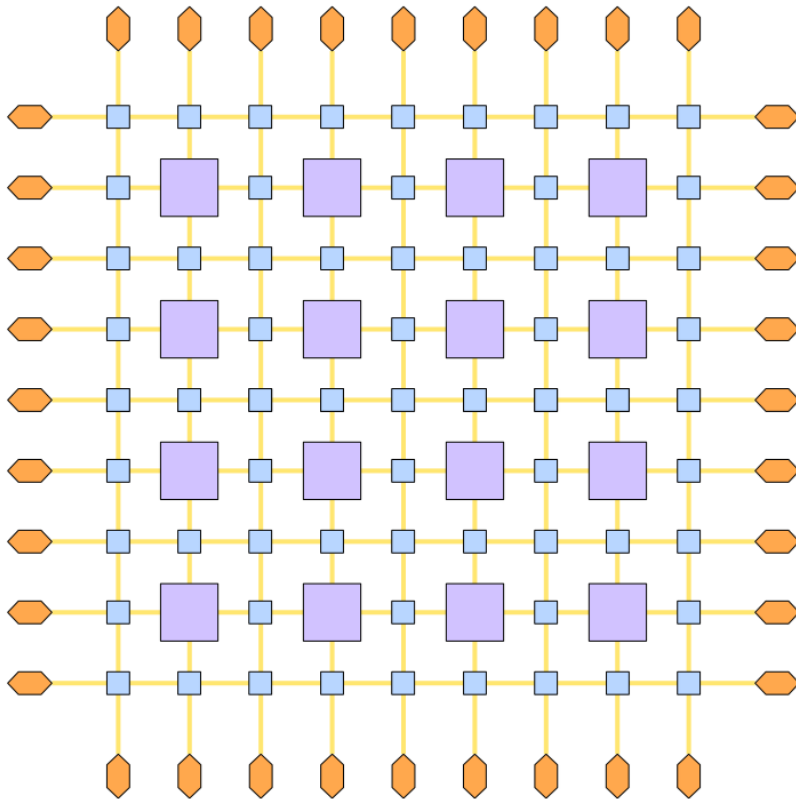
# Lógica programada



La expresión «lógica programada» (o «lógica programable») hace referencia a la posibilidad de establecer la operativa lógica de un circuito integrado definiéndola por medio de un lenguaje que permite describirla a un nivel más o menos alto y que puede posteriormente ser traducida para implementarla en el dispositivo lógico, bien durante su fabricación o bien posteriormente.



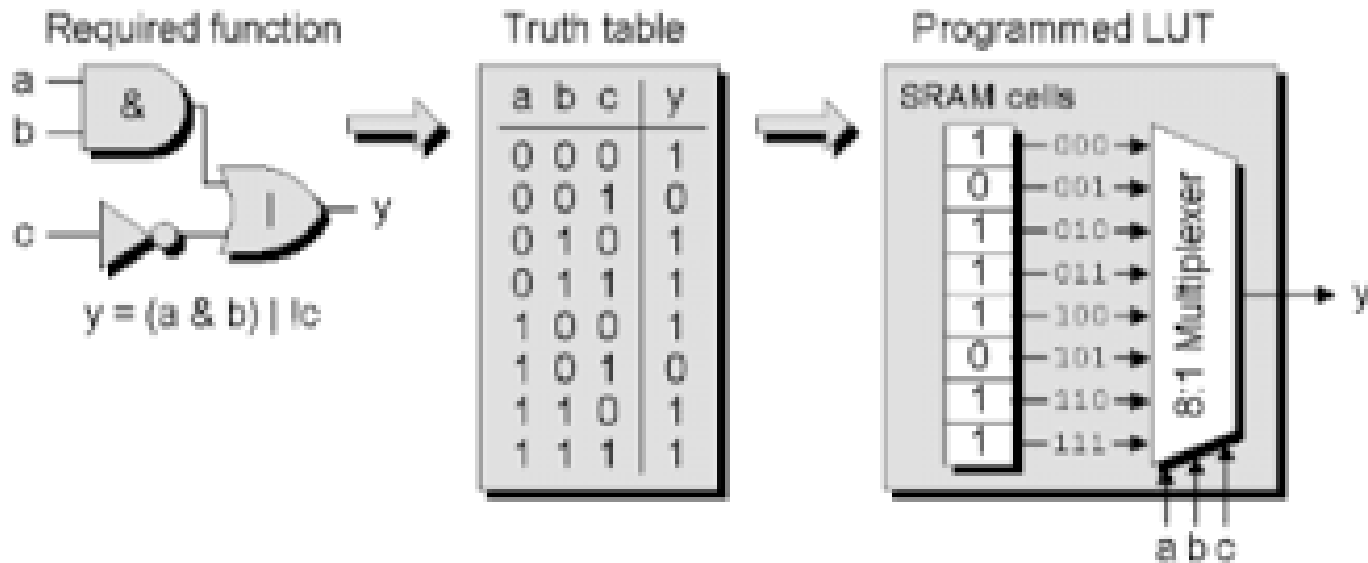
# Bloques Lógicos Configurables



leyenda    bus de interconexión    matriz de conmutación programable    bloque de entrada-salida    bloque lógico configurable

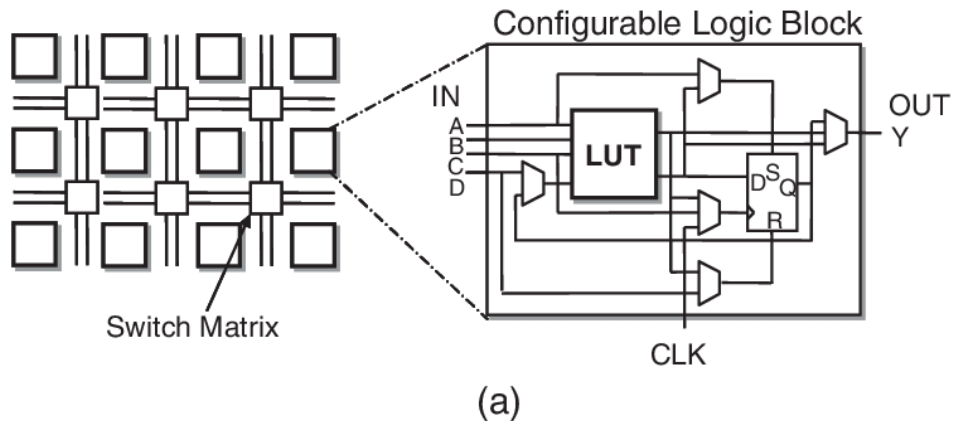
- Son el núcleo de la FPGA. Cada bloque lógico contiene varias LUTs (Look-Up Tables), flip-flops y otros elementos básicos de lógica.
- LUTs se usan para implementar funciones lógicas combinacionales.
- Flip-flops se utilizan para almacenar estados y crear circuitos secuenciales.
- Cada CLB puede ser configurado para realizar una variedad de funciones lógicas, desde puertas simples hasta operaciones más complejas.

# Bloques Lógicos Configurables (LUT's)

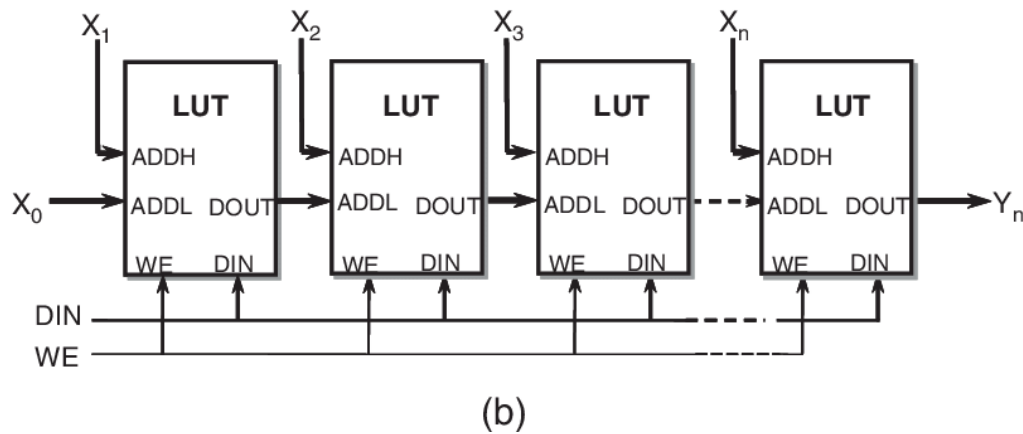


Una tabla de consulta es un componente dentro de las FPGAs que se utiliza para implementar funciones lógicas combinacionales. Un LUT funciona como una pequeña tabla de verdad que puede ser programada para realizar cualquier función lógica con un número fijo de entradas.

# Bloques Lógicos Configurables (LUT's)

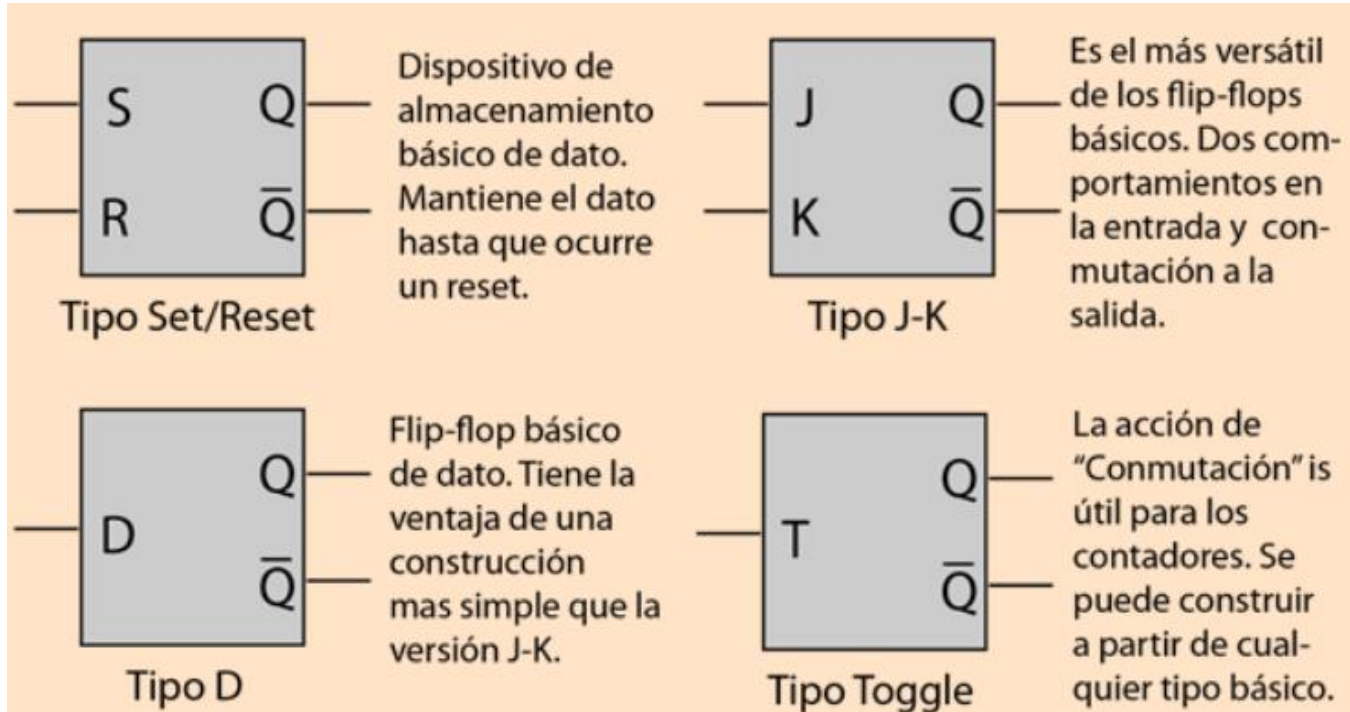


- **Función Lógica Programable:** Un LUT tiene un número definido de entradas. Puede implementar cualquier función lógica combinacional para esas entradas.
- **Operación Interna:** Internamente, un LUT funciona como una pequeña memoria que almacena los resultados de todas las posibles combinaciones de las entradas. Cuando se aplican valores a las entradas del LUT, esta consulta la tabla y da como salida el valor correspondiente almacenado.





