METODOLOLOGÍAS DE DISEÑO Y HERRAMIENTAS

TEMA 1- INTRODUCCIÓN

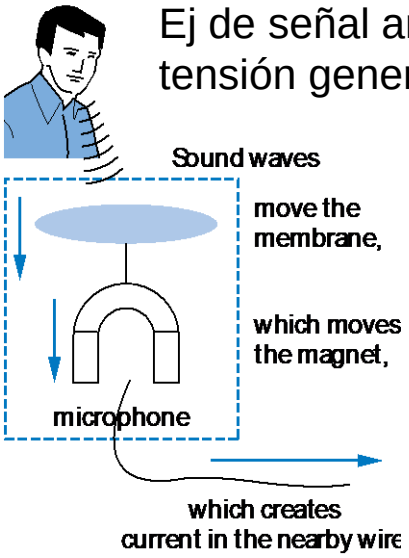
Índice

1. Concepto de sistema digital

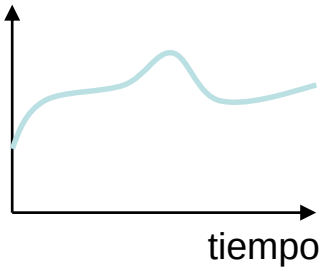
- Sistema electrónico: transforma (procesa) una información (entrada) convirtiéndola en otra (salida) siendo el soporte de esta información una variable eléctrica.

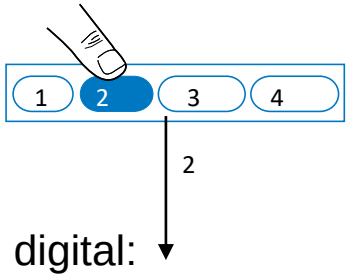
- Procesa señales digitales (toman un conjunto finito de valores) frente a las señales analógicas (pueden tomar infinitos valores)

Ej de señal analógica:
tensión generada por un micrófono



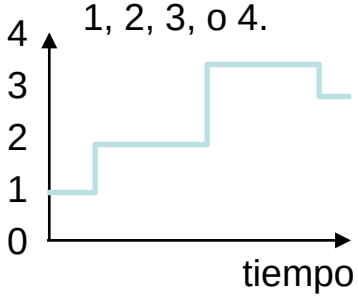
Valores posibles:
1.00, 1.01, 2.09,
... infinitos





Ej de señal digital:
Botón sobre un teclado

Valores posibles:
1, 2, 3, o 4.



Señales digitales binarias

- Sólo consideraremos sistemas que trabajan con señales digitales binarias
- Las señales digitales binarias toman sólo dos posibles valores, representados típicamente como 0 y 1.

  - Un dígito binario se llama bit.
- La mayoría de los sistemas digitales trabajan con señales binarias
  - almacenar y transmitir uno de dos valores es más fácil que tres o más
  - el transistor (el componente básico de los circuitos digitales) tiene dos modos fundamentales de operación: conducción (ON) y no conducción (OFF).

- **¿Cómo codificar en binario las señales?**

  - Muchas señales son inherentemente binarias (botones que se presionan o no).

  - Otras señales son inherentemente digitales (formato de texto o numérico) y sólo necesitan codificación en binario (por ej. la codificación ASCII).

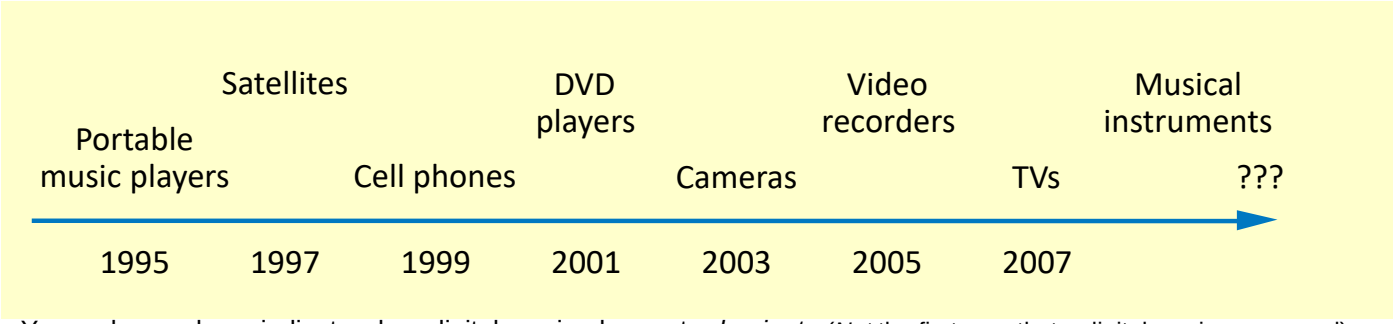
  - Otras señales son analógicas y requieren conversión A-D (analógico-digital).

Sistemas digitales hoy

- En todas partes a nuestro alrededor
  - No son sólo los PCs y portátiles
  - Smart phones, tablets
  - Multimedia: cámaras, videos, audio…..
  - Equipos médicos, electrodomésticos, transportes
  - Para ser llevados/implantados: gafas Google, smart watches, implantes médicos….IoT

Sistemas digitales hoy

- La sociedad de la información y las comunicaciones cada vez emplea más sistemas digitales. Los existentes se mejoran y aparecen otros nuevos.
- Están en todas partes porque ha sido posible hacer cosas muy complejas con tamaños, consumo de potencia y costes muchos menores que en el pasado
  - Herramientas de CAD de diseño, fabricación y test cada vez más sofisticadas.
  - Mejores tecnologías de fabricación (se reducen las dimensiones mínimas, se aumenta el tamaño del dado, se incrementa el número de capas de interconexión, …).
- Muchas aplicaciones que antes se hacían en analógico ahora se hacen en digital

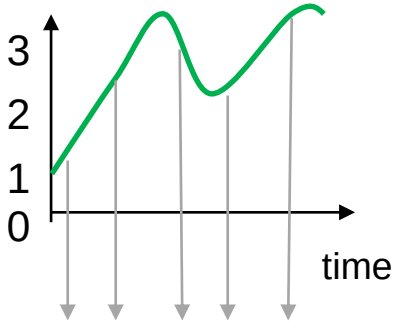


Years shown above indicate when digital version began to *dominate* (*Not* the first year that a digital version appeared)

