University of Bridgeport

INTRO TO VLSI DESIGN
CPE 448

VHDL
Tutorial-IV

Components and
Essentials of Libraries

For: April 2, 2002 (Tuesday) By: Prakash S. Thapa

LIBRARY

Each design unit – such as an entity, architecture, package body – is analyzed (compiled) and placed in a design library. Libraries are generally implemented as directories and are referenced by logical name. In the implementation of the VHDL simulator, this logical name maps to a physical path to the corresponding directory and this mapping is maintained by the host implementation. However, just like variables and signals, before we can use a design library we must declare the library we are using by specifying the library's logical name.

In VHDL, the libraries STD and WORK are implicitly declared. Therefore user programs do not need to declare these libraries. The former contains standard packages provided with VHDL distributions. The latter refers to the working directory, which can be set within the simulation environment you are using.

Example 1:

library *IEEE;*
use *IEEE.std_logic_1164.all***;**

COMPONENTS:

Within an architecture body, we can write component instantiation statements that describe instances of an entity and connect signals to the ports of the instances. This simple approach to building a hierarchical design works well if we know in advance all the details of the entities we want to use. However, this is not always the case, especially in a large design project.

The first thing we need to do to describe an interconnection of subsystems in a design is to describe the different kinds of components used. We can achieve this by writing entity declarations for each of the subsystems. Each entity declaration is a separate design unit and has corresponding architecture bodies that describe implementations.

```
component_declaration
      component identifier is
                  [ generic (generic_interface_list ); ]
                  [ port (port_interface_list ); ]
      end component [ identifier ];

component flipflop is
      generic (Tprop, Tsetup, Thold : delay_length);
      port (clk, clr, d   :      in bit;
            q             :     out bit );
end component flipflop;
```

Example 2:

```
entity reg4 is
      port (clk, clr      : in   bit;
            d             : in   bit_vector(0 to 3);
            q             : out bit_vector(0 to 3);
end entity reg4;

architecture struct of reg4 is
      component flipflop is
            generic (Tprop, Tsetup, Thold      : delay_length);
            port ( clk, clr, d                 : in bit;
                        q                      : out bit);
      end component flipflop;
begin

bit0: component flipflop
      generic map ( Tprop => 2 ns, Tsetup => 2ns, Thold => 1ns)
      port map ( clk => clk, clr => clr, d => d(0), q => q(0) );

bit1: component flipflop
      generic map ( Tprop => 2 ns, Tsetup => 2ns, Thold => 1ns)
      port map ( clk => clk, clr => clr, d => d(1), q => q(1) );

bit2: component flipflop
      generic map ( Tprop => 2 ns, Tsetup => 2ns, Thold => 1ns)
      port map ( clk => clk, clr => clr, d => d(2), q => q(2) );

bit3: component flipflop
      generic map ( Tprop => 2 ns, Tsetup => 2ns, Thold => 1ns)
      port map ( clk => clk, clr => clr, d => d(3), q => q(3) );

end architecture struct;
```