VHDL:
Packages and Components
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
A PACKAGE may contain several packages.

**PACKAGE**
- COMPONENT
- FUNCTION
- PROCEDURE
- TYPE
- CONSTANT
- SIGNAL
- etc.

**LIBRARY**
- ENTITY
- ARCHITECTURE
- CONFIGURATION
- LIBRARY declarations

Main code

PACKAGE to LIBRARY: compile

A LIBRARY may contain several packages.
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.
To make use of the package in the VHDL main code we need:
1. Compile the VHDL package, so the package becomes part of the WORK LIBRARY.
2. Add a new USE clause to the main code.
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  -----------------------------------------------------------------------------
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.

talarico@gonzaga.edu
COMPONENT

• To use a component it must be:
  – Declared
  – Instantiated

COMPONENT declaration:

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

COMPONENT instantiation:

```text
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)

```vhdl
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;
```

```vhdl
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;
```

```vhdl
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;
```

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 !!!
Component declared in the main (2):

```vhdl
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;

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;
```
Example:
Components declared in a package (1)

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)

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