**FPGA from Zero to Hero**

**Repository Linting Rules**

* Start your codes with the following template

A black square with green lines

Description automatically generated

* Conform to the standard library usage. For VHDL, only 3 lines below other than custom libraries and packages

**library IEEE;**

**use IEEE.std\_logic\_1164.all;**

**use IEEE.numeric\_std.all;**

* Avoid codes, files or folder names with spaces or uppercases, use underscores and lowercase letters
* Use uppercase names only for IEEE, generics, constants, enumerated types and attributes
* Use clk\_, rst\_ prefixes for signals
* Use \_tb.vhd \_pkg.vhd suffixes for testbench and package files
* Do not use unnecessary prefixes and suffixes for signals such as c\_, s\_, \_i, \_o, \_c, \_s, tb\_, \_tb etc.
* Check indentation rules inside the code to improve design readability, replace tab with 3 spaces in your editor
* Do not exceed 200 characters (including spaces) per line
* Avoid writing very long testbenches, prefer packages instead
* Avoid the signals which do not have any relation but are used in the same process and use them in separate processes
* Remove signals and variables that are never written or read, and delete them
* Give high importance to code comments (definition/template at the top of code, explanation at the beginning of each process)
* Check if case statements cover every state defined, avoid dead/unnecessary states
* Reset control signals and outputs properly (data valid signals, FSM signals, output signals etc.)
* Check that signals and ports of HDL entity are initialized (only control signals and output ports are to be initialized with a reset)
* Discover the null range problems where the vector size of one of the arguments is strictly larger than the other’s
* Avoid duplicate file names (for instance many “frame\_writer.vhd” files in the repository) and entity name mismatches and clean less preferred ones
* Be careful with clock domain crossings and apply proper CDC solutions (2FF/3FF/handshake/FIFO etc.) if CDC is inevitable
* Analyze reset assertions/de-assertions if asynchronous reset is used, If synchronous. reset is used and it checks its proper usage

***Note 1: Linting Tool (LT) is a kind of automatic review. LT shall be supported by manual reviews (a.k.a code reviews/peer reviews) so that each code file in a company reaches a good level of quality***

***Note 2: In case of a Linting Rule Exception explain it with a comment in the code and persuade the reviewers about that***