Golden rules:

1. Keep it simple, stupid (KISS)
2. One .vhd file per one module
3. Use clear indentation

VHDL:

* NOT case sensitive,
* NOT sensitive to white space
* Module names start and end with char
* Signal names start with char

Comments

--This is a VHDL comment  
tmp\_output <= a; --this is also a comment

Constant definitions

**constant** tmp\_i **:** integer **:=** 0**;**

**constant** tmp\_s **:** std\_logic **:=** '0'**;**

**constant** tmp\_s **:** std\_logic\_vector **(**3 **downto** 0**)** **:=** "0101"**;**

Signal definitions

--no init **signal** tmp\_output **:** std\_logic**;**   
**:=** '0'**;** --init as low   
**:=** '1'**;** --init as high

-- no init  **signal** tmp**:** std\_logic\_vector **(**3 **downto** 0**);**

**:=** "0000"**;** --init as array   
**:=** x"0"**;** --init as hex (/4)  
**:=** **(others** **=>** '0'**);** --init wide

Entity declaration

**entity** entity\_name **is**   
**generic** **(** **);**   
**port** **(** **);**   
**end** entity\_name**;**

port directions = {in, out, inout}

**architecture** beh **of** entity\_name **is**

--define inside use signals

--define components to use

**begin**

--operation of the module!

**end** beh**;**

Component declaration / instantiation

In architecture, before **begin**

**component** COMP\_NAME   
**generic** **( );**   
**port ( );**   
**end** **component;**

In architecture, after **begin**

INST\_NAME **:** COMP\_NAME   
**generic** **map**   
**(** generic1 **=>** generic1**)**   
**port** **map**   
**(** port1 **=>** signal1**,**   
 port2 **=>** signal2 **);**

Logic operations (bitwise)

**b\_out <= b\_in; --assignment  
b\_out <= b\_in & b\_in; --concatenate**   
**b\_out <= not b\_in; --not  
b\_out <= a\_in operator b\_in; --and**

**where operator = {and, or, nand, nor, xor, xnor}**

Conditional signal assignments

out **<=** **<**expr**>** **when** **<**cond**>** **else** **<**expr**>** **when** **<**cond**>** **else** **<**expr**>;**

**with** **<**sig\_name**>** **select**   
 out **<=** **<**expr**>** **when** **<**choice1**>,**   
 **<**expr**>** **when** **<**choice2**>,**   
 **<**expr**>** **when** **others;**

Process body

proc\_name **:** **process(**s1**,** s2**,** s3**)**   
--variable declarations (optional)   
**begin**   
  
**end** **process;**

Process - if elsif else

**if** **(**condition**)** **then**   
 **<**expression**>**   
**elsif** **(**condition**)** **then**   
 **<**expression**>  
else**   
 **<**expression**>**   
**end** **if;**

Process - case

**case** **(**signal\_name**)** **is**   
 **when** choice1 **=>**   
 **<**expressions**>**   
 **when** choice2 **=>**   
 **<**expressions**>**   
 **when** **others =>**   
 **<**expressions**>**   
**end** **case;**



