Universidad de Granada
Departamento de Electrónica y Tecnología de Computadores

TEMA 1

Metodologías y herramientas de diseño

Diseño de Circuitos y Sistemas Electrónicos

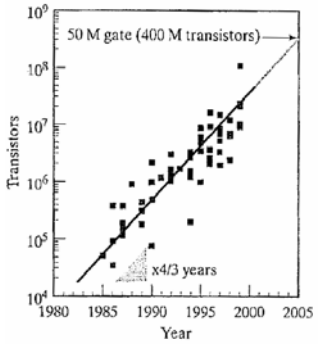
Ing. Telecomunicación

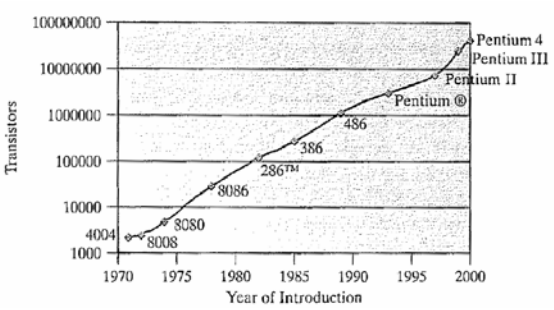
Sumario

- Introducción
- Metodologías de diseño de sistemas electrónicos integrados
- Descripción de sistemas electrónicos integrados
  - ✓ Diseño *full-custom*
  - ✓ Lenguajes de descripción de hardware (HDL)
  - ✓ Descripción de sistemas digitales: VHDL
- Flujos de diseño
  - ✓ Síntesis comportamental
- Verificación

Introducción

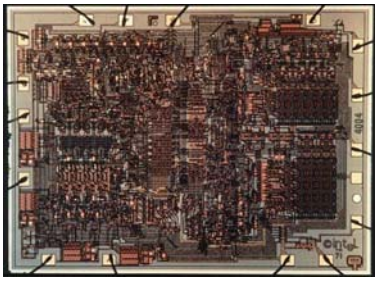
- Las prestaciones y la capacidad de integración de los circuitos integrados crecen con la ley de Moore:





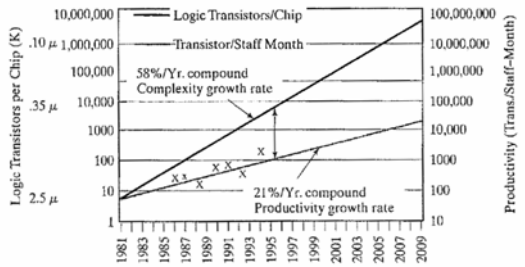
Introducción

- Esta evolución ha tenido un impacto directo en el diseño de circuitos integrados:
  - ✓ los dispositivos modernos requieren una aproximación jerárquica al diseño, empleando herramientas automáticas



Introducción

- El incremento de complejidad de los circuitos integrados implica retos en diferentes niveles:
  - ✓ **tecnológico:** consumo, fiabilidad, procesos de fabricación, etc.
  - ✓ **diseño:** imposible sin herramientas CAD y metodologías adecuadas
    - ❍ aumento del tamaño de los grupos de trabajo



Introducción

- La metodología de diseño ha sufrido diferentes evoluciones durante la era electrónica:
  - ✓ durante los 1970s todo el diseño era completamente *full-custom*
  - ✓ posteriormente, la llegada de las primeras tecnologías programables, como PLAs y PALs, supuso un cierto cambio en sistemas digitales
  - ✓ durante los 1990s el desarrollo de las herramientas CAD y las implementaciones basadas en celdas estándar (*semi-custom*) permitió un incremento en densidad
  - ✓ en años recientes, la aparición de macroceldas y tecnologías programables avanzadas, como las FPGAs, junto al desarrollo de las herramientas CAD y los lenguajes HDL han permitido:
    - ❍ reducción de ciclos de diseño
    - ❍ reutilización de módulos (*IP cores*)

Sumario

- Introducción
- Metodologías de diseño de sistemas electrónicos integrados
- Descripción de sistemas electrónicos integrados
  - ✓ Diseño *full-custom*
  - ✓ Lenguajes de descripción de hardware (HDL)
  - ✓ Descripción de sistemas digitales: VHDL
- Flujos de diseño
  - ✓ Síntesis comportamental
- Verificación

Metodologías de diseño de ICs

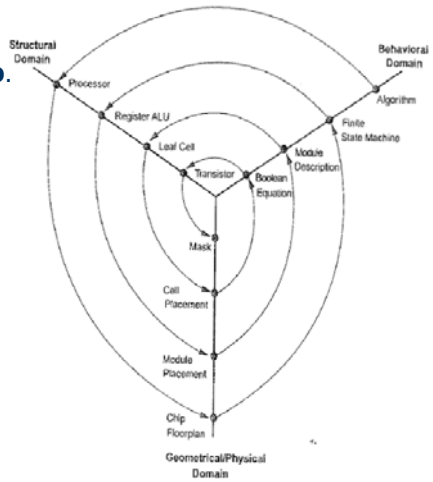
- El diseño de circuitos integrados ha adoptado técnicas de otros campos, como el desarrollo de *software*, para adaptarse a las nuevas tecnologías y demandas.
- Mientras los principios básicos han permanecido estables, el estilo de diseño y las herramientas han evolucionado siguiendo al avance tecnológico y la demanda de mayor productividad.
- A medida que aumenta la complejidad de los sistemas, factores contradictorios se convierten en parámetros de diseño:
  - ✓ prestaciones (velocidad)
  - ✓ consumo
  - ✓ coste y volumen de producción

Metodologías de diseño de ICs

- El diseño de sistemas digitales VLSI puede dividirse en cinco niveles de diseño interrelacionados:
  - ✓ arquitectura (especificaciones de un microprocesador)
  - ✓ microarquitectura (particionamiento de la arquitectura en unidades funcionales)
  - ✓ diseño lógico (descripción de las unidades funcionales)
  - ✓ diseño del circuito (implementación de la lógica con transistores)
  - ✓ diseño físico (*layout* del circuito integrado)
- Dependiendo de la tecnología objetivo y del flujo de diseño, alguno de los niveles anteriores queda oculto o el objeto de su descripción cambia.