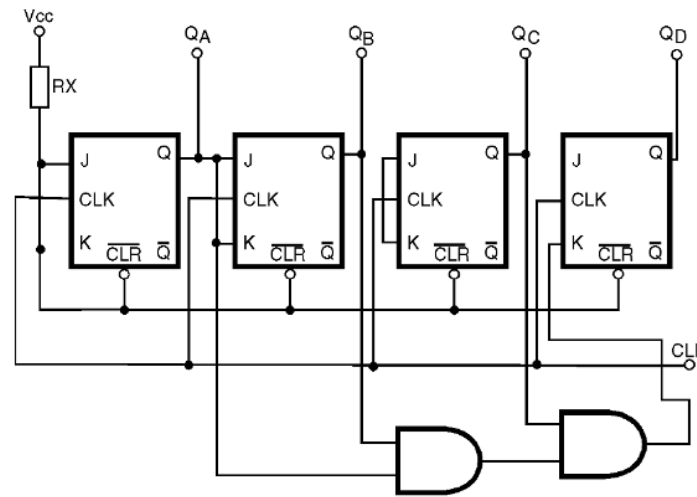
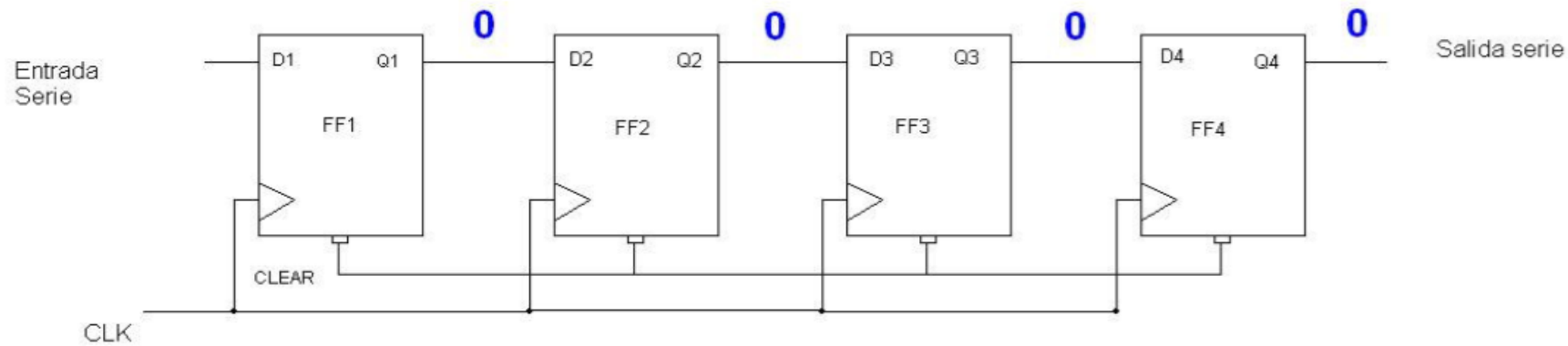
# Bloques Lógicos Configurables (flip-flops)



Son componentes fundamentales en la electrónica digital, usados para almacenar información y sincronizar el comportamiento de los circuitos. A diferencia de los circuitos combinacionales, los flip-flops son circuitos secuenciales, lo que significa que dependen tanto de las entradas actuales como de los valores almacenados previamente.

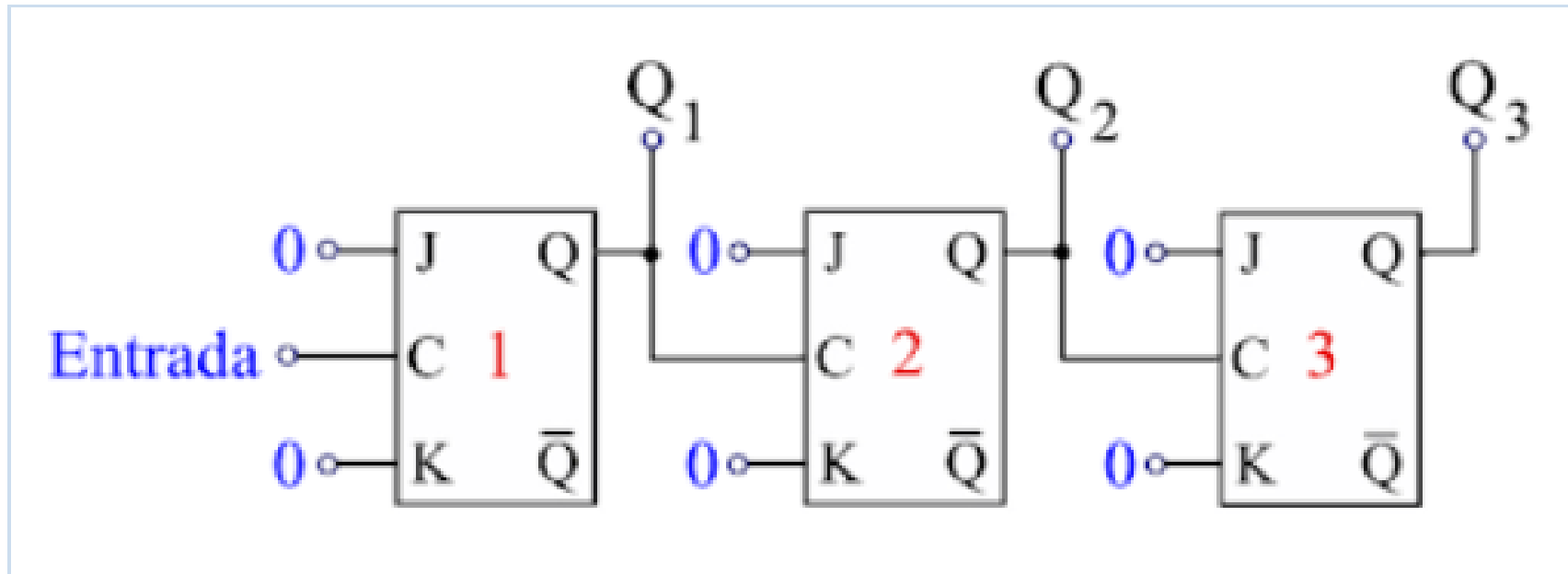
Un flip-flop almacena un bit de información (0 o 1) y cambia su estado en función de una señal de control, típicamente un reloj (clock).

# Bloques Lógicos Configurables (flip-flops)

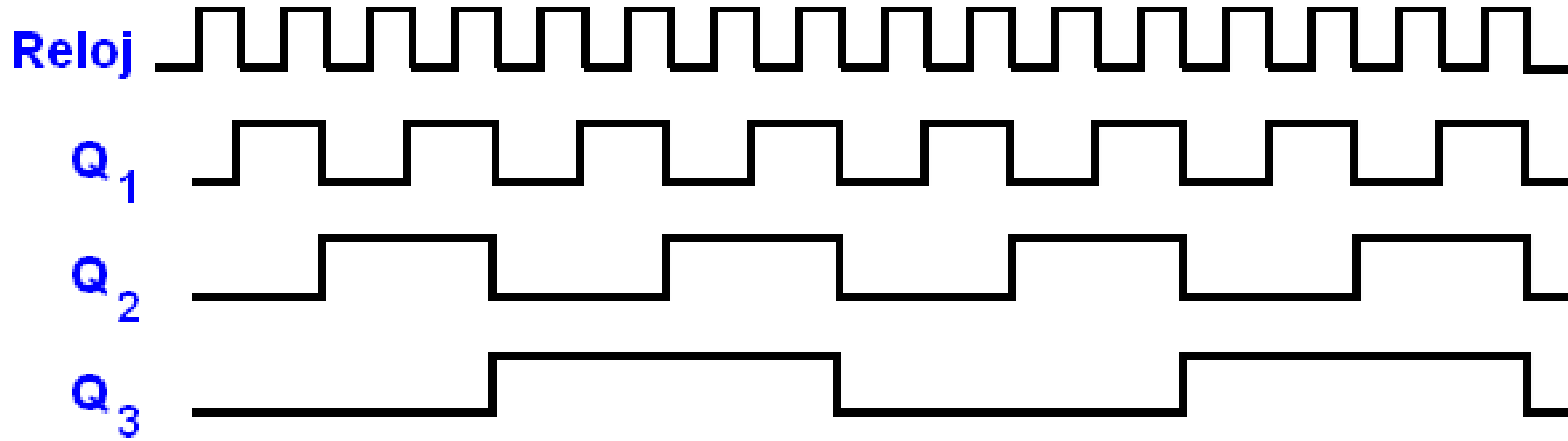




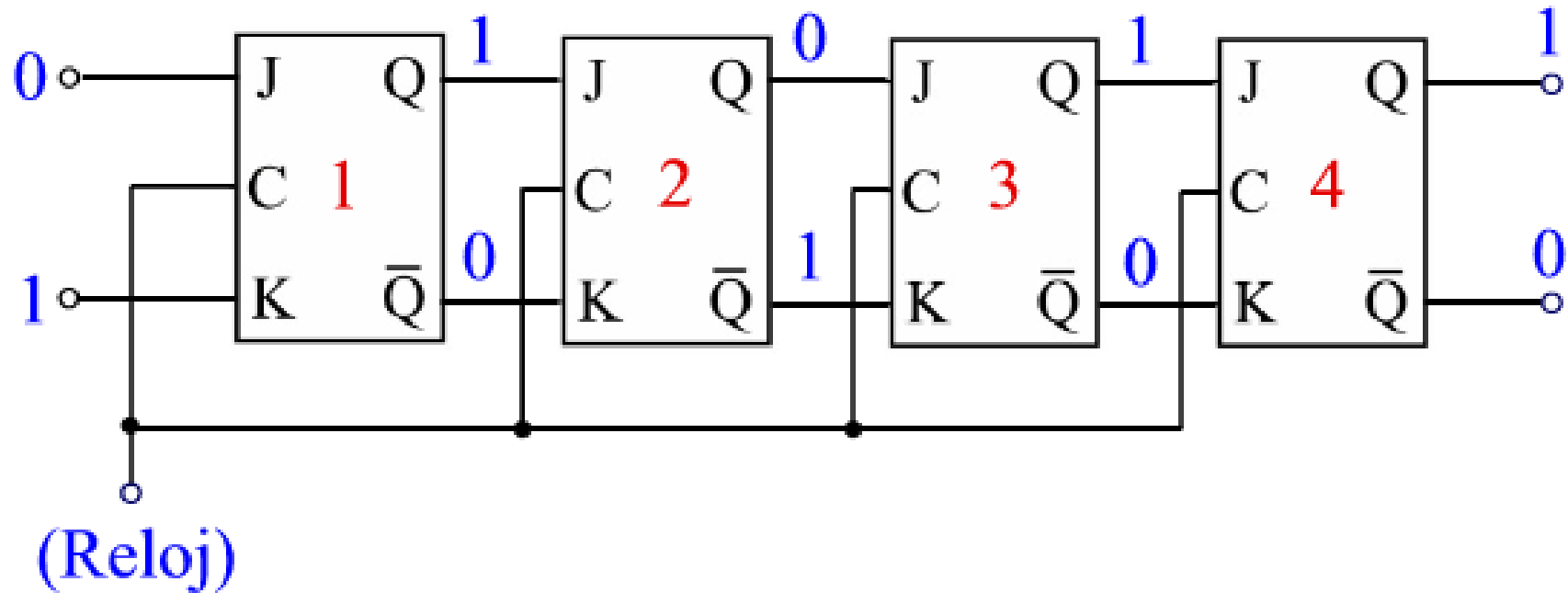
# Bloques Lógicos Configurables (flip-flops)



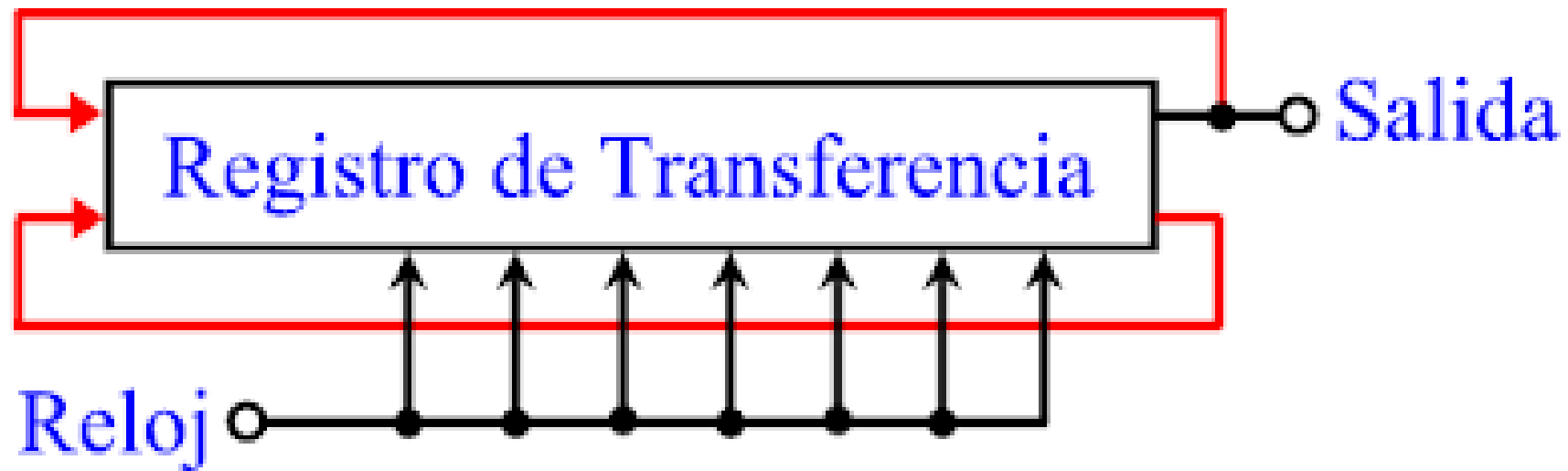
# Bloques Lógicos Configurables (flip-flops)



