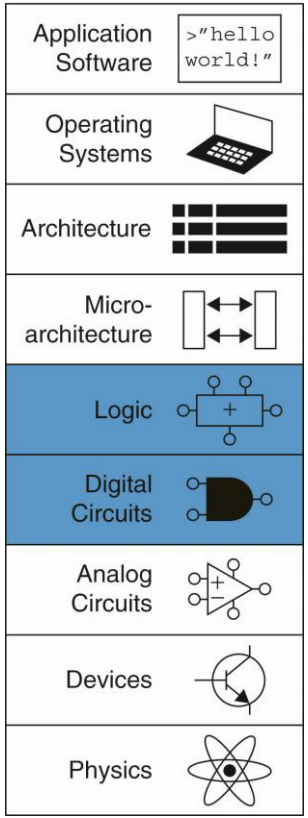
Introducción a VHDL (I)

UNIVERSIDAD NEBRIJA

VHDL

- VHDL lenguaje para descripción hardware usado en la industria en todo el mundo.
  - **VHDL** acrónimo de **V**HSIC (**V**ery **H**igh **S**peed **I**ntegrated **C**ircuit) **H**ardware **D**escription **L**anguage



Génesis de VHDL

Estado del arte en torno a 1980

- Múltiples métodos de diseño y lenguajes de descripción hardware en uso
- Portabilidad de diseño entre herramientas CAD de diferentes vendedores nula o limitada
- Objetivo: acortar el tiempo desde el concepto de diseño a la implementación de 18 a 6 meses

Historia breve de VHDL

- Junio 1981: Woods Hole Workshop
- Julio 1983: contrato para desarrollar VHDL
  - Intermetrics
  - IBM
  - Texas Instruments
- Agosto 1985: VHDL Version 7.2 liberada
- Diciembre 1987:

  VHDL se convierte en un estándar de IEEE: Standard 1076-1987 y en 1988 en estándar de ANSI.

Versiones de VHDL

- Versiones de VHDL:
  - IEEE-1076 1987
  - IEEE-1076 1993 (más implementada en herramientas CAD)
  - IEEE-1076 2000 (cambios menores)
  - IEEE-1076 2002 (cambios menores)
  - IEEE-1076 2008

IEEE Extensions

- IEEE standard 1076.1 Analog and Mixed Signal Extensions (VHDL-AMS)
- IEEE standard 1076.2 VHDL Mathematical Packages
- IEEE standard 1076.3 Synthesis Packages
- IEEE standard 1076.4 VHDL Initiative Towards ASIC Libraries (VITAL)
- IEEE standard 1076.6 VHDL Register Transfer Level (RTL) Synthesis
- IEEE standard 1164 Multivalue Logic System for VHDL Model Interoperability
- IEEE standard 1029 VHDL Waveform and Vector Exchange to Support Design and Test Verification (WAVES)

UNIVERSIDAD
NEBRIJA

Verilog

- Más simple y sintácticamente diferente
  - C-like

- Gateway Design Automation Co., 1985
- Gateway comprador por Cadence en 1990
- IEEE Standard 1364-1995
- Estándar *de facto* para programación ASIC

- Interfaz de lenguaje de programación para permitir conectar con código no Verilog.

VHDL vs. Verilog

Desarrollado por el gobierno	Desarrollado comercialmente
Basado en Ada	Basado en C
Tipado fuerte	Tipado medio
Insensible a mayúsculas	Sensible a mayúsculas
Difícil de aprender	Fácil de aprender
Más potente	Menos potente

Características de VHDL y Verilog

- Independiente de vendedor y tecnología
- Portable
- Reusable

Nombrado y Etiquetado (1)

- VHDL es insensible a mayúsculas (Case-insensitive)

  Ejemplo:

  Los nombres o etiquetas

  `databus`
  `Databus`
  `DataBus`
  `DATABUS`

  son equivalentes.

Nombrado y Etiquetado(2)

Reglas genéricas (según VHDL-87)

1. Todos los nombres deberían empezar con un carácter alfabético (a-z o A-Z)
2. Usar solo caracteres alfabéticos (a-z or A-Z), dígitos (0-9) y el carácter subrayado (_).
3. No usar ninguna puntuación o caracteres reservados dentro de un nombre (!, ?, ., &, +, -, etc.)
4. No usar dos o más subrayados consecutivos dentro de un nombre (por ejemplo Sel__A es inválido)
5. Todos los nombres y etiquetas en una entidad y arquitectura dada deben ser únicos.