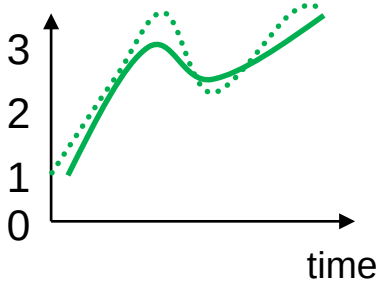
Ventajas de los sistemas digitales

- La digitalización hace posible que no se pierda **calidad** en el "procesado"

Señal original



Señal recibida



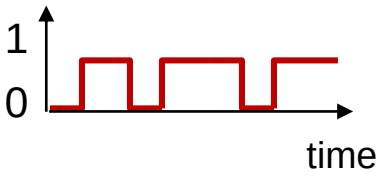
Las señales analógicas (p. ej. de audio o vídeo) pueden perder calidad porque los niveles no pueden almacenarse / copiarse / transmitirse/procesarse perfectamente

¿Se ajustan a mayor o menor valor?

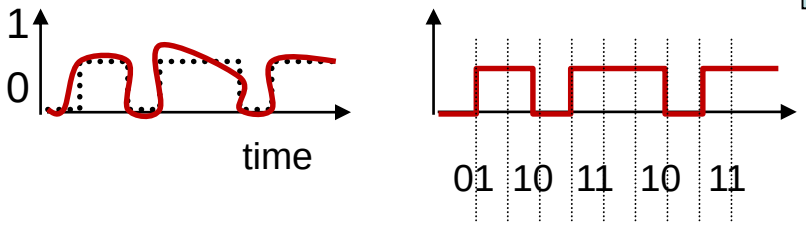
01 10 11 10 11

A/D

Señal digital

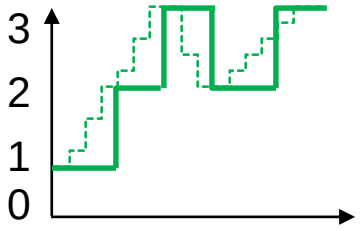


Se restablecen 0s y 1s



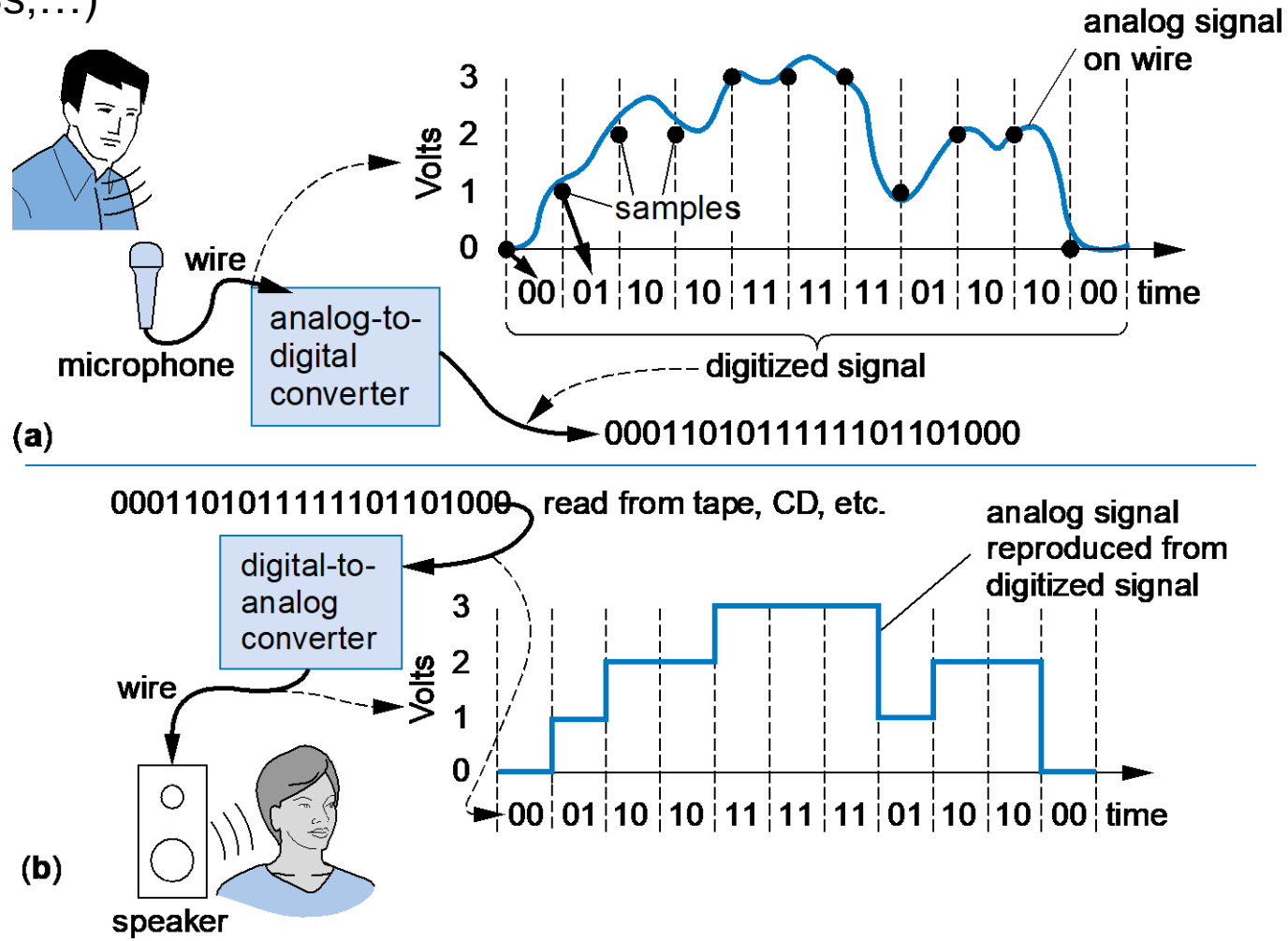
D/A

Señal reconstruida
(mejor cuanto mayor razón de muestreo y más bits por muestra)

