Integrated Circuits Design by FPGA

م.م. أحمد مؤيد عبدالحسين

جامعة الفرات الأوسط التقنية / الكلية التقنية الهندسية / نجف

Lecture 10

Packages and Components

Objectives of this Lecture

- To define the new terms **Packages** and **Components**
- To understand the code structure of **Packages** and **Components**.
- To implement examples using **Packages** and **Components**.

Contents of this Lecture

- Introduction
- PACKAGE
- COMPONENT

Introduction

- Code structure: library declarations, entity, architecture (Lecture 1)
- VHDL Data Classes and Data Types (Lecture 2)
- VHDL Parallel Code (Lecture 4)
- VHDL Sequential Code (Lecture 5 & Lecture 6)
- Design of Finite State Machines (FSM) (Lecture 8)

Introduction

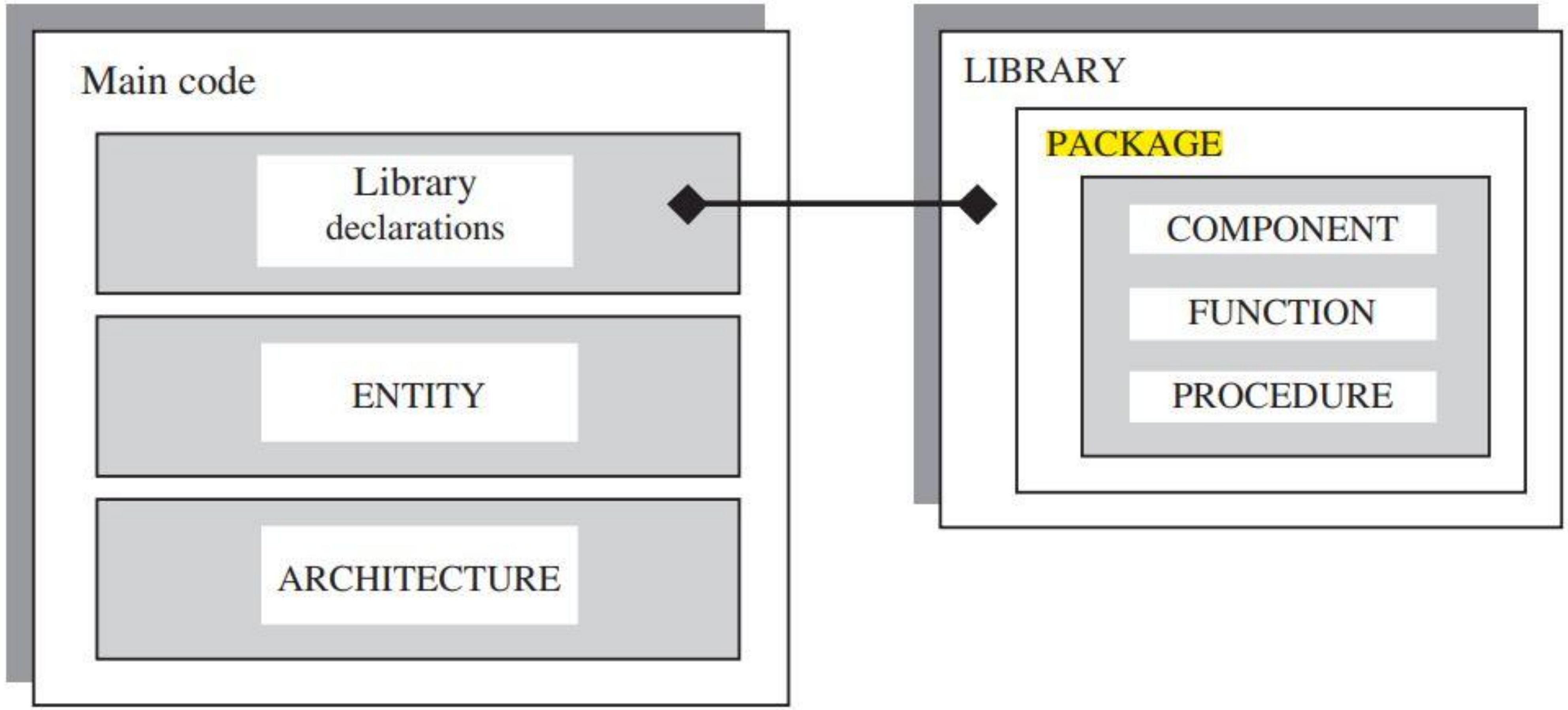


Figure 10.1
Fundamental units of VHDL code.