Nombres reservados

A
ABS
ACCESS
AFTER
ALIAS
ALL
AND
ARCHITECTURE
ARRAY
ASSERT
ATTRIBUTE

B
BEGIN
BLOCK
BODY
BUFFER
BUS

C
CASE
COMPONENT
CONFIGURATION
CONSTANT

D
DISCONNECT
DOWNTO

E
ELSE
ELSIF
END
ENTITY
EXIT

F
FILE
FOR
FUNCTION

G
GENERATE
GENERIC
GUARDED

I
IF
IN
INOUT
IS

L
LABEL
LIBRARY
LINKAGE
LOOP

M
MAP
MOD

N
NAND
NEW
NEXT
NOR
NOT
NULL

O
OF
ON
OPEN
OR
OTHERS
OUT

P
PACKAGE
PORT
PROCEDURE
PROCESS

R
RANGE
RECORD
REGISTER
REM
REPORT
RETURN

S
SELECT
SEVERITY
SIGNAL
SUBTYPE

T
THEN
TO
TRANSPORT
TYPE

U
UNITS
UNTIL
USE

V
VARIABLE

W
WAIT
WHEN
WHILE
WITH

X
XOR

Formato libre

- VHDL en un lenguaje de formato libre (“free format”)

  Los compiladores VHDL no imponen convenciones de formato como espaciado o indentación. Los espacios y los saltos de línea se tratan igual.

  Ejemplo:

```
if (a=b) then
```

o

```
if  (a=b)   then
```

o

```
if (a =
b) then
```

son equivalentes.

Estándares de fiabilidad y estilos de codificación

Hay recomendaciones como:

OpenCores Coding Guidelines

Comentarios

- Los comentarios en VHDL se indican con un guion doble “--”
  - En cualquier sitio de la línea
  - Cualquier texto que lo siga en la línea es un comentario
  - El comentario acaba con el salto de línea

Ejemplo:

```
-- main subcircuit
Data_in <= Data_bus;     -- reading data from the input FIFO
```

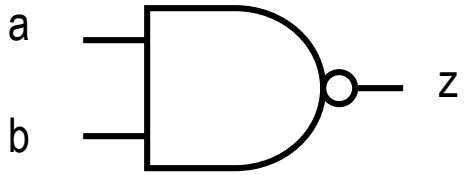
Comentarios

- Para explicar la función de un modulo a otros diseñadores o recordarla para el mantenimiento.
- Explicar, no reescribir lo mismo que dice el código
- Localizado junto al código descrito.

Introducción a VHDL (II)

UNIVERSIDAD NEBRIJA

Ejemplo: NAND Gate



a	b	z
0	0	1
0	1	1
1	0	1
1	1	0

Ejemplo de código VHDL

- 3 secciones en cada pieza de código VHDL
- Extensión de fichero VHDL file .vhd
- El nombre del fichero debería ser el mismo que el nombre de la entidad(nand_gate.vhd) [OpenCores Coding Guidelines]

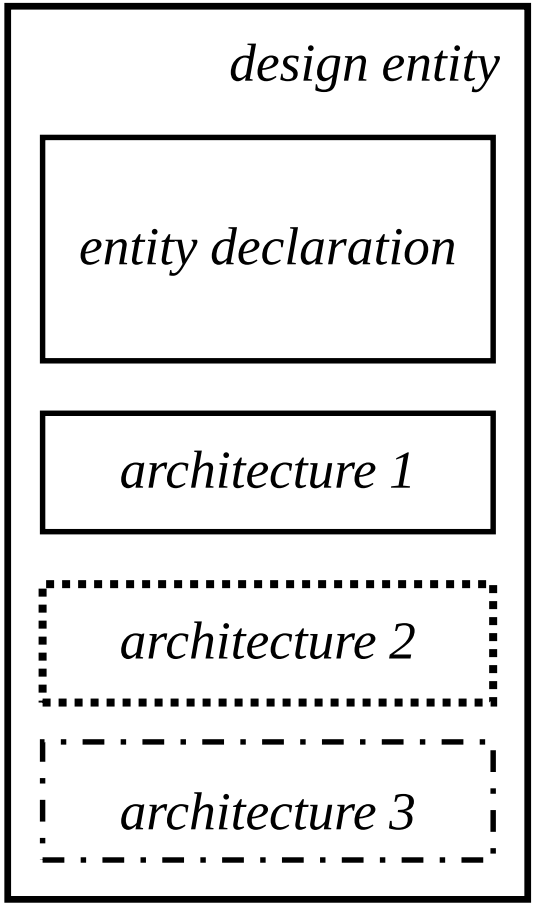
```
LIBRARY ieee;
USE ieee.std_logic_1164.all;


ENTITY nand_gate IS
      PORT(
            a    : IN STD_LOGIC;
            b    : IN STD_LOGIC;
            z    : OUT STD_LOGIC);
END nand_gate;


ARCHITECTURE model OF nand_gate IS
BEGIN
      z <= a NAND b;
END model;
```