# Bloques Lógicos Configurables (flip-flops)



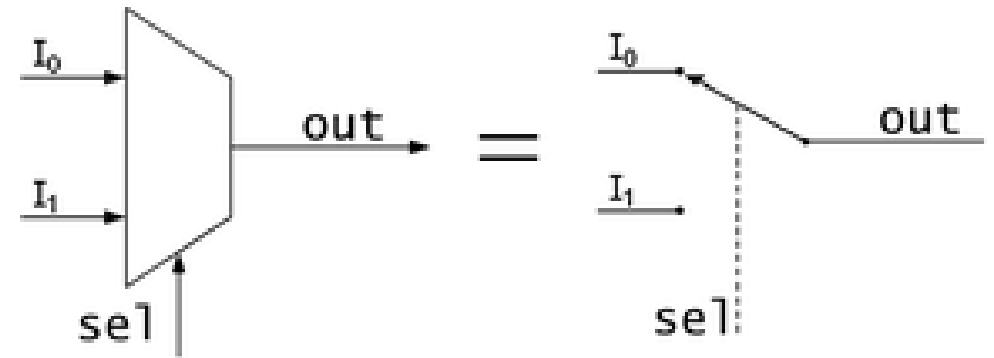
# Bloques Lógicos Configurables (flip-flops)



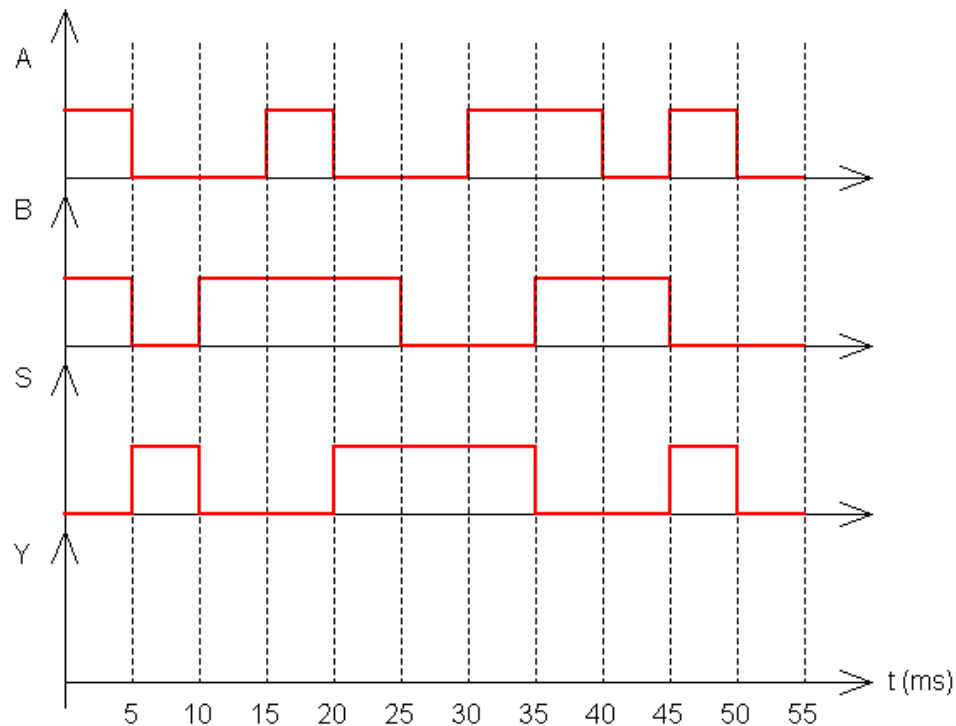
# Bloques Lógicos Configurables (Multiplexores)

Es un dispositivo lógico digital que selecciona una de varias señales de entrada y la dirige a una única línea de salida. Su principal función es permitir que varias señales compartan un único recurso o línea de comunicación.

- Entradas de datos
- Entradas de selección



# Bloques Lógicos Configurables (Multiplexores)



Intervalo de tiempo (ms)	Entrada de selección	Salida
	S	Y
0-5	0	A
5-10	1	B
10-15	0	A
15-20	0	A
20-25	1	B
25-30	1	B
30-35	1	B
35-40	0	A
40-45	0	A
45-50	1	B
50-55	0	A

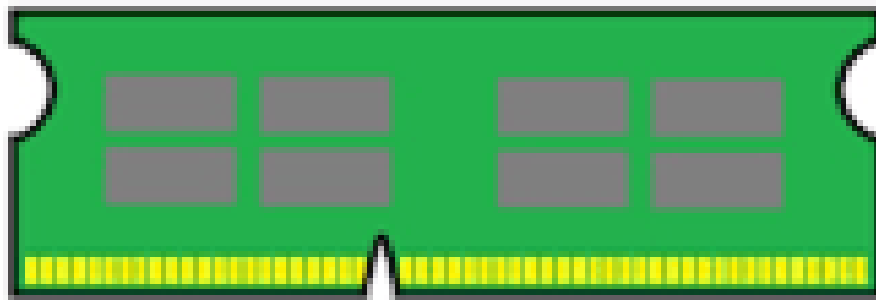
# Arquitectura de las FPGA's

- **Interconexiones Programables:** Las FPGA tienen una red extensa de interconexiones que permiten conectar los bloques lógicos de forma flexible. Estas conexiones son configurables, lo que permite modificar cómo se conectan los CLBs entre sí y con otros elementos. Existen caminos cortos y largos dentro de la FPGA para optimizar la latencia y el rendimiento.
- **Bloques de Entradas/Salidas:** Estos bloques permiten la comunicación entre la FPGA y el mundo exterior, ya sea con otros circuitos, sensores, o actuadores. Pueden ser configurados para manejar diferentes estándares de señal (como LVDS, TTL, etc.) y ajustar los niveles de voltaje. Los bloques I/O también permiten la adaptación de las señales externas para trabajar con las señales internas de la FPGA.

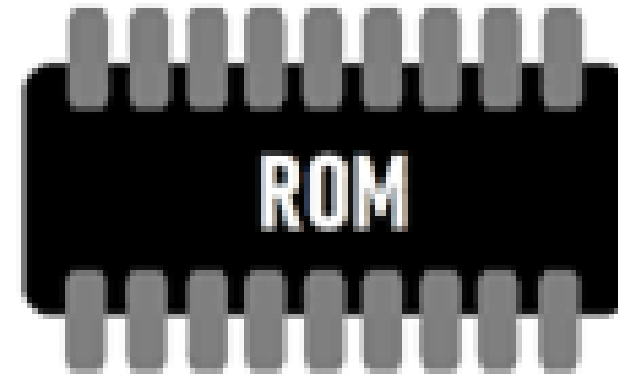


# Arquitectura de las FPGA's ( )

- **Memoria RAM y ROM Integrada:** Las FPGA suelen tener bloques de BRAM (Block RAM) integrados que permiten el almacenamiento temporal de datos. Los bloques de RAM son configurables y pueden ser utilizados para crear memorias de diferentes tamaños y configuraciones (ROM, RAM dual-port, etc.).



RAM - Memory Chip



ROM - Memory

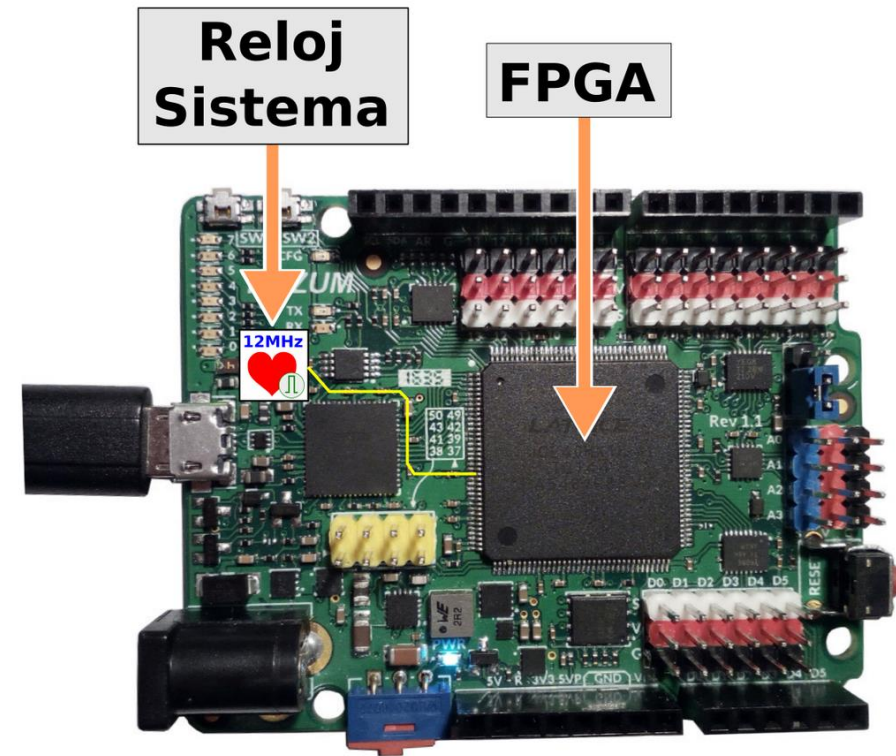
# Arquitectura de las FPGA's ()

- **DSPs:** Algunos FPGAs incluyen bloques DSP especializados para realizar operaciones aritméticas de alta velocidad, como multiplicaciones y sumas. Estos bloques son muy útiles para aplicaciones de procesamiento de señales, procesamiento de imágenes o machine learning.



# Arquitectura de las FPGA's (Relojes y Circuitos de Control)

- **Relojes y Circuitos de Control:** Las FPGA incluyen redes de distribución de reloj para sincronizar las operaciones en diferentes bloques lógicos. Pueden tener varios osciladores o ser configuradas para trabajar con señales de reloj externas. También pueden incluir PLLs (Phase-Locked Loops) o DLLs (Delay-Locked Loops) para gestionar la señal del reloj.



# Arquitectura de las FPGA's (Módulos de Comunicación)



- **Módulos de Comunicación:** En aplicaciones de alto rendimiento, las FPGA pueden incluir transceptores de alta velocidad, utilizados para interfaces de comunicación como Ethernet, PCIe, o USB. Estos transceptores permiten la transmisión y recepción de datos a altas velocidades, haciendo que las FPGAs sean ideales para aplicaciones de red.

# Arquitectura de las FPGA's

- **Soft Cores y Hard Cores:** Las FPGA pueden tener soft cores (núcleos de procesadores definidos en lógica programable) que se implementan en los bloques lógicos de la FPGA, o hard cores (núcleos predefinidos de procesadores) que están físicamente integrados en el silicio. Los soft cores son más flexibles, pero los hard cores son más eficientes en cuanto a recursos y rendimiento.

# Elementos del lenguaje

VHDL es un lenguaje de descripción de hardware que se utiliza para modelar, diseñar y simular circuitos digitales. Su propósito principal es describir el comportamiento y la estructura de sistemas electrónicos, permitiendo la creación de diseños que pueden ser sintetizados en hardware, como FPGAs o ASICs.

Aquí están los elementos más importantes del VHDL:

<https://www.youtube.com/watch?v=g7Z-sBpmCJU>

```
1 library ieee;
2 use ieee.std_logic_1164.all;
3 use ieee.numeric_std.all;
4
5 entity signed_adder is
6   port
7   (
8     aclr : in    std_logic;
9     clk  : in    std_logic;
10    a     : in    std_logic_vector;
11    b     : in    std_logic_vector;
12    q     : out   std_logic_vector
13  );
14 end signed_adder;
15
16 architecture signed_adder_arch of signed_adder is
17   signal q_s : signed(a'high+1 downto 0); -- extra bit wide
18
19 begin -- architecture
20   assert(a'length >= b'length)
21     report "Port A must be the longer vector if different sizes!"
22     severity FAILURE;
23   q <= std_logic_vector(q_s);
24
25   adding_proc:
26   process (aclr, clk)
27   begin
28     if (aclr = '1') then
29       q_s <= (others => '0');
30     elsif rising_edge(clk) then
31       q_s <= ('0'&signed(a)) + ('0'&signed(b));
32     end if; -- clk'd
33   end process;
34
35 end signed_adder_arch;
```

# Elementos del lenguaje (Entidad)

La entidad define la interfaz de un componente de hardware, describiendo sus entradas y salidas. Es el bloque externo del diseño y describe qué hace el sistema sin entrar en detalles de cómo lo hace. Dentro de la entidad, se declara el puerto (las señales de entrada y salida).

**Module Definition**

Entity name:

Architecture name:

**I/O Port Definitions**

Port Name	Direction	Bus	MSB	LSB
	<div><div>in</div><div>in</div><div>out</div><div>inout</div></div>	<input type="checkbox"/>	0	0

?

OK Cancel

```
entity AND_GATE is
    port(
        A : in  std_logic;
        B : in  std_logic;
        Y : out std_logic
    );
end AND_GATE;
```



# Elementos del lenguaje (Arquitectura)

La arquitectura describe el funcionamiento interno o la implementación de la entidad. Aquí se detalla el cómo se implementa el comportamiento del diseño especificado en la entidad. Puede contener descripciones comportamentales, estructurales o una mezcla de ambas.

```
architecture Behavioral of AND_GATE is
begin
    Y <= A and B;  -- Comportamiento de la compuerta AND
end Behavioral;
```

# Elementos del lenguaje (Proceso)

```
process (A, B)
begin
    if (A = '1' and B = '1') then
        Y <= '1';
    else
        Y <= '0';
    end if;
end process;
```

Los procesos son bloques secuenciales dentro de una arquitectura que permiten describir comportamientos que dependen del tiempo o de eventos. Un proceso puede contener sentencias condicionales (if, case), bucles y otras estructuras secuenciales. Los procesos son muy útiles para describir circuitos secuenciales como flip-flops o máquinas de estados.

