



CSCE 612 - Spring 2003

Dr. James P. Davis

VHDL Style Guide

Version 2.1 - 14 February 2003

Project Organization

- A file should contain only one design unit. A design unit is defined as any of the following:
 - entity
 - architecture
 - entity - architecture pair
 - package
 - package body
 - test bench
 - configuration (may be combined with one of the above design units if it improves the readability of the design).
- The structure of the file system hierarchy shall mirror the logical structure of the system being modeled.

File Naming Convention

- All files that contain VHDL source should have a .vhd extension.
- Each file should be named according to the design unit(s) it contains:
 - entity => `basename_e.vhd`
 - architecture => `basename_a.vhd`
 - entity & architecture => `basename_ea.vhd`
 - configuration => `basename_cfg.vhd`
 - package => `basename_p.vhd`
 - package body => `basename_pb.vhd`
 - test bench => `basename_tb.vhd`
- The base name should be the same as the design unit name the file contains.

File Prolog

The following prolog should be included at the top of every VHDL source file.

```
-----  
-- File name  
-----
```

```
-- This VHDL code was developed by  
-----
```

```
-- Description  
-----
```

Code Organization

- **Indentation**
 - Indentation should be used to show logical structure of code.

- 3 spaces should be used for each level of indentation.
 - Do not use tab characters.
 - Code should start in the left-most column and be indented for each block.
 - Indentation levels in sequential statements shall not exceed 4. If more levels are required, the design should be separated into more manageable parts.
 - Indented regions in sequential statements shall not have more than 60 lines.
 - Labels should be placed in the left-most column.
- **Comments**
 - Comments are used to provide information that a person could not discern simply by reading the code. The purpose of comments is to allow the function of a design to be understood by a designer not involved in the development of the code.
 - Multi-line comments shall start and end with an empty comment line.
- **Managing File Size**
 - Each file should contain no more than one entity-architecture pair.
 - Lines shall not exceed 80 characters in length. If more than 80 characters are required continue the statement on the next line.
 - There shall be no more than 2 process statements in an architecture. Use a structural decomposition if necessary.
- **White Space**
 - Add white space in the form of blank lines, spaces, and indentation to improve the readability of code.
 - Groups of logically related statements and declarations shall be separated by 1 blank line.
 - Each statement shall start on a new line.
- **Labels**
 - Concurrent statements shall be labeled.
 - Loop statements shall be labeled.
 - Next and exit statements shall specify the loop label they control.
- **Object Naming Conventions**
 - The *name* should be a meaningful, descriptive term for the object. Use underscores to separate multiple words within the name.
 - `_S => signal`
 - `_C => constant`
 - `_V => variable`
 - `_IDX => loop index`
 - `_I => internal signal`
 - `_F => file`
 - `_G => generic`

- `_LIB => library`
 - `_LBL => label`
 - `_A => architecture`
The following may be used in place of `_A` for architectures:
 - ▲ `_BEH => behavioral architecture`
 - ▲ `_RTL => register-transfer level (synthesizable) architecture`
 - ▲ `_STR => structural architecture`
 - ▲ `_CFG` for configuration
 - `_TYP` for user-defined type and subtype identifiers
 - `_PKG` for user-defined package identifiers
- Use lower case letters for all VHDL reserved words, attributes, etc.

VHDL Coding Style

• Entity

The input and output ports and generic parameters of the module can be described in the entity. There can not be more than one entity in a module.

• Generics

Usually generics are declared as integer or natural type values.

• Processes

Processes have a sensitivity list for signals and they execute every time one of the signals in the sensitivity list is assigned a value. In synthesizable code all the functionality of the design is enclosed within processes and the processes are connected to each other by signals. Processes can be divided into sequential or combinatorial processes, where sequential processes are sensitive only to clock and reset signals and combinatorial are sensitive to every signal that is referred to them.

• Signals

Signals are used to transfer values between processes or processes and ports. They can also trigger the execution of different processes (clock, reset). For synthesizable code usually only discrete data types or their composites is recommended to be used.