Introduction

- Packages
- Components

Lecture 10 & 11

- Functions
- Procedures

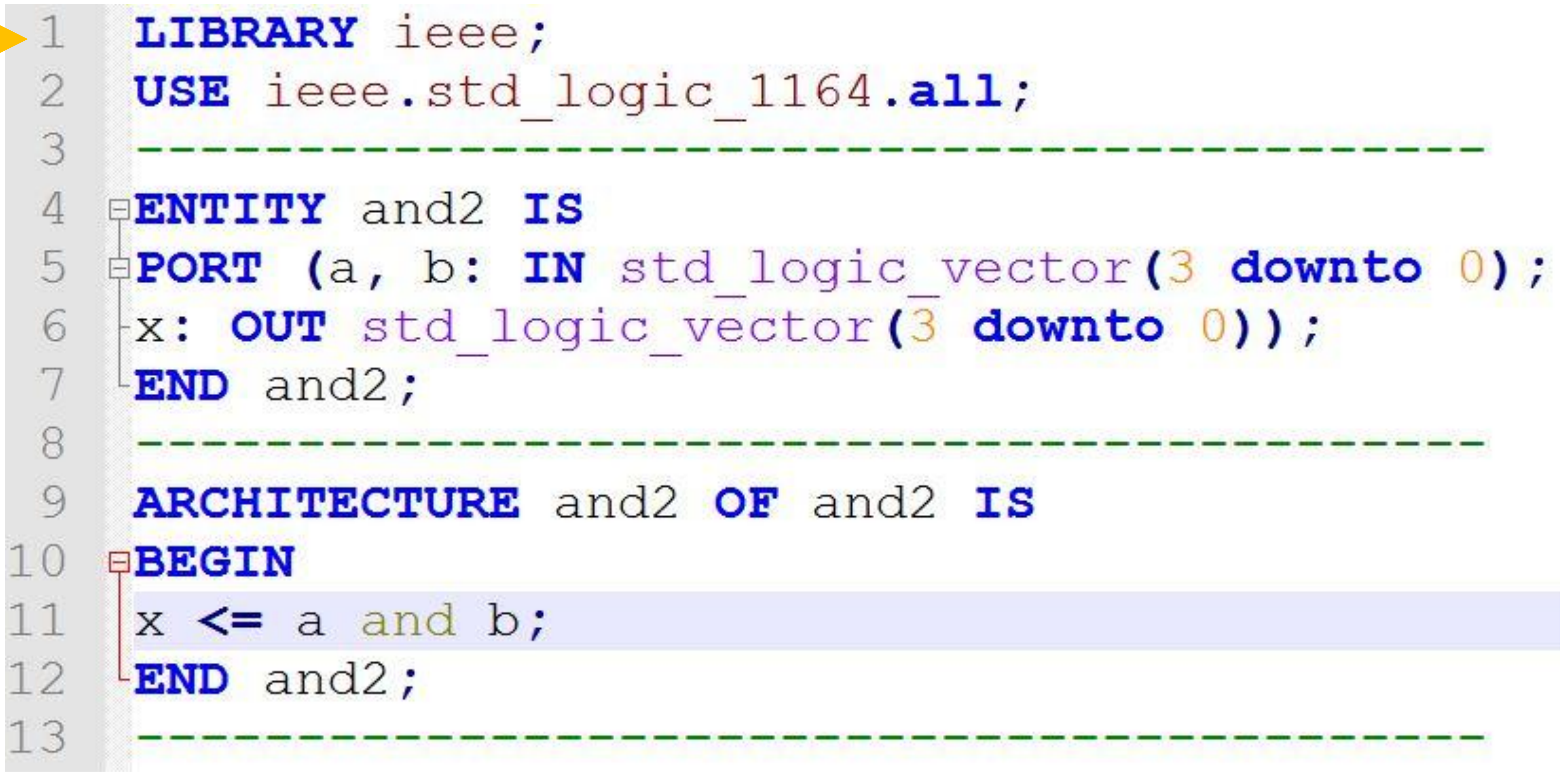
Lecture 12 & 13

Introduction

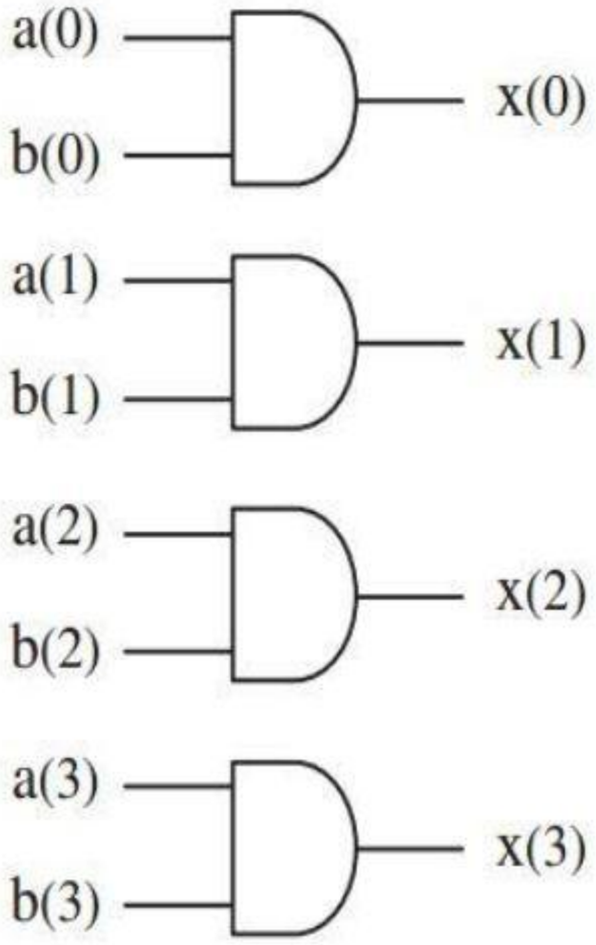
- These new units can be located in the main code itself (that is, on the left-hand side of figure 10.1)
- However, since their main purpose is to allow common pieces of code to be reused and shared, it is more usual to place them in a LIBRARY.
- This also leads to code partitioning, which is helpful when dealing with long codes.
- In summary, frequently used pieces of code can be written in the form of COMPONENTS, FUNCTIONS, or PROCEDURES, then placed in a **PACKAGE**, which is finally compiled into the destination LIBRARY.
- We have already seen that at least two LIBRARIES are generally needed in a design: **ieee**, and **work.**
- After studying lectures 10 to 13, we will be able to construct our own libraries, which can then be added to the list above.

Introduction

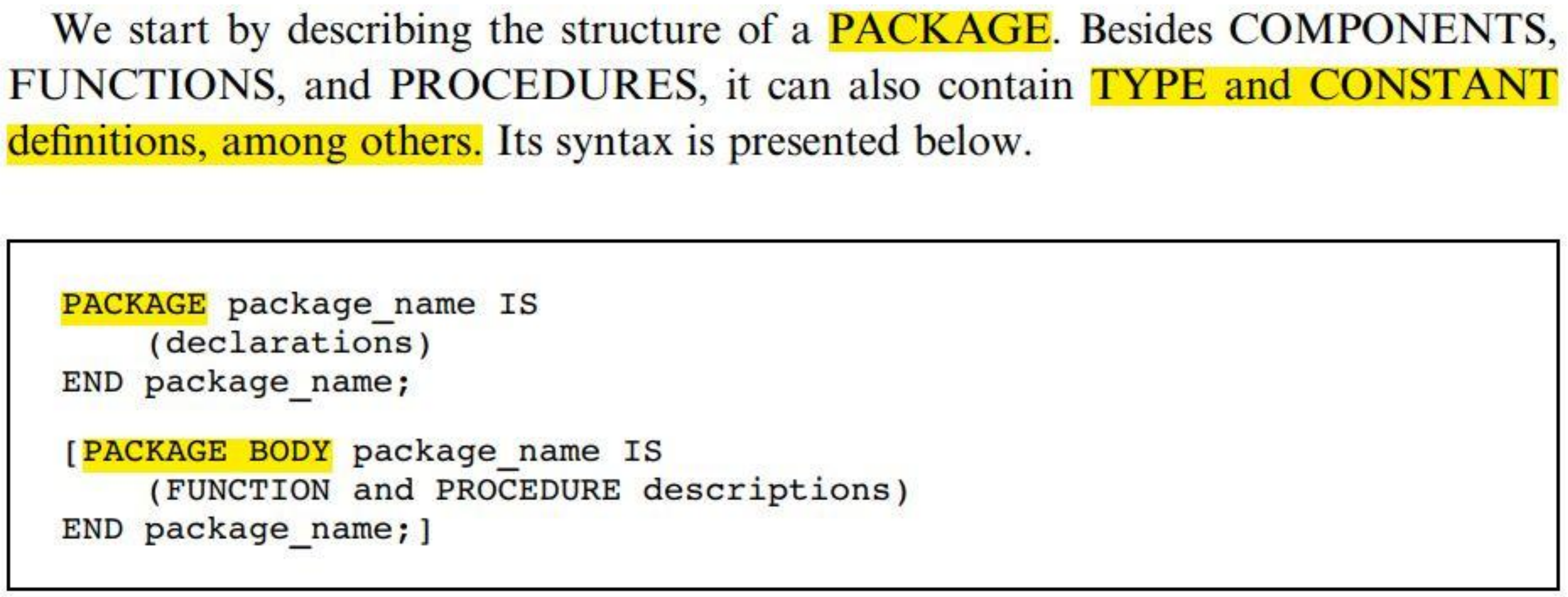
- For example, commonly used circuits, like flip-flops, multiplexers, adders, logic gates, etc., can be placed in a LIBRARY, so any project can make use of them without having to explicitly rewrite such codes.



```
LIBRARY ieee;
USE ieee.std_logic_1164.all;
------------------------------------------
ENTITY and2 IS
PORT (a, b: IN std_logic_vector(3 downto 0);
x: OUT std_logic_vector(3 downto 0));
END and2;
------------------------------------------
ARCHITECTURE and2 OF and2 IS
BEGIN
x <= a and b;
END and2;
------------------------------------------
```



PACKAGE



We start by describing the structure of a PACKAGE. Besides COMPONENTS, FUNCTIONS, and PROCEDURES, it can also contain TYPE and CONSTANT definitions, among others. Its syntax is presented below.

```
PACKAGE package_name IS
    (declarations)
END package_name;

[PACKAGE BODY package_name IS
    (FUNCTION and PROCEDURE descriptions)
END package_name;]
```