# Elementos del lenguaje (Señales y Variables)

Las señales se utilizan para la comunicación entre diferentes procesos o entre la entidad y la arquitectura. Las variables (variable), por otro lado, son locales a un proceso y se usan para almacenar valores temporales dentro de un proceso. Las señales tienen un retraso inherente (simulan el retardo de propagación en el hardware), mientras que las variables son instantáneas dentro del proceso.

```
signal temp: std_logic;  
variable counter: integer := 0;
```

# Elementos del lenguaje (Tipos de Datos)

VHDL incluye diversos tipos de datos para modelar señales y variables, los más comunes son:

- `std_logic`: Representa un bit y puede tomar varios valores como '0', '1', 'Z' (impedancia alta), 'X' (indeterminado), etc.
- `std_logic_vector`: Un vector de varios bits.
- `integer`: Números enteros.
- `boolean`: Verdadero o falso.

```
architecture ejemplo of tipos_de_datos is
    -- Declaración de señales con diferentes tipos de datos

    -- BIT: un único bit ('0' o '1')
    signal bit_example : BIT := '1';

    -- BIT_VECTOR: vector de bits
    signal bit_vector_example : BIT_VECTOR(3 downto 0) := "1010";

    -- STD_LOGIC: más flexibilidad que BIT ('0', '1', 'Z', 'X', etc.)
    signal std_logic_example : STD_LOGIC := '1';

    -- STD_LOGIC_VECTOR: vector de señales STD_LOGIC
    signal std_logic_vector_example : STD_LOGIC_VECTOR(7 downto 0) := "11001100";

    -- INTEGER: enteros
    signal integer_example : INTEGER := 10;

    -- REAL: número real con decimales
    signal real_example : REAL := 3.14;

    -- SIGNED: número con signo (positivo o negativo) en forma binaria
    signal signed_example : SIGNED(7 downto 0) := to_signed(-15, 8);

    -- UNSIGNED: número sin signo (solo positivo) en forma binaria
    signal unsigned_example : UNSIGNED(7 downto 0) := to_unsigned(15, 8);

    -- BOOLEAN: valores lógicos (TRUE o FALSE)
    signal boolean_example : BOOLEAN := TRUE;
```

# Elementos del lenguaje (Concurrente vs Secuencial)

Concurrente: Las sentencias concurrentes se ejecutan simultáneamente y son naturales para describir lógica combinacional. Ejemplos de sentencias concurrentes incluyen asignaciones simples o instancias de componentes.

Secuencial: Dentro de los procesos, las sentencias se ejecutan de manera secuencial, lo cual es útil para describir el comportamiento temporal o secuencial de un sistema.

# Elementos del lenguaje (Asignación de Señales)

La asignación de valores a las señales se hace usando el operador `<=` (asignación concurrente). Dentro de un proceso, las variables se asignan con el operador `:=`, pero las señales siguen usando `<=`.

```
signal Result: std_logic;  
Result <= A and B;  -- Asignación de señal
```

- Concurrente: Las sentencias concurrentes se ejecutan simultáneamente y son naturales para describir lógica combinacional. Ejemplos de sentencias concurrentes incluyen asignaciones simples o instancias de componentes.
- Secuencial: Dentro de los procesos, las sentencias se ejecutan de manera secuencial, lo cual es útil para describir el comportamiento temporal o secuencial de un sistema.

# Elementos del lenguaje (Componentes e Instancias)

Los componentes permiten la reutilización de otros diseños dentro de un diseño más grande, similar a módulos en otros lenguajes de hardware. Puedes declarar e instanciar componentes dentro de la arquitectura.

```
component AND_GATE is
    port (A : in std_logic; B : in std_logic; Y : out std_logic);
end component;

U1: AND_GATE port map (A => signal_A, B => signal_B, Y => signal_Y);
```



# Elementos del lenguaje (ejemplo)

- Entidad compuertas\_and\_or:
  - Define las entradas (a, b, c, d) y salidas (and\_out, or\_out, sum\_out, real\_val).
  - a, b, c, d son señales de tipo STD\_LOGIC que se usan para las compuertas lógicas.
  - and\_out y or\_out son salidas de tipo STD\_LOGIC para los resultados de las compuertas AND y OR, respectivamente.
  - sum\_out es una salida de tipo INTEGER que mostrará el resultado de una operación aritmética.
  - real\_val es una salida de tipo REAL que mostrará el resultado de una operación con números reales.
- Arquitectura behavioral:
  - Se definen señales internas and\_result, or\_result, integer\_sum, y real\_example.
  - Compuerta AND: Calcula el resultado de la operación a and b y lo asigna a and\_result.
  - Compuerta OR: Calcula el resultado de c or d y lo asigna a or\_result.

# Elementos del lenguaje (ejemplo)

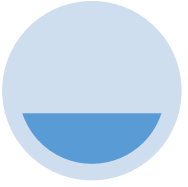
- Proceso:
  - Dentro del proceso: Se asignan los resultados de las compuertas AND y OR a las salidas `and_out` y `or_out`, respectivamente.
  - Se realiza una operación de suma con valores enteros concatenando `a` y `b`, convirtiendo la concatenación a un entero y asignando el resultado a `sum_out`.
  - Se realiza una operación aritmética con valores reales y se asigna el resultado a `real_val`.

# Filtros

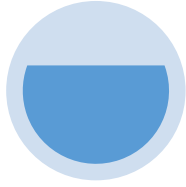
Es un circuito electrónico que posee una entrada y una salida  
En la entrada se introduce señales alternas de diferentes frecuencias y en la salida se extraen señales atenuadas en mayor o menor medida según la frecuencia de la señal.



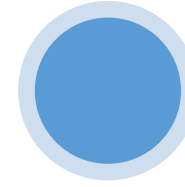
# Filtros



Activos  
Pasivos



Analógicos  
Digitales

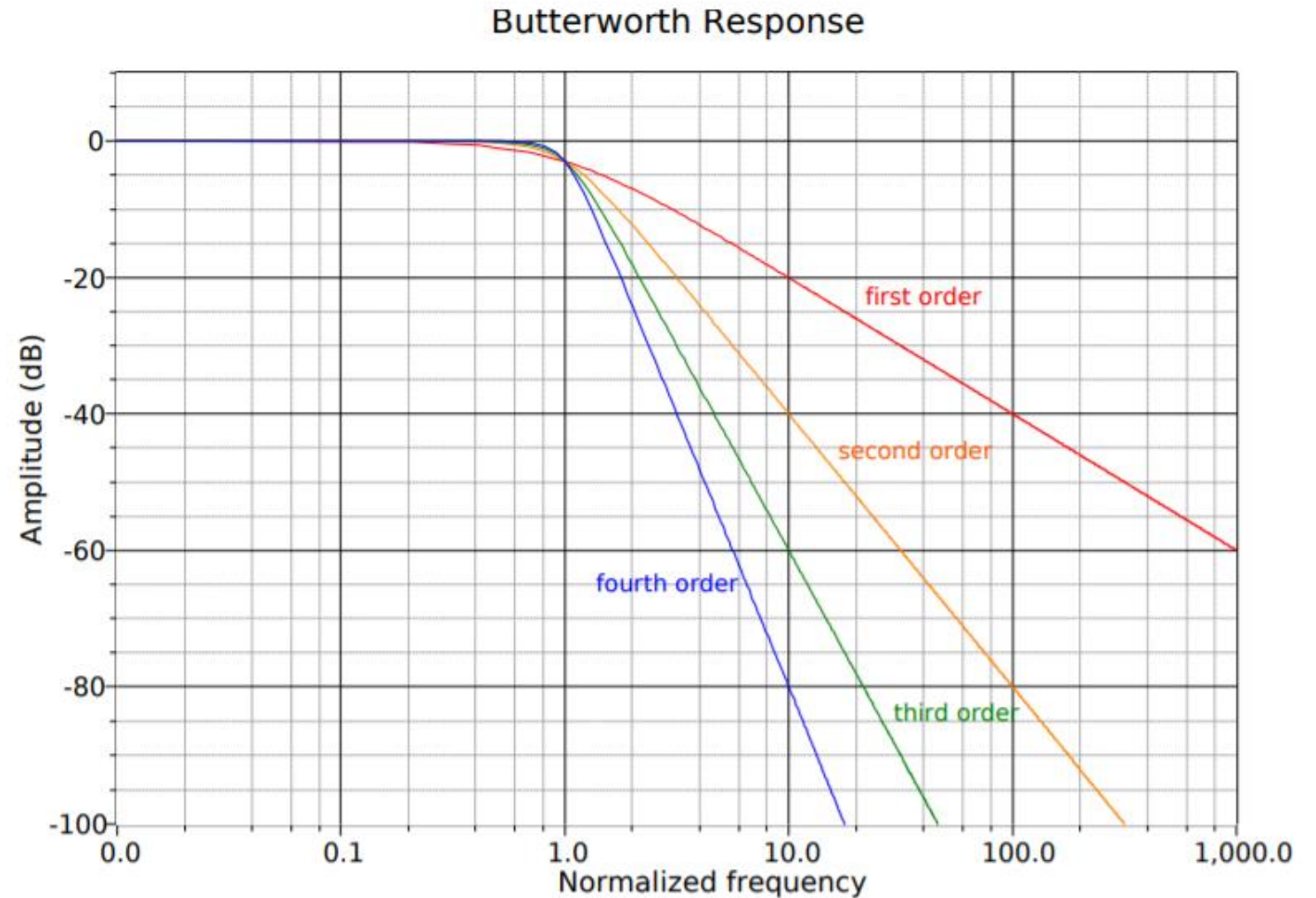


Filtro de paso bajo  
Filtro de paso alto  
Filtro de paso  
banda  
Filtro de banda  
eliminada  
Filtro multibanda  
Filtro variable

# Filtros - Orden

Un filtro es un circuito con al menos un elemento reactivo. Un circuito con solo un elemento reactivo es un “filtro de primer orden”, si el circuito tiene dos elementos reactivos es un “filtro de segundo orden”...

La diferencia que existe entre un filtro de primer orden y un filtro de orden mayor es la curva de respuesta de frecuencia.

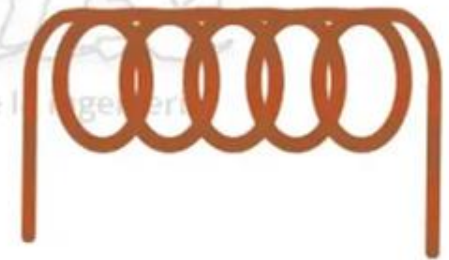


# Filtros - Orden

Resistencia

Capacitor

Bobina



Ingeniería  
Mezclada  
La enciclopedia de la ingeniería

# Filtros - Frecuencia

Se refiere a la cantidad de repeticiones de un fenómeno periódico en una unidad de tiempo.

$$1 \text{ Hz} = \left[ \frac{1}{\text{s}} \right]$$



$$f = 0.5 \text{ Hz}$$
$$T = 2.0 \text{ s}$$



$$f = 1.0 \text{ Hz}$$
$$T = 1.0 \text{ s}$$

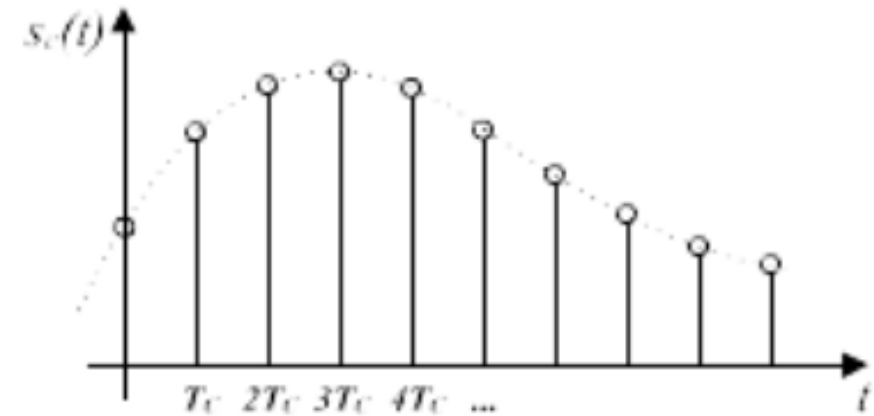
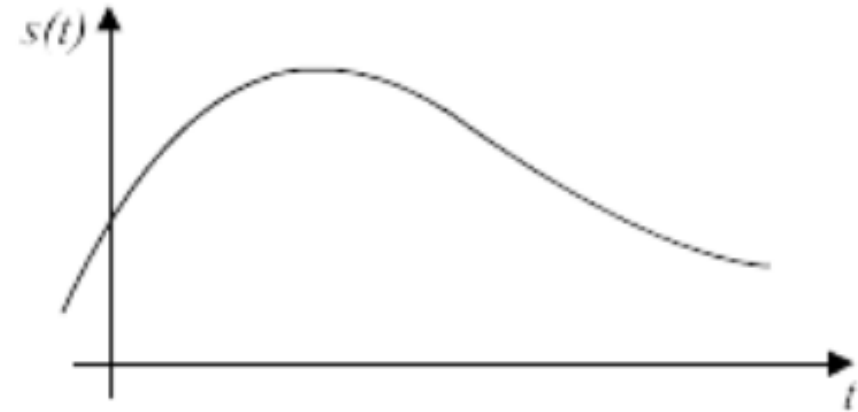