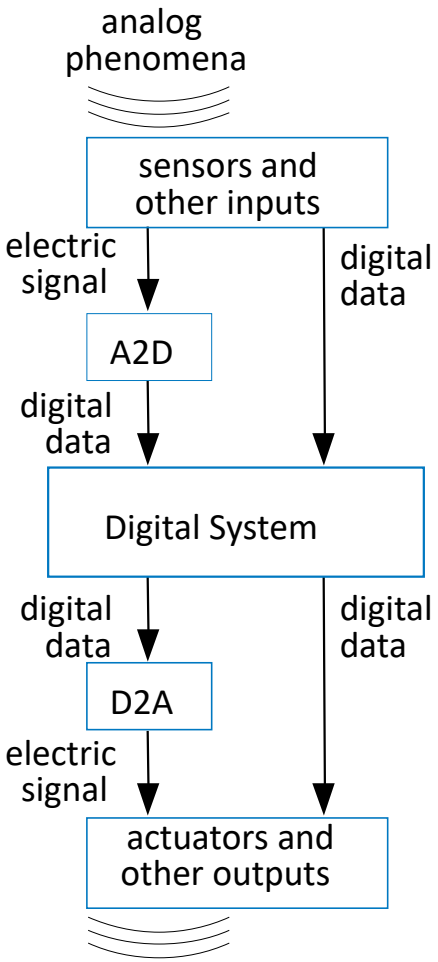


Ventajas de los sistemas digitales

- Ejemplo: Permite el **almacenamiento sin deterioro** (en CDs, DVDs, USBs,…)



Sistemas digitales



- En todas partes
- La codificación papel central
- El mundo es analógico, es necesario conversión AD y DA

Fenómenos físicos digitales

- Algunos sensores producen como salida una señal eléctrica con dos posibles valores: pulsador, sensor de luz,…
- Otros fenómenos digitales pueden tomar varios valores y se requiere un circuito de codificación

Fenómenos físicos analógicos

- **Sensor**: mide el fenómeno físico y lo convierte en una señal eléctrica analógica. E.j. Micrófono (sonido),
- **Analog-to-digital converter**: convierte la señal eléctrica a binario
- **Digital-to-analog converter:** convierte los bits en una señal eléctrica analógica
- **Actuador:** convierte la señal eléctrica en fenómeno físico. Ej. altavoz

Índice TEMA 1

1. Concepto de sistema digital

2. Sistema digitales. Opciones de implementación
   - Software versus hardware
   - Requerimientos de la aplicación

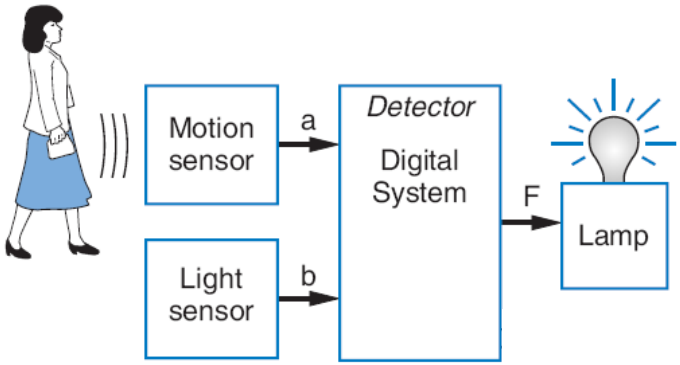
3. Realización del hardware

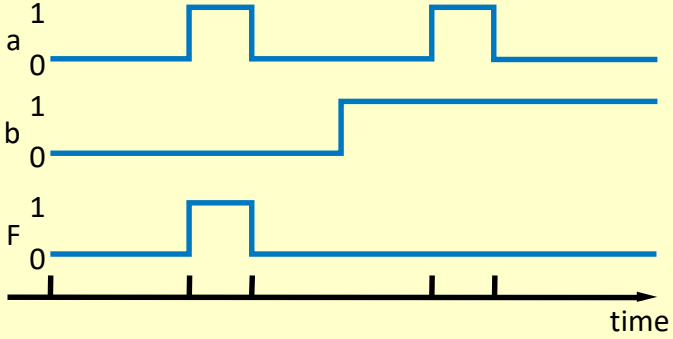
4. Diseño hardware

2. Sistemas digitales: opciones de implementación

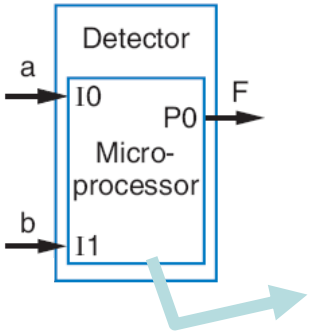
Software versus hardware

Ej. muy sencillo: diseñar un detector de movimiento que se active sólo por la noche



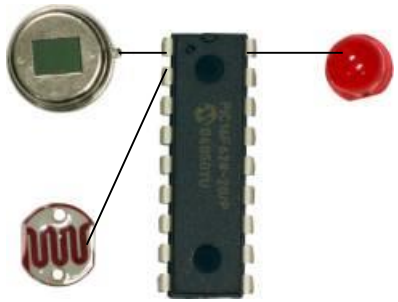


Solución software: fácil y barata pero bajas prestaciones

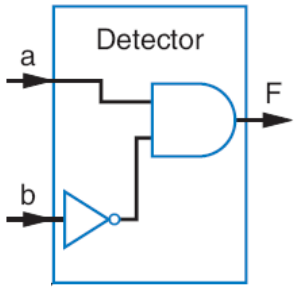


I0	P0
I1	P1
I2	P2
I3	P3
I4	P4
I5	P5
I6	P6
I7	P7

```
void main()
{
   while (1) {
      P0 = I0 && !I1;
      // F = a and !b,
   }
}
```



Solución hardware: altas prestaciones



2. Sistemas digitales: opciones de implementación

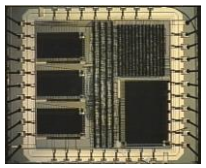
Software para ejecutar en procesadores de propósito general: C, C++, Java, …

Aumento de flexibilidad y rapidez del diseño

Software empotrado para procesadores específicos: microcontroladores, DSP, GPUs, ...

Hardware sintetizado sobre dispositivos programables: FPGAs (Field Programmable Gate Arrays)

Reducción de área, potencia y tiempo de respuesta



Hardware sobre circuitos integrados: ICs (Integrated Circuits) analógicos, digitales o de señal mixta

- La mayoría de los sistemas combinan software y hardware: **codiseño hardware-software**.
- **En esta asignatura nos centraremos en el diseño hardware**.