As can be seen, the syntax is composed of two parts: PACKAGE and PACKAGE BODY

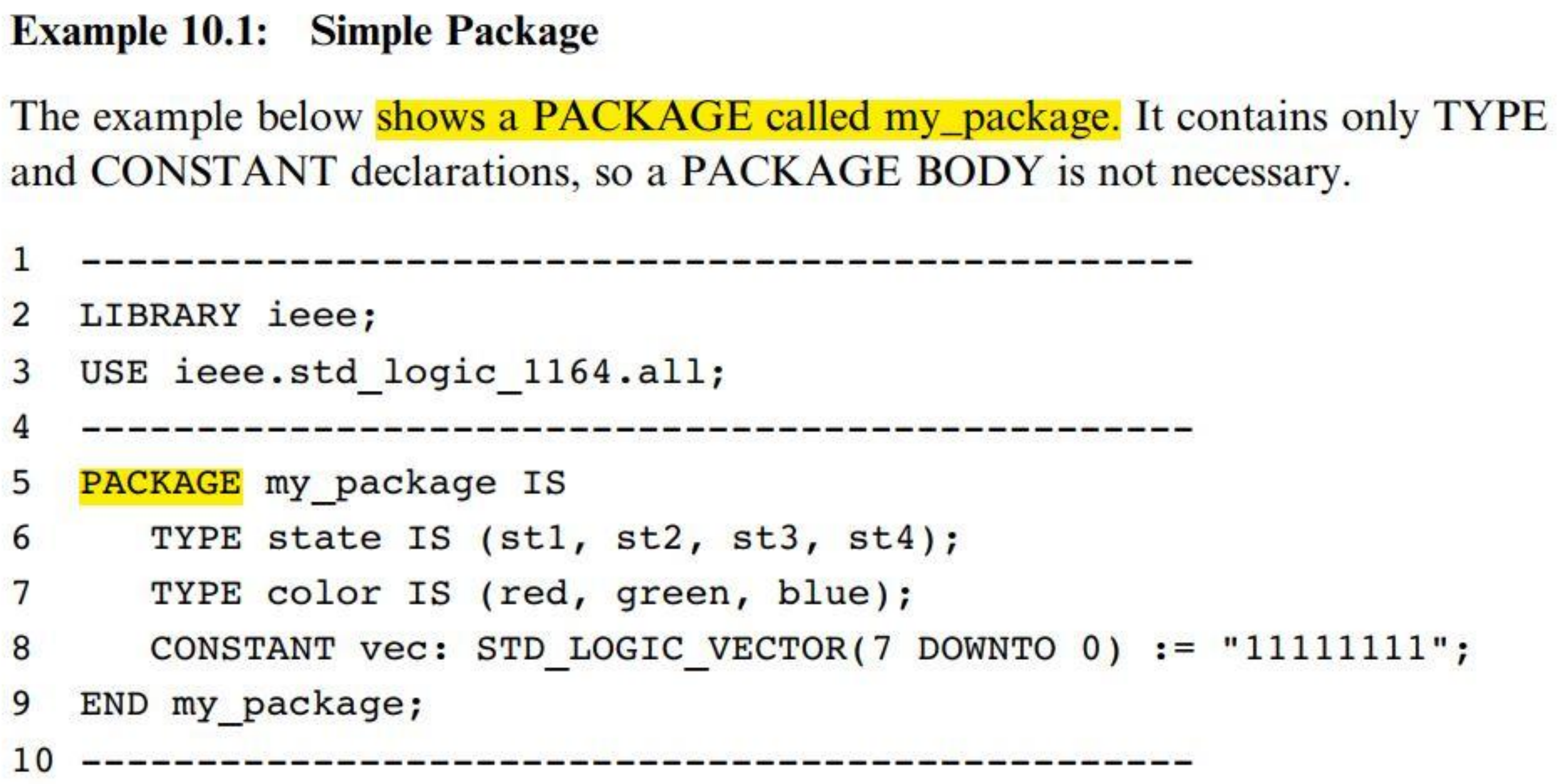
PACKAGE

- The first part (PACKAGE) is mandatory and contains all declarations, while the second part (PACKAGE BODY) is necessary only when one or more subprograms (FUNCTION or PROCEDURE) are declared in the first upper part, in which case it must contain the descriptions (bodies) of the subprograms.

- PACKAGE and PACKAGE BODY must have the same name.

- The declarations list can contain the following: COMPONENT, FUNCTION, PROCEDURE, TYPE, CONSTANT, etc.

PACKAGE



Example 10.1: Simple Package

The example below shows a PACKAGE called my_package. It contains only TYPE and CONSTANT declarations, so a PACKAGE BODY is not necessary.

```
---------------------------------------------
LIBRARY ieee;
USE ieee.std_logic_1164.all;
---------------------------------------------
PACKAGE my_package IS
   TYPE state IS (st1, st2, st3, st4);
   TYPE color IS (red, green, blue);
   CONSTANT vec: STD_LOGIC_VECTOR(7 DOWNTO 0) := "11111111";
END my_package;
---------------------------------------------
```

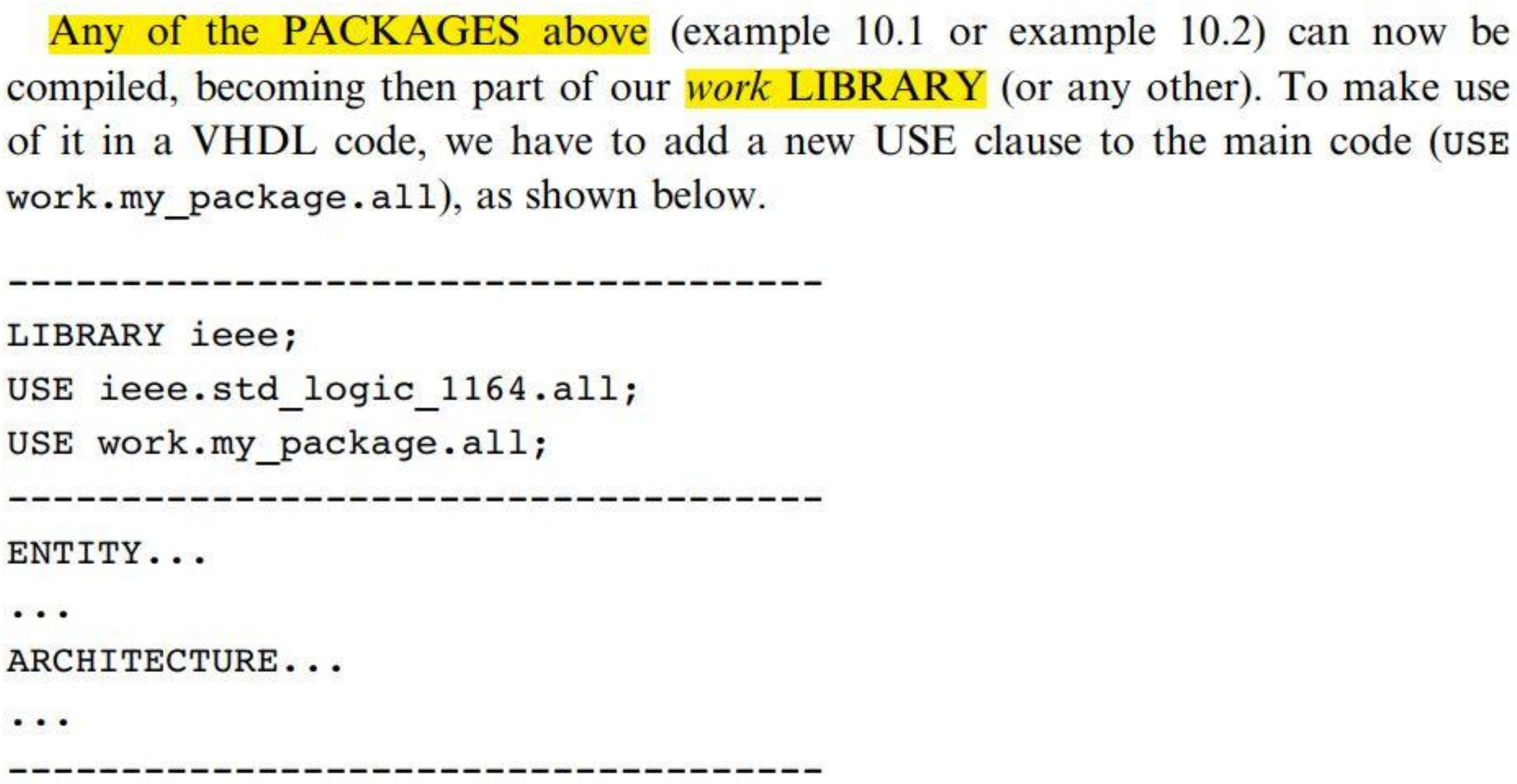
PACKAGE

- The next example (example 10.2) contains, besides TYPE and CONSTANT declarations, a FUNCTION. Therefore, a PACKAGE BODY is now needed (details on how to write a FUNCTION will be seen in lecture 12). This function returns TRUE when a positive edge occurs on clk.

