GONZAGA UNIVERSITY

VHDL:
Packages and Components

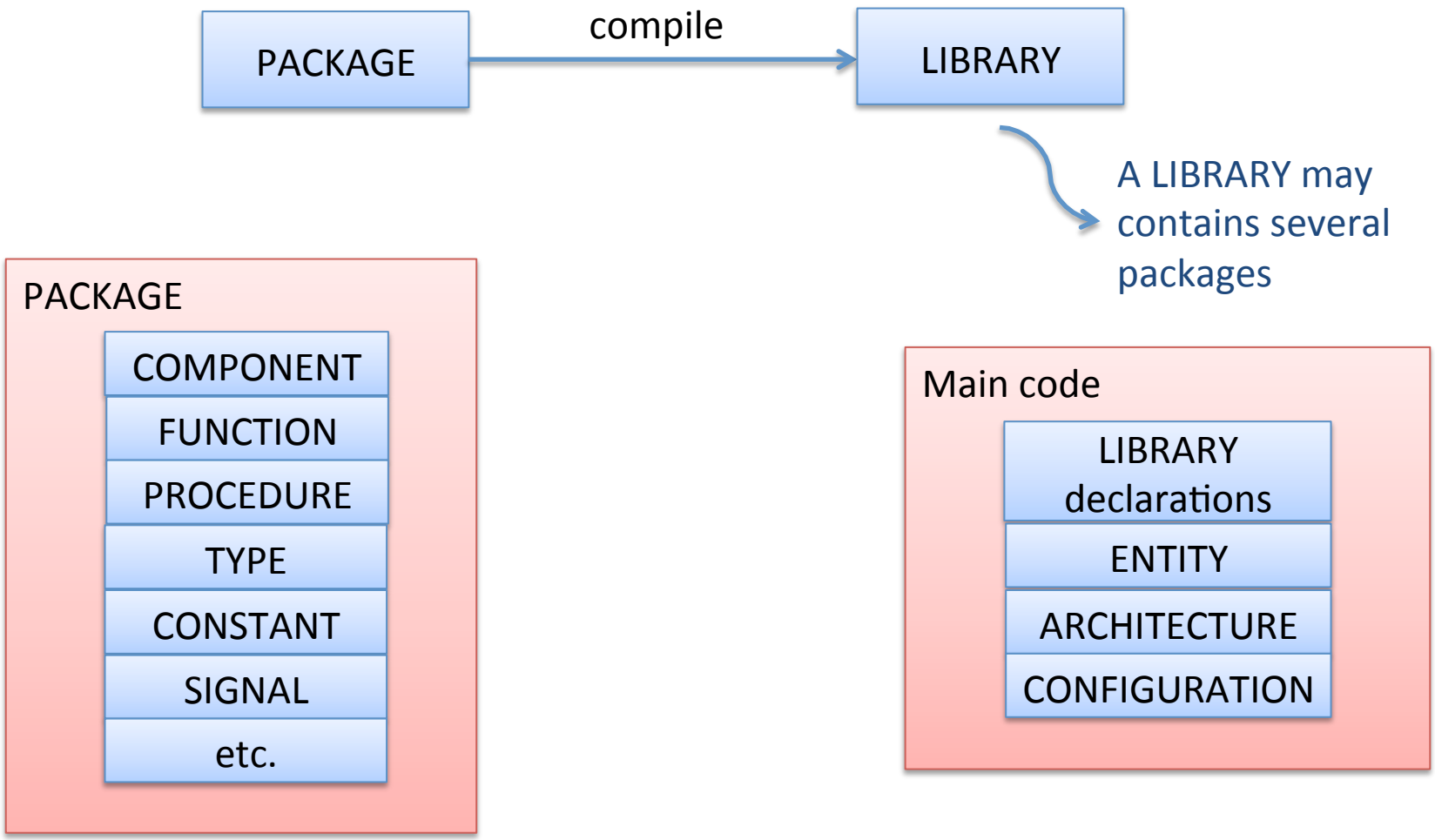
GONZAGA UNIVERSITY

Code Reuse

- Frequently used pieces of VHDL code can be written in the form of:
  - COMPONENTs
  - FUNCTIONs
  - PROCEDUREs
- These “units” can be located in the MAIN code

