

VHDL Style File

The main goal of your VHDL source code is to model a digital circuit. This means that your only mission is to satisfy the VHDL synthesizer. But in reality, you and other humans are going to need to read and understand your VHDL source code. The single most important factor in creating readable and understandable VHDL source code is to make sure your VHDL source code follows certain appearance guidelines. The file listed below shows some more of the more important attributes and rules that all VHDL source code should adhere to. The overriding factor with your VHDL source code is to make it neat, organized, and readable; any specific items not listed in this style files should adhere to these principles.

```
-----
-- Company:
-- Engineer: Manuel Laybor, Ima Goodstudent
--
-- Create Date: 09/10/2007
-- Design Name: VHDL Style File
-- Module Name: comp2
-- Project Name: style file example
-- Target Devices: Digilent Nexys Board
-- Tool versions:
-- Description: Describe the purpose of this file. This should be a high-level
--              description but it should be complete. The low-level implementation
--              details should appear in the body of the VHDL code in the form of comments.
--
-- Dependencies: If the synthesis of this file depends on other file, be
--              sure to list them here.
--
-- Revision:
-- Revision 0.01 - File Created
-- Additional Comments: Say stuff that may be helpful to others who may be
--              wanting to use this file for their own projects.
-----
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

-- entity declaration should appear neat; much of the declaration
-- is aligned to increase readability.
entity comp2 is
    Port ( A,B : in STD_LOGIC_VECTOR (9 downto 0);
          EQ : out STD_LOGIC);
end comp2;

-- code is indented;
-- comments align with code indentations
architecture my_comp2 of comp2 is
begin
    -- all items in architecture body are indented

    -- label is included with process statements
    my_comp: process(A,B)
    begin
        if (A = B) then
            EQ <= '1';
        else
            EQ <= '0';
        end if;
    end process my_comp;
end my_comp2;
```

```

library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

entity comp3
  Port ( A,B,C : in  STD_LOGIC_VECTOR (9 downto 0);
        EQ3 : out STD_LOGIC);
end comp3;

architecture my_comp3 of comp3 is
  -- all items in declarative region of architecture are indented

  -- component declaration
  component comp2
    Port ( A,B : in  STD_LOGIC_VECTOR (9 downto 0);
          EQ : out STD_LOGIC);
  end component;

  -- intermediate signals
  signal eq1_s, eq2_s : std_logic;

begin
  -- blank lines are used to delineate instantiations
  -- and signal assignment statements

  -- component instantiations:
  -- label and entity name one first line
  -- port mappings on subsequent lines
  --
  -- port mappings are neat and aligned
  eq1: comp2
  port map (A => A,
           B => B,
           EQ => eq1_s);

  -- component instantiation
  -- another form of alignment used
  eq2: comp2
  port map ( A => A,
           B => C,
           EQ => eq2_s);

  -- basic logic for final output
  EQ3 <= eq1_s AND eq2_s;

end;

```