PACKAGE

```
LIBRARY ieee;
USE ieee.std_logic_1164.all;
---------------------------------------------
PACKAGE my_package IS
   TYPE state IS (st1, st2, st3, st4);
   TYPE color IS (red, green, blue);
   CONSTANT vec: STD_LOGIC_VECTOR(7 DOWNTO 0) := "11111111";
   FUNCTION positive_edge(SIGNAL s: STD_LOGIC) RETURN BOOLEAN;
END my package;
---------------------------------------------
PACKAGE BODY my_package IS
   FUNCTION positive_edge(SIGNAL s: STD_LOGIC) RETURN BOOLEAN IS
   BEGIN
      RETURN (s'EVENT AND s='1');
   END positive_edge;
END my_package;
---------------------------------------------
```

PACKAGE



Any of the PACKAGES above (example 10.1 or example 10.2) can now be compiled, becoming then part of our *work* LIBRARY (or any other). To make use of it in a VHDL code, we have to add a new USE clause to the main code (`USE work.my_package.all`), as shown below.

```
------------------------------------
LIBRARY ieee;
USE ieee.std_logic_1164.all;
USE work.my_package.all;
------------------------------------
ENTITY...
...
ARCHITECTURE...
...
------------------------------------
```

COMPONENT

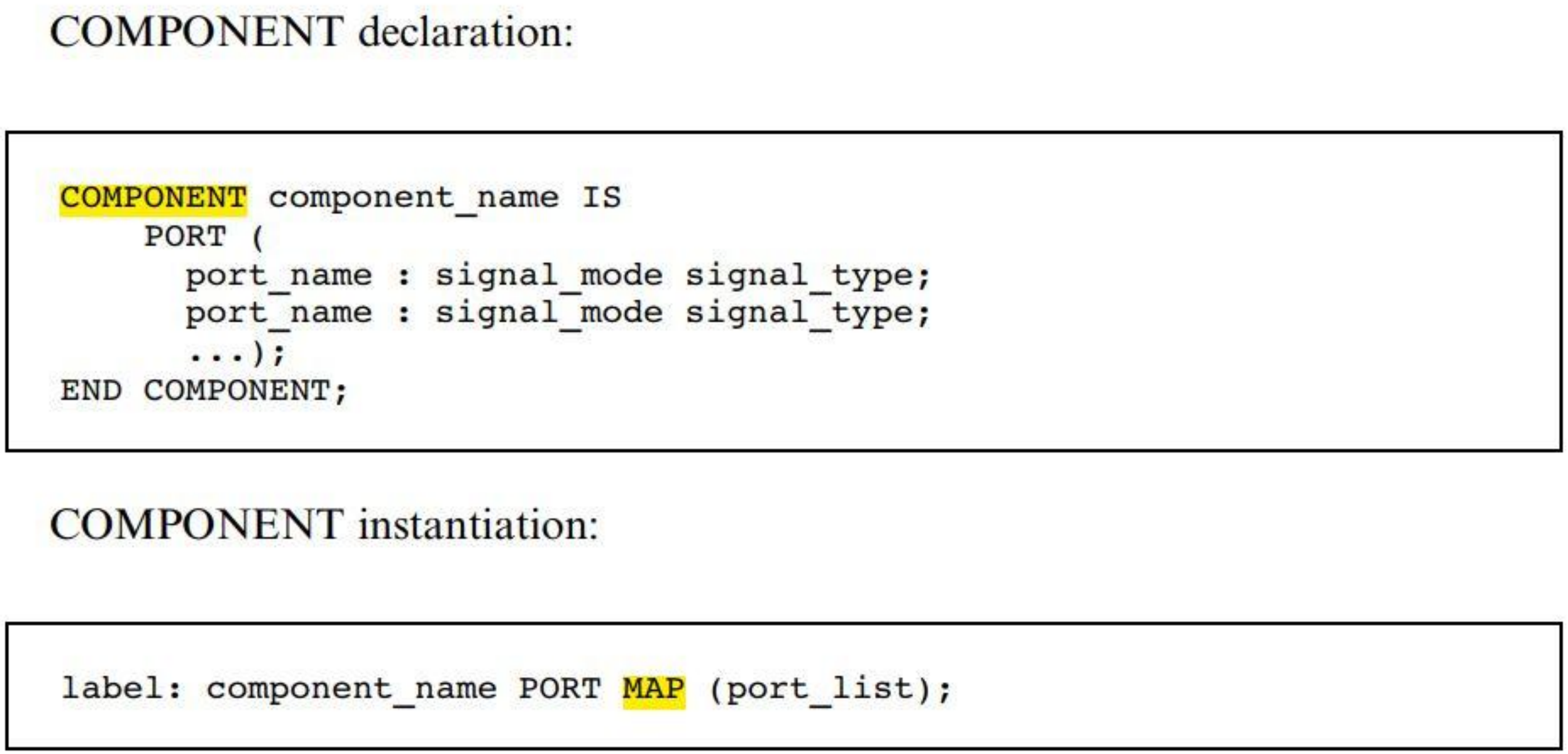
- A **COMPONENT** is simply a piece of conventional code (that is, LIBRARY declarations + ENTITY + ARCHITECTURE, as seen in previous lectures).

- However, by declaring such code as being a **COMPONENT**, it can then be used within another circuit, thus allowing the construction of hierarchical designs.

- A **COMPONENT** is also another way of partitioning a code and providing code sharing and code reuse.

COMPONENT

- To use (instantiate) a COMPONENT, it must first be declared. The corresponding syntaxes are shown below.



COMPONENT declaration:

```
COMPONENT component_name IS
    PORT (
      port_name : signal_mode signal_type;
      port_name : signal_mode signal_type;
      ...);
END COMPONENT;
```

COMPONENT instantiation:

```
label: component_name PORT MAP (port_list);
```

COMPONENT

```
----- COMPONENT declaration: -----------
COMPONENT inverter IS
      PORT (a: IN STD_LOGIC; b: OUT STD_LOGIC);
END COMPONENT;

----- COMPONENT instantiation: -----------
U1: inverter PORT MAP (x, y);
```