- … However since the main purpose is to allow common pieces of code to be reused and shared it is more usual to place them in a PACKAGE which is finally compiled in a LIBRARY

Package


PACKAGE
compile
LIBRARY
A LIBRARY may contains several packages
PACKAGE
COMPONENT
FUNCTION
PROCEDURE
TYPE
CONSTANT
SIGNAL
etc.
Main code
LIBRARY declarations
ENTITY
ARCHITECTURE
CONFIGURATION

GONZAGA UNIVERSITY

Package syntax

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

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

Package Declaration List

Package Body = descriptions

- The PACKAGE declaration list can contain:
  - TYPE
  - CONSTANT
  - SIGNAL
  - FUNCTION
  - PROCEDURE
  - etc.

See sections 4.7-4.9 of IEEE Std 1076-2008 for more details

GONZAGA UNIVERSITY

PACKAGE: Example

```
1  ------------------------------------------------
2  LIBRARY ieee;
3  USE ieee.std_logic_1164.all;
4  ------------------------------------------------
5  PACKAGE my_package IS
6     TYPE state IS (st1, st2, st3, st4);
7     TYPE color IS (red, green, blue);
8     CONSTANT vec: STD_LOGIC_VECTOR(7 DOWNTO 0) := "11111111";
9  END my_package;
10 ------------------------------------------------
```

To make use of the package in the VHDL main code we need:

1. Compile the VHDL package

   the package become part of the WORK LIBRARY

2. add a new USE clause to the main code.

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

GONZAGA UNIVERSITY

PACKAGE: Example

```
1  ------------------------------------------------
2  LIBRARY ieee;
3  USE ieee.std_logic_1164.all;
4  ------------------------------------------------
5  PACKAGE my_package IS
6     TYPE state IS (st1, st2, st3, st4);
7     TYPE color IS (red, green, blue);
8     CONSTANT vec: STD_LOGIC_VECTOR(7 DOWNTO 0) := "11111111";
9     FUNCTION positive_edge(SIGNAL s: STD_LOGIC) RETURN BOOLEAN;
10 END my_package;
11 ------------------------------------------------
12 PACKAGE BODY my_package IS
13    FUNCTION positive_edge(SIGNAL s: STD_LOGIC) RETURN BOOLEAN IS
14    BEGIN
15       RETURN (s'EVENT AND s='1');
16    END positive_edge;
17 END my_package;
18 ------------------------------------------------
```

GONZAGA UNIVERSITY

COMPONENT

- A COMPONENT is simply a piece of **conventional code** (= LIBRARY declarations + ENTITY + ARCHITECTURE)

- However, by declaring the code as a COMPONENT, it can be used within another circuit

Allowing the construction of **hierarchical** designs.

GONZAGA UNIVERSITY

COMPONENT

- To use a component it must be:
  - **Declared**
  - **Instantiated**

COMPONENT declaration:

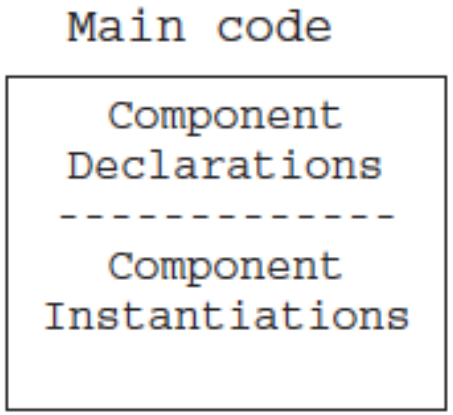
```
COMPONENT component_name IS
    PORT (
      port_name : signal_mode signal_type;
      port_name : signal_mode signal_type;
      ...);
END COMPONENT;
```

