EmLogic

VHDL Conventions

General rules

Use English for all code, names and comments

Never use reserved words (VHDL, Verilog) as names

Write names that are meaningful to anyone

Separate words with underscore. (But skip underscore where English may use hyphen)

Single character before first underscore or after last underscore is not allowed other than for reserved prefixes or suffixes (or if dictated by ext. IP)

Make lots of good comments (On Why, - and normally not on What)

Always write a header for code segments (incl. processes, loops and multiple lines doing a common funct.)

Labelling is mandatory for processes, generates, blocks,
- Closing labels are also mandatory

Use positive logic only (i.e. not cs_n or similar - other than as connection to external IP or PCB)

Use positive naming when feasible (enable rather than disable)

Compound Functional names should normally be written with the most important subname first (e.g. uart_tx)
- Exceptions to this rule could be made for clocks, resets and interrupts; - like clk_rtc and irq_uart_tx_ready. Sometimes it also makes sense to write the interface name first - if a signal is clearly a part of that interface (e.g. axi_clk, but normally not similar for SBI)

For signals going between modules, use one of
- <name>_<source>2<dest>
- <name>2<dest>
- <source>_<name>

lowercase / UPPERCASE

Lowercase to be used for all code apart from
- Uppercase for constants defined in Entities and Packages
- Uppercase for constants in procedures
- Uppercase for Generic Constants
- Uppercase for enumerated literals

Underscores should be used for clarity

Code layout

Use sectioning and space to increase readability

Use a tabular layout, aligning groups of statements

Entity and Instance naming

Module entity Stand alone function	`uart`
FPGA top level entity	`<the FPGA name>`
Submodule entity Either functional name or use module-name as prefix	`baudrate_ctrl, uart_rx`
Instance Use functional name as suffix as default. May also prefix by i[#]_ as an alternative. *(Exception for generate)*	`i_uart, i3_uart, i_uart_host_if, i_fifo_addresses, i_cmd_queue`

Testbenches and harness
a) TB for uart
b) TB for uart_rx
c) TB for testing RX in uart
d) TB with special purpose
e) Test harness for the same as above
	`a) uart_tb`
	`b) uart_rx_tb (uart_<func inside uart>_tb)`
	`c) uart_tb_rx (uart_tb_<func to test in full uart>)`
	`d) uvvm_tb_demo`
	`e) same as above but ‘th’ rather than ‘tb’`

VIP variants	`uart_vvc, uart_tx_vvc`

Architecture naming

Functional / RTL (including hierarchical): Multiple architectures:	`rtl` `rtl_func`
Behavioural	`bhv - or other function name (`**`not`**` RTL)`
Testbench architecture or harness	`Functional name: e.g. func, corner, tx, rx_test`
Special purpose (e.g. netlist, low power, device spesific)	`Any meaningful name (rtl_<func>) for RTL`

Package naming

Module or entity spesific	`uart_pkg, uart_pif_pkg, uart_pif_priv_pkg`
	`(Use ‘priv’ if private. Default is public)`

General packages	`<name>_pkg (e.g. common_methods_pkg)`
VIP variants	`uart_bfm_pkg` `vvc_methods_pkg`

Library naming

Company dedicated library	`emlogic_supports_comps emlogic_space_wire emlogic_vip_space_wire`
Project dedicated library	`<module/functionality-name> or [<project-name>_]<module/functionality-name>`
Vendor dedicated library	`<venfor-name>_name, unless already given a library name`

File naming

Default	`<most primary unit in file>.vhd (see Entity name)`
	`uart.vhd, uart_rx.vhd, uart_tb.vhd`

If additional architectures	`Entity: <entity-name>_ent.vhd`
	`Arch: <entity-name>_<arch-name>.vhd`
	`e.g. uart_ent, uart_bhv.vhd, uart_rtl_low_power.vhd`