LIBRARY DECLARATION

ENTITY DECLARATION

ARCHITECTURE BODY

Entidad de diseño (Design entity)

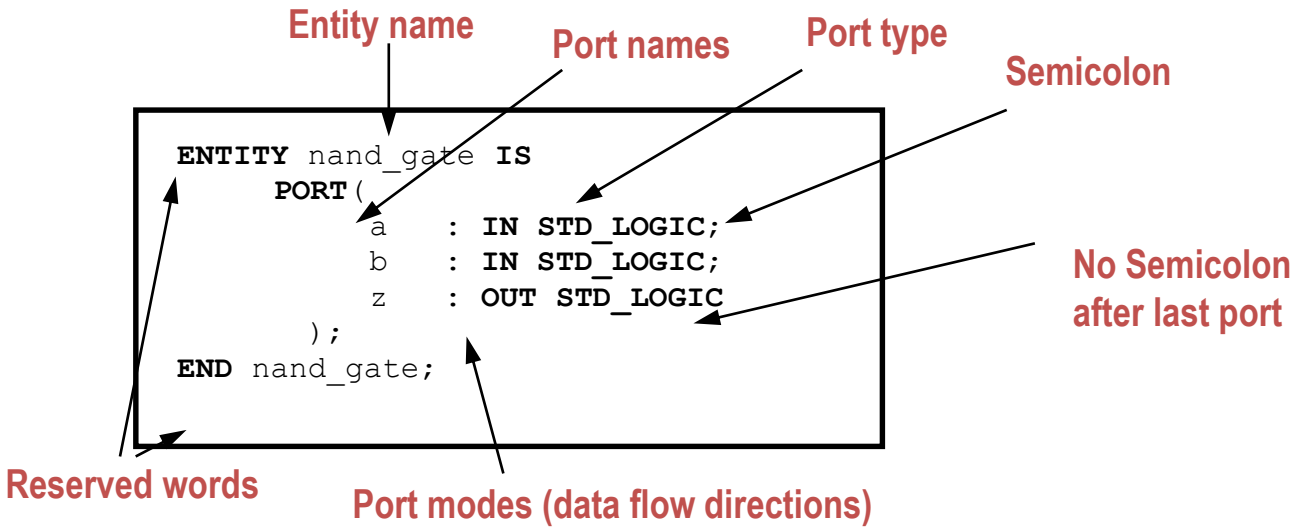


Design Entity – bloque más básico de un diseño.

Una entidad puede tener muchas arquitecturas diferentes.

Declaración de Entity

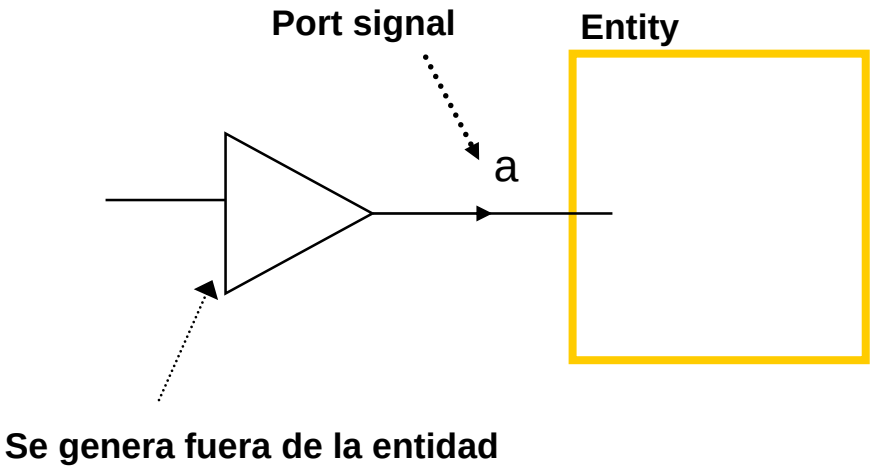
- Entity describe la interfaz de los componentes, es decir, puertos de entrada y salida.



