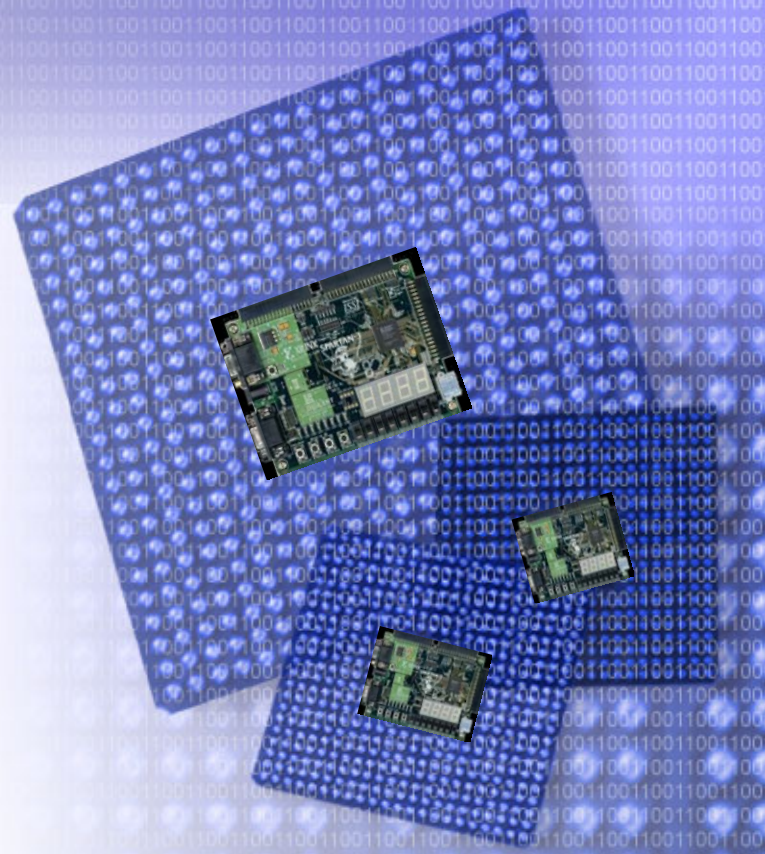
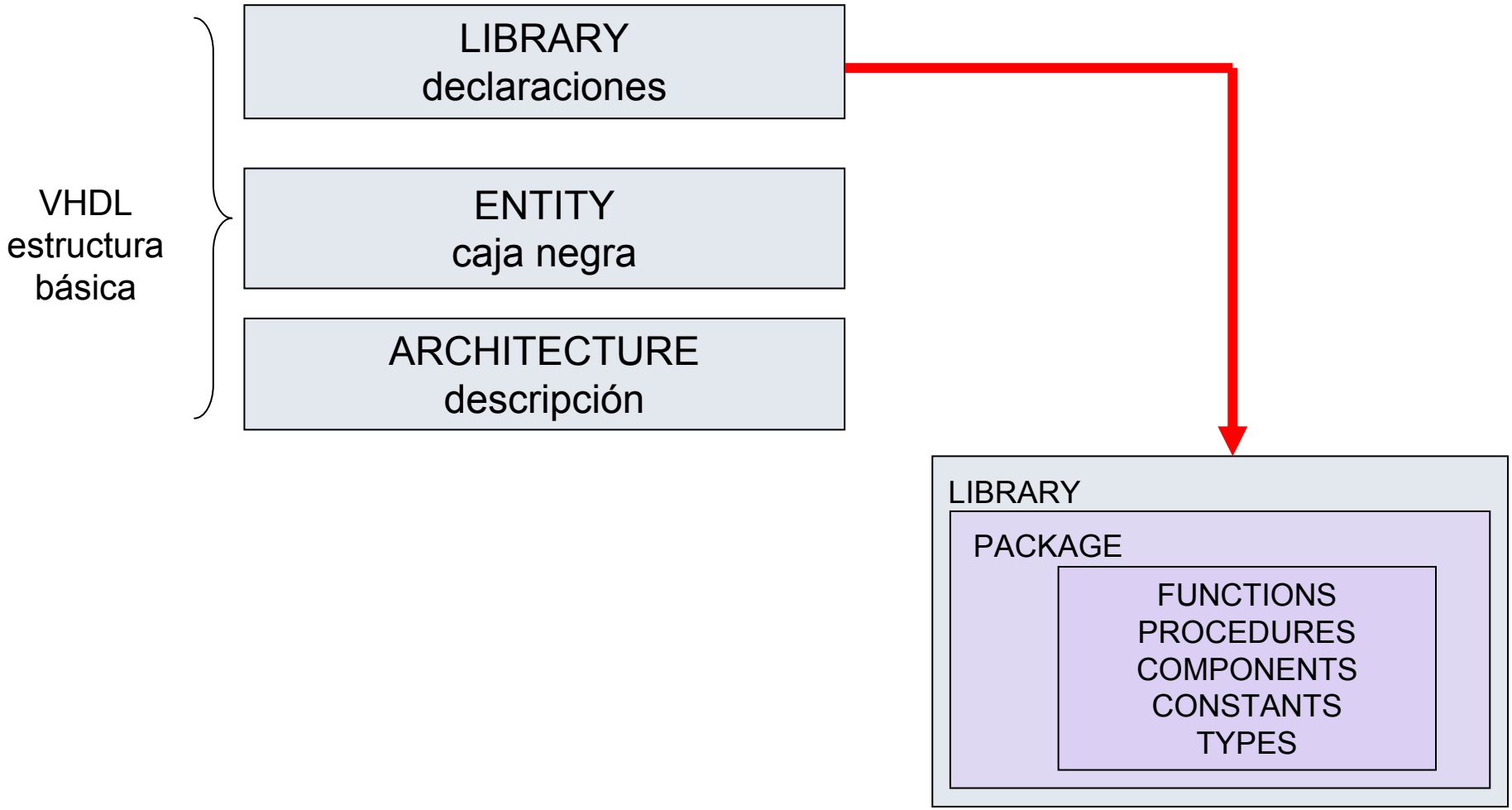
Universitat d'Alacant
Universidad de Alicante



VHDL. Lenguaje de descripción hardware

Estructura Básica de diseño

Estructura de un diseño en VHDL

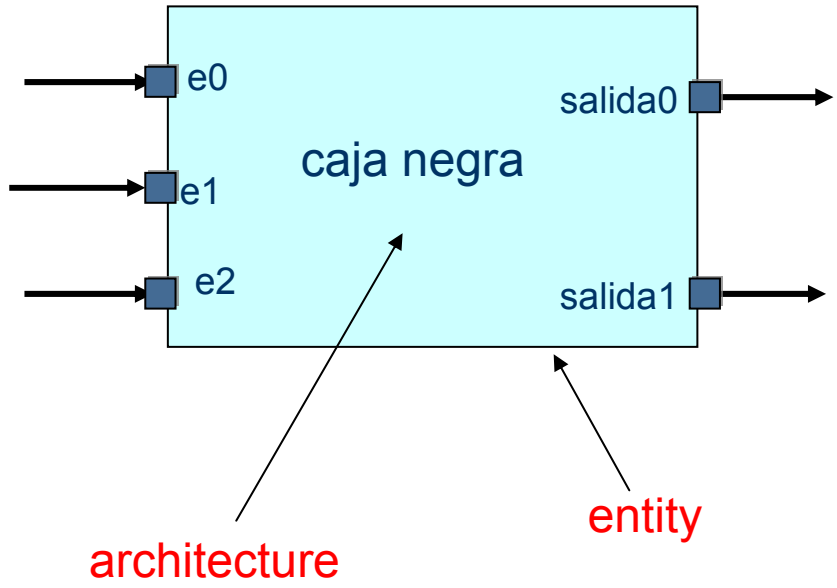


Elementos de descripción VHDL

- Bibliotecas (Library)
  - Almacenan los elementos de diseño: tipo de datos, operadores, componentes, objetos, funciones,...
  - Esos elementos de diseño se organizan en Paquetes
    - Packages: son unidades de almacenamiento de elementos y tienen que hacerse “visibles” para poder ser utilizados .
  - Hay 2 bibliotecas que siempre son visibles por defecto: std (la standard) y work (la de trabajo) y que no es necesario declarar
- Entidades (Entity)
  - Es el modelo de interfaz de un circuito con el exterior mediante unos terminales de entrada y de salida.
  - Es la caja negra que define las entradas y salidas.
- Arquitectura (Architecture)
  - Describe el funcionamiento del circuito.

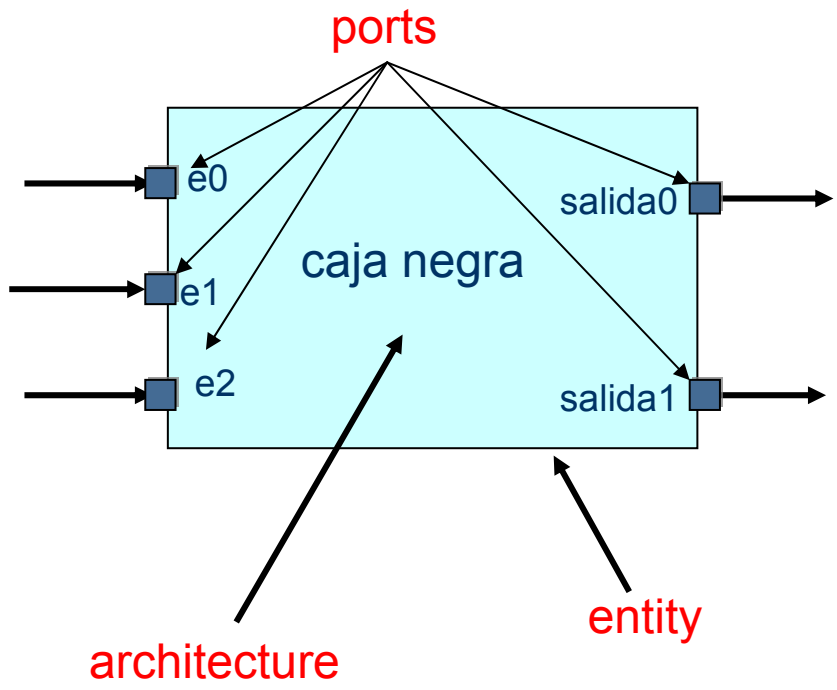
La entidad y la arquitectura

- Una unidad hardware se puede visualizar como una caja negra
  - El interfaz de la “caja negra” está completamente definido
  - El interior está oculto
- En VHDL la caja negra se denomina entidad
  - La ENTITY describe la E/S del diseño
- Para describir su funcionamiento se asocia una implementación que se denomina arquitectura
  - La ARCHITECTURE describe el contenido del diseño.



Puertos de una entidad. PORT

- Cada una de las posibles conexiones se denomina PORT y consta de:
  - Un **nombre**, que debe ser único dentro de la entidad.
  - Una **lista de propiedades**, como:
    - la dirección del flujo de datos, entrada, salida, bidireccional y se conoce como **MODO** del puerto.
    - los valores que puede tomar el puerto: '0', '1' o ('Z'), etc.,los valores posibles dependen de lo que se denomina **TIPO** de señal.
  - Los puertos son una clase especial de **señales** que además añade el modo al tipo de señal



Port = Canal de Comunicación

Modos de un Puerto

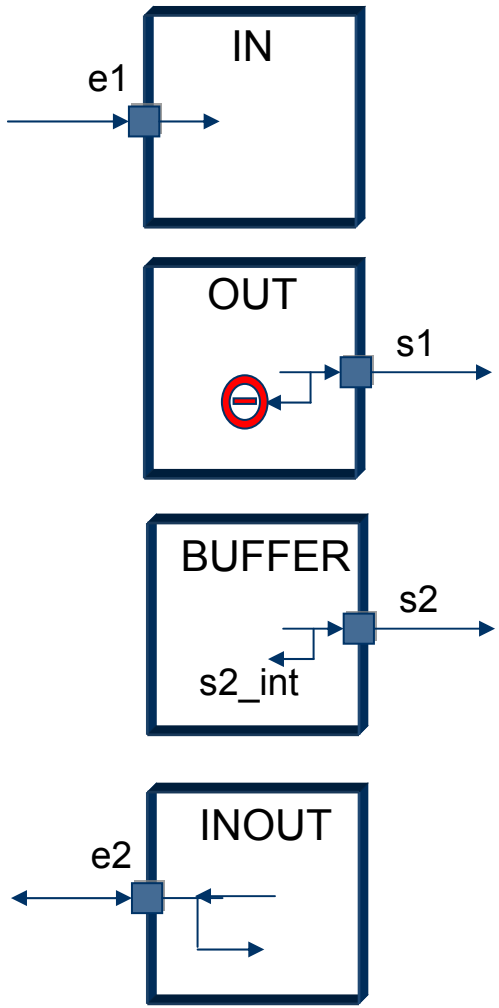
Indican la dirección y si el puerto puede leerse o escribirse dentro de la entidad

IN: Una señal que entra en la entidad y no sale. La señal puede ser leída pero no escrita.

OUT: Una señal que sale fuera de la señal y no es usada internamente. La señal no puede ser leída dentro de la entidad.

BUFFER: Una señal que sale de la entidad y también es realimentada dentro de la entidad.

INOUT: Una señal que es bidireccional, entrada/salida de la entidad.



Estructura de un diseño en VHDL

```
LIBRARY ieee;
USE ieee.std_LOGIC_1164.all;

ENTITY nombre_entidad IS
[GENERIC(   )];
PORT(

   );
END nombre_entidad;

ARCHITECTURE nombre_arquitectura OF nombre_entidad IS


BEGIN


END nombre_arquitectura;
```

declaraciones de puertos

Nombre de la entidad

parte declarativa de la arquitectura

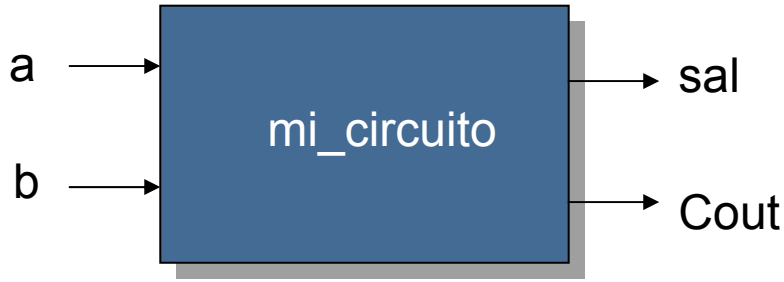
cuerpo de la arquitectura

Nombre de la arquitectura

Declaración de entidad

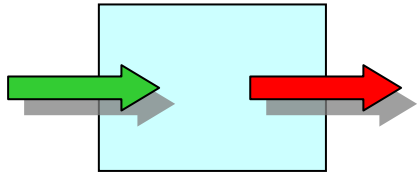
```
LIBRARY ieee;
USE ieee.std_logic_1164.ALL;
ENTITY mi_circuito IS
PORT (
        a       : IN  std_logic_vector(7 DOWNTO 0);
        b       : IN  std_logic_vector(7 DOWNTO 0);
        sal     : OUT std_logic_vector(7 DOWNTO 0);
        Cout    : OUT std_logic
      );
END mi_circuito ;
```

modo tipo



Resumen: Entidad y Arquitectura

- La entidad se utiliza para hacer una descripción "caja negra" del diseño, sólo se detalla su interfaz



- Los contenidos del circuito se modelan dentro de la arquitectura

Una entidad puede tener varias arquitecturas

- Por ejemplo, la descripción de comportamiento que ha hecho el diseñador y el modelo post-layout obtenido después de implementar el chip

Invariancias

- VHDL presenta ciertas invariancias que conviene conocer antes de su utilización.
  - Invariante a mayúsculas, es decir, dos expresiones iguales conteniendo mayúsculas y minúsculas son idénticas.

```
Salida <= A and B;   ≡   SALIDa <= a AND b;
```

  - Invariante a los espacios, es decir, dos expresiones iguales conteniendo más o menos espacios son idénticas

```
Salida <= A     and B;   ≡   Salida <= A     and     B;
```

  - Los comentarios van detrás de dos rayas “- -” y convienen que sean claros para que las descripciones puedan ser fácilmente utilizadas pr otras personas o por ti mismo.

Invariancias

- VHDL es relativamente laxo con la utilización de paréntesis, una buena idea es utilizar los paréntesis de manera que una persona la pueda entender con facilidad

if x='0' and y='0' or z='1' then
  bla;
  bla;
end if;
	if (((x='0') and (y='0')) or (z='1')) then
	  bla;
	  bla;
	end if;

- Cada asignación termina con ";"
- Cada **"if"** tiene el correspondiente **"then "**
- Cada **"if"** termina con el correspondiente **"end if"**
- Si se necesita **"else if"** se utilizará **"elsif"**
- Cada **"case"** termina con el correspondiente **"end case"**
- Cada **"loop"** termina con el correspondiente **"end loop"**

Identificadores

- **Identificadores**. Son las palabras que se utilizan para identificar a las funciones, señales, puertos, variables, etc. Es conveniente que dichas palabras proporcionen información suficiente para que la descripción sea fácilmente reutilizable.
  - El identificador debe dar suficiente información para su uso .
  - El identificador puede ser tan largo como se quiera, pero un nombre demasiado largo es complicado de utilizar, y demasiado corto quizá proporcione poca información.
  - El identificador puede contener cualquier combinación de las letras (A-Z y a-z) números (0-9) y el sub-guión (“_”)
  - El identificador debe empezar por un carácter alfabético.
  - El identificador no puede termina con el sub-guión (“_”)

Palabras reservadas

- Son palabras que no se pueden utilizar como identificadores

accses	exit	mod	return	while
after	file	new	signal	with
alias	for	next	shared	
all	function	null	then	
atribute	generic	of	to	
block	group	on	type	
body	in	open	until	
buffer	is	out	use	
bus	label	range	variable	
constant	loop	rem	wait	