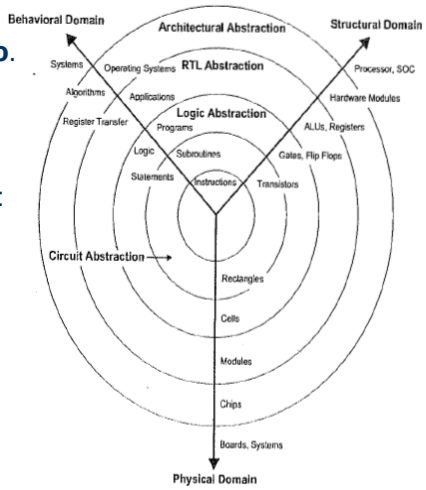
Metodologías de diseño de ICs

- Esta división en el diseño se ve plasmada en el **diseño jerárquico**.
- Esta división se plasma en la descripción a través de los diferentes niveles de **abstracción**:
  - ✓ arquitectura
  - ✓ RTL (*Register Transfer Level*)
  - ✓ lógica
  - ✓ circuito
  - ✓ se extienden a los diferentes tipos de descripción



Metodologías de diseño de ICs

- Esta división en el diseño se ve plasmada en el **diseño jerárquico**.
- Esta división se plasma en la descripción a través de los diferentes niveles de **abstracción**:
  - ✓ arquitectura
  - ✓ RTL (*Register Transfer Level*)
  - ✓ lógica
  - ✓ circuito
  - ✓ se extienden a los diferentes tipos de descripción

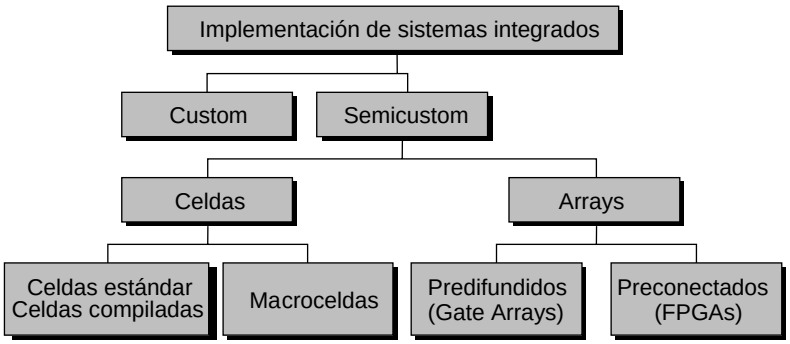


Metodologías de diseño de ICs

- Un mismo sistema puede ser descrito utilizando alguno de los dominios anteriores, o varios simultáneamente:
  - ✓ **descripción comportamental (*behavioral*):** especifica o describe el algoritmo que realiza el sistema
  - ✓ **descripción estructural (*structural*):** especifica los componentes necesarios para formar el sistema y la manera en que han de interconectarse
  - ✓ **descripción física:** especifica la disposición física en la que se han de situar los componentes anteriores
- Dependiendo de la tecnología objetiva y del flujo de diseño, alguno de los niveles se ven modificados o quedan ocultos.

Metodologías de diseño de ICs

- Actualmente se dispone de diferentes alternativas para la implementación de circuitos integrados digitales:



Diseño *full-custom*

- El diseño *full-custom* es la única opción posible cuando las restricciones básicas del sistemas son:
  - ✓ velocidad
  - ✓ densidad
- Su naturaleza implica costes muy elevados de diseño y *time-to-market* considerable, con lo que sólo se justifica cuando:
  - ✓ el bloque diseñado podrá ser reutilizado (librerías de celdas)
  - ✓ el coste puede amortizarse con un gran volumen de producción (microprocesadores y memoria son los principales ejemplos)
  - ✓ el coste no es el factor determinante del diseño (aplicaciones de supercomputación, sistemas militares, etc.)

Diseño *full-custom*

- El diseño *full-custom* cada vez se utiliza menos, incluso en diseño de gran volumen como microprocesadores:
  - ✓ bloques críticos (PLLs, *buffers* de reloj, etc.)
  - ✓ desarrollo de librerías de celdas estándar para diseño *semi-custom*
- Este tipo de diseño implica un uso mínimo de herramientas automáticas, aunque requiere el apoyo de flujos de diseño muy complejos:
  - ✓ herramientas para verificación y simulación
  - ✓ editores del *layout*
  - ✓ comprobación de reglas de diseño (DRC) y reglas eléctricas (ERC)
  - ✓ extracción de elementos parásitos

Diseño basado en celdas

- El uso de diferentes metodologías de diseño basadas en celdas trata de mantener las ventajas del diseño *full-custom* pero acortando y automatizando el proceso:
  - ✓ **"cuanto menor es el tiempo de diseño, mayor es la penalización"**
- El diseño basado en celdas trata de reducir el esfuerzo de diseño reutilizando un conjunto reducido de celdas (librería o biblioteca):
  - ✓ dichas celdas sólo han de diseñarse y verificarse una vez
  - ✓ la reutilización permite amortizar el coste de diseño
  - ✓ la limitación en el número de celdas disponibles reduce la flexibilidad del diseño (penalización en área, consumo, velocidad, etc.)