$$f = 2.0 \text{ Hz}$$
$$T = 0.5 \text{ s}$$



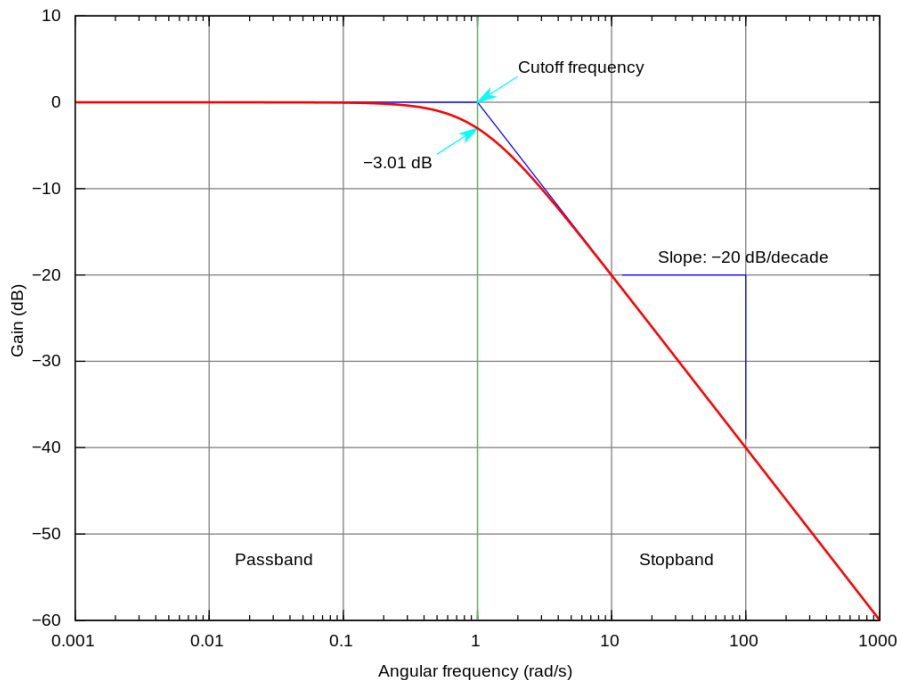
# Filtros – S/s

Es el número de muestras por unidad de tiempo que se toman de una señal continua para producir una señal discreta,



# Filtros - Frecuencia de corte

Es un límite en la respuesta frecuencial de un sistema en el cual la energía que fluye a través de este se comienza a reducir (atenuar o reflejar) en lugar de pasar a través de él.

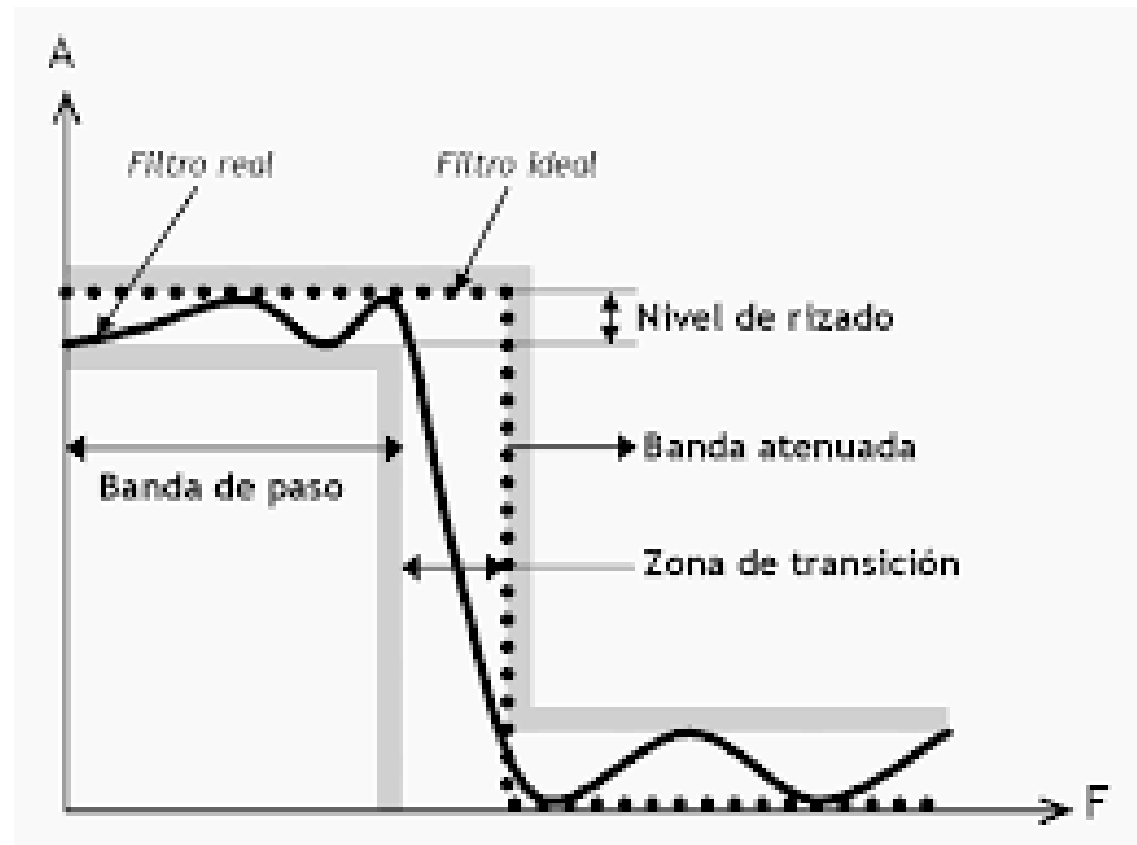


Es la frecuencia para la que la ganancia en tensión del filtro cae de 1 a 0.707 (en decibelios (dB) se diría que la ganancia del filtro se reduce en 3dB)

En los filtros pasa banda y elimina banda existirán dos frecuencias de corte diferentes, la inferior y la superior.

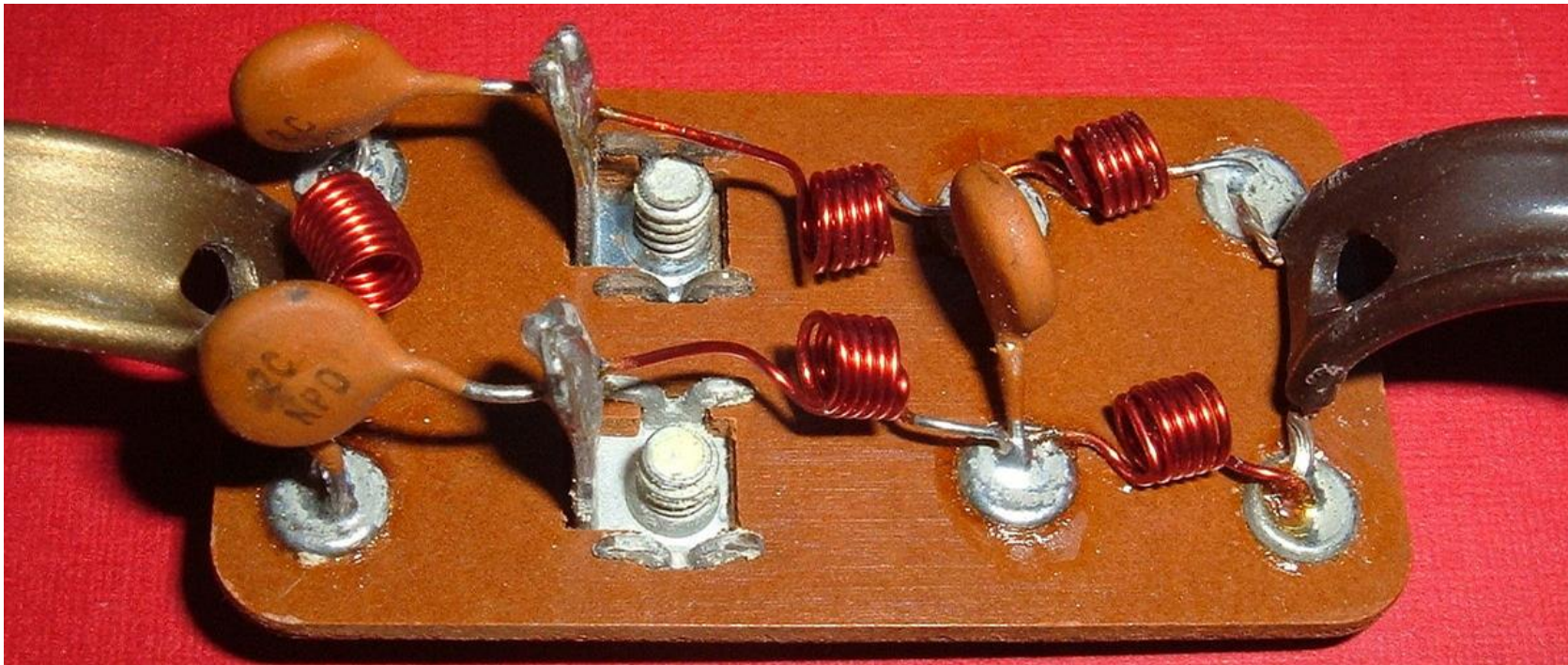
# Filtros - Banda de Paso

Es el margen de frecuencias para las cuales la señal puede pasar

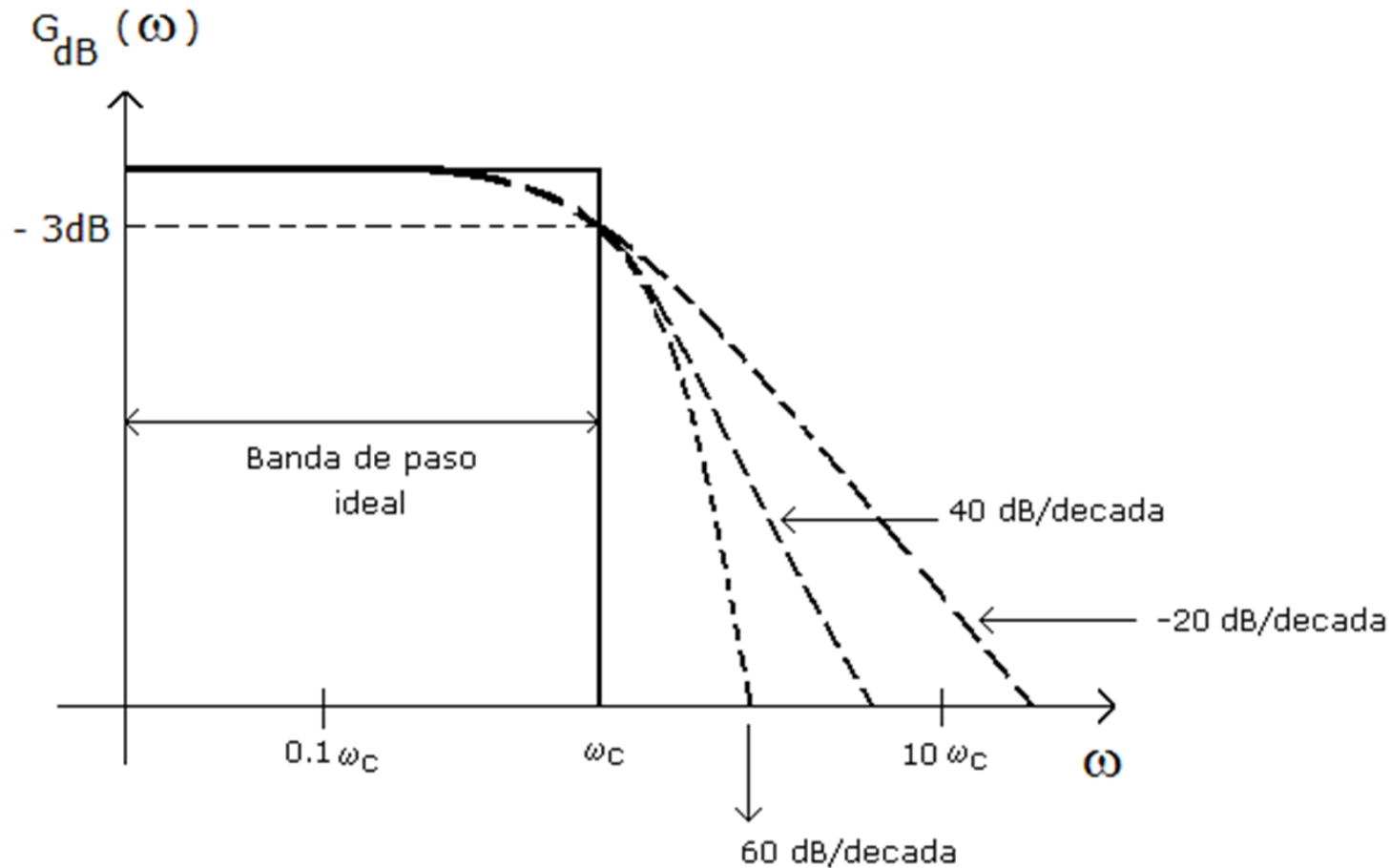


# Filtros - Varios

- **Banda de rechazo:** Rango de frecuencias de un filtro en el cual las frecuencias de una señal no pueden pasar.
- **Década:** Dos frecuencias están separadas una década si una de ellas es de valor diez veces mayor que la otra.
- **Banda atenuada:** Es el rango de frecuencias que el filtro atenúa más de 3dB.