Hardware versus software

– La solución software puede no ser suficientemente buena: muy lenta, muy costosa, o consumir mucha potencia

Calculo para controlar el ala de un avion:

• 50 ms en microprocesador

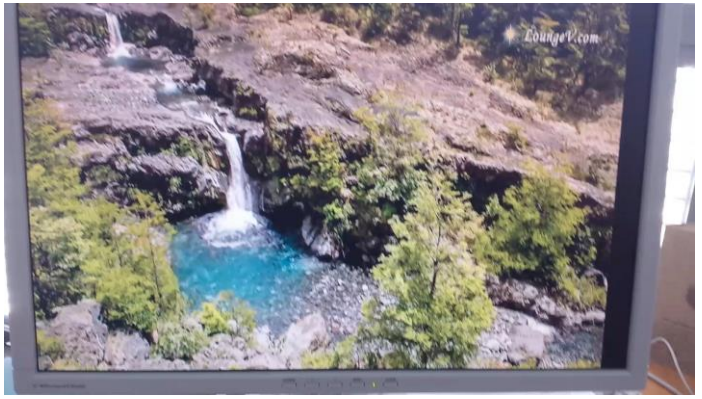
• 5 ms como circuito a medida

Debe ejecutarse 100 veces por segundo:

• 100 * 50 ms = 5000 ms = 5 segundos

• 100 * 5 ms = 500 ms = 0.5 segundos

Microprocessor demasiado lento, circuito OK.



Índice

1. Concepto de sistema digital

2. Sistema digitales. Opciones de implementación

3. Realizacion del hardware
   - ASICs
   - Hardware sobre FPGAs

4. Diseño hardware

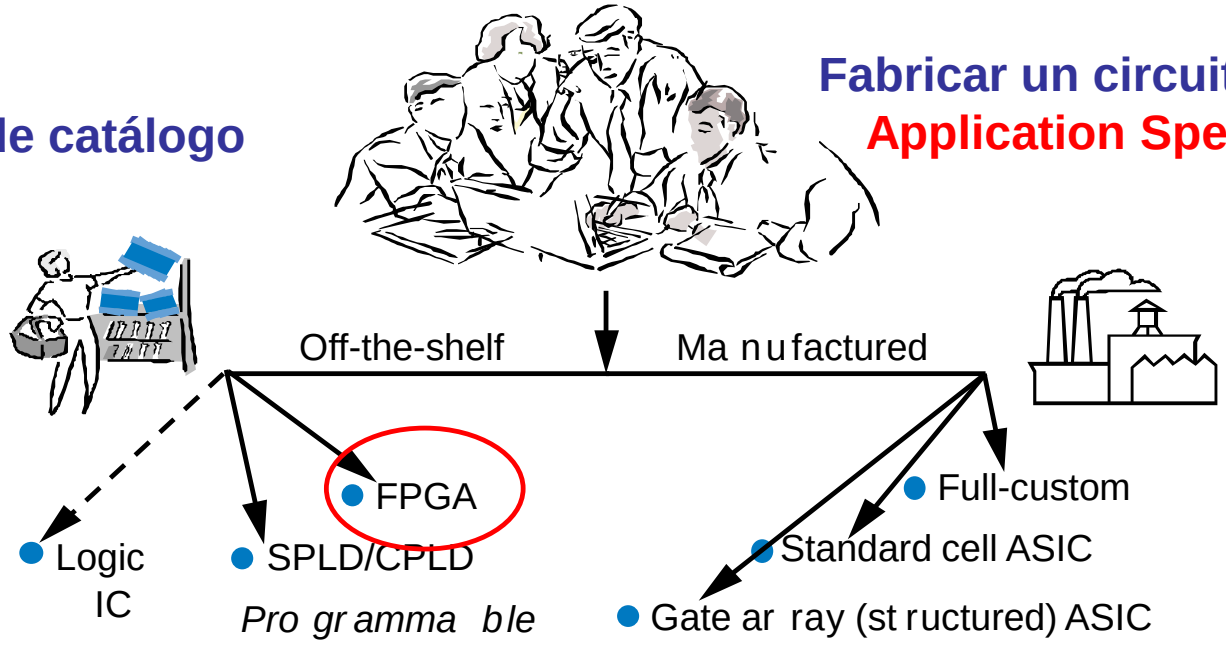
3. Realización del hardware

- Distintas opciones para realizar físicamente el sistema

  Diferentes prestaciones del sistema resultante, coste o tiempo de diseño y fabricación

Usar ICs de catálogo

Fabricar un circuito integrado (IC)
Application Specific IC (ASIC)



ASICs

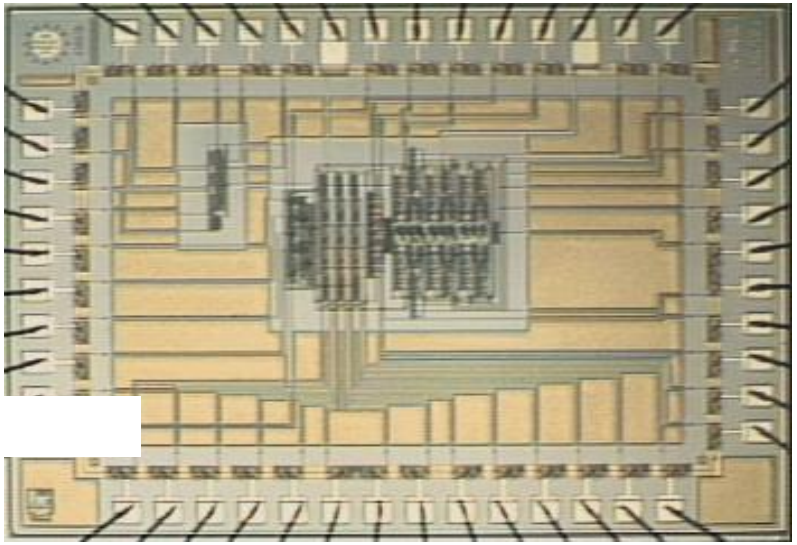
El mayor coste se invierte en lanzar al mercado por primera vez un producto

Fase de diseño, fase de fabricación y fase de test (prototipos y posterior producción en serie).