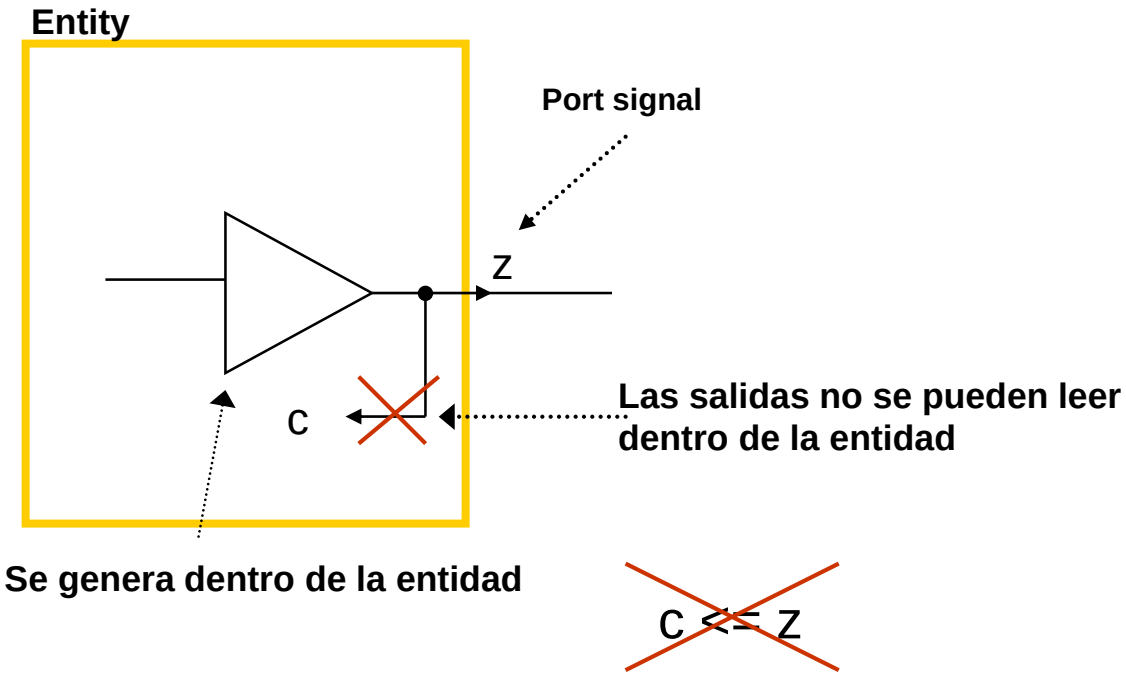
Sintaxis simplificada

```
ENTITY entity_name IS
   PORT (
      port_name :  port_mode  signal_type;
      port_name :  port_mode  signal_type;
      ……..
      port_name :  port_mode  signal_type);
END entity_name;
```