COMPONENT

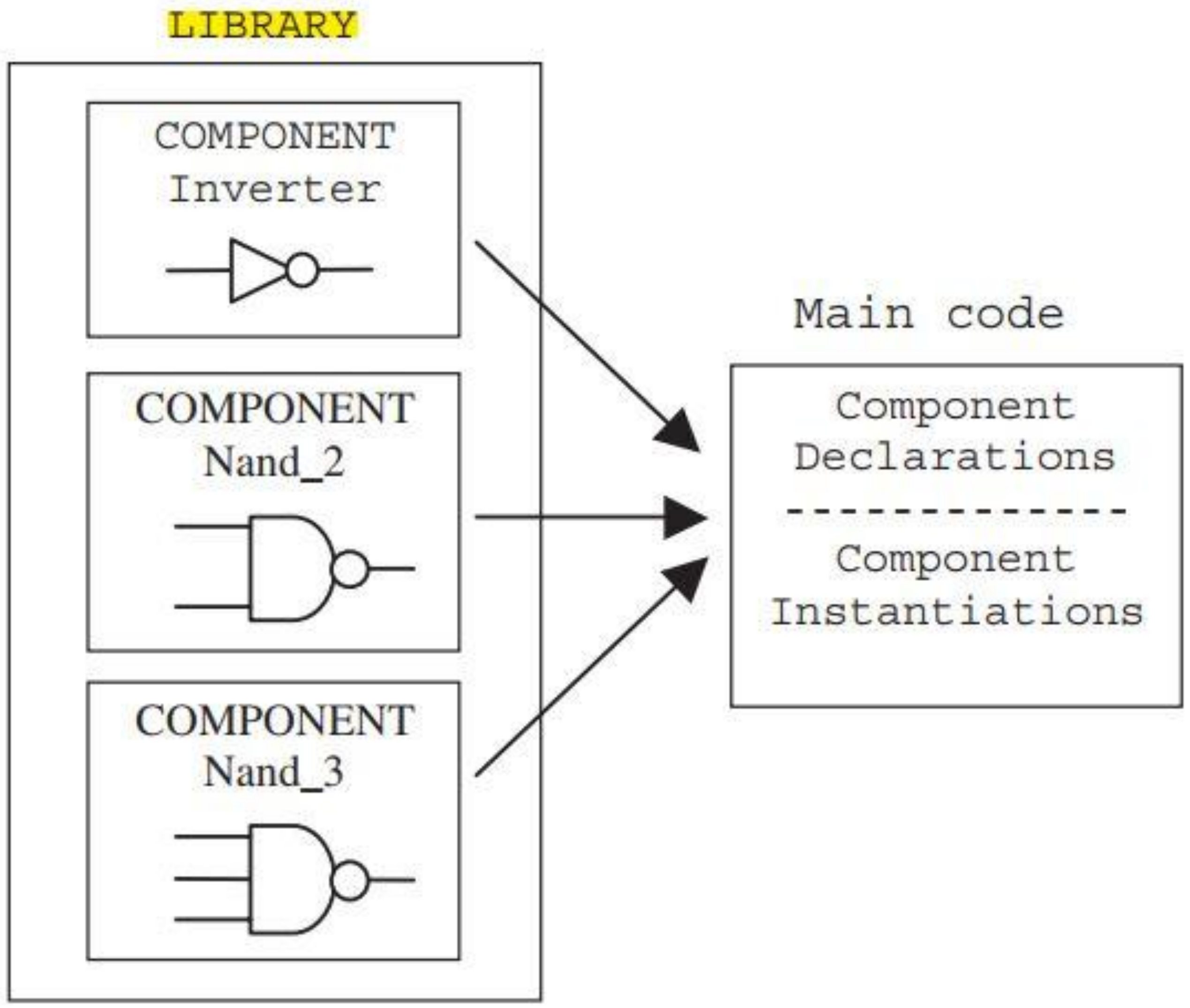


Figure 10.2 a. : Declarations in the main code itself

COMPONENT

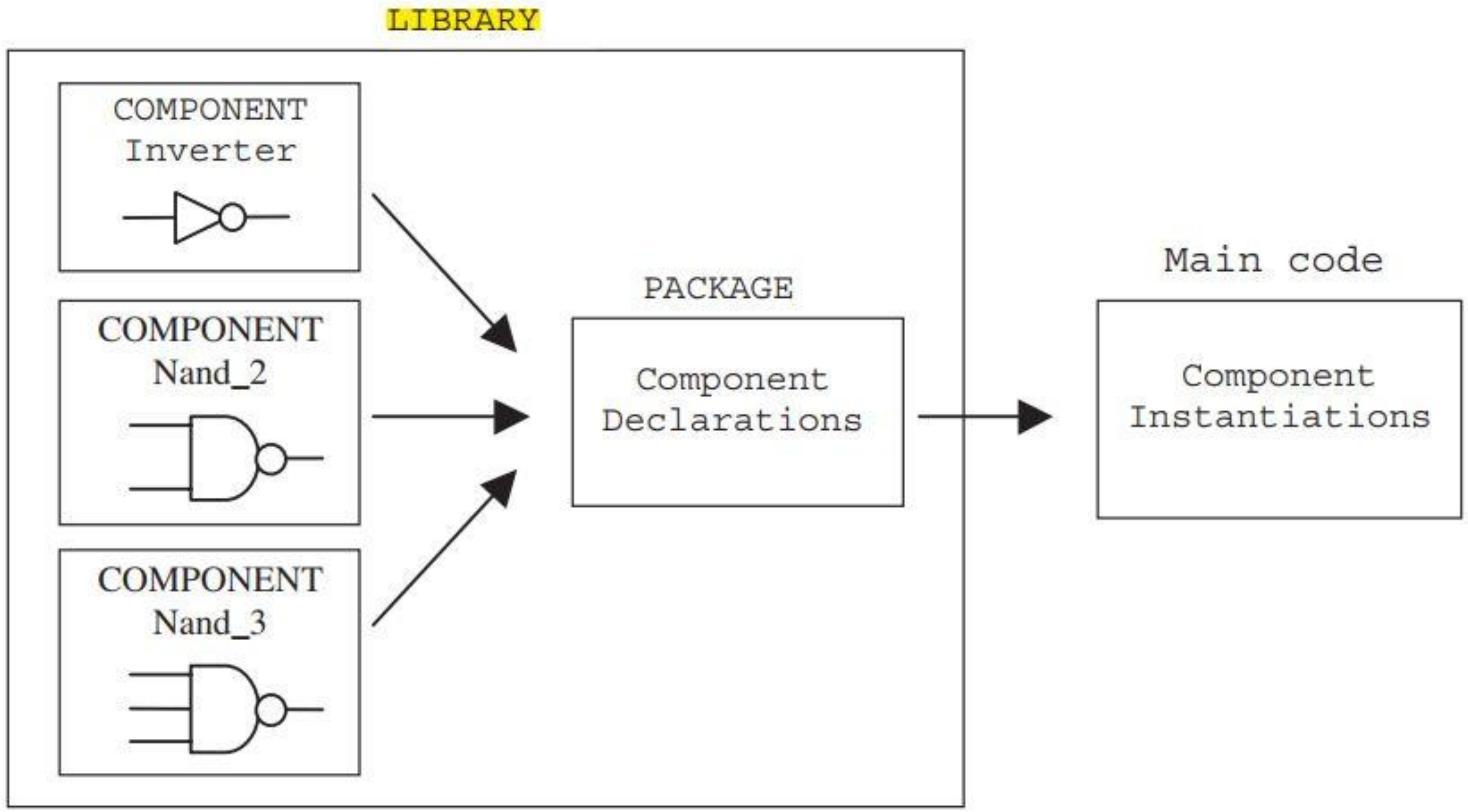


Figure 10.2 b. : Declarations in a PACKAGE

COMPONENT

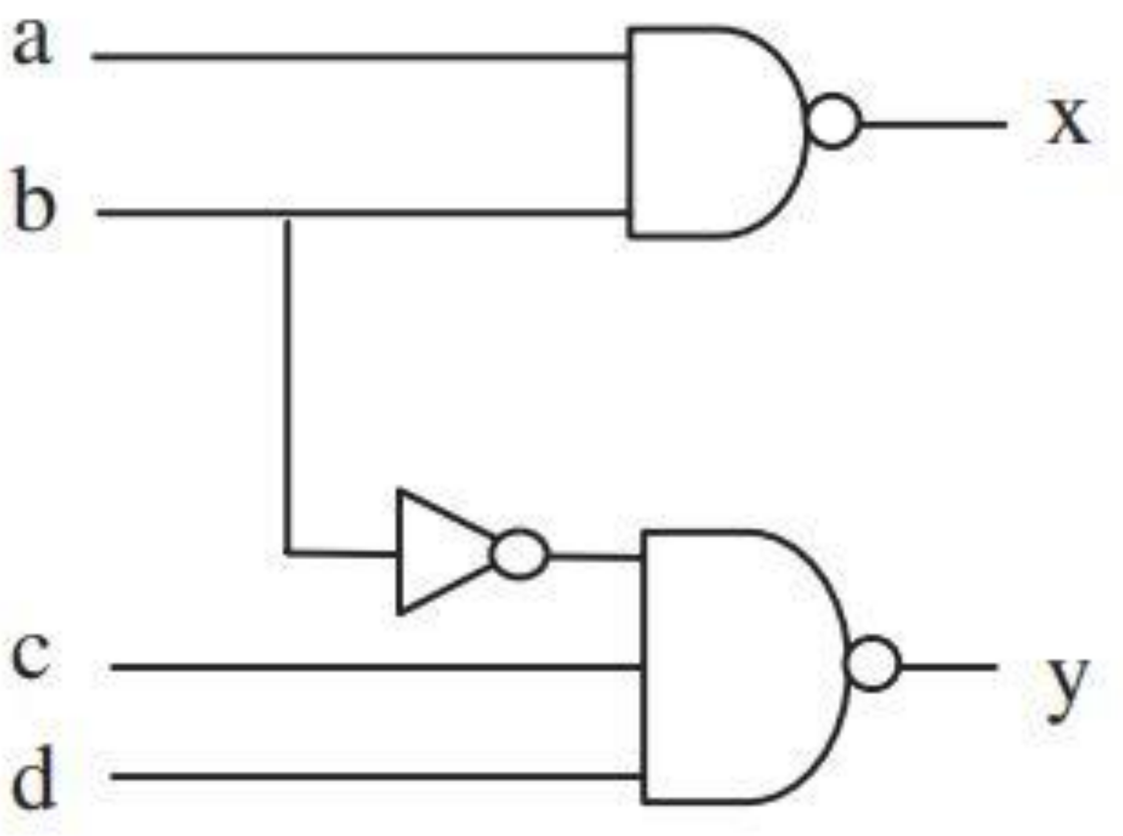


Figure 10.3
Circuit of example 10.3.