- Fase de diseño: Termina con una **layout** (patrones geométricos para cada una de las capas físicas que componen un IC)

ASICs

- Fase de fabricación: Generación de máscaras, preparación de obleas de silicio, pasos del proceso tecnológico (dispositivos, contactos, metalizaciones).



- Fase de test:

- Existen equipos especializados (máquinas y software) para testar automáticamente ICs
- Test de prototipos: validación del diseño
- Test de producción: Puede haber unidades defectuosas (defectos de fabricación)

- Los costes (diseño, fabricación y test) son asumibles para fabricantes de equipos originales (OEMs) que emplean miles de circuitos integrados por año (los costes de diseño y de generación de máscaras para la fabricación se reparten entre todas las unidades producidas).
- Permite obtener las mejores prestaciones

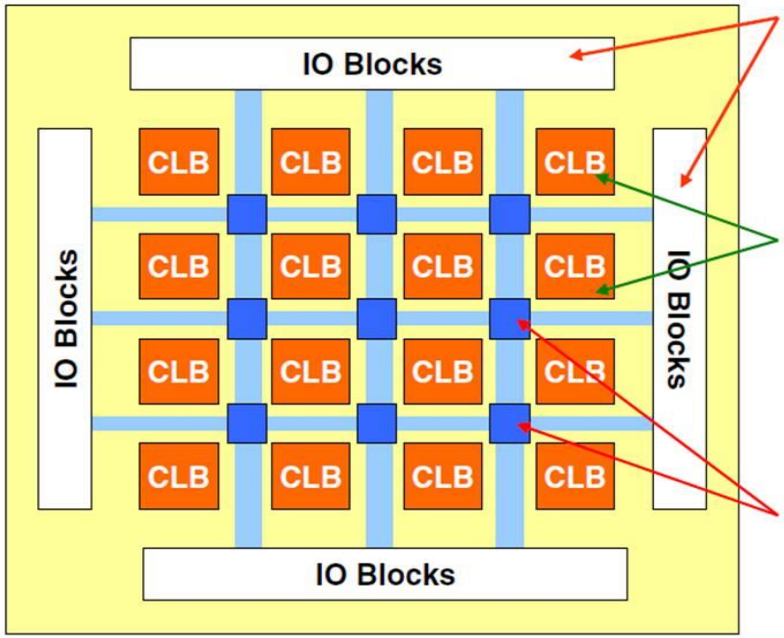
Hardware sobre FPGAs

Una FPGA (Field Programmable Gate Array) es un chip de silicio (hardware) en el que se puede implementar cualquier diseño digital configurando adecuadamente sus recursos

- La fase de diseño termina con un **bitstream** (secuencia de 0s y 1s para configurarlo).
- No hay fabricación sino que se compra y se configura "en campo"
  - Segundos/minutos frente a semanas de fabricación de un ASIC
  - En general se pueden configurar muchas veces
  - Muchas veces decimos programar en vez de configurar pero no hay que confundirlo con que un FPGA ejecute un programa software
- Acorta el tiempo y el coste de desarrollo/prototipado. No hay que esperar semanas, ni fabricar un circuito, para validar que el diseño es adecuado a nuestra aplicación
- De hecho, los FPGAs también se usan para prototipar diseños que luego se van a implementar como ASICs
- Cada vez son más competitivos y están expandiendo significativamente sus áreas de aplicación

Hardware sobre FPGAs. Descripción

Una FPGA (Field Programmable Gate Array) es un chip de silicio (hardware) en el que se puede implementar cualquier diseño digital programando adecuadamente sus recursos. Los recursos estándares son:



(IOBs) Bloques de E/S

- Interfaz con los terminales del dispositivo

(CLBs) Bloques lógicos configurables

- Elementos funcionales para implementar la lógica del usuario

Red de interconexión

- Caminos para interconectar las entradas y salidas de CLBs e IOBs

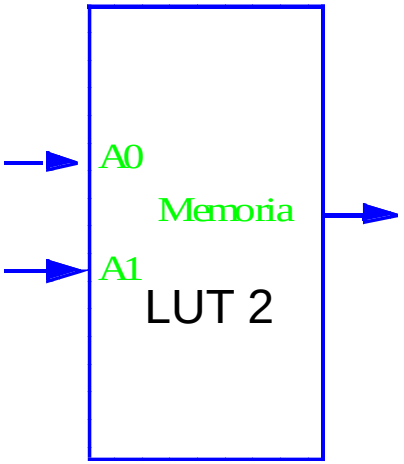
- Además las FPGAs disponen de otros recursos (según el fabricante y la familia): RAM (en bloques y distribuida), multiplicadores, o incluso microprocesadores (ej. ARM en los zynq de Xilinx)

- Distintos fabricantes de FPGAs. Los principales Xilinx(ahora parte de AMD) y Altera (ahora Intel). En las prácticas de la asignatura trabajaremos con un FPGA de la familia Artix7 de Xilinx

- CLB es el nombre que Xilinx da a los bloques lógicos configurables. Otros fabricantes usan otros.

Hardware sobre FPGAs. Xilinx. Configurabilidad

- La configurabilidad de los CLBs de Xilinx se basa en el uso de LUTs (Look Up Tables)
  - un LUT es una memoria para almacenar una tabla de verdad
  - cambiando los valores almacenados se cambia la función implementada
  - una memoria de *n entradas puede implementar cualquier función de hasta n variables*
  - el retraso es independiente de la función implementada



