CO102
Computer Hardware Experiment

Lecture 03: VHDL – Using Port Map

Liang Yanyan

澳門科技大學
Macau of University of Science and Technology

A Big Design

- If we are going to implement a big design, one way is to put everything in one VHDL file.
- E.g. a design with an “and” gate and a “or” gate.

```
entity topdesign is
port(
        d0 : in std_logic_vector(31 downto 0);
        d1 : in std_logic_vector(31 downto 0);
        out1 : out std_logic_vector(31 downto 0)
        );
end topdesign;
architecture topdesign_arch of topsign is
signal tmp : std_logic_vector(31 downto 0);
begin
        tmp <= d0 and d1;
        out1 <= tmp or d0;
end topdesign_arch;
```

How about in C?

- If we are going to implement a big project in C:
  - Divide the project into a number of parts, and put each part in an individual function or file.
  - Invoke those functions in the main function.
- Advantages
  - Improve the readability of code.
  - Easier for maintenance.
  - Easier for code reuse in other projects.

How about in VHDL?

- We can divide the big design into a number of smaller parts, each part is called a COMPONENT.
- Implement each COMPONENT in a VHDL file.
- Invoke (connect) all COMPONENTs in the main VHDL file using PORT MAP statement.
- Then we can use ISE to import all VHDL files (the main VHDL and components) and simulate the entire design.

Divide the big design

```
entity topdesign is
   port(
          d0 : in std_logic_vector(31 downto 0);
          d1 : in std_logic_vector(31 downto 0);
          out1 : out std_logic_vector(31 downto 0)
          );
   end topdesign;

architecture topdesign_arch of topsign is
   signal tmp : std_logic_vector(31 downto 0);
   begin
          tmp <= d0 and d1;
          out1 <= tmp or d0;
   end topdesign_arch;
```

Divide the design into two smaller components, one performs "and" function, the other performs "or'.

VHDL for “and”: *myand.vhd*

```
entity myand is
   port(
           d0 : in std_logic_vector(31 downto 0);
           d1 : in std_logic_vector(31 downto 0);
           out1 : out std_logic_vector(31 downto 0)
           );
   end myand;

architecture myand_arch of myand is
begin
           out1 <= d0 and d1;
end myand_arch;
```

VHDL for “or”: *myor.vhd*

```
entity myor is
   port(
            d0 : in std_logic_vector(31 downto 0);
            d1 : in std_logic_vector(31 downto 0);
            out1 : out std_logic_vector(31 downto 0)
            );
   end myor;

architecture myor_arch of myor is
begin
            out1 <= d0 or d1;
end myor_arch;
```

VHDL for main: topdesign.vhd

```
entity topdesign is
   port(
            d0 : in std_logic_vector(31 downto 0);
            d1 : in std_logic_vector(31 downto 0);
            out1 : out std_logic_vector(31 downto 0)
            );
   end topdesign;
```

VHDL for main: topdesign.vhd

```
architecture topdesign_arch of topsign is
component myand
        port(
                 d0 : in std_logic_vector(31 downto 0);
                 d1 : in std_logic_vector(31 downto 0);
                 out1 : out std_logic_vector(31 downto 0)
                 );
end component;
component myor
        port(
                 d0 : in std_logic_vector(31 downto 0);
                 d1 : in std_logic_vector(31 downto 0);
                 out1 : out std_logic_vector(31 downto 0)
                 );
end component;

signal tmp : std_logic_vector(31 downto 0);
```

VHDL for main: topdesign.vhd

```
begin
    myand1 : myand PORT MAP (
                    d0 => d0,
                    d1 => d1,
                    out1 => tmp
    );
    myor1 : myor PORT MAP (
                    d0 => tmp,
                    d1 => d0,
                    out1 => out1
    );
end topdesign_arch;
```

VHDL for main: topdesign.vhd

```
 begin
      myand1 : myand PORT MAP (d0, d1, tmp);
      myor1 : myor PORT MAP (tmp, d0, out1);
end topdesign_arch;
```