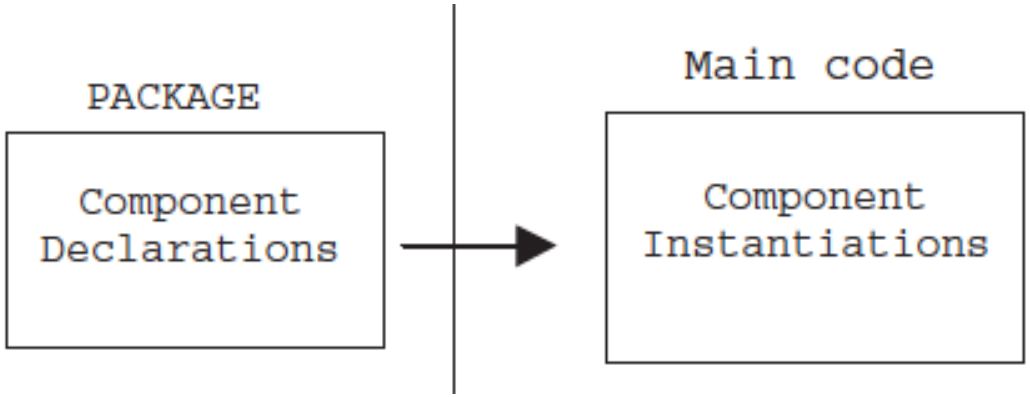
COMPONENT instantiation:

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

Basic ways of declaring COMPONENTs

- Declaring COMPONENTS
  - in the main code
  - in a package





Example: Components declared in the main (1)

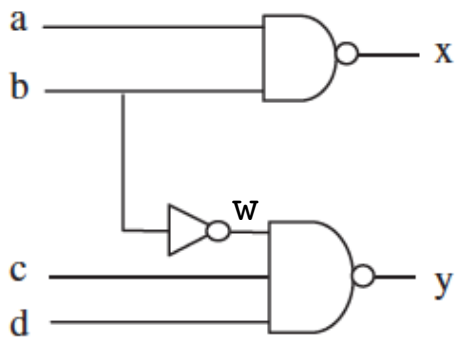
```
1  ------- File inverter.vhd: ------------------
2  LIBRARY ieee;
3  USE ieee.std_logic_1164.all;
4  ------------------------------------
5  ENTITY inverter IS
6     PORT (a: IN STD_LOGIC; b: OUT STD_LOGIC);
7  END inverter;
8  ------------------------------------
9  ARCHITECTURE inverter OF inverter IS
10 BEGIN
11    b <= NOT a;
12 END inverter;
13 ----------------------------------------------
```

```
1  ------ File nand_2.vhd: --------------------
2  LIBRARY ieee;
3  USE ieee.std_logic_1164.all;
4  ------------------------------------
5  ENTITY nand_2 IS
6     PORT (a, b: IN STD_LOGIC; c: OUT STD_LOGIC);
7  END nand_2;
8  ------------------------------------
9  ARCHITECTURE nand_2 OF nand_2 IS
10 BEGIN
11    c <= NOT (a AND b);
12 END nand_2;
13 ----------------------------------------------
```

```
1  ------ File nand_3.vhd: --------------------
2  LIBRARY ieee;
3  USE ieee.std_logic_1164.all;
4  ------------------------------------
5  ENTITY nand_3 IS
6     PORT (a, b, c: IN STD_LOGIC; d: OUT STD_LOGIC);
7  END nand_3;
8  ------------------------------------
9  ARCHITECTURE nand_3 OF nand_3 IS
10 BEGIN
11    d <= NOT (a AND b AND c);
12 END nand_3;
13 ----------------------------------------------
```

Disclaimer:
The purpose of this example is to show VHDL syntax. It is NOT an endorsement to work at such a low level of abstraction !!!

Example: Components declared in the main (2)




```
1  ---------- File project.vhd: ----------
2  LIBRARY ieee;
3  USE ieee.std_logic_1164.all;
4  ------------------------------------
5  ENTITY project IS
6     PORT (a, b, c, d: IN STD_LOGIC;
7           x, y: OUT STD_LOGIC);
8  END project;
9  ------------------------------------
```

```
10 ARCHITECTURE structural OF project IS
11    -------------
12    COMPONENT inverter IS
13       PORT (a: IN STD_LOGIC; b: OUT STD_LOGIC);
14    END COMPONENT;
15    -------------
16    COMPONENT nand_2 IS
17       PORT (a, b: IN STD_LOGIC; c: OUT STD_LOGIC);
18    END COMPONENT;
19    -------------
20    COMPONENT nand_3 IS
21       PORT (a, b, c: IN STD_LOGIC; d: OUT STD_LOGIC);
22    END COMPONENT;
23    -------------
24    SIGNAL w: STD_LOGIC;
25 BEGIN
26    U1: inverter PORT MAP (b, w);
27    U2: nand_2 PORT MAP (a, b, x);
28    U3: nand_3 PORT MAP (w, c, d, y);
29 END structural;
30 -----------------------------------------
```