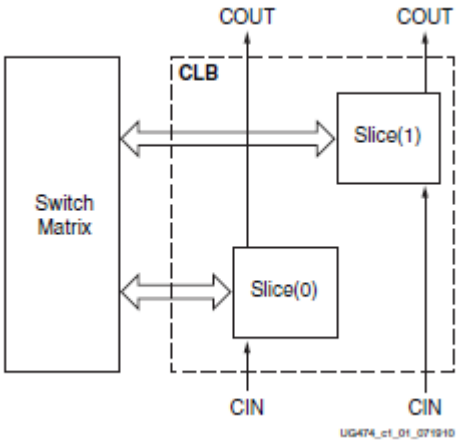
programación para implementar una EXOR

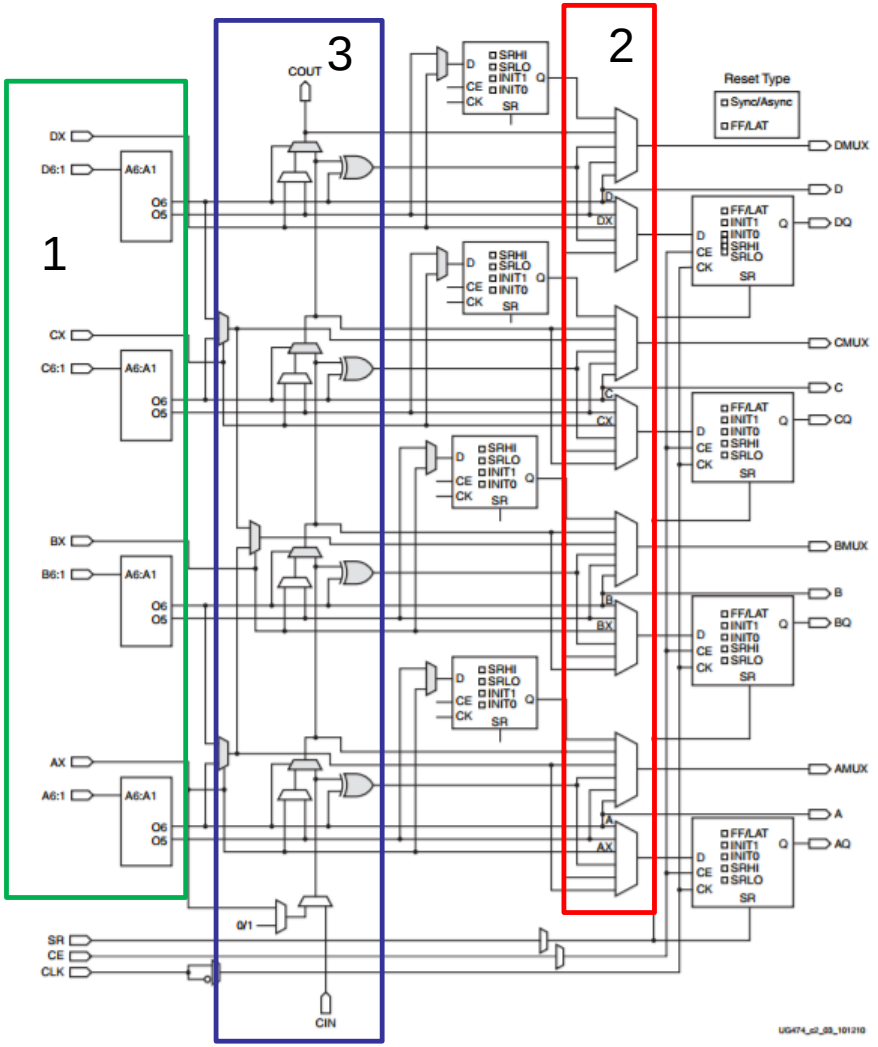
dirección	
0	0
1	1
2	1
3	0

Hardware sobre FPGAs. Xilinx. Arquitectura CLBs

- Los CLBs contienen slices. Cada slice incluye:
  - LUTs: Implementa lógica combinacional
  - Multiplexores: permiten combinar varios LUTs para realizar funciones más complejas
  - Elementos de memoria: indispensables en la realización de sistemas digitales
  - Lógica dedicada para funciones aritméticas: unas pocas puertas para optimizar la implementación de operaciones aritméticas
  - Interconexiones configurables: permite conectar de distintas maneras los elementos anteriores entre si y con el exterior
- La arquitectura de los CLBs ha ido evolucionando. No es igual un CLB de un Spartan-3 (obsoleto) que el de un dispositivo actual (ejemplos: familias Artix7, Kintex7, …)
  - Distinto número de slices
  - LUTs de más entradas
  - Más LUTs/multiplexores/elementos de memoria por slice

Hardware sobre FPGAs. Xilinx. Arquitectura CLBs. Serie 7





- CLB contiene 2 slices
- cada slice:
  - 4 LUTs: cada una puede usarse como una LUT de 6 entradas o como 2 LUTs de 5 entradas (1)
  - 8 Flip-Flops: se puede configurar el tipo de disparo, el tipo de reset….
  - Multiplexores (2)
  - Lógica dedicada (3)

- Los LUTs de algunos slices pueden usarse también como memoria (distributed memory) o para implementar registros de desplazamiento

Índice

1. Concepto de sistema digital
2. Sistema digitales. Opciones de implementación
3. Realizacion del hardware
4. Diseño hardware

- Flujo de diseño
  - Nivel lógico y nivel RT
- Flujo RT -> Hardware
- Otras consideraciones

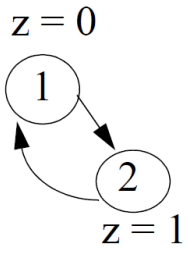
Diseño lógico

- Se codifican los estados.
- Se obtienen (o manipulan) las expresiones Booleanas.
- Se mapean las expresiones a puertas lógicas.

Para circuitos secuenciales:

Nivel lógico de comportamiento:

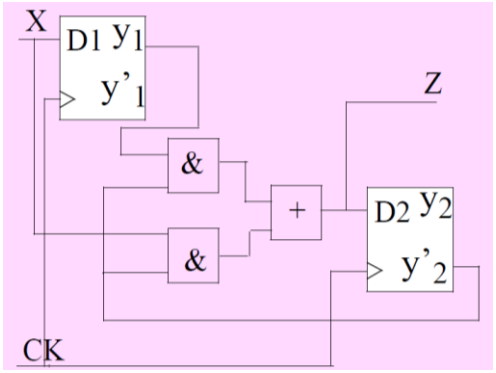
- Diagramas de estado
- Tablas de transición de estados



Estado presente	Próximo estado, salida
1	2, 0
2	1, 1

Nivel lógico estructural:

- Red de puertas y Flip-Flops



Para circuitos combinacionales:

Nivel lógico de comportamiento:

- Funciones Booleanas (tablas de verdad, mapas de Karnaugh)

Nivel lógico estructural:

- Red de puertas

Diseño RT

- Se asignan operaciones a ciclos de reloj
- Se asignan operaciones a los recursos disponibles.

