1 ENTITIES AND ARCHITECTURES

✍ 1.1 ENTITIES

Entities describe the core interfaces and generic constants.

Rule

? ENTITY name should exactly describe the functionality of the block.
? Its name should start with an upper case letter.
? The name should not exceed 15 letters.

EG. entity CNTR_4BIT is

✍ 1.2 PORTS

Ports describe the directions of core interfaces.

Rule

? PORT name should follow SIGNAL.

Example:

```
Data_In  : in STD_LOGIC_VECTOR (3 downto 0);   -- I/P Data Lines
Data_Out : out STD_LOGIC_VECTOR (3 downto 0);  -- O/P Data lines
```

? If the PORT describes a port of a standard device or component, its name should follow that standard.

Example :

```
Rts      : in STD_LOGIC;                -- Request To Send I/P
```

? Start with upper case letter and do not exceed 15 characters.
? Each port declaration should be followed by a small comment that describes its uses

✍ 1.3 ARCHITECTURES

Architectures define how the system behaves. This description can be in different levels or for different purposes. Since architectures are tied to entities, their names should describe the method of its description method.

Rule

? ARCHITECTURE names should be ``behavioral", ``structural" or ``rtl".
? Structural and behavioral code should not be mixed.
? In case of multiple architectures for same entity and same description method, add a three letters suffix that describes exactly the difference between the architectures.
? Before the ARCHITECTURE statement a single line comment should describe the uses of this architecture and if it is synthesizable or for simulation only.

Example :

```
---------------------- Synthesizable Core Model For ALU -----------------------------------
---------------------- Stimulus Vector Generator For ALU Core Model -------------------
```

✍ 1.4 COMPONENTS

Components are used to define hierarchal levels of the VHDL design.

Rule

? Start COMPONENT name with either the short name of the PACKAGE or the ENTITY.
? Component instantiations should not exceed 4 levels of hierarchy.

Example:

? Component DIVIDER_CNTR is

✍ 1.5 CONFIGURATIONS

Configurations perform the mapping between components and its corresponding architecture.

Rule

? CONFIGURATION name should explicitly describe the top design name.
? Start with upper case letter and do not exceed 15 characters.

2 PACKAGES, FUNCTIONS AND PROCEDURES

✍ 2.1 PACKAGES

Packages contain all generic design statements.

Rule

? Packages should hold all system-defined constants, data types, components, procedures and functions.
? Start with upper case letter and do not exceed 15 characters.
? Add prefix (**pkg**_) for Packages.

✍ 2.2 FUNCTIONS AND PROCEDURES

Functions and procedures are used to describe set of operations or logic that is going to be repeated in many places in the code.

Rule

? Start with upper case letter and do not exceed 10 characters.
? The name should describe the functionality of the procedure or the function.
  ? Local variables should have the prefix [***l***_].
  ? Local signals should have in its characteristics field.
? Add prefix (**func**_) for Function and (**proc**_) for Procedure.

3 CONSTANTS AND GENERIC

Constants and generics are used to make the design more readable and easy to maintain. Since they define system parameters at one place and by readable names.

Rule

? Constants and generic should be all uppercase letters.
? The name should describe explicitly the usage of the constant.
? Generics must have type integer for synthesis.
? Use suffix (**_g**) for generics.

4 SIGNALS AND VARIABLES

✍ 4.1 SIGNALS

Signals describe wires between components and storage elements.
There are two types of signals according to their location in the code.

1. Connecting signals that connects between components or processes
2. Internal signals that are inside the architectures or processes.

Rule

- The name of the SIGNAL should define its functionality, and should not exceed 15 characters.
- After its definition a single line comment should describe its functionality.
- Connecting signals should have at the beginning of their names the short name of the driving ENTITY, COMPONENT or PROCESS.
- The following attributes should be added to the SIGNAL name to describe its characteristics.
  - SIGNAL activity, high or low. **P** "default" for positive activity and **N** for negative.
  - Global or local, which is important in functions and procedures. **G** "default" for global or **L** for local.
  - Add (**_Z**) for Tri-stated signals.
  - Use suffix [**_S**] for general signals.
  - If the SIGNAL gets its value in a clocked process only it should have the suffix [***_q***] or [***_reg***] if it is a bus.