Diseño basado en celdas

- El diseño basado en **celdas estándar** (***standard cell***) emplea librerías que normalmente contienen:
  - ✓ funciones lógicas sencillas y biestables
  - ✓ bloques básicos (sumadores, contadores)
- El diseño se realiza en un esquemático basado en las celdas a emplear o a partir de descripciones de alto nivel, generándose el *layout* automáticamente:
  - ✓ la automatización es posible por las restricciones impuestas al layout
  - ✓ normalmente, se emplazan las celdas en filas separadas por canales para el rutado
  - ✓ las herramientas CAD actuales (síntesis y *place&route*) han convertido a esta opción en la más usada hoy en día

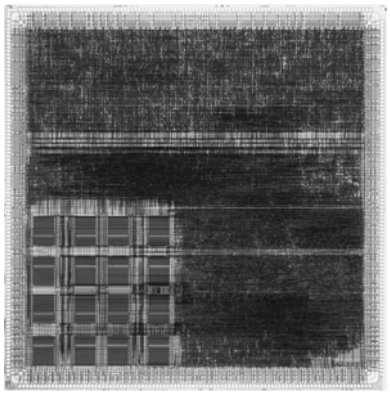
Diseño basado en celdas

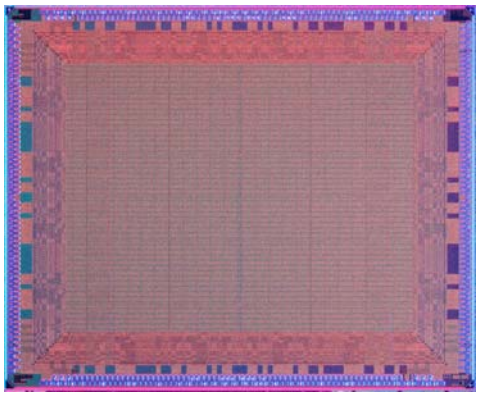
- El diseño basado en celdas estándar resulta ineficiente para estructuras complejas:
  - ✓ es necesario disponer de bloques más complejos que las celdas estándar: **macroceldas**
- **Macrocelda *hard***: representa un módulo con una estructura física predeterminada (diseño *full-custom*):
  - ✓ no pueden trasladarse a otras tecnologías
  - ✓ sólo se emplean si la síntesis automática es insuficiente
- **Macrocelda *soft***: no incluye estructura física predeterminada, sólo restricciones para emplazado y rutado:
  - ✓ pueden implementarse en diferentes tecnologías (*IP cores*)

Diseño basado en *arrays*

- El diseño automatizado no reduce el tiempo de fabricación del circuito integrado ni elimina dicha fabricación:
  - ✓ puede retrasar la introducción del producto
  - ✓ coste elevado sólo asequible para grandes volúmenes de producción
- Para solventar este problema, se han desarrollado diferentes tecnologías que permiten reducir el tiempo de fabricación:
  - ✓ ***Arrays* predifundidos (*sea-of-gates*):** el chip incluye diferentes tipos de celdas estándar y la oblea se almacena a falta de los últimos pasos de interconexión
  - ✓ ***Arrays* preconectados:** el chip incluye una matriz de elementos lógicos conectados por conexiones programables (PALs, PLAs, FPGAs)

Diseño basado en *arrays*





Diseño de Circuitos y Sistemas Electrónicos – Ingeniería de Telecomunicación

Sumario

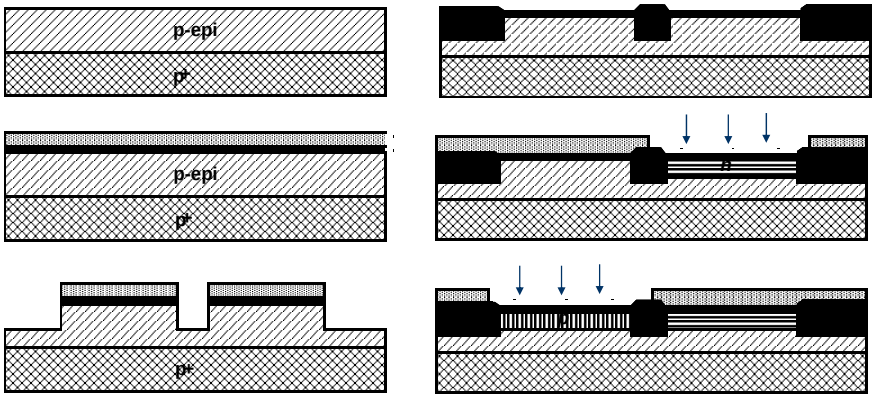
- Introducción
- Metodologías de diseño de sistemas electrónicos integrados
- Descripción de sistemas electrónicos integrados
  - ✓ Diseño *full-custom*
  - ✓ Lenguajes de descripción de hardware (HDL)
  - ✓ Descripción de sistemas digitales: VHDL
- Flujos de diseño
  - ✓ Síntesis comportamental
- Verificación

Descripción de sistemas electrónicos

- La complejidad de los circuitos integrados hace que la tarea de diseño dependa de herramientas CAD y de los procesos de automatización del mismo:
  - ✓ elevados niveles de abstracción
  - ✓ estructura jerárquica
- La descripción de los sistemas ha de estar en consonancia con las herramientas y flujos de diseño:
  - ✓ todas la metodologías anteriores, salvo el diseño *full-custom*, pueden hacer uso de descripciones abstractas o algorítmicas
  - ✓ el diseño *full-custom*, por naturaleza, requiere un tratamiento manual de la topología y estructura del circuito:
    - ❍ herramientas y flujos basados en el proceso de fabricación