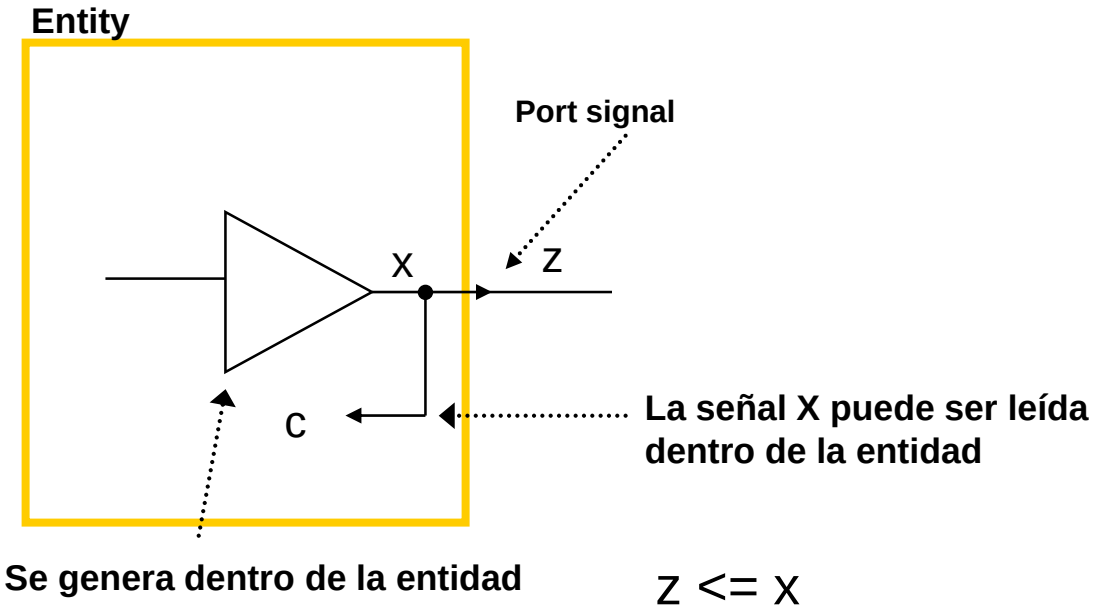
Port Mode IN



Port Mode OUT

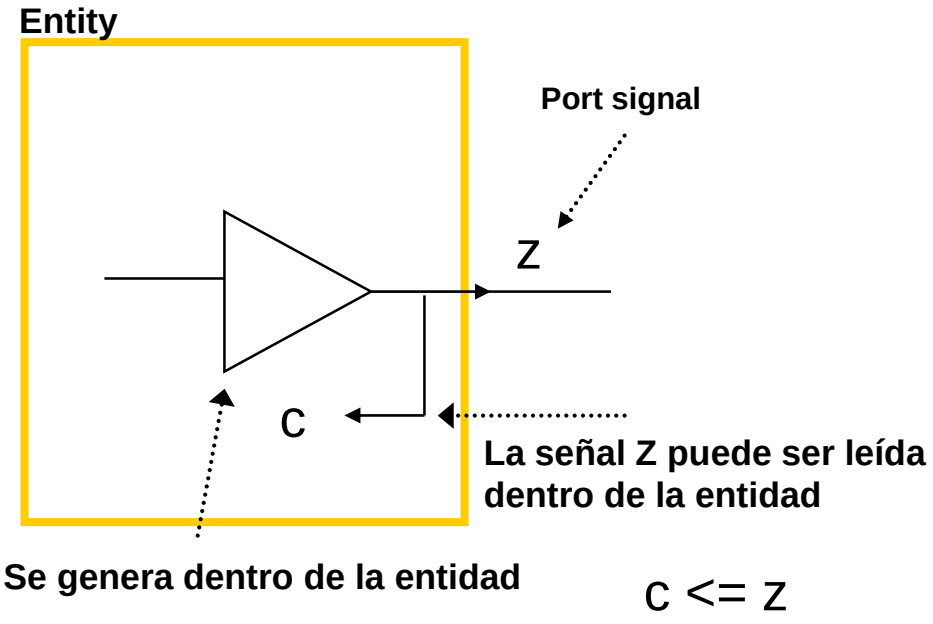


Port Mode OUT (con señal extra)