Components Declared in the Main Code

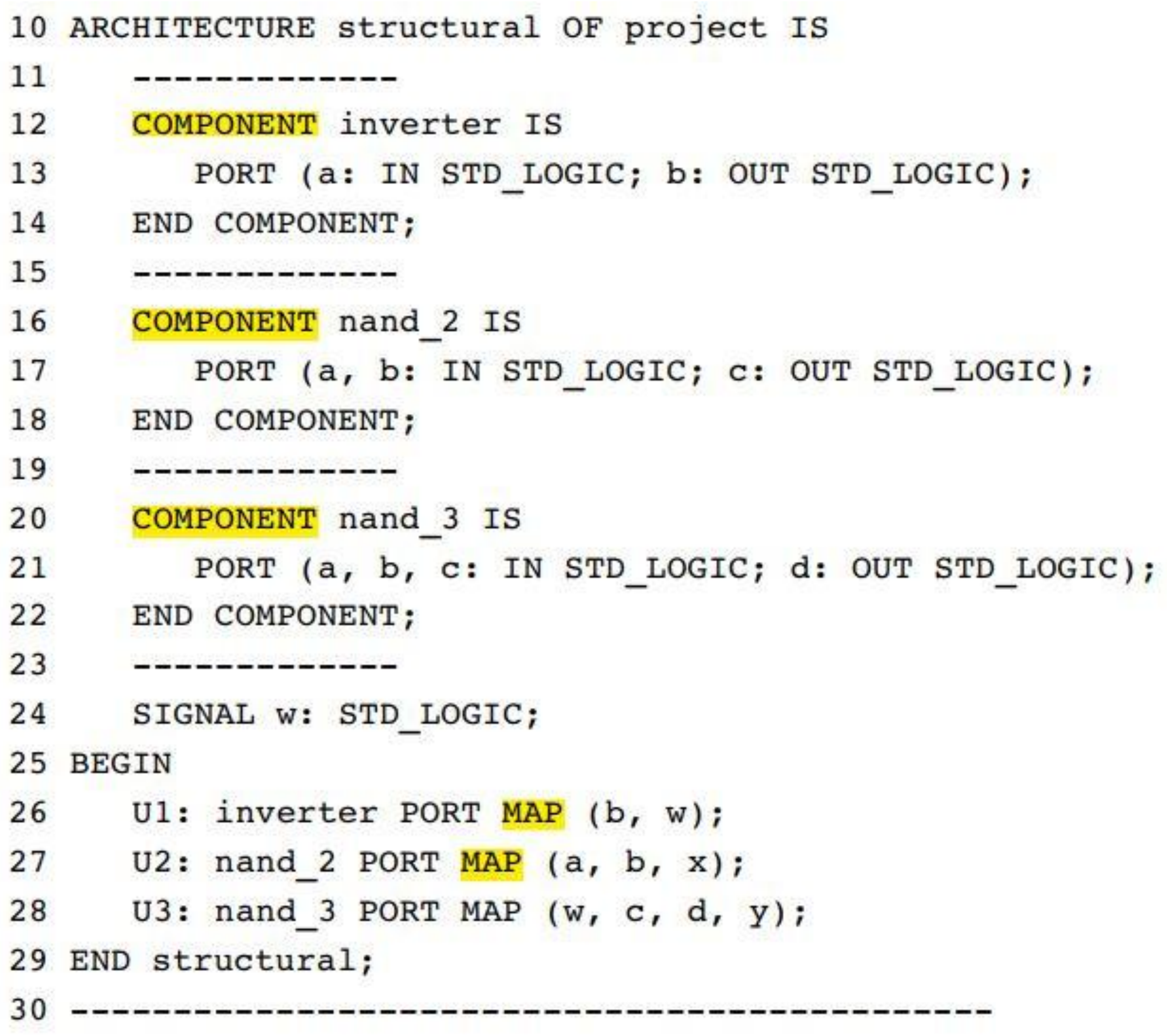
```
------ File inverter.vhd: -------------------
LIBRARY ieee;
USE ieee.std_logic_1164.all;
------------------------------------
ENTITY inverter IS
   PORT (a: IN STD_LOGIC; b: OUT STD_LOGIC);
END inverter;
------------------------------------
ARCHITECTURE inverter OF inverter IS
BEGIN
   b <= NOT a;
END inverter;
---------------------------------------------
```

```
------ File nand_2.vhd: --------------------
LIBRARY ieee;
USE ieee.std_logic_1164.all;
------------------------------------
ENTITY nand_2 IS
   PORT (a, b: IN STD_LOGIC; c: OUT STD_LOGIC);
END nand_2;
------------------------------------
ARCHITECTURE nand_2 OF nand_2 IS
BEGIN
   c <= NOT (a AND b);
END nand_2;
---------------------------------------------
```

```
----- File nand_3.vhd: ---------------------
LIBRARY ieee;
USE ieee.std_logic_1164.all;
------------------------------------
ENTITY nand_3 IS
   PORT (a, b, c: IN STD_LOGIC; d: OUT STD_LOGIC);
END nand_3;
------------------------------------
ARCHITECTURE nand_3 OF nand_3 IS
BEGIN
   d <= NOT (a AND b AND c);
END nand_3;
---------------------------------------------
```

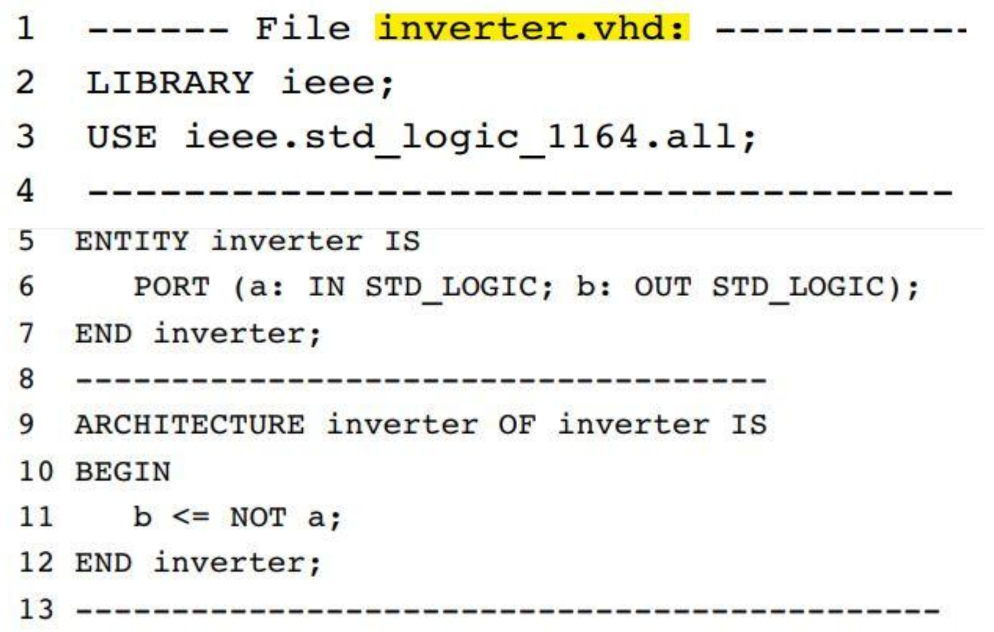
```
----- File project.vhd: --------------------
LIBRARY ieee;
USE ieee.std_logic_1164.all;
------------------------------------
ENTITY project IS
   PORT (a, b, c, d: IN STD_LOGIC;
         x, y: OUT STD_LOGIC);
END project;
------------------------------------
```

