VHDL Programming in CprE 381

Henry Duwe
Spring 2020
Originally Zhao Zhang
CprE 381
Iowa State University

VHDL Notes in 381

- I will provide a little bit of VHDL instruction just-in-time
- I will focus on the minimum VHDL to complete 381
  - This is not intended to teach you how to be a VHDL expert
- Take the notes as exposure to new concepts/techniques
  - The notes will only cover the essential of each concept
  - Search Internet or get a good VHDL book, for the complete description and details
  - The “Free Range VHDL” is a good starting point but may not be sufficient

Structural, Dataflow, and Behavioral

- Those are different ways to *model* a hardware component

```
entity my_circuit is
    port(i1, i2, i3, i4 : in std_logic;
         o : out std_logic);
end my_circuit;
```

Structure, Dataflow, and Behavior

```
architecture mixed of my_circuit is
  signal wire1, wire2 : std_logic;
  component nand2 is
    port(A : in std_logic;
         B : in std_logic;
         O : out std_logic);
  end component;
begin
  nand_gate1 : nand2
    port map( A => i1,
              B => i2,
              o => wire1);
    -- this is a comment line
    -- the following works, but is harder to read
    -- port map (i1, i2, wire1);

  nand_gata2 : block
  begin
    wire2 <= i1 nand i2;
  end block;

  nand_data3 : process (wire1, wire2)
  begin
    if (wire1 = '1') and (wire2 = '1') then
      o <= '0';
    else
      o <= '1';
    end if;
  end process;
end
```

Structure, Data Flow, and Behavior

- The previous example mixes three modeling styles: structure, data flow, and behavior
- It uses three VHDL programming constructs (complex statements), each for a NAND gate
  - Component/entity instantiation
  - Concurrent signal assignment
  - Process statement
- The logic circuits can be the same – Different modeling styles may or may not lead to the same circuit

Test Bench

- Test bench provides stimulus to the simulation
- Its entity statement is usually empty

```
entity tb_one_complement is
end tb_one_complement;
```

Test Bench

```
architecture behavior of tb_one_complement is
  -- components, signals, component instances
  component one_complement is
    port(A : in std_logic_vector(31 downto 0);
         O : out std_logic_vector(31 downto 0));
  end component;
  signal s_A, s_O : std_logic_vector(31 downto 0);
  …
begin

-- Instantiate design to test
DUT: one_complement
port map(A => s_A,
         O => s_O);

-- The test sequence
process
  begin
    s_A <= X"00000001";
    wait for 100 ns;

    s_A <= X"00001000";
    wait for 100 ns;
  end process;  -- it repeats
end behavior;
```