Example: Components declared in a package (1)

```
1  ------- File inverter.vhd: ------------------
2  LIBRARY ieee;
3  USE ieee.std_logic_1164.all;
4  ------------------------------------
5  ENTITY inverter IS
6     PORT (a: IN STD_LOGIC; b: OUT STD_LOGIC);
7  END inverter;
8  ------------------------------------
9  ARCHITECTURE inverter OF inverter IS
10 BEGIN
11    b <= NOT a;
12 END inverter;
13 ----------------------------------------------
```

```
1  ------ File nand_2.vhd: --------------------
2  LIBRARY ieee;
3  USE ieee.std_logic_1164.all;
4  ------------------------------------
5  ENTITY nand_2 IS
6     PORT (a, b: IN STD_LOGIC; c: OUT STD_LOGIC);
7  END nand_2;
8  ------------------------------------
9  ARCHITECTURE nand_2 OF nand_2 IS
10 BEGIN
11    c <= NOT (a AND b);
12 END nand_2;
13 ----------------------------------------------
```

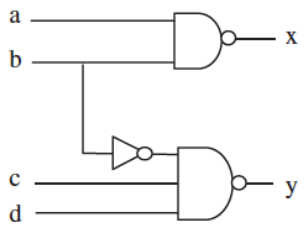
```
1  ------ File nand_3.vhd: --------------------
2  LIBRARY ieee;
3  USE ieee.std_logic_1164.all;
4  ------------------------------------
5  ENTITY nand_3 IS
6     PORT (a, b, c: IN STD_LOGIC; d: OUT STD_LOGIC);
7  END nand_3;
8  ------------------------------------
9  ARCHITECTURE nand_3 OF nand_3 IS
10 BEGIN
11    d <= NOT (a AND b AND c);
12 END nand_3;
13 ----------------------------------------------
```

Disclaimer:
The purpose of this example is to show VHDL syntax. It is NOT an endorsement to work at such a low level of abstraction !!!

Example: Components declared in a package (2)

```
1  ----- File my_components.vhd: ---------------
2  LIBRARY ieee;
3  USE ieee.std_logic_1164.all;
4  ------------------------
5  PACKAGE my_components IS
6     ------ inverter: -------
7     COMPONENT inverter IS
8        PORT (a: IN STD_LOGIC; b: OUT STD_LOGIC);
9     END COMPONENT;
10    ------ 2-input nand: ---
11    COMPONENT nand_2 IS
12       PORT (a, b: IN STD_LOGIC; c: OUT STD_LOGIC);
13    END COMPONENT;
14    ------ 3-input nand: ---
15    COMPONENT nand_3 IS
16       PORT (a, b, c: IN STD_LOGIC; d: OUT STD_LOGIC);
17    END COMPONENT;
18    ------------------------
19 END my_components;
20 ----------------------------------------
```

```
1  ----- File project.vhd: --------------------
2  LIBRARY ieee;
3  USE ieee.std_logic_1164.all;
4  USE work.my_components.all;
5  --------------------------------
6  ENTITY project IS
7     PORT ( a, b, c, d: IN STD_LOGIC;
8            x, y: OUT STD_LOGIC);
9  END project;
10 --------------------------------
11 ARCHITECTURE structural OF project IS
12    SIGNAL w: STD_LOGIC;
13 BEGIN
14    U1: inverter PORT MAP (b, w);
15    U2: nand_2 PORT MAP (a, b, x);
16    U3: nand_3 PORT MAP (w, c, d, y);
17 END structural;
18 ----------------------------------------
```



GONZAGA UNIVERSITY

PORT MAPPING

- There are two ways to map the PORTS of a COMPONENT during its instantiation:
  - positional mapping
  - nominal (by name) mapping

```
COMPONENT inverter IS
   PORT (a: IN STD_LOGIC; b: OUT STD_LOGIC);
END COMPONENT;
...
U1: inverter PORT MAP (x, y); -- positional mapping
...
U2: inverter PORT MAP (x => a, y =>b);  -- nominal mapping
```

- Ports can also be left unconnected (using the keyword OPEN)