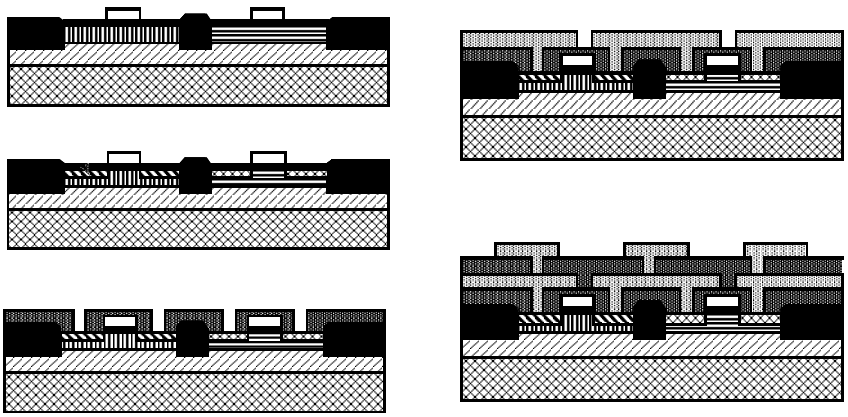
Diseño *full-custom*

- Descripción basada en la especificación de la topología para la fabricación del circuito integrado:



Diseño *full-custom*

- Descripción basada en la especificación de la topología para la fabricación del circuito integrado:



Diseño de Circuitos y Sistemas Electrónicos – Ingeniería de Telecomunicación

Diseño *full-custom*

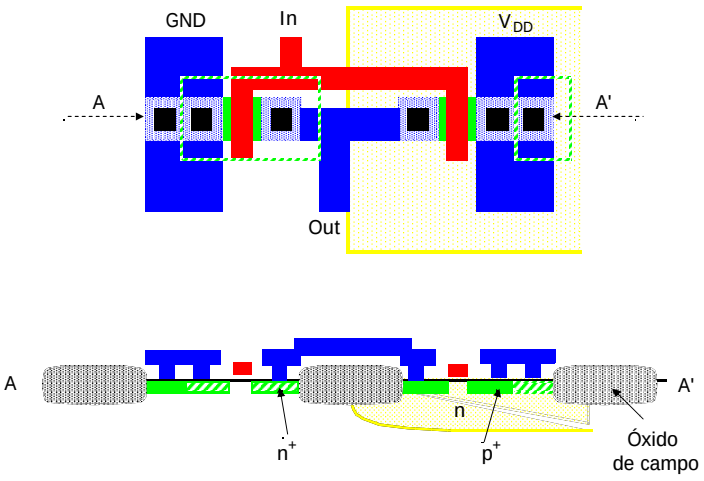
- La descripción *full-custom* está basada en dos conceptos básicos:
  - ***layer*:** traslada las máscaras empleadas en la fabricación de circuitos integrados a un conjunto de niveles de *layout* que permiten la visualización de la estructura del circuito:
    - sustratos y pozos
    - regiones de difusión o activas (n^+, p^+): definen las áreas donde se forman los transistores
    - polisilicio: forma la puerta de los transistores e interconexiones
    - metales: interconexiones
    - vías y contactos: conexiones entre *layers*
    - un *layout* consiste en una combinación de polígonos, cada uno de un cierto *layer*

Diseño *full-custom*

- La descripción *full-custom* está basada en dos conceptos básicos:
  - **reglas de diseño:** conjunto de reglas para el trazado de objetos en el *layout* de un circuito integrado:
    - reflejan las restricciones físicas para la creación de las máscaras de fabricación
    - su unidad básica es la dimensión mínima, que responde a la dimensión mínima en la máscara que puede trasladarse por fotolitografía al circuito integrado
    - principales conjuntos de reglas:
      - reglas para transistores
      - reglas para vías y contactos
      - reglas para pozos y sustratos

Diseño de Circuitos y Sistemas Electrónicos – Ingeniería de Telecomunicación

Diseño *full-custom*



Diseño *full-custom*

- El diseño *full-custom*, aparte de las herramientas de edición de *layout*, depende de dos herramientas básicas:
  - ✓ **DRC (*Design-Rule Check*):** comprueba que el *layout* trazado satisface las reglas de diseño, ya que de lo contrario su funcionamiento no será correcto
  - ✓ **extracción del circuito:** extrae un circuito esquemático del *layout* físico del circuito integrado mediante el análisis de los objetos trazados y la interacción entre los diferentes *layers*:
    - ❍ extracción de la red de transistores y sus interconexiones: permite la simulación y verificación funcional del circuito
    - ❍ extracción de elementos parásitos: simulación y análisis preciso del comportamiento real del circuito

Lenguajes de descripción de hardware

- A medida que aumenta la complejidad de los sistemas integrados, la descripción puramente estructural se hace inabordable:
  - ✓ es necesario trasladar a esquemáticos las estructuras lógicas, proceso laborioso y **propenso a errores**
  - ✓ estructuras complejas son difícilmente abordables sin herramientas de síntesis automática
  - ✓ es imprescindible verificar mediante simulación el correcto funcionamiento del sistema durante el diseño
- El aumento del nivel de **abstracción** y la **jerarquización** en la descripción de los sistemas integrados requiere el uso de medios de descripción unificados.