# Filtros - dB



Es una unidad que se utiliza para expresar la relación entre dos valores de presión sonora, o tensión y potencia eléctrica. La unidad básica es el belio (o bel) de símbolo B, pero dada la amplitud de los campos que se miden en la práctica, se utiliza su submúltiplo, el decibelio. El nombre se le ha dado en homenaje a Alexander Graham Bell. Es una expresión que no es lineal, sino logarítmica, adimensional y matemáticamente escalar. Ni el belio, ni el decibelio son unidades del Sistema internacional de unidades.<sup>4</sup>

# Filtros - dB

$$\text{dB} = 10 \log_{10} \frac{P_1}{P_2} \text{ (si lo que se comparan son potencias),}$$

$$\text{dB} = 20 \log_{10} \frac{V_1}{V_2} \text{ (si lo que se comparan son voltajes),}$$

$$\text{dB} = 20 \log_{10} \frac{I_1}{I_2} \text{ (si lo que se comparan son intensidades de corriente).}$$

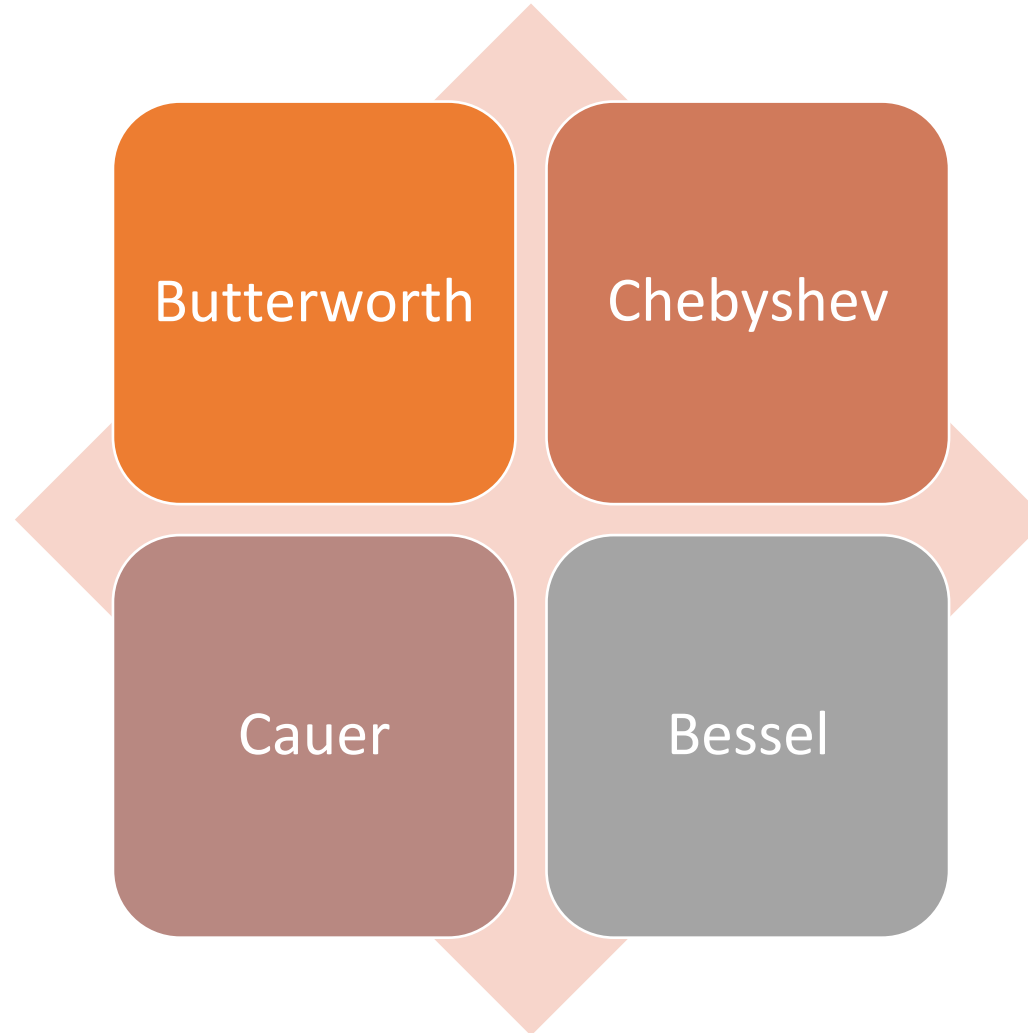
# Filtros - Función de transferencia

Con independencia de la realización concreta del filtro, salvo que debe ser lineal, (analógico, digital o mecánico) su forma de comportarse se describe por su función de transferencia. Esta determina la forma en que la señal aplicada cambia en amplitud y en fase, para cada frecuencia, al atravesar el filtro.

Se puede llegar a expresar matemáticamente la función de transferencia en forma de fracción mediante las transformaciones en frecuencia adecuadas. Se dice que los valores que hacen nulo el numerador son los ceros y los que hacen nulo el denominador son polos.

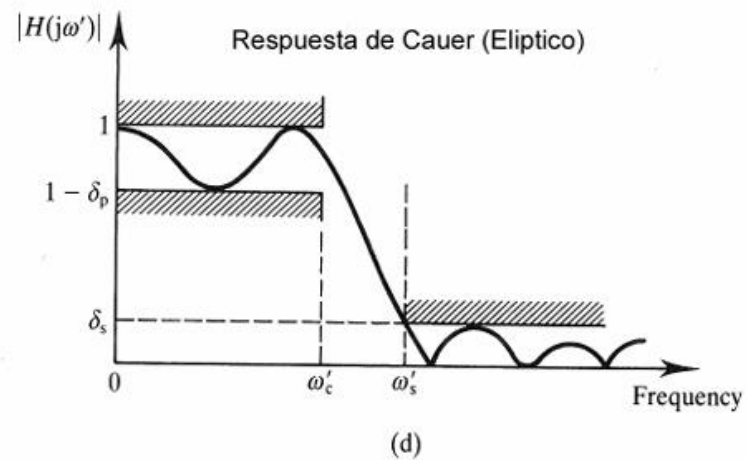
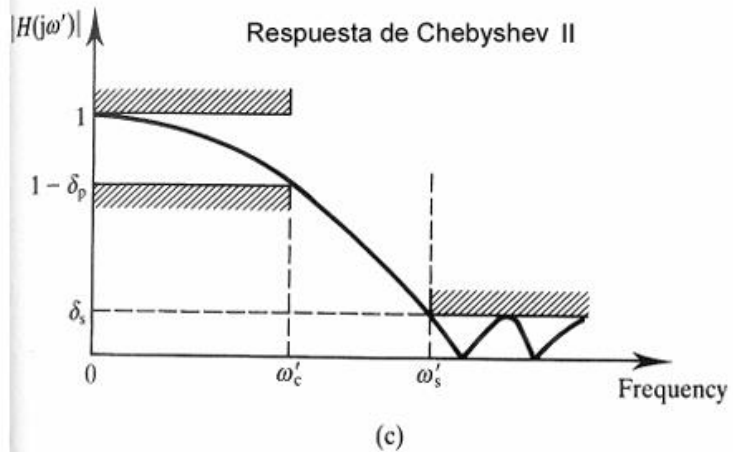
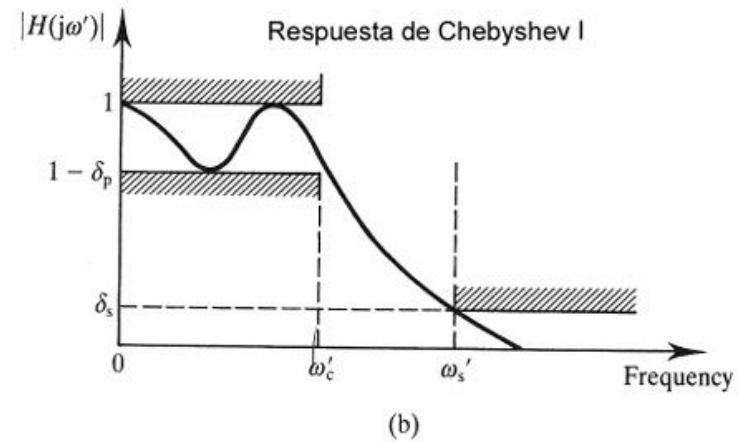
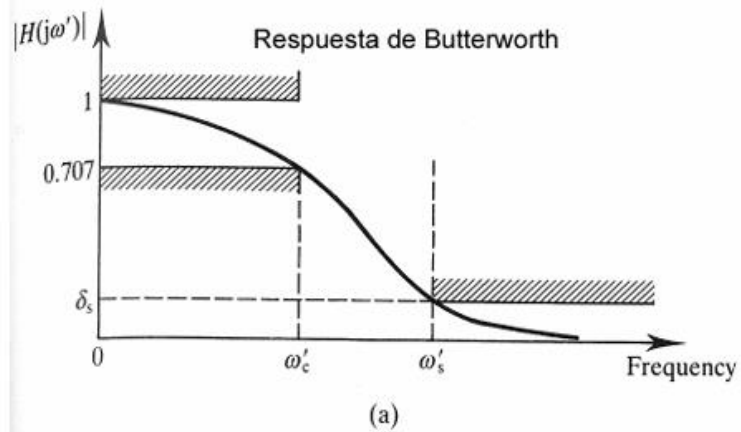
$$H(f) = \frac{\textit{numerador}(f)}{\textit{denominador}(f)}$$