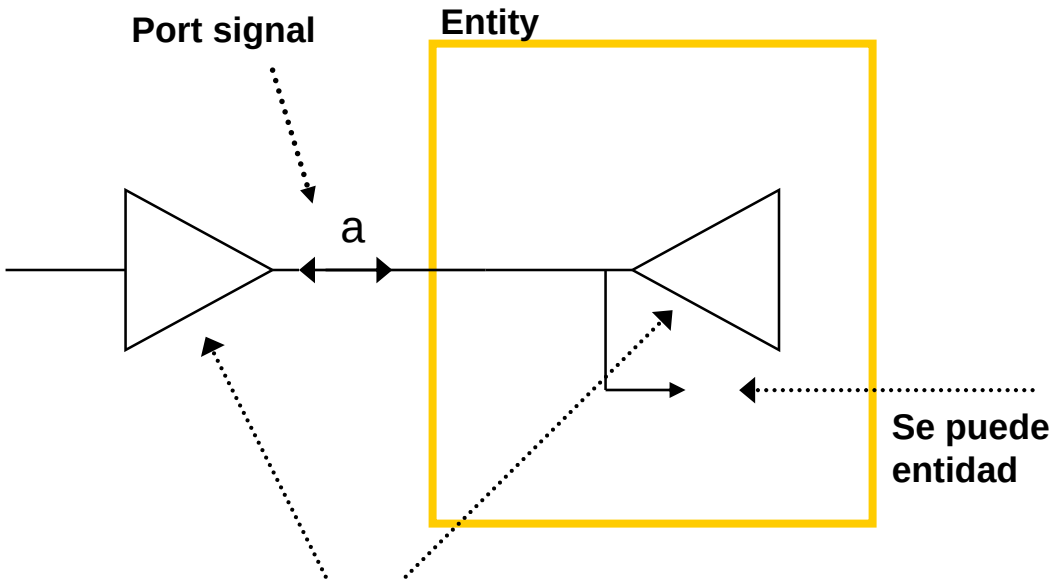

```
z <= x
c <= x
```

Port Mode BUFFER



No recomendado por problemas a la hora de sintetizar.

Port Mode INOUT



Port Modes - Resumen

Describe la dirección en que viaja la información respecto al componente

- **In**: La información entra a través del puerto y solo se puede leer. Aparece solo en la parte derecha de una asignación de variable o señal.
- **Out**: el valor de un puerto de salida solo puede se actualizado dentro de la entidad. No puede ser leído. Solo puede aparecer en el lado izquierdo de una asignación a una señal.
- **Inout**: El valor de un puerto bidireccional se puede leer y actualizar en la entidad. Puede aparecer a ambos lados de la asignación.
- **Buffer:** No se recomienda su uso.

Architecture

- Describe una implementación de una entidad de diseño.
- Ejemplo

```
ARCHITECTURE model OF nand_gate IS
BEGIN
      z <= a NAND b;
END model;
```

Architecture – sintaxis simplificada

```
ARCHITECTURE architecture_name OF entity_name IS
    [ declarations ]
BEGIN
     code
END architecture_name;
```

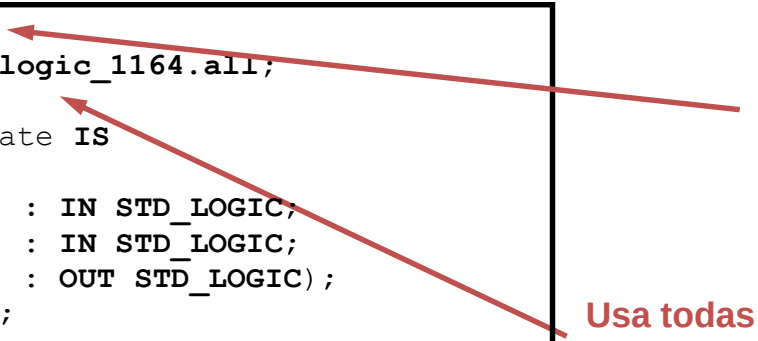
Declaración y arquitectura

nand_gate.vhd

```
LIBRARY ieee;
USE ieee.std_logic_1164.all;

ENTITY nand_gate IS
     PORT(
          a    : IN STD_LOGIC;
          b    : IN STD_LOGIC;
          z    : OUT STD_LOGIC);
END nand_gate;

ARCHITECTURE dataflow OF nand_gate IS
BEGIN
     z <= a NAND b;
END dataflow;
```