Components Declared in the Main Code

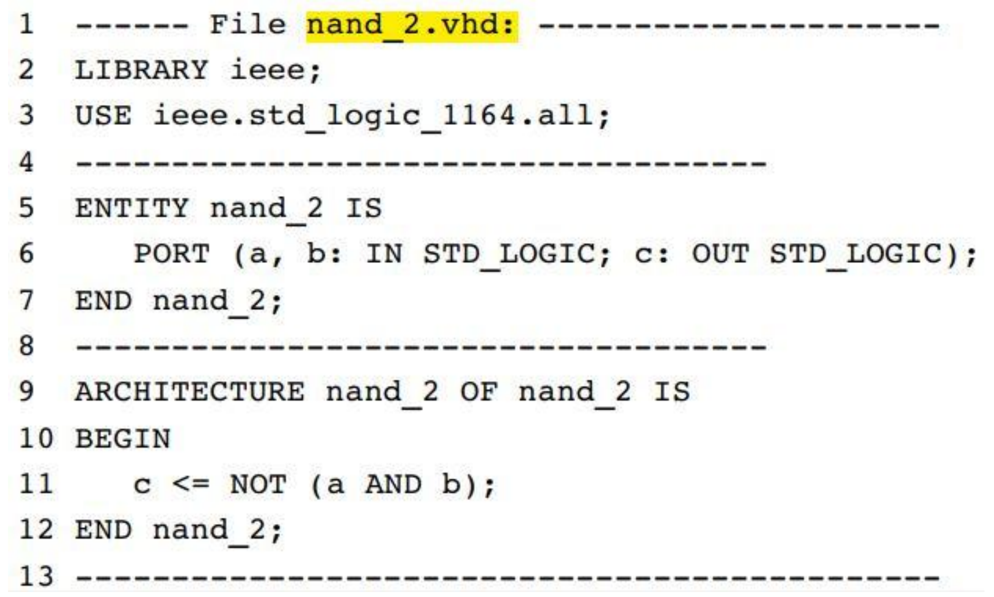


```
ARCHITECTURE structural OF project IS
   -------------
   COMPONENT inverter IS
      PORT (a: IN STD_LOGIC; b: OUT STD_LOGIC);
   END COMPONENT;
   -------------
   COMPONENT nand_2 IS
      PORT (a, b: IN STD_LOGIC; c: OUT STD_LOGIC);
   END COMPONENT;
   -------------
   COMPONENT nand_3 IS
      PORT (a, b, c: IN STD_LOGIC; d: OUT STD_LOGIC);
   END COMPONENT;
   -------------
   SIGNAL w: STD_LOGIC;
BEGIN
   U1: inverter PORT MAP (b, w);
   U2: nand_2 PORT MAP (a, b, x);
   U3: nand_3 PORT MAP (w, c, d, y);
END structural;
---------------------------------------------
```

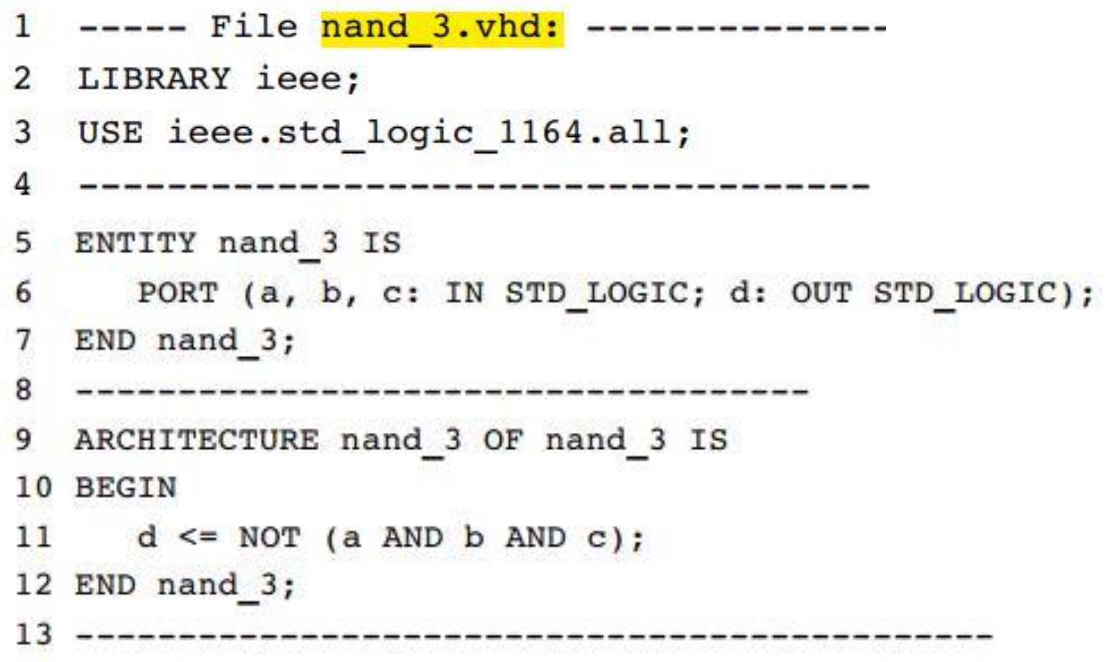
Components Declared in a Package



```
------ File inverter.vhd: -----------
LIBRARY ieee;
USE ieee.std_logic_1164.all;
------------------------------------
ENTITY inverter IS
   PORT (a: IN STD_LOGIC; b: OUT STD_LOGIC);
END inverter;
------------------------------------
ARCHITECTURE inverter OF inverter IS
BEGIN
   b <= NOT a;
END inverter;
--------------------------------------------
```

