VHDL Coding Style

When working in teams it is useful to obey some coding styles in order to improve readability and maintainability. Furthermore you usually get good structured VHDL code and better synthesis results!

Therefore, *thou shalt* obey the following commandments:

1. Every Design/ Test bench should begin with a pre-defined header template. The header contains comments, revision control codes and the name of the authors.

```
-----------------------------------------------------------------------------
-- Title       : <title>
-- Project     : <project title>
-----------------------------------------------------------------------------
-- File        : <filename>
-- Author      : <authors>
-- Company     : <company, department>
-- Last update: <date>
-- Platform    : <tools: simulator, synthesis tool, platform>
-----------------------------------------------------------------------------
-- Description: <What is this code for?>
-----------------------------------------------------------------------------
-- Revisions   :
-- Date         Version  Author     Description
-- <date>       <nr.>    <author>    <changes done>
-----------------------------------------------------------------------------
```

2. Everything is lowercase unless otherwise specified. Verilog is case-sensitive, VHDL is NOT.

3. The filename name has to match the module/entity name. For example, if the entity name is ecu, the filename should be ecu.vhd. Next the testbench should use the extension <unit_name>_tb.vhd, e.g. ecu_tb.vhd, and packages should be named <package_name>_pkg.vhd (ecu_pkg.vhd).

4. Architecture names should reflec the coding style:
   behavior – behavioral description
   rtl – register transfer level description
   structural – structural code
   sim – non synthesizable code

5. Configurations should be named <configuration_name>_cfg, example: xmit_cfg

6. Use 2 spaces for indentation (in place of tab stops). Max. 80 characters per line.

7. Designs shall provide sufficient comments to enable self-documenting Code.

8. Sub module names should contain parent module root name as prefix. E.g. if a module ecu.vhd uses a sub-module incorporating a counter, than the sub-module should be named ecu_counter.vhd.

9. Use explicit port mapping when instantiating sub modules. Use "Named" association instead of "positional" association.

10. Document all processes, signals, and variables. Enforce documentation of major design elements.

11. Instance names have to be <block_unit>_ins.e.g. if name of the unit is cnt_unit, the name of the instance should be cntr_unit_ins.

12. Busses are always declared from high to low
    e.g. data : std_logic_vector (10 downto 0)

13. Do not leave state assignments to the compiler – do it yourself and use one-hot encoding preferable!

14. Standard logic (`std_logic`) used everywhere!

15. Use only IEEE packages in general. Some designers re-create functions present in the IEEE version, save it as a separate package and name the package in their own style. Please avoid this.

16. If you have a top level file that consists of 5 sub-units instantiated under it, I want you to include all the 5 component declarations in one single package.
    Example: If the top_level file called dsp_unit.vhd has 2 sub-components under it (namely, dsp_cnt8.vhd and dsp_cnt16.vhd), you should include the component declarations of these 2 sub-components in a package file called dsp_comps_pkg.vhd. Obviously, you should call the compile and call the

dsp_comps_pkg.vhd file at the top most line (near library declaration area) of tyhe dsp_unit.vhd file.

17. Each port declaration should be on a separate line.

18. All ports in VHDL are in, out, or inout (no buffer).

19. Do not use Verilog, VHDL and SDF keywords for naming signals or variables.

20. Signal and variable names must start with an alphanumeric character – names with a leading number are not allowed!

21. Naming conventions for signals, constants, processes, etc.

    - internal signals use a prefix **s_**
    - variables use a prefix **v_**
    - constants use a prefix **C_**
    - file definitions use a prefix **f_**
    - type definitions use a prefix **t_**
    - process label use a prefix **P_**
    - block label use a prefix **B_**
    - low active signals use the appendix **_n**
    - unit input signals use the appendix **_i**
    - unit output signals use the appendix **_o**
    - unit bidirectional signals use the appendix **_b**

22. Use proper indentaion

not recommended:

```
if (a='0') then
  y <= '1'; else
     y <= '0';
  end if;
```

recommended:

```
if (a='0') then
  y <= '1';
else
  y <= '0';
end if;
```

23. Only integer types are accepted in Generics. Any other type is usually not accepted by the synthesizer.

24. Do not rely on operator precedence.

25. Avoid internal three-state signals.

26. Do not use bussed ports with width one.

27. Do not mix component instantiation and logic description.

28. Partition synchronous and asynchronous logic in different units.

29. Keep levels of hierarchy to a maximum of 3-5. Too many levels of hierarchy become difficult to manage, understand and analyze.

30. No clock gating done at the unit level, all clock gating done at a central unit (Generally, gating the clock is a <u>bad</u> idea. However, in certain cases, designers intentionally gate the clock either to alter the clock's skew or to insert delay or to produce an asymmetrical clock. Even in these justified clock-gating situations, do not do it at the sub-block level. Perform all the clock-gatings at the top-most level of the chip)
    Reasons for gating the clock at top_level:
    - Industry standard process
    - Enables skew and insertion delay control by layout tool
    - Clock gating in lower-level modules provides challenges to synthesis & simulation tools
    - Isolating to a central module enables design team to manage the implementation

31. Clock / reset blocks instanced at top level of core hierarchy.

32. Pads are located at top level of chip.

33. VHDL keywords should be followed by <u>one</u> space.

34. Combinational processes must define all of it's sensitive signals.

35. Clocked processes must define the clock and its activity (rising or falling). They must also define the reset signal –if it exists--, it's activity and relation to the clock.

36. Expressions within parentheses are written with no space after the opening parentheses and no space before the closing parentheses.
e.g.: **`y <= (a and b) or c;`**

37. Error reporting and warnings using the assert statement using severity and report should provide details about where and when it occurred, should give an error ID or name and should give a hint on the cause of the errror.

38. Maintain your VHDL, Script and Documentation files in a clean directory structure. For example, place sub-modules in subdirectories that are named the same way as the sub-modules. E.g. place the sub-module exu.vhd and it's according files in a subdirectory exu.