# Filtros - Tipologías

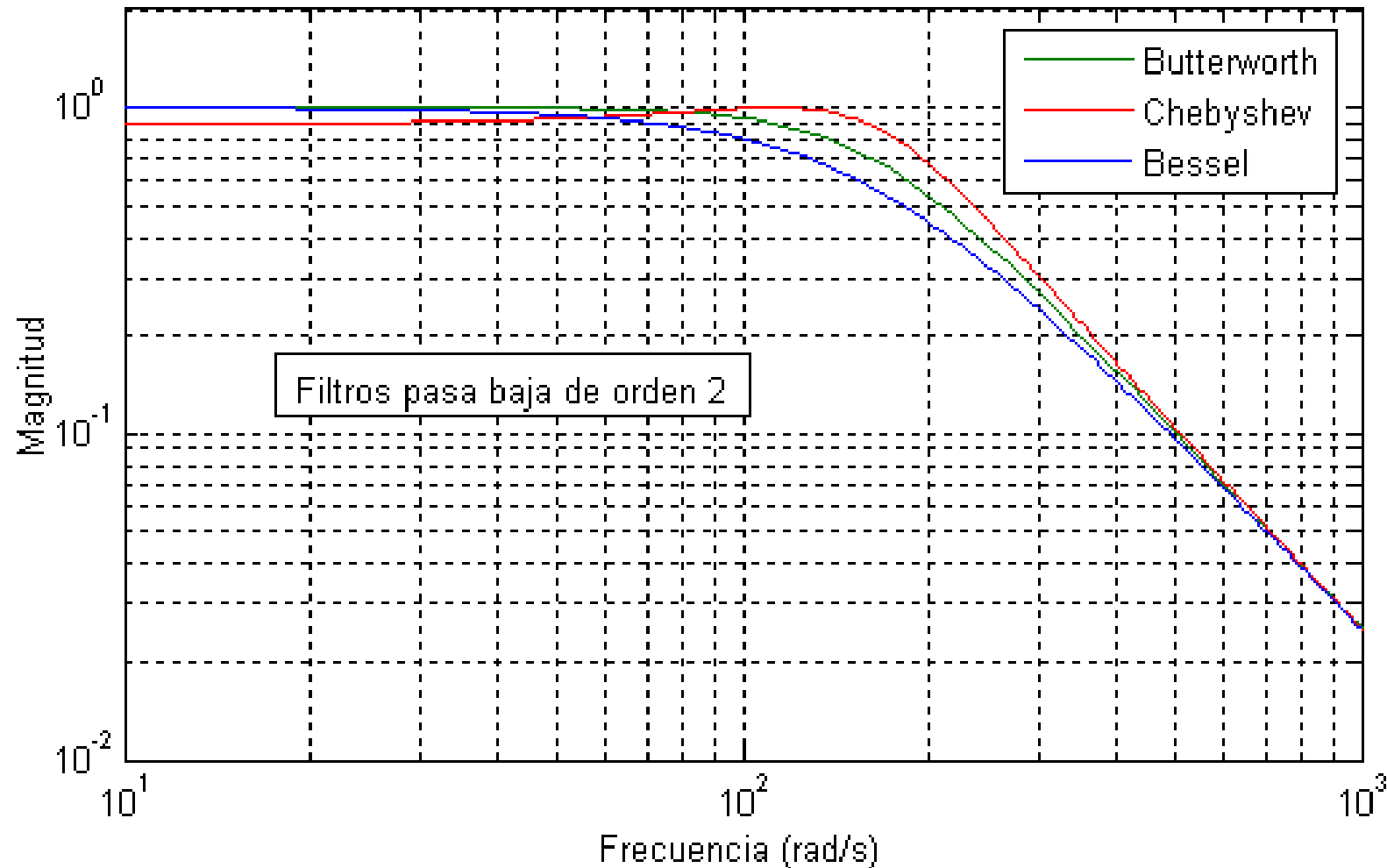




# Filtros - Tipologías

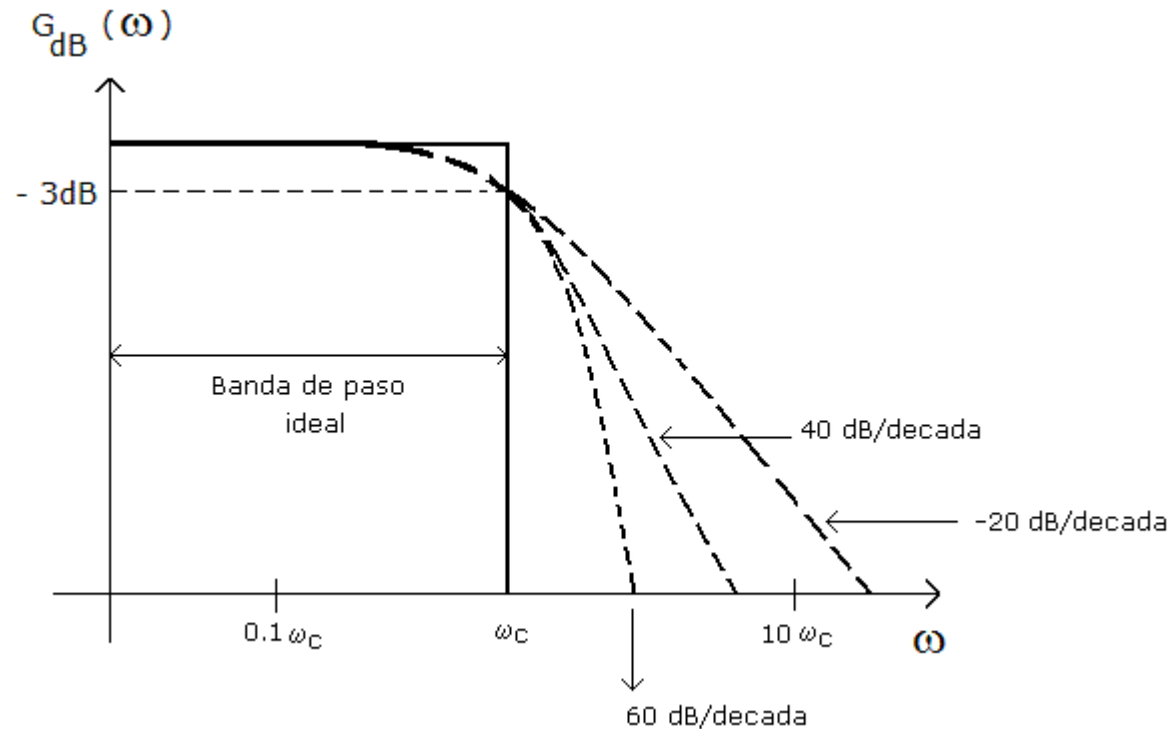


# Filtros – Tipos de filtros



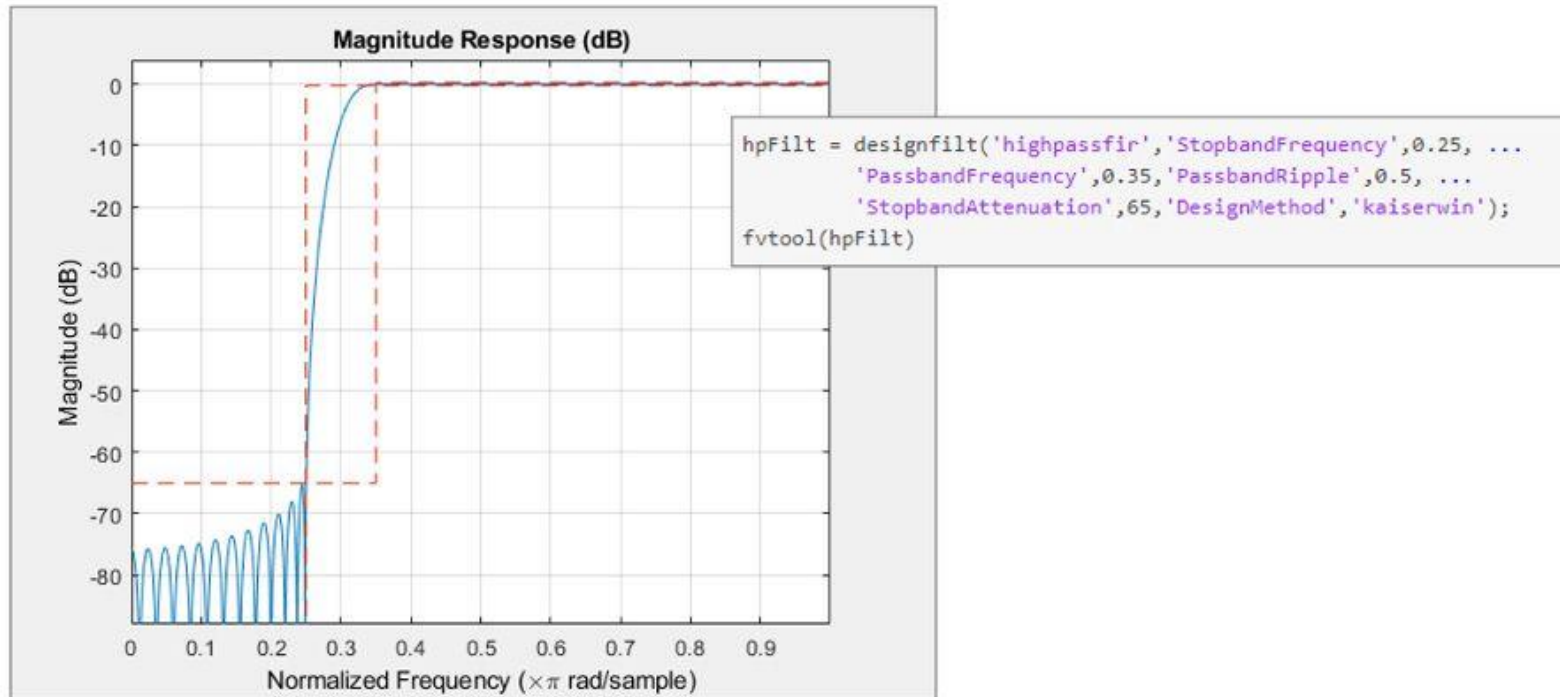
# Filtros – Tipos de filtros

**Filtro pasa bajos:** Son aquellos que introducen muy poca atenuación a las frecuencias que son menores que la frecuencia de corte. Las frecuencias que son mayores que la de corte son atenuadas fuertemente.



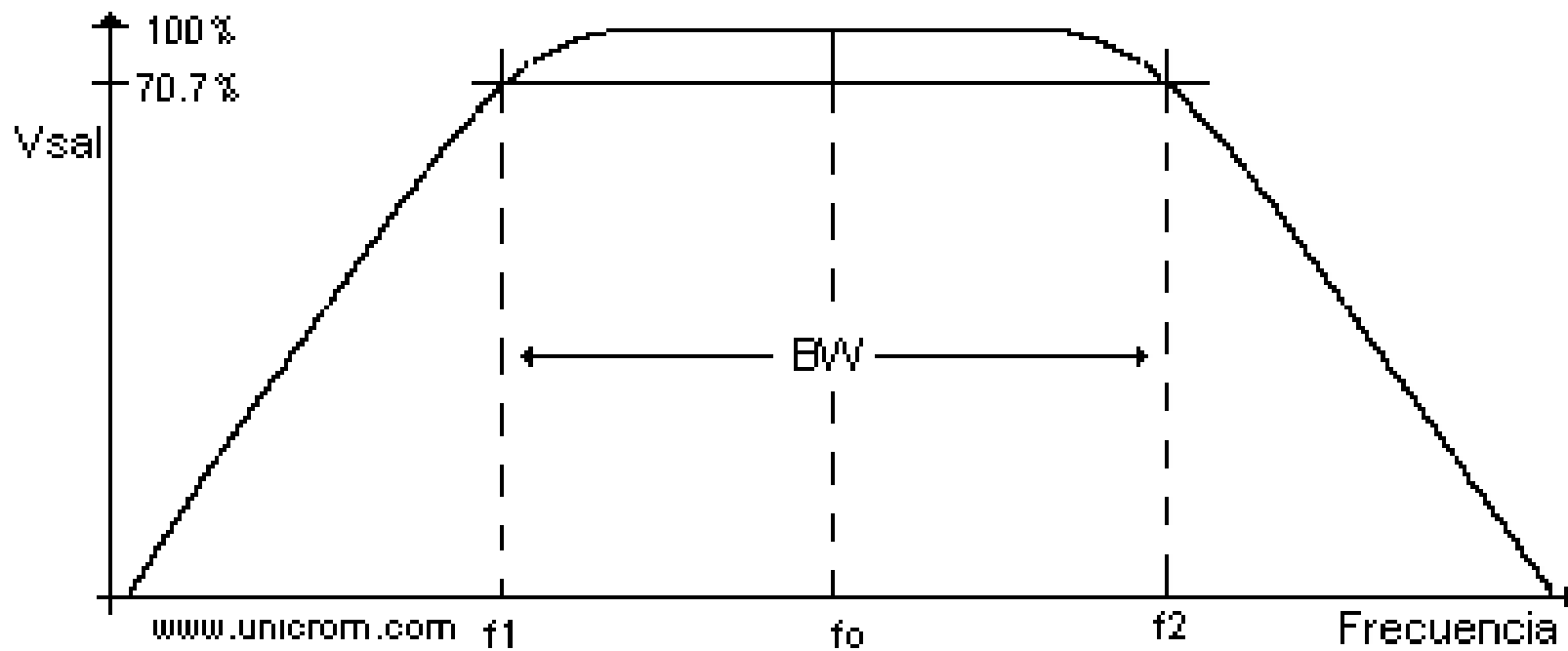
# Filtros – Tipos de filtros

**Filtro pasa altos:** Este tipo de filtro atenúa levemente las frecuencias que son mayores que la frecuencia de corte e introducen mucha atenuación a las que son menores que dicha frecuencia.



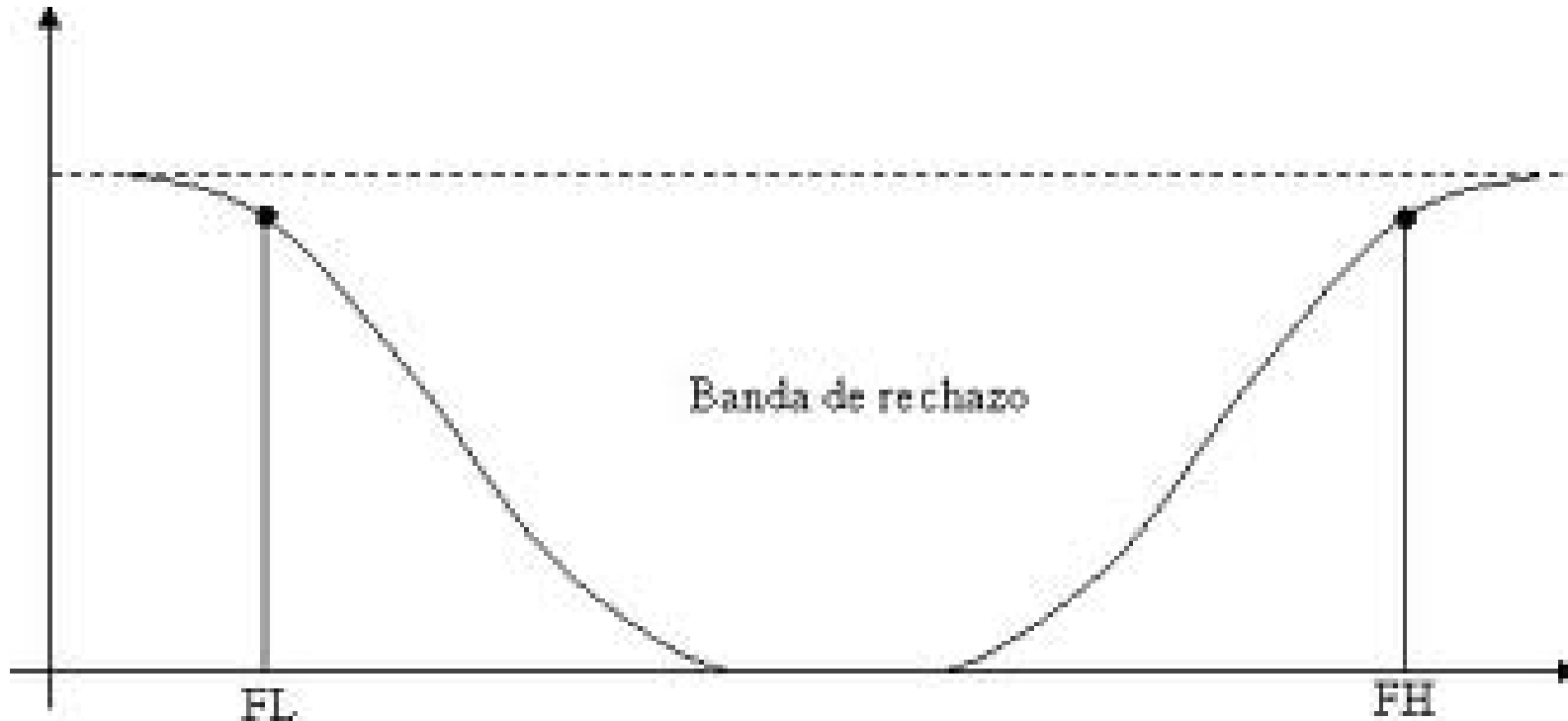
# Filtros – Tipos de filtros

**Filtro pasa banda:** En este filtro existen dos frecuencias de corte, una inferior y otra superior. Este filtro sólo atenúa grandemente las señales cuya frecuencia sea menor que la frecuencia de corte inferior o aquellas de frecuencia superior a la frecuencia de corte superior. por tanto, sólo permiten el paso de un rango o banda de frecuencias sin atenuar.