Lenguajes de descripción de hardware

- Los lenguajes de descripción de hardware (HDL: *Hardware Description Language*) permiten:
  - ✓ describir sistemas en un nivel muy elevado de abstracción
  - ✓ facilitar la simulación y verificación del sistema durante el diseño
  - ✓ crear descripciones portables e independientes de la tecnología de implementación
- La síntesis basada en descripciones HDL produce circuitos menos densos y más lentos que los resultantes del diseño *full-custom* por diseñadores expertos:
  - ✓ el estado de desarrollo tecnológico permite que estas implementaciones basten para la mayoría de ASICs actuales

Lenguajes de descripción de hardware

- A pesar de su similitud con los lenguajes de programación, existe una diferencia fundamental:
  - ✓ en *software* la ejecución del algoritmo descrito es secuencial
  - ✓ en *hardware*, un circuito está formado por bloques que funcionan simultáneamente
- Su uso, especialmente para síntesis, depende de las herramientas a utilizar, con lo que es preciso describir el sistema teniendo en mente el tipo de circuito que se va a generar:
  - ✓ pueden obtenerse resultados no deseados o incorrectos (introducción de registros y *latches*, imposibilidad de síntesis, etc.)
  - ✓ el resultado puede ocupar mucha más área de la requerida o ser excesivamente lento

Lenguajes de descripción de hardware

- Los lenguajes más usados actualmente son:
  - ✓ Verilog (IEEE1364-01):
    - ❍ desarrollado en 1984 por Gateway Design Automation para simulación lógica
    - ❍ convertido en estándar abierto en 1990 tras la compra de Gateway por Cadence
    - ❍ compacto y de sintaxis similar a C
  - ✓ VHDL (IEEE1076-02, *Very high-speed integrated circuit* HDL):
    - ❍ desarrollado en 1981 por el Departamento de Defensa y estandarizado por IEEE en 1987
    - ❍ menos compacto y más verboso que Verilog
    - ❍ adecuado para grandes equipos y proyectos

Descripción de sistemas digitales: VHDL

- La descripción con VHDL se basa en la definición de unidades de código que se agrupan en diferentes ficheros.
- Los tipos de unidades posibles en VHDL son:
  - ✓ **entidad (`entity`):** describe la interfaz con el exterior de un elemento
    - ❍ equivale al símbolo de un elemento en un esquemático
  - ✓ **arquitectura (`architecture`):** describe la estructura o el comportamiento interno de una entidad
    - ❍ es posible definir diferentes arquitecturas para cada entidad
    - ❍ incluye la declaración de componentes, señales y variables internas, etc.

Descripción de sistemas digitales: VHDL

- La descripción con VHDL se basa en la definición de unidades de código que se agrupan en diferentes ficheros.
- Los tipos de unidades posibles en VHDL son:
  - ✓ **configuración (`configuration`):** especifica la arquitectura, si existen varias, que se asocia en cada momento a una entidad dependiendo de ciertas condiciones (simulación, síntesis, etc.)
  - ✓ **paquete (`package`):** incluye un conjunto de declaraciones (tipos, subtipos, etc.) compartidos por varias unidades de diseño
  - ✓ **cuerpo de paquete (`package body`):** define los elementos declarados en el paquete correspondiente
    - ❍ existen paquetes predefinidos y el usuario puede crear otros

Diseño de Circuitos y Sistemas Electrónicos – Ingeniería de Telecomunicación

Descripción de sistemas digitales: VHDL

- Los principales paquetes predefinidos son:
  - ✓ `std.standard`: define los tipos básicos de VHDL como `boolean`, `bit`, `bit_vector`, `character`, `string`, `integer`, `real`, `time`, etc.
  - ✓ `std.textio`: define los tipos `line` y `text` y procedimientos de lectura y escritura en ficheros
  - ✓ `IEEE.std_logic_1164`: define los tipos `std_logic` y `std_logic_vector`
  - ✓ `IEEE.std_logic_arith`: define los tipos `signed` y `unsigned` y operaciones aritméticas con estos tipos
  - ✓ `IEEE.std_logic_signed`: define operaciones aritméticas con el tipo `std_logic_vector` considerado codificado con signo

Descripción de sistemas digitales: VHDL

- Los principales paquetes predefinidos son:
  - ✓ `IEEE.std_logic_unsigned`: define operaciones aritméticas con el tipo `std_logic_vector` considerado codificado sin signo
  - ✓ `IEEE.std_logic_textio`: define procedimientos de lectura y escritura en ficheros para los tipos `std_logic` y `std_logic_vector`.
- VHDL es un lenguaje **fuertemente tipado**, a diferencia de muchos lenguajes de programación:
  - ✓ no es posible realizar asignaciones entre señales o variables de tipos diferentes
  - ✓ requiere funciones de conversión entre tipos

Diseño de Circuitos y Sistemas Electrónicos – Ingeniería de Telecomunicación

Descripción de sistemas digitales: VHDL

```
LIBRARY ieee;                                   -- Declaracion librerias
USE ieee.std_logic_1164.ALL;                    -- Paquetes en uso de libreria
USE ieee.std_logic_arith.ALL;

ENTITY biestableD IS                            -- Declaracion de entidad
    PORT (
          D,clk : IN  STD_LOGIC;                -- Entradas
          q     : OUT STD_LOGIC                 -- Salida
          );
END biestableD;

ARCHITECTURE proceso OF biestableD IS           -- Declaracion de arquitectura
BEGIN                                           -- Cuerpo de la arquitectura
proceso_biestableD : PROCESS (clk)              -- Lista de sensibilidad
    BEGIN
          IF clk'event AND CLK='1' THEN         -- Flanco de subida de clk
                    q <= d;                     -- Asignación concurrente
          END IF;
    END PROCESS proceso_biestableD;             -- Fin del proceso
END proceso;
```


Descripción de sistemas digitales: VHDL

- Los puertos de las entidades pueden ser de cuatro tipos, atendiendo al tipo de E/S salida físicas:
  - ✓ `in`: puerto de entrada
    - ❍ sólo puede ser leído en el interior de la entidad
  - ✓ `out`: puerto de salida
    - ❍ la entidad sólo puede actualizar su valor, nunca leerlo
  - ✓ `buffer`: puerto de salida
    - ❍ la entidad puede actualizar su valor y leerlo internamente
    - ❍ puede cambiarse a `out` añadiendo una señal intermedia
  - ✓ `inout`: puerto de entrada/salida
    - ❍ la entidad puede actualizar su valor y leerlo (fijado externamente)
    - ❍ normalmente corresponde a una E/S triestado