.

✍ 4.2 VARIABLES

Variables are the sequential storage elements in VHDL.

- VARIABLE name should be simple and should describe its functionality.
- Use suffix [**_V**] for variables.

5 PROCESSES AND BLOCKS

Processes and blocks enable the designers to write sequential statements inside them.

✍ 5.1 PROCESSES

There are two types of processes: clocked and combinational processes.

Rule

- All processes should have names describing the functionality of the process.
- Processes should be labeled.
- The followings should be clarified in the comments section of the PROCESS
- Combinational or clocked PROCESS.
  - Combinational processes should define all signals that it is sensitive to.
  - Clocked processes should define the clock and its activity "rising or falling".
  - Clocked processes should also define the reset signal -if it exists-, its activity and its relation to the clock.

6 TEST BENCHES

Test benches should have special consideration because of their importance in the design flow. A testbench has neither ports nor generics.

Rule

- Test bench entity should have the same name of the ENTITY it tests with the suffix [_***TB***].
- Architectures, Processes, Variables and Signals should follow the same previous rules.
- Memory storage and stimulus generation should be done through functions and procedures.
- Error reporting and warnings (using SEVERITY and REPORT keywords) should provide all details about:
  - Entity or component name
  - Signal or variable name
  - Function or procedure name (if used)
  - Current time
  - Error ID or name
  - Possible cause of the error (Optional)
  - Source of error or warning (RTL, structural or behavioral code)

7 FILES AND DIRECTORY STRUCTURES

7.1 FILES

Rule

- Each entity and all its architectures should be in the same file.
- Package and package body should be in the same file.
- Configuration should be in separate files.
- File names should have the same name of as that of the entity, package or configuration.
- Log files should hold the name of the data to be logged.

7.2 DIRECTORIES

Directories or storage locations is a very important factor to ease the searching and types of files.

Rule

- For each design create 4 main directories. Core, sim, Syn and Library directories.
- The **Core** directory holds all design files and components.
- The **Library** directory holds all packages and external files. This directory can has as much directories inside it one for each used library and it should be named with the name of library.
- The **Syn** directory should hold all output files generated from the synthesis and P&R tool. If the code is targeted to many technologies, each technology should has its own directory and named with its name. Scripts directory should be included to store all scripts for synthesis tool.
- The **Sim** directory should hold 4 main directories. **TB**, which hold the test bench file and any input files or processing files. **Scripts** directory that holds all tools dependent scripts. **RTL or preSYN** directory that holds all results from simulating the RTL or behavioral design. A directory for each structural simulation named the technology used.
- Each version of the design should have the same directory structure as described earlier.

8 FINITE STATE MACHINES

FSM is a system whose outputs depend on the present state of the machine. The present state of the system goes through finite cyclic states.

8.1 SPECIFICATIONS

Rules

- Specify your design requirements properly.
- Draw state diagrams and state tables.
- Take care of invalid states.

8.2 CODING

Rules

- The name of state should imply its functionality.
- Do not use FSM attributes within the VHDL code.

Example:

Type state is (got1, got10, got101); -- States of Sequence Detector

- Name Present state as (PRstate) and next state as (NXstate).
- Write three processes for the design.
- Write comments explaining the functionality of the states.
- Use binary encoding for CPLD target devices.

8.3 TIPS FOR SYNTHESIZABILITY

- Run synthesis early in the design process.
- Don't give default values for ports, signals and variables.
- Use only synthesizable data types.
- Array ranges must be of type integer.
- Integer type objects must be constrained with respect to their range.
- Use standard templates for clocked processes.
- Generics must have type integer.
- For soft-core development use generics, don't use constants.
- Add an **"others"** statement to the FSM-switch.
- All conditional assignments should be checked for completeness.
- The sensitivity list for combinational processes must be complete.
- Use gate instantiation only at few instances.
- Take care for requirements of resource sharing.

9 "PREFERENCES" FOR HDL EDITOR

- Use "Fixedsys" font with size '9'.
- Tabulation = 4.
- Use default Colour setting of XILINX HDL editor.