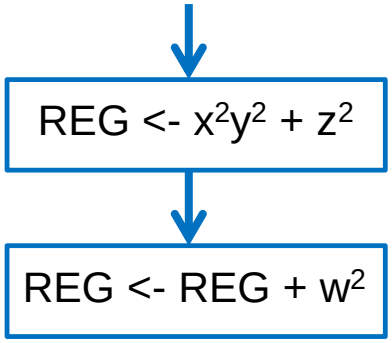
Ejemplo sencillo:

Realizar la operación $x^2y^2 + z^2 + w^2$

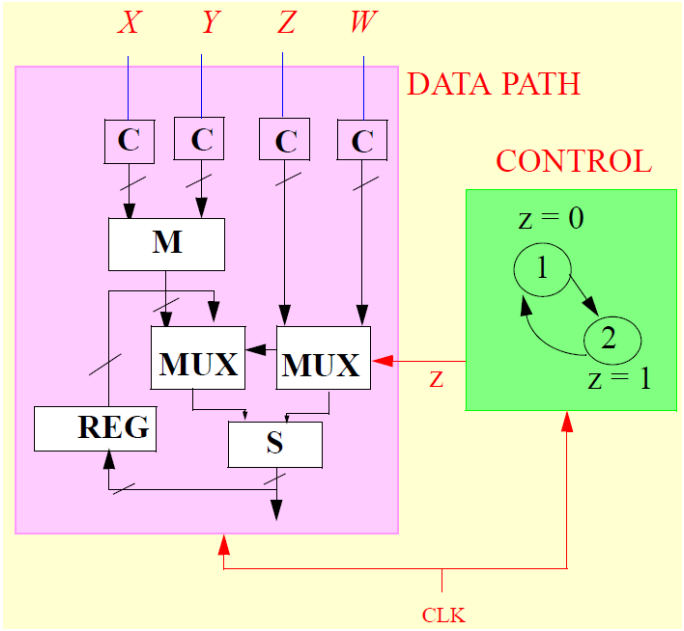
Se dispone de elevadores al cuadrado (C), multiplicadores (M) y sumadores (S)

RT de comportamiento:

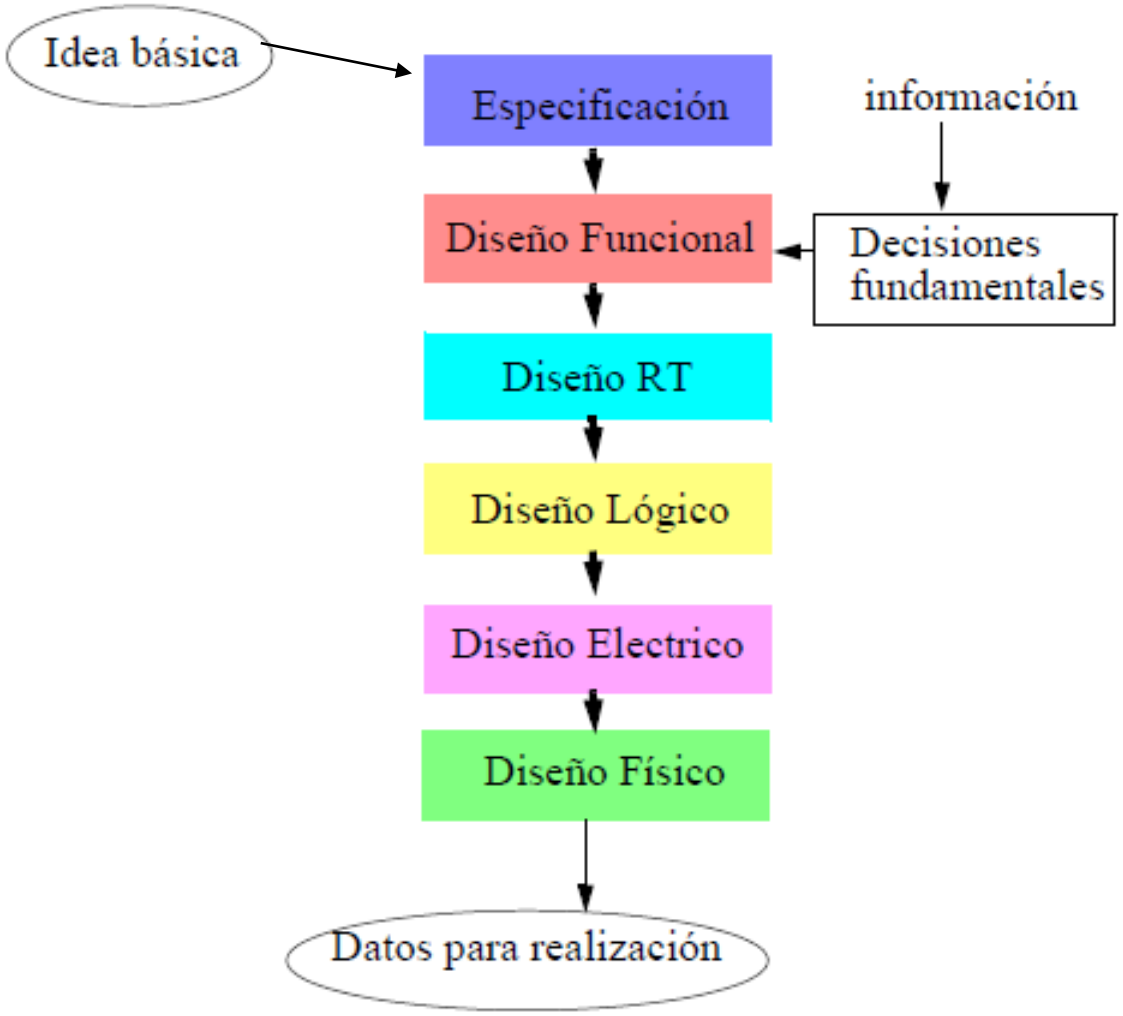
Se decide realizar la operación en 2 ciclos de reloj



RT estructural:



Flujo de diseño



Flujo de diseño

- diseño funcional

  determinar un algoritmo o paradigma computacional que resuelve el problema

- diseño a nivel RT

  asignar operaciones a ciclos de reloj y asignar operadores a componentes hardware