Descripción de sistemas digitales: VHDL

- En una arquitectura se puede definir:
  - ✓ tipos (`type`) y constantes (`constant`): enumerados, físicos, compuestos, etc.
  - ✓ señales (`signal`): corresponden a las interconexiones en el interior de la entidad
- El cuerpo de la arquitectura de una entidad está compuesto por:
  - ✓ asignaciones concurrentes: se sintetizan como lógica combinacional
  - ✓ procesos: a través de la lista de sensibilidad permiten describir elementos secuenciales de un sistema
  - ✓ instanciación de componentes: permite estructurar jerárquicamente el diseño a través de una descripción estructural

Descripción de sistemas digitales: VHDL

- Asignaciones concurrentes:

```
comp <= '1' WHEN a=b ELSE '0';            -- Comparador
addsub <= a+b WHEN sel='1' ELSE a-b;      -- Sumador/restador
z <= x WHEN tri='0' ELSE 'Z';             -- Buffer triestado

SIGNAL code   : STD_LOGIC_VECTOR(1 DOWNTO 0);
SIGNAL decode : STD_LOGIC_VECTOR(3 DOWNTO 0);
...
WITH code SELECT
     decode   <=   "0001" WHEN "00",
                   "0010" WHEN "01",
                   "0100" WHEN "10",
                   "1000" WHEN "11",
                   "0000" WHEN OTHERS;
```

Descripción de sistemas digitales: VHDL

- Los procesos requieren:
  - ✓ lista de sensibilidad: señales que lo activan
  - ✓ sentencia `wait`: suspende la ejecución del proceso
    - ❍ `wait on` lista de sensibilidad
    - ❍ `wait until` expresión booleana
    - ❍ `wait for` expresión temporal
- Dentro de los procesos se pueden definir variables (`variable`):
  - ✓ se declaran y se asignan dentro del proceso
  - ✓ se evalúan en el momento en el que la sentencia de asignación se ejecuta y el resultado se asigna inmediatamente a la variable

Descripción de sistemas digitales: VHDL

- Los procesos permiten diferentes estructuras secuenciales:

```
WHILE condicion booleana LOOP
...                          -- Sentencias secuenciales
END LOOP;

FOR identificador IN rango LOOP
...                          -- Sentencias secuenciales
END LOOP;

LOOP
...                          -- Sentencias secuenciales
END LOOP;
```

Descripción de sistemas digitales: VHDL

- Los procesos permiten diferentes estructuras secuenciales:

```
IF condicion booleana THEN
...                          -- Sentencias secuenciales
ELSIF condicion booleana THEN
...                          -- Sentencias secuenciales
END IF;

CASE (señal,variable)IS
  WHEN valor1=>
        ...                  -- Sentencias secuenciales
  WHEN valor2=>
        ...                  -- Sentencias secuenciales
  ...
  WHEN OTHERS=>              -- Especificar todas las opciones
        ...                  -- Sentencias secuenciales
END CASE;
```


Descripción de sistemas digitales: VHDL

- Palabras reservadas en VHDL-87:

ABS	CASE	GENERIC	NOR	REGISTER	VARIABLE
ACCESS	COMPONENT	GUARDED	NOT	REM	WAIT
AFTER	CONFIGURATION	IF	NULL	REPORT	WHEN
ALIAS	CONSTANT	IN	OF	RETURN	WHILE
ALL	DISCONNECT	INOUT	ON	SELECT	WITH
AND	DOWNTO	IS	OPEN	SEVERITY	XOR
ARCHITECTURE	ELSE	LABEL	OR	SIGNAL	
ARRAY	ELSIF	LIBRARY	OTHERS	SUBTYPE	
ASSERT	END	LINKAGE	OUT	THEN	
ATTRIBUTE	ENTITY	LOOP	PACKAGE	TO	
BEGIN	EXIT	MAP	PORT	TRANSPORT	
BLOCK	FILE	MOD	PROCEDURE	TYPE	
BODY	FOR	NAND	PROCESS	UNITS	
BUFFER	FUNCTION	NEW	RANGE	UNTIL	
BUS	GENERATE	NEXT	RECORD	USE	

- Palabras reservadas añadidas en VHDL-93:

GROUP	LITERAL	REJECT	SHARED	SRA	XNOR
IMPURE	POSTPONED	ROL	SLA	SRL	
INERTIAL	PURE	ROR	SLL	UNAFFECTED	

Descripción de sistemas digitales: VHDL

- Operadores lógicos:

  `and  or  nand  nor  xor  xnor  not`

- Operadores de relación:

  `=  /=  <  <=  >  >=`

- Operadores de desplazamiento:

  `sll  srl  sla  sra  rol  ror`

- Operadores aditivos:

  `+  -  &`

- Operadores multiplicativos:

  `*  /  mod  rem`

- Otros operadores:

  `abs  **`

Descripción de sistemas digitales: VHDL

- **Ejemplo:** descripción estructural

```
ENTITY half_adder IS                -- Declaracion de entidad
     PORT (
               a,b : IN  BIT;       -- Sumandos
               c,s : OUT BIT        -- Suma y acarreo
               );
END half_adder;
ARCHITECTURE dcse OF half_adder IS
   COMPONENT xor2                   -- Puerta EXOR
               PORT (    x,y : IN  BIT;
                         z   : OUT BIT);
   END COMPONENT;
   COMPONENT and2                   -- Puerta AND
               PORT (    x,y : IN  BIT;
                         z   : OUT BIT);
   END COMPONENT;
BEGIN
   C1: xor2 PORT MAP (x => a, y => b, z => s);-- s <= a XOR b
   C2: and2 PORT MAP (x => a, y => b, z => c);-- c <= a AND b
END dcse;
```

