University of Bridgeport

INTRO TO VLSI DESIGN
CPE 448

VHDL
Tutorial-II

Basic Language Concepts
&
Design Simulation

For:

I. Simple System Design

The primary programming abstraction in VHDL is design entity. Example of design entity includes a chip, board, and transistor. Design entity consists of two sections:

1. Entity Declaration:
   This contains interface declaration.

Example 1:

```
entity half_adder is
port (
        a     :   in bit;
        b     :   in bit;
        sum   :   out bit;
        carry :   out bit );
end half_adder;
```

Example 2:

```
entity eq_comp4 is
port (
        a     :   in bit_vector(3 downto 0);
        b     :   in bit_vector(3 downto 0);
        equals:   out bit );
end eq_comp4;
```

2. Architecture Body:
   This contains functional declaration.

Example 1:

```
architecture bool of half_adder is
begin
        sum   <=  (a xor b);
        carry <=  (a and b);
end bool
;
```

Example 2:

```
architecture dataflow of eq_comp4 is
begin
        equals<= '1'
              when (a=b)
                   else '0';
end dataflow;
```

II. Basic System Descriptions

From the level of abstraction systems can be described in three types:

1. STRUCTURAL:
   One way to describe a system is to describe component chips and the interconnections assuming that the user is familiar with it. This kind of definition is the structural definition.

Example 1:

```
library ieee;
use ieee.std_logic_1164.all;
entity  full_adder is
port  (
        a       :       in std_ulogic;
        b       :       in std_ulogic;
        ci      :       in std_ulogic;
        sum     :      out std_ulogic;
        co      :      out std_ulogic );
end full_adder;
architecture  bool of full_adder is
signal s1, s2, s3          :       std_ulogic;
begin
        u0:   s1<= (a xor b);
        u1:   s2<= (ci and s1);
        u2:   s3<= (a and b);
        u3:   sum<= (s1 xor c_in);
        u4:   co<= (s2 or s3);
end dataflow;
```

Example 2:

```
library ieee;
use ieee.std_logic_1164.all;
entity  eq_comp4 is
port (
      a      : in std_logic_vector(3 down_to 0);
      b      : in std_logic_vector(3 down_to 0);
      equals: out std_logic );
end eq_comp4;
architecture  struct of eq_comp4 is
      signal x     : std_logic_vector(0 to 3);

begin
      u0:   xnor_2      port map (a(0), b(0), x(0));
      u1:   xnor_2      port map (a(1), b(0), x(0));
      u2:   xnor_2      port map (a(2), b(0), x(0));
      u3:   xnor_2      port map (a(3), b(0), x(0));
      u4:and_4  port  map  (x(0),  x(1),  x(2),  x(3),
equals );
end struct;
```

This example assumes that xnor_2 and and_4 are already defined components.

2. BEHAVIORAL:

We can describe a system in terms of processing it performs on its input signals and the type of output it signals it produces.

Example 1:

```
library ieee;
use ieee.std_logic_1164.all;
entity eq_comp4 is
port (
      a      : in std_logic_vector(3 down_to 0);
      b      : in std_logic_vector(3 down_to 0);
      equals: out std_logic );
end eq_comp4;
architecture behavioral of eq_comp4 is
begin
comp:      process (a,b)
      begin
                  if (a=b) then
                        equals<= '1';
                  else
                        equals<= '0';
                  end if;
            end process comp;
end behavioral;
```

3. DATAFLOW:

Dataflow architecture specifies how data will be transferred from signal to signal and input to input without the sequential statements.

Some distinguish between dataflow and behavioral others lump them together in behavioral description. Primary difference is that behavioral uses processes while dataflow does not.

Example 1:

```
library ieee;
use ieee.std_logic_1164.all;
entity eq_comp4 is
port (
      a      : in std_logic_vector(3 down_to 0);
      b      : in std_logic_vector(3 down_to 0);
      equals: out std_logic );
end eq_comp4;

architecture dataflow of eq_comp4 is
begin
            equals<= '1'
                  when (a=b)
                        else '0';
end dataflow;
```

Example 2:

```
library ieee;
use ieee.std_logic_1164.all;
entity eq_comp4 is
port (
      a      : in std_logic_vector(3 down_to 0);
      b      : in std_logic_vector(3 down_to 0);
      equals: out std_logic );
end eq_comp4;
architecture bool of eq_comp4 is
begin
      equals<=          not ( a(0) xor b(0))
                  and  not ( a(1) xor b(1))
                  and  not ( a(2) xor b(2))
                  and  not ( a(3) xor b(3));
end bool;
```