- diseño lógico

  codificar los estados, obtener (o manipular) expresiones booleanas y mapear expresiones a puertas lógicas

  muy automatizado en el diseño de sistemas digitales

- diseño eléctrico

  seleccionar topologías de circuitos y dimensionar transistores

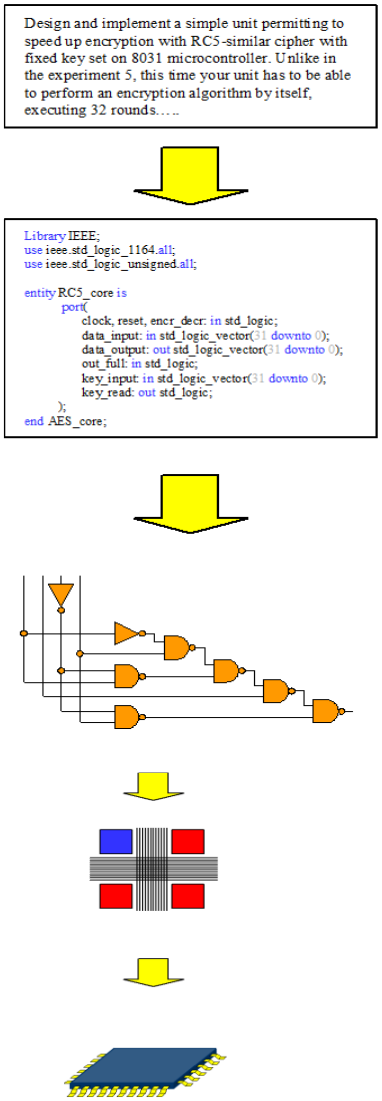
  No existe cuando se diseña sobre FPGAs. Normalmente no es necesario en el diseño de ASICs digitales (ya hay una librería de celdas diseñada)

- diseño físico
  - Generar lo necesario para realizar físicamente el sistema
  - muy dependiente del estilo de implementación
  - muy automatizado en el diseño de sistemas digitales

Metodologías de diseño

- Las tareas de diseño de sistemas digitales se automatizan mediante herramientas de CAD para reducir el tiempo de diseño y los errores, así como explorar rápidamente diferentes compromisos de velocidad, área, consumo, etc.
- El flujo RT -> hardware está muy bien establecido (maduro) y se utiliza extensivamente. (Este flujo se utilizará en las prácticas de laboratorio)
  - El diseñador es responsable de derivar una descripción a nivel RT
  - Utiliza un lenguaje de descripción de hardware para codificarla y poderla introducir en una herramienta de CAD
  - Las herramientas de CAD generan automáticamente (pero bajo el control del diseñador) lo necesario para realizar físicamente el diseño (layout o bitstream)

Flujo RT a hardware



Especificación

En las primeras sesiones de laboratorio
lo veremos con más detalle y cómo hacerlo con **Vivado**

Descripción HDL

Síntesis RTL
automatizado

Síntesis Física
automatizado

Realización Física

Simulación del HDL
Validación funcional

Simulación del circuito sintetizado
Validación funcional
Validación temporal

Validación temporal

Flujo RT a hardware

- Escribir HDL
  - Descripción del comportamiento (funcionalidad) del sistema
- Simulación del HDL
  - Determina si el código HDL describe el comportamiento deseado
- Sintesis del HDL (ó sintesis RTL)
  - obtiene una red de componentes lógicos (puertas, flip-flops y otros más complejos) que implementa la funcionalidad descrita en el código HDL
- Simulación del circuito sintetizado
  - Determina si el circuito sintetizado implementa la funcionalidad deseada y se analiza su comportamiento temporal
- Síntesis física
  - Genera el layout del circuito para que pueda ser fabricado o el bitstream para programar el FPGA
- Validación post diseño físico
  - Validación temporal más precisa

Otras consideraciones

¿ Cómo se deriva la descripción HDL a nivel RT ?

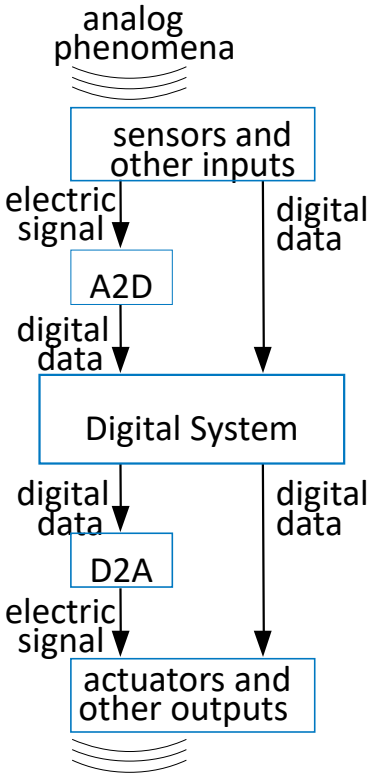
• Hay etapas de diseño anteriores: diseño funcional (o del algoritmo) y diseño arquitectural (de la descripción RT)

- Estas etapas tienen un gran impacto en las prestaciones y el coste del sistema
- En muchos casos se deriva manualmente, aunque se está avanzado mucho en metodologías y herramientas para abordar estas fases de diseño.
- Por ejemplo, existen herramientas para automatizar el diseño arquitectural (**herramientas de síntesis de alto nivel**). El diseño se describe en C/C++ o Matlab a nivel algorítmico

• Es usual que el diseño de los sistemas complejos comience particionando su funcionalidad en sub_tareas y derivando especificaciones detalladas para cada una de ellas. Es decir, es necesario diseñar una colección de módulos hardware complejos. Sin embargo, cada vez menos se diseña de cero sino que se reusan o reutilizan bloques (**diseño basado en IPs**)

–Normalmente se venden mediante el correspondiente pago de licencias. En tal caso son como “bloques negros” que se pueden usar (se conocen sus entradas y salidas) pero no se sabe cómo son internamente. También existen de libre distribución (iniciativa OpenCores, similar a la iniciativa de software libre).

Resumen



- Visión más amplia del sistema digital
- El sistema digital se puede implementar en hardware, en software o ambas cosas (co-diseño)
- En esta asignatura nos centramos en diseño hardware
- Realización del hardware: ASICs y FPGAs
- Diseño: Flujo RT -> hardware muy utilizado hoy en día