Descripción de sistemas digitales: VHDL

- **Ejemplo:** *latch* y biestable con reset síncrono

```
ENTITY latch IS
PORT (
    d,clk : IN  BIT;
    q     : OUT BIT );
END latch;

ARCHITECTURE dcse OF latch IS
BEGIN
  PROCESS (d,clk)
      BEGIN
      IF clk= '1' THEN
          q <= d;
      END IF;
  END PROCESS;
END dcse;
```

```
ENTITY flip-flop IS
PORT (
    d,rst,clk : IN  BIT;
    q         : OUT BIT );
END flip-flop;

ARCHITECTURE dcse OF flip-flop IS
BEGIN
  PROCESS (clk)
      BEGIN
      IF clk'event AND clk= '1' THEN
          IF rst= '0' THEN
             q <= '0';
          ELSE
             q <= d;
          END IF;
      END IF;
  END PROCESS;
END dcse;
```


Descripción de sistemas digitales: VHDL

- **Ejemplo:** registro de 32 bits

```
ENTITY reg_32 IS
    PORT (
            d   : IN         STD_LOGIC_VECTOR(31 DOWNTO 0);
            q   : OUT        STD_LOGIC_VECTOR(31 DOWNTO 0);
            clk : IN         STD_LOGIC
            );
END reg_32;

ARCHITECTURE dcse OF reg_32 IS
BEGIN
   PROCESS(clk)
   BEGIN
     IF clk'event AND clk= '1' THEN
               q <= d;
     END IF;
   END PROCESS;
END dcse;
```

Descripción de sistemas digitales: VHDL

- **Ejemplo:** máquina de estados finitos con reset asíncrono

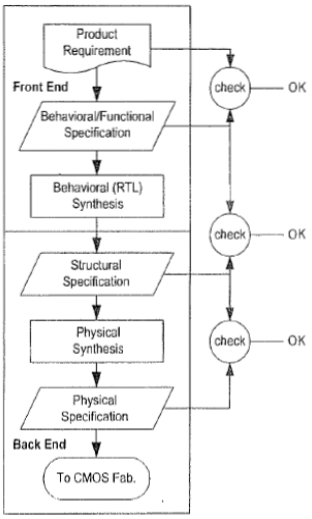
```
ENTITY fsm IS
PORT (
    clk     : IN STD_LOGIC;
    rst     : IN STD_LOGIC;
    entrada : IN STD_LOGIC;
    salida  : OUT STD_LOGIC
);
END fsm;

ARCHITECTURE dcse OF fsm IS

TYPE estado IS (init,exec,idle);
SIGNAL estado_actual : estado;

BEGIN
   PROCESS (clk, rst)
   BEGIN
     IF rst = '1' THEN
       estado_actual <= init;
ELSIF clk'event AND clk ='1' THEN
  CASE estado_actual IS
     WHEN init =>
        IF condicion THEN
          estado_actual <= exec;
        END IF;
     WHEN exec =>
        IF condicion THEN
          estado_actual <= idle;
        END IF;
     WHEN idle =>
        IF condicion THEN
          estado_actual <= init;
        END IF;
  END CASE;
 END IF;
END PROCESS;
END dcse;
```


Sumario

- Introducción
- Metodologías de diseño de sistemas electrónicos integrados
- Descripción de sistemas electrónicos integrados
  - ✓ Diseño *full-custom*
  - ✓ Lenguajes de descripción de hardware (HDL)
  - ✓ Descripción de sistemas digitales: VHDL
- Flujos de diseño
  - ✓ Síntesis comportamental
- Verificación

Flujos de diseño

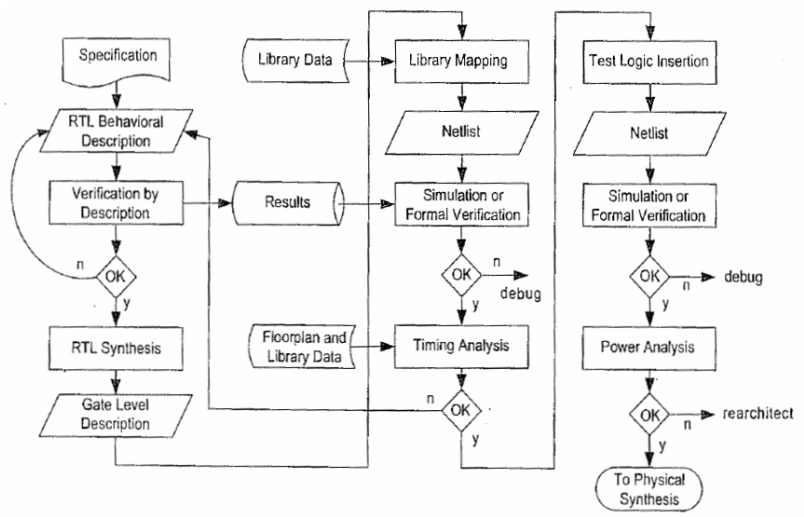
- **Flujo de diseño:** conjunto de procedimientos y herramientas que permiten realizar libre de errores un determinado diseño partiendo desde una especificación inicial hasta conseguir una implementación física
- A pesar de las diferentes tecnologías finales, la mayoría de flujos comerciales son similares:
  - ✓ *front-end*: síntesis comportamental
  - ✓ *back-end*: síntesis física



Síntesis comportamental

- En el nivel comportamental, la especificación del sistema no depende de la implementación:
  - ✓ proporciona independencia frente a detalles concretos de la implementación
  - ✓ se depende completamente de las herramientas del flujo para garantizar un diseño óptimo
- Para el diseño de ASICs, las herramientas más usadas traducen la descripción comportamental a una *netlist* estructural basada en puertas:
  - ✓ para otras tecnologías, esta *netlist* estructural estará basada en las **primitivas** disponibles en dicha tecnologías

Síntesis comportamental



Síntesis comportamental

- **Diseño lógico y verificación:** se simula la descripción HDL del sistema utilizando bancos de prueba (*testbench*) adecuados:
  - ✓ pueden emplearse simuladores *ad hoc* (ModelSim, ActiveHDL)
  - ✓ permite depurar el sistema a nivel algorítmico
  - ✓ en ocasiones, pueden desarrollarse arquitecturas sólo para simulación
- **Síntesis RTL:** la descripción comportamental se traduce a una *netlist* de puertas y registros genéricos, a partir de una descripción RTL (*Register Transfer Level*) intermedia:
  - ✓ descomposición de máquinas de estado, optimización de la ruta de datos, optimización de consumo
  - ✓ Design Compiler (Synopsys), BuildGates (Cadence), Synplify (Synplicity)

Síntesis comportamental

- **Mapeo:** la descripción genérica en puertas se traduce a una *netlist* basada en primitivas propias de la tecnología:
  - ✓ ciertos bloques (memorias, aritmética, DSP) se trasladan a recursos específicos
  - ✓ dependiendo del flujo de diseño concreto, puede incluir la fase de emplazamiento y enrutamiento (*place&route*)
- **Verificación funcional:** idealmente, la *netlist* resultante del mapeo ha de implementar correctamente el sistema descrito:
  - ✓ código poco adecuado o ambiguo puede provocar que los sintetizadores generen implementaciones incorrectas
  - ✓ el sistema sintetizado puede someterse a los *testbenches* usados para la verificación lógica (requiere patrones de test exhaustivos)

Síntesis comportamental

- **Análisis temporal:** el resultado de la síntesis no sólo de ser funcionalmente correcto sino cumplir las especificaciones temporales del diseño:
  - ✓ las herramientas de análisis requieren una modelización detallada de todos los retardos en la tecnología de implementación
  - ✓ comprueban los retardos máximos (t_{su}) y mínimos (t_{hold})
- **Análisis de consumo:** es un aspecto cada vez más importante en el sistema final y se incorpora a más flujos de diseño:
  - ✓ depende de la actividad del circuito y los patrones de entrada
  - ✓ puede realizarse empleando un determinado *testbench* y analizando los resultados de la simulación

Sumario

- Introducción
- Metodologías de diseño de sistemas electrónicos integrados
- Descripción de sistemas electrónicos integrados
  - ✓ Diseño *full-custom*
  - ✓ Lenguajes de descripción de hardware (HDL)
  - ✓ Descripción de sistemas digitales: VHDL
- Flujos de diseño
  - ✓ Síntesis comportamental
- Verificación

Verificación

- La **verificación** o **test** (durante las etapas de diseño, tras la fabricación y durante la vida útil) del sistema puede suponer un esfuerzo mayor que el propio diseño:
  - ✓ **verificación lógica y funcional:** asegura que el resultado del diseño se ajusta al funcionamiento y requerimientos previstos
  - ✓ **pruebas físicas (*silicon debug*):** realizadas sobre los primeros prototipos, aseguran que la implementación física opera correctamente:
    - ❍ mucho más compleja que la verificación en la etapa de diseño
  - ✓ **tests de fabricación:** realizados sobre cada chip fabricado, aseguran que éste funciona correctamente:
    - ❍ durante la vida útil permiten verificar fallos de un componente

Verificación

- En sistemas complejos, la verificación del circuito integrado es una tarea compleja que requiere el uso de técnicas especiales durante la fase de diseño.
- La verificación de un circuito integrado requiere asegurar:
  - ✓ **controlabilidad:** posibilidad de fijar el valor lógico de cada nodo interno del circuito integrado
  - ✓ **observabilidad:** posibilidad de examinar, directa o indirectamente, el estado de cualquier nodo interno del circuito integrado
- Principales aproximaciones al **diseño para testeabilidad**:
  - ✓ técnicas basada en *scan*
  - ✓ *Built-In-Selft-Test* (BIST)

Verificación

- En las técnicas basadas en *scan* los registros cuentan con un modo especial de operación (*scan mode*):
  - ✓ todos los registros del sistemas forman un registro de desplazamiento
  - ✓ es posible examinar el estado interno del sistema o inducir un cierto estado operando sobre dicho registro
- Para reducir el tamaño del registro de *scan* se pueden emplear:
  - ✓ ***scan* paralelo:** se usan varios registros de *scan*, bien por módulos o bien de manera automática para igualar sus longitudes
  - ✓ ***scan* parcial:** solo se incluyen partes críticas del sistema
- Dado que su inclusión requiere modificar la estructura del sistema, ha de minimizarse el impacto (área y velocidad)

Verificación

- Las técnicas de *scan* pueden extenderse para la verificación de sistemas con varios chips:
  - ✓ *Boundary-scan* (JTAG): los pines de E/S de todos los chips se conectan formando un registro de *scan*
- Las técnicas BIST aumentan el área del sistema incluyendo recursos para realizar ciertas operaciones que los autoverifiquen:
  - ✓ análisis de firma o comprobación de redundancia cíclica (CRC)
  - ✓ inclusión de BILBOs (*Built-In Logic Block Observation*)
- Otras técnicas examinan parámetros físicos del chip:
  - ✓ test IDDQ: detecta faltas de anclaje en lógica CMOS estática examinando la corriente en las líneas de alimentación del chip