Declaraciones de bibliotecas/librerías

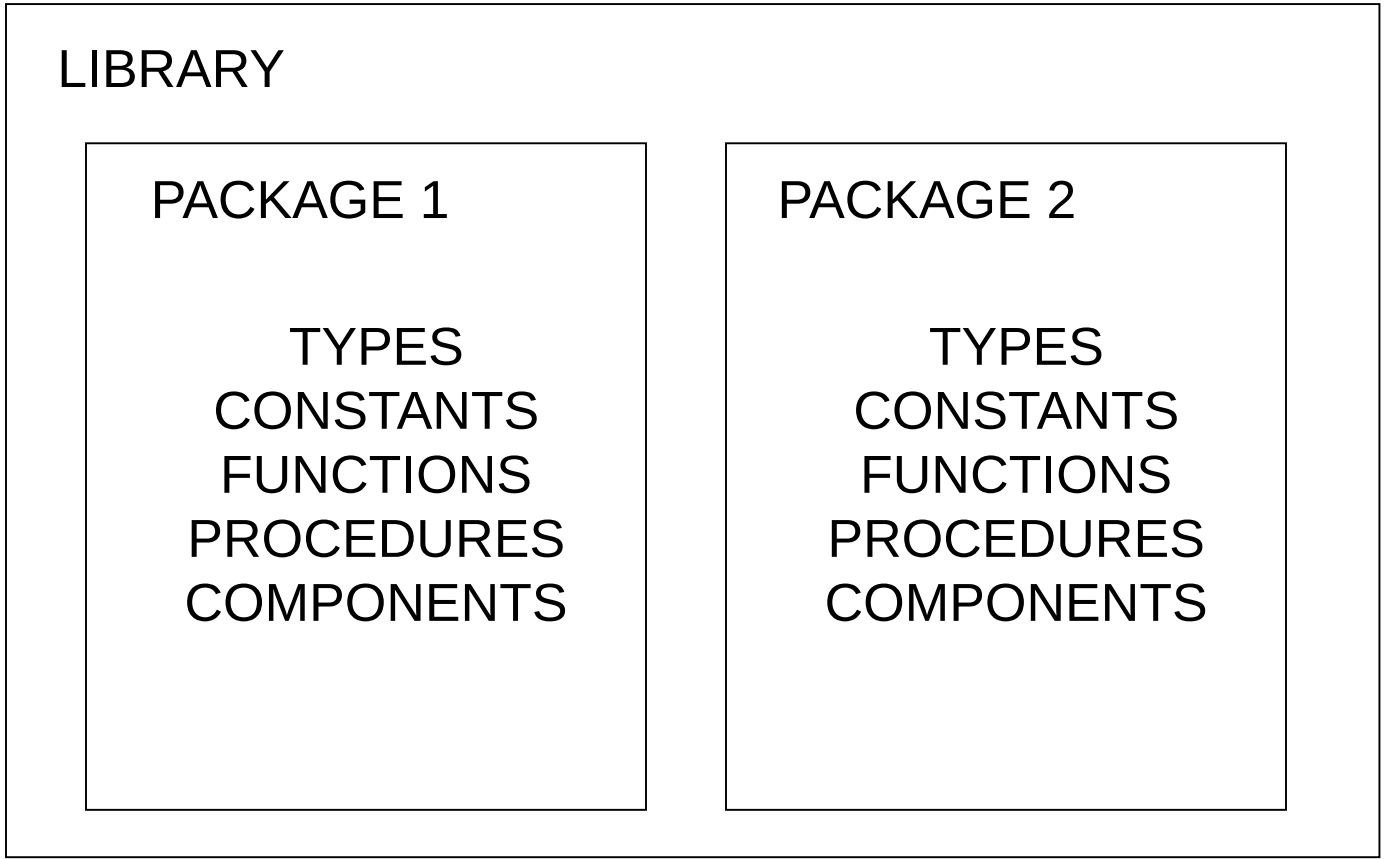
```
LIBRARY ieee;
USE ieee.std_logic_1164.all;

ENTITY nand_gate IS
     PORT(
          a    : IN STD_LOGIC;
          b    : IN STD_LOGIC;
          z    : OUT STD_LOGIC);
END nand_gate;

ARCHITECTURE model OF nand_gate IS
BEGIN
     z <= a NAND b;
END model;
```



Library - syntaxis

```
LIBRARY  library_name;
USE library_name.package_name.package_parts;
```

Partes fundamentales de una librería



- ieee

  Especifica lógica multi-nivel, Incluidos los tipos STD_LOGIC, y STD_LOGIC_VECTOR

Ha de ser declarada explícitamente

- std

  Especifica tipos de datos predefinidos (BIT, BOOLEAN, INTEGER, REAL, SIGNED, UNSIGNED, etc.), operaciones aritméticas, funciones de conversión de tipos básicos, funciones de entrada/salida básica, etc.

- work

  Guarda el diseño actual al compilarlo

Visible por defecto