Packages	`uart_pkg.vhd e.g. common_methods_pkg.vhd`

Type range restrictions

Vector (any kind)	`Range N downto M, (M=0 or justify other)`
Single dimensional array	`Same as vector, so also name <element>_vector, always range N downto M. Exception for 'string', which is normally range 1 to N. Other exceptions using range N to M should be named <element>_array`
Multi dimensional array	`Use good explanatory names. Dimension 2 is often range N to M, but a word-array (of SLV) would often be N downto M`
Number (any kind)	`Must define range`

Type usage restrictions

std_logic_vector	`Never use if object is always representing a number`
unsigned	`Use for ALL objects representing an unsigned number (unless natural is better)`
signed	`Use for ALL objects representing a signed number (unless integer is better)`
Integer, natural, positive	`Typically use for indexes and pointers ONLY use when really well understood. Always restrict range. Never use for primary I/O (for synthesis). Use strictest possible type (i.e. positive if only positives if using unconstrained)`
Enumerated, Boolean, Records	`Use anywhere, but never use for FPGA primary I/O (for synthesis)`

Prefixes - For Signal, Var., Const., etc.

Signal	`<name> (no prefix)`
Global signal def. in pkg	`global_<name> (May skip 'global_' for very well known names, like VVC signals in UVVM)`

Variable	`v_<name> (def. in process NOT register)`
	`vr_<name> (def. in process. Intended register)`
	`<name> (formal parameter in subroutine)`
	`v_<name> (def. inside protected type)`

Shared varrable (of protected type) Unprotected is no longer allowed	`shared_<name>`

Constant	`C_<NAME> (normal constants)`
	`<name> (formal param. subroutine)`
	`C_<NAME> (defined in subroutine)`

Generic constant	`GC_<NAME>`
Enumeration literals	`[<TYPE>_]<NAME> (TYPE = User type name) or S_<NAME> (recommended for FSM, but not mandatory. Typically very useful when referring to states from outside the FSM, but less useful for plain and simple FSMs)`
Alias	`a_<name>`
Alias Hierarchical reference	`ha_<name>`

Register address constants locally	`C_ADDR_<reg-name>`
	`e.g. C_ADDR_ERROR_FLAGS`

Register address constants Icentrally/top-level	`C_ADDR_<module-name>_<reg-name>`
	`e.g. C_ADDR_UART1_ERROR_FLAGS`

Address offset	`C_ADDR_OFFSET_<module-name>`
	`e.g. C_ADDR_OFFSET_UART1`

Prefixes - other

Process	`p_<name>`
Procedure	`<name>`
Function	`<name> (min 1 param otherwise use 'VOID')`
Type	`t_<name>`
Generate	`g_<name>`
Loop label (optional)	`l_<name>`

Suffixes

Intended purely as extra info for signals/variables/const. If multiple suffixes apply, add them in alphabetical order (e.g. <name>_a_n) Note a number is mandatory for their given # below

Active low avoid	`-n`
Asynchronous	`_a`
Synchronized	`_s# (1..N)`
Delayed i.e. all in the same block domain	`-d# (1..N)` (may also sometimes use sr for shift reg.)
pipeline stage _p#	`_p# (1..N)` (May also sometimes use sr for shift reg.)
Differential pair	`_dp and _dn`
Toggle-signal to indicate valid on toggle, i.e. not a boolean signal	`_tgl`

Fixed names and abbreviations

clk **clk_<funct-name>**	'clk' may only be used for signals going to flop or memory clock inputs. E.g. a clock just going out of the FPGA to for instance an external DAC should NOT use 'clk' in the name, but rather for instance 'clock'
clock_<funct-name> **<funct-name>_clock**	Never use this for a signal going to a flop or memory clock inputs (see clk above)
rst, arst **rst<_clk-name>, rst_30, rst_uart** **<interface prefix>_rst**	Reset (rst: synchronous, i.e. not immediate) (arst: asynchronous reset, i.e. immediate) See 'Compound Functional names' under 'General rