Estructura de un Código VHDL

Unidades Fundamentales

Un código VHDL se compone de tres secciones:

- Library/Package (declaración de bibliotecas o paquetes)

  Contiene una lista de todas las bibliotecas o paquetes necesarios en el diseño.

- ENTITY (entidad)

  Especifica principalmente los puertos de entrada/salida, más constantes genéricas (opcional).

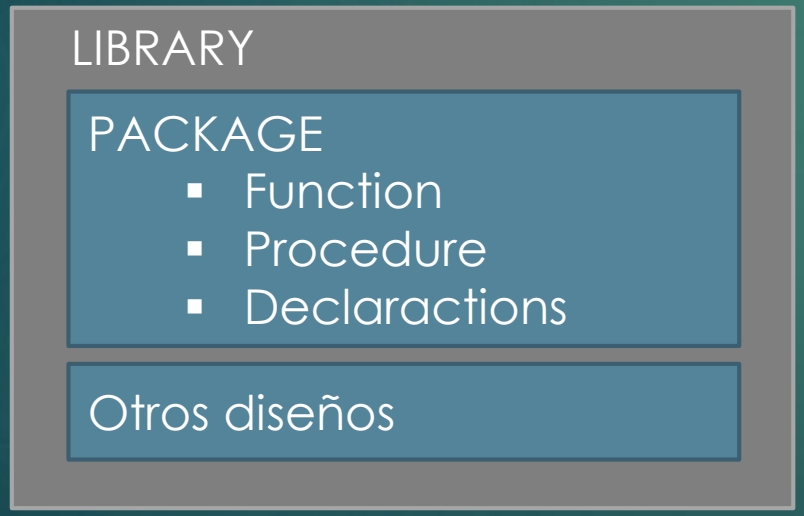
- ARCHITECTURE (arquitectura)

  Contiene el código que describe como debe funcionar el circuito, a partir del cual las estructuras de hardware son inferidas.

Bibliotecas y Paquetes

Una biblioteca es una colección de piezas de código usadas regularmente.

La estructura típica de una biblioteca se ilustra en la figura siguiente.



Bibliotecas y Paquetes

Cualquier circuito previo puede ser parte de una biblioteca. El circuito puede usarse (instanciación) en otros diseños usando la palabra reservada COMPONENT.

Otra opción es escribir las piezas de código usadas regularmente en la forma de una FUNCTION o PROCEDURE (a esto se le conoce como subprograma), y colocarlas dentro de un paquete.

Las declaraciones generales de los tipos de datos son usualmente hechas en bibliotecas.

Bibliotecas y Paquetes

Las bibliotecas estándar en VHDL son *std* y *ieee*. A continuación se enlistan sus paquetes principales.

- Bibliotecas *std*
  - Paquete *standard*. Contiene la definición de varios tipos de datos y varias operaciones lógicas.
  - Paquete *textio*. Define el manejo de textos y archivos.

- Bibliotecas *ieee*
  - Paquete *std_logic_1164*. Define nueve valores para los tipos de datos STD_ULOGIC y STD_LOGIC.
  - Paquete *numeric_std*. Introduce los tipos SIGNED y UNSIGNED junto con sus operadores, teniendo como base el tipo de dato STD_LOGIC.

Bibliotecas y Paquetes

- Paquete *std_logic_arith*. Define los tipos SIGEND y UNSIGNED junto con sus operadores. Este paquete es parcialmente equivalente a *numeric_std*.
- Paquete *std_logic_unsigned*. Introduce funciones que permiten operaciones aritméticas, corrimientos y comparación con señales de tipo STD_LOGIC_VECTOR que actúan como números sin signo.
- Paquete *std_logic_signed*. Las misma funcionalidad que en el caso anterior pero sobre señales STD_LOGIC_VECTOR que actúan como números con signo.

Nota. Los tres últimos paquetes no son parte del estándar. Sin embargo son ampliamente usados.

Bibliotecas y Paquetes

Para hacer visible un paquete en un diseño, son necesarias dos declaraciones, una corresponde a la biblioteca a la que pertenece y la otra es incluirlos usan la palabra reservada *use*.

Sintaxis

```
LIBRARY library_name;
USE library_name.package_name.all
```

Bibliotecas y Paquetes

Ejemplo

```
LIBRARY std;                        -- Declaración opcional
USE std.standard.all;               -- Declaración opcional
LIBRARY work;                       -- Declaración opcional
USE work.all                        -- Declaración opcional
LIBRARY ieee;
USE ieee.std_logic_1164.all;
USE work.my_package;
```

Nota. En VHDL el punto y coma (;) indica el fin de una declaración o sentencia, mientras que el doble guion (--) indica comentario.

Entidad

La parte principal de una entidad (ENTITY) es PORT, que es una lista de especificaciones de todos los puertos de entrada y salida del circuito.

Sintaxis

```
ENTITY entity_name IS
        PORT(
                port_name: port_mode signal_type;
                port_name: port_mode signal_type;
                …
        );
END [ENTITY] [entiy_name];
```

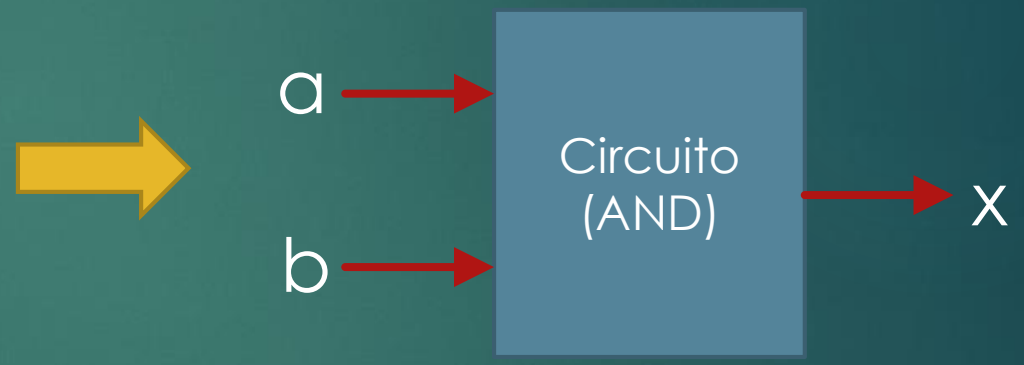
Entidad

- El nombre de la entidad (ENTITY) puede ser cualquiera palabra excepto las palabras reservadas de lenguaje.
- En la sintaxis los corchetes ([ ]) indican que el campo es opcional.
- Todos los miembros del campo PORT de la sintaxis antes mostrada son señales. Es decir, cables que entran y salen del circuito.
- Los puertos pueden ser IN (entrada), OUT (salida) y INOUT (entrada/salida).
- El tipo de señal puede ser BIT, INTEGER, STD_LOGIC, etc.

Entidad

Ejemplo

```
ENTITY compuerta_and IS
        PORT(
                a: in std_logic;
                b: in std_logic;
                x: out std_logic;
        );
END ENTITY;
```



Arquitectura

La arquitectura (ARCHITECTURE) contiene una descripción de como el circuito debe funcionar.

Sintaxis

```
ARCHITECTURE architecture_name OF entity_name IS
      [architecture_declarative_part]
BEGIN
      architecture_statements_part
END [ARCHITECTURE] [architecture_name];
```

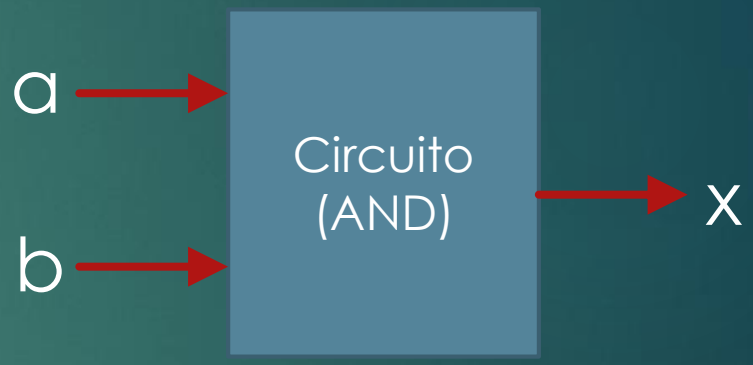
Arquitectura

- La arquitectura (ARCHITECTURE) tiene dos partes: un parte declarativa (opcional) y otra de sentencias (código) que inicia a partir de BEGIN.
- La parte declarativa puede contener las siguientes declaraciones: subprogramas, tipos, señales, variables, archivos, atributos, entre otros.
- La parte de sentencias son donde las sentencias VHDL (código) se colocan.
- El nombre de la arquitectura puede ser cualquiera (excepto palabras reservadas), incluso el mismo nombre de la entidad.

Arquitectura

Ejemplo

```
ARCHITECTURE archi OF compuerta_and IS
BEGIN
        x <= a AND b;
END ARCHITECTURE;
```



El circuito que describe la arquitectura archi es una compuerta AND de dos entradas.