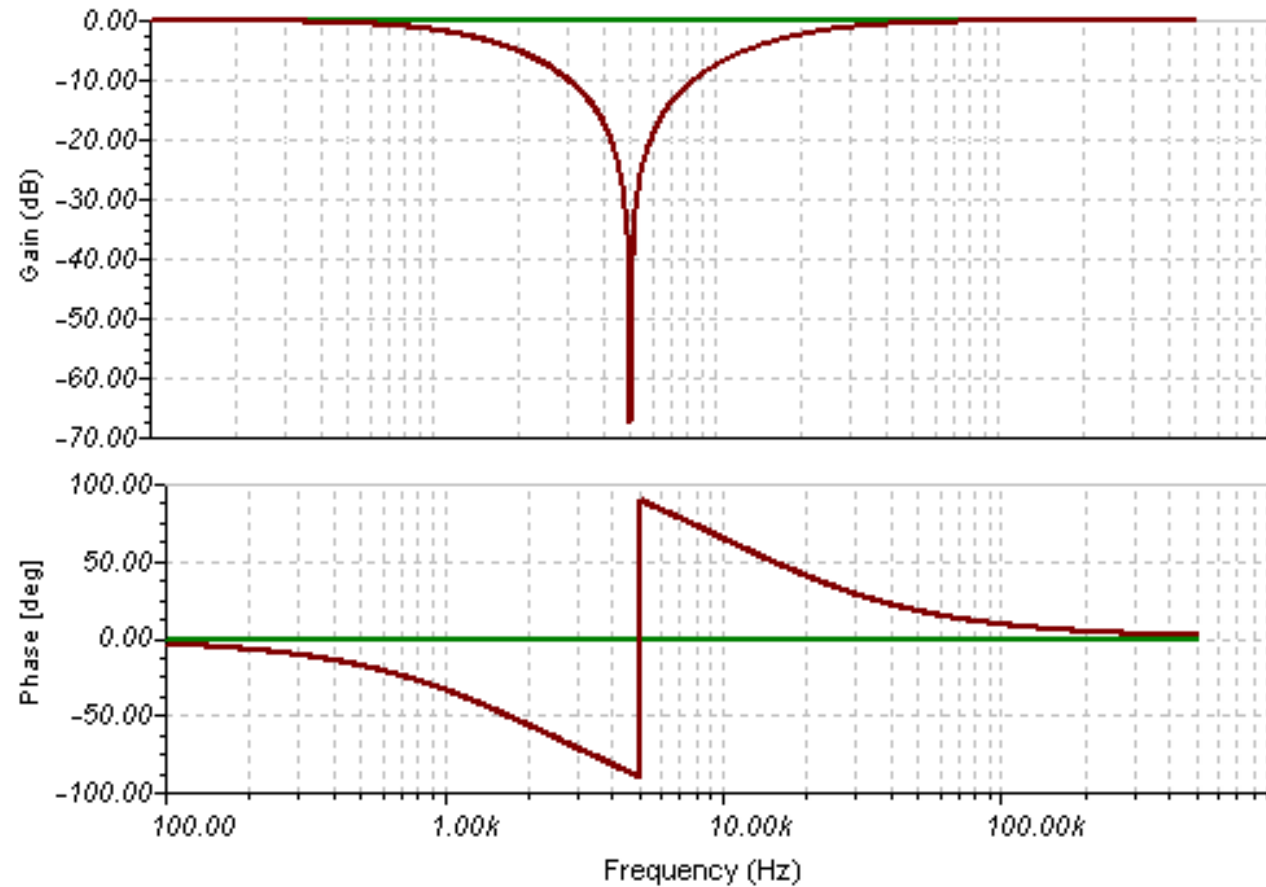


# Filtros – Tipos de filtros

**Filtro elimina banda:** Este filtro elimina en su salida todas las señales que tengan una frecuencia comprendida entre una frecuencia de corte inferior y otra de corte superior. Por tanto, estos filtros eliminan una banda completa de frecuencias de las introducidas en su entrada



# Filtros – Tipos de filtros






# Filtros – Tipos de filtros

## Filtro eléctrico:

- Elemento que discrimina una determinada frecuencia o gama de frecuencias de una señal eléctrica que pasa a través de él, pudiendo modificar tanto su amplitud como su fase

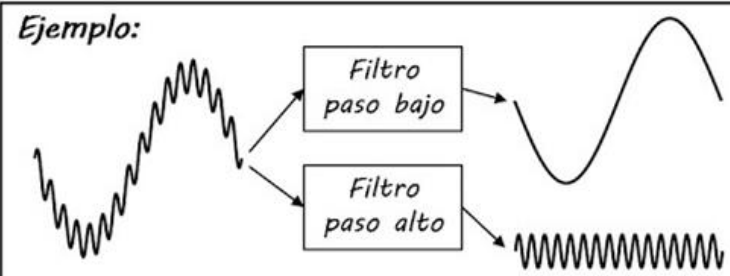
## Señal eléctrica a distintas frecuencias:

- Frecuencia  $f < f_{ci}$  
- Frecuencia  $f_{ci} < f < f_{cs}$  
- Frecuencias  $f_{cs} < f$  

## Clases previas recomendadas:

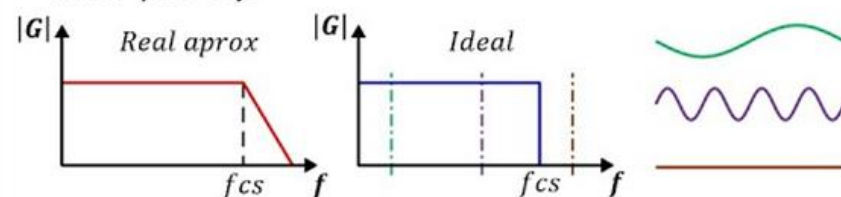
- Diagramas de bode
- Amplificadores operacionales (Realimentación negativa)

## Ejemplo:

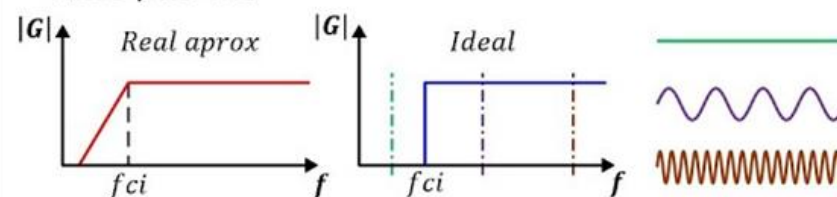


## Filtros en función de la respuesta en frecuencia:

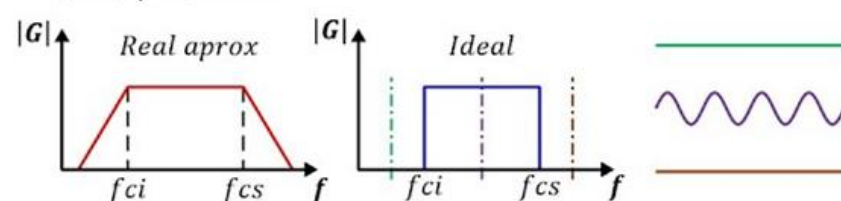
### - Filtro paso bajo



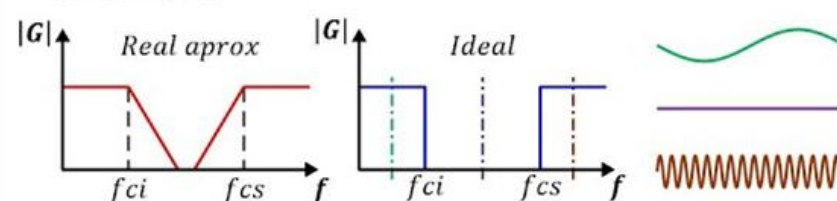
### - Filtro paso alto



### - Filtro paso banda



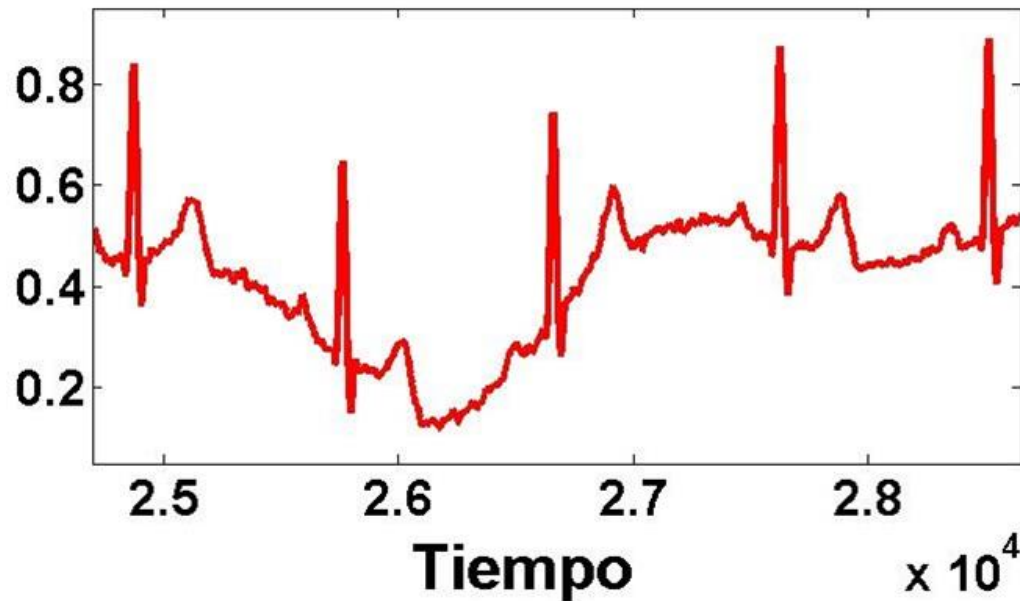
### - Filtro Notch



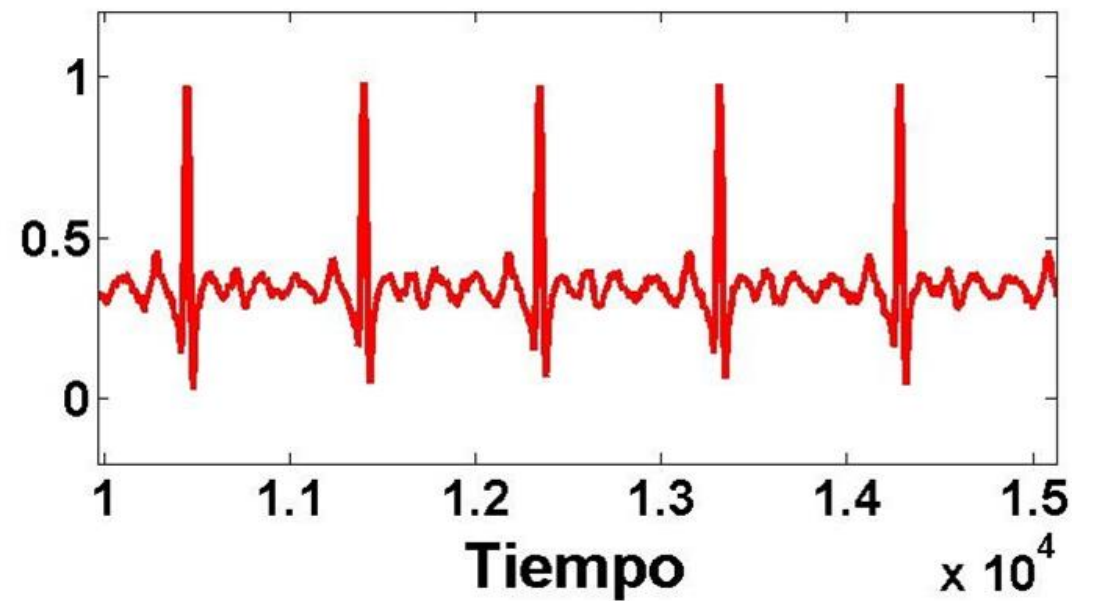


# Filtros – Tipos de filtros

1. ECG Original



2. ECG Filtrado por FFT



<https://pysdr.org/es/content-es/filters.html>

<https://es.mathworks.com/help/signal/filter-design.html>



Esta licencia permite a otros distribuir, remezclar, retocar, y crear a partir de esta obra de manera no comercial y, a pesar que sus nuevas obras deben siempre mencionar a la IU Digital y mantenerse sin fines comerciales, no están obligados a licenciar obras derivadas bajo las mismas condiciones.



# ¡GRACIAS

**IU**Digital  
de Antioquia  
INSTITUCIÓN UNIVERSITARIA DIGITAL DE ANTIOQUIA