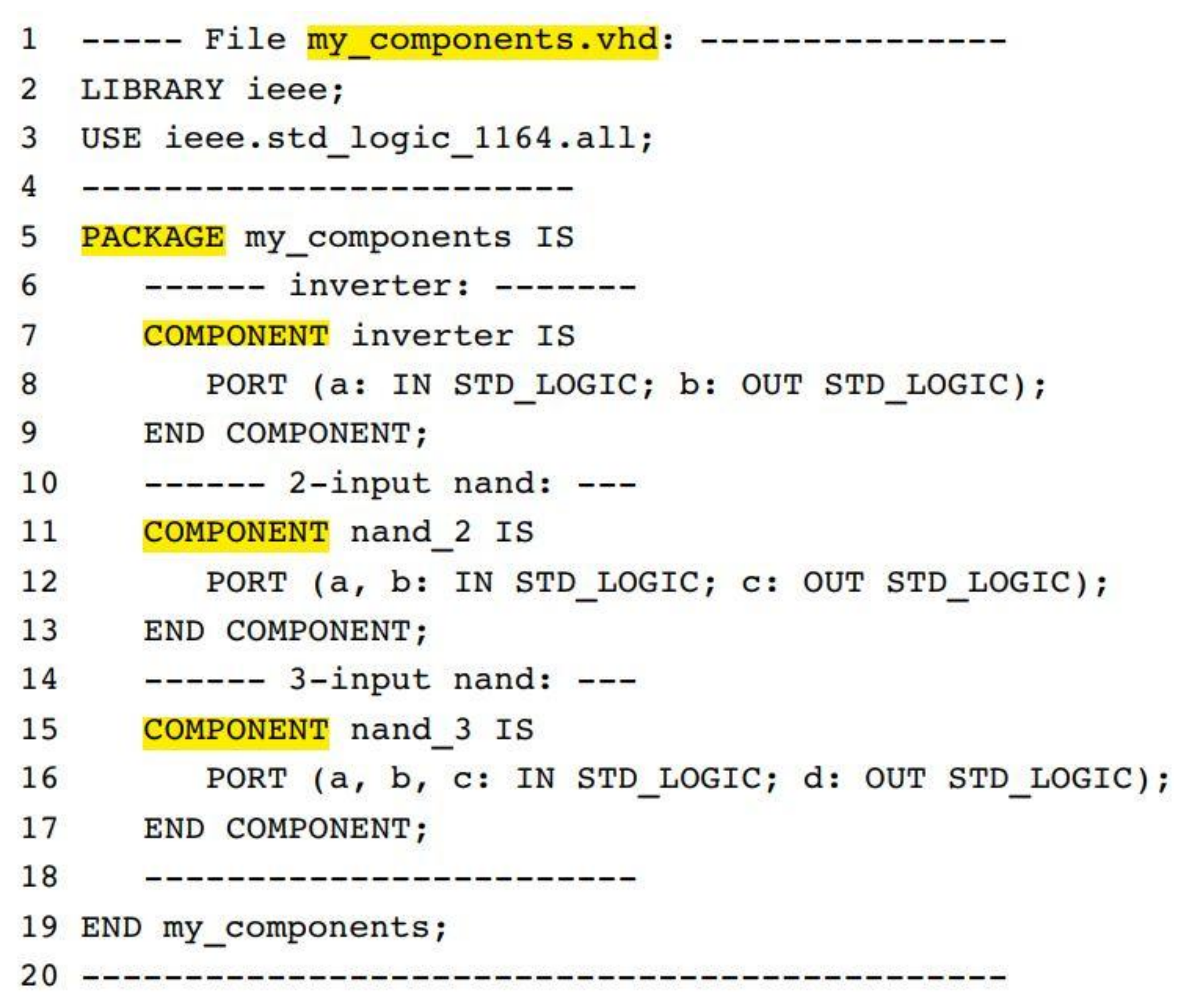


```
------ File nand_2.vhd: --------------------
LIBRARY ieee;
USE ieee.std_logic_1164.all;
------------------------------------
ENTITY nand_2 IS
   PORT (a, b: IN STD_LOGIC; c: OUT STD_LOGIC);
END nand_2;
------------------------------------
ARCHITECTURE nand_2 OF nand_2 IS
BEGIN
   c <= NOT (a AND b);
END nand_2;
--------------------------------------------
```

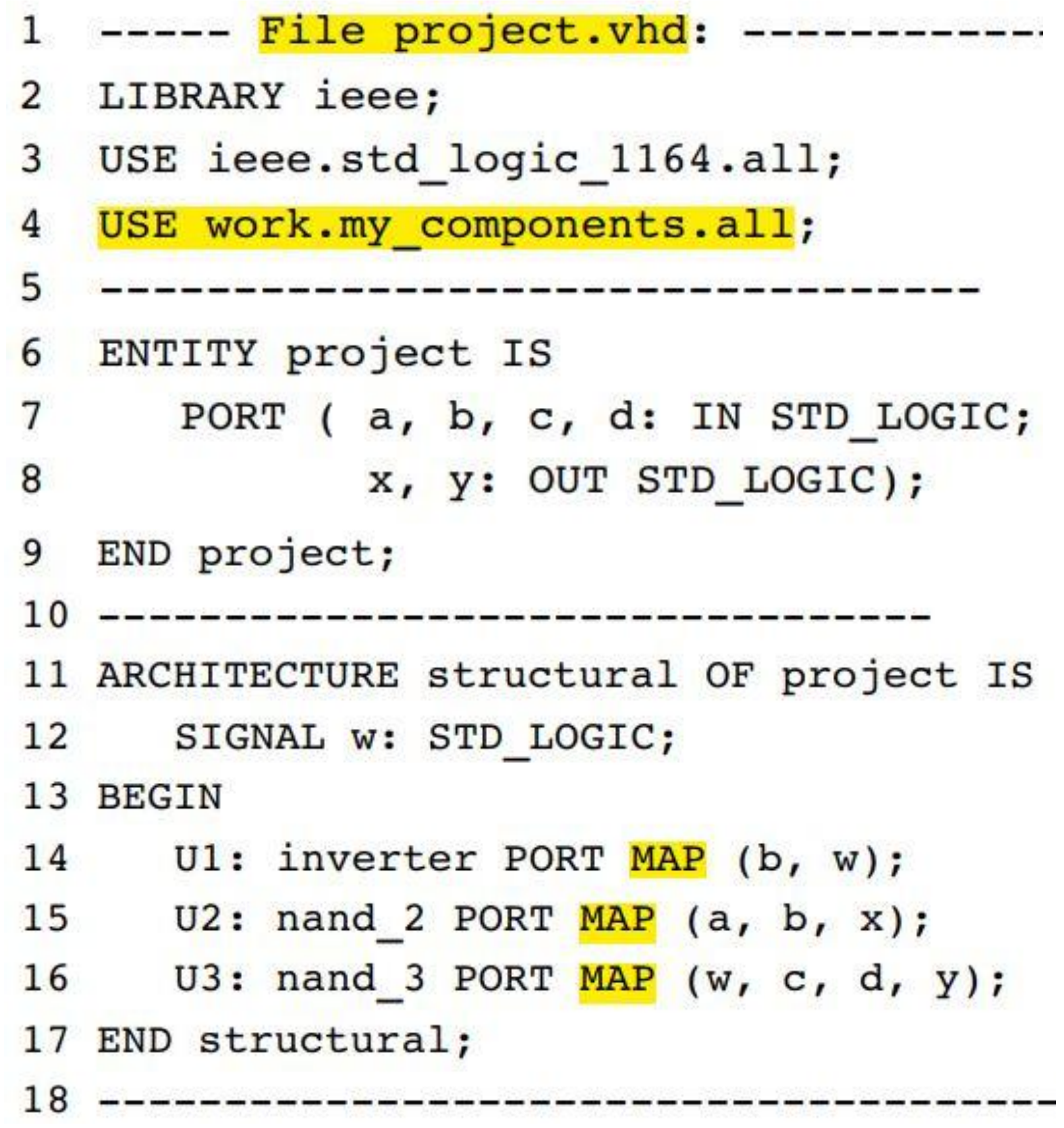


```
----- File nand_3.vhd: -------------
LIBRARY ieee;
USE ieee.std_logic_1164.all;
------------------------------------
ENTITY nand_3 IS
   PORT (a, b, c: IN STD_LOGIC; d: OUT STD_LOGIC);
END nand_3;
------------------------------------
ARCHITECTURE nand_3 OF nand_3 IS
BEGIN
   d <= NOT (a AND b AND c);
END nand_3;
--------------------------------------------
```

Components Declared in a Package



```
----- File my_components.vhd: ---------------
LIBRARY ieee;
USE ieee.std_logic_1164.all;
------------------------
PACKAGE my_components IS
   ------ inverter: -------
   COMPONENT inverter IS
      PORT (a: IN STD_LOGIC; b: OUT STD_LOGIC);
   END COMPONENT;
   ------ 2-input nand: ---
   COMPONENT nand_2 IS
      PORT (a, b: IN STD_LOGIC; c: OUT STD_LOGIC);
   END COMPONENT;
   ------ 3-input nand: ---
   COMPONENT nand_3 IS
      PORT (a, b, c: IN STD_LOGIC; d: OUT STD_LOGIC);
   END COMPONENT;
   ------------------------
END my_components;
---------------------------------------------
```



```
----- File project.vhd: -----------
LIBRARY ieee;
USE ieee.std_logic_1164.all;
USE work.my_components.all;
-------------------------------
ENTITY project IS
   PORT ( a, b, c, d: IN STD_LOGIC;
            x, y: OUT STD_LOGIC);
END project;
------------------------------
ARCHITECTURE structural OF project IS
   SIGNAL w: STD_LOGIC;
BEGIN
   U1: inverter PORT MAP (b, w);
   U2: nand_2 PORT MAP (a, b, x);
   U3: nand_3 PORT MAP (w, c, d, y);
END structural;
---------------------------------
```

Assignments

- The assignments will be attached to your class room.

End of lecture 10
Any Questions ?