Código VHDL

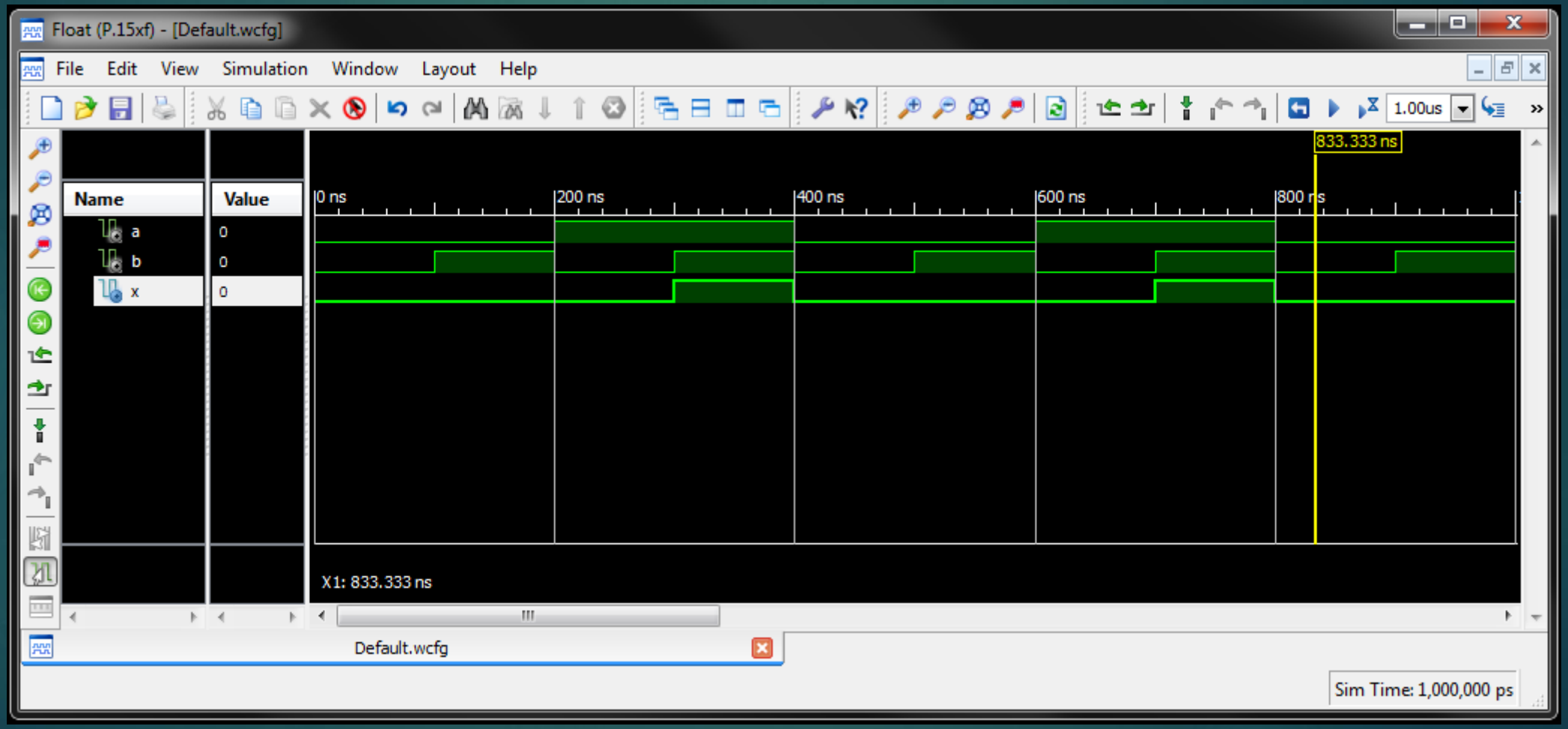
```
 1  ----------------------------------------
 2  library IEEE;
 3  use IEEE.STD_LOGIC_1164.ALL;
 4  ----------------------------------------
 5  entity compuerta_and is
 6      Port ( a : in  STD_LOGIC;
 7             b : in  STD_LOGIC;
 8             x : out STD_LOGIC
 9           );
10  end compuerta_and;
11  ----------------------------------------
12  architecture archi of compuerta_and is
13
14  begin
15
16     x <= a AND b;
17
18  end archi;
```

Bibliotecas/Paquetes

Entidad

Arquitectura

Código VHDL - Simulación



GENERIC

- Las declaraciones GENERIC permiten la especificación de parámetros genéricos. Por ejemplo, una constante genérica que puede ser fácilmente modifica para diferentes aplicaciones.

- Su propósito es parametrizar el diseño, brindándole al código más flexibilidad y reusabilidad.

- GENERIC es la única declaración que se permite en la entidad (ENTITY) y va antes de PORT, lo que causa que las constantes sean globales. Las constantes declaras en GENERIC se pueden usar en PORT.

GENERIC

Sintaxis

```
GENERIC (
    constant_name: constant_type := constant_value;
    constant_name: constant_type := constant_value;
    ...
    );
```

Nota. Si un COMPONENT contiene una declaración GENERIC y se instancia en otro diseño, los valores de las constantes genéricas de la componente instanciada pueden sobrescribirse a través del diseño principal.

GENERIC

Ejemplo

```
ENTITY my_entity IS
	GENERIC(
		m: INTEGER := 8;
	);
	PORT(
		a: in   std_logic_vector(m-1 downto 0);
		x: out std_logic_vector(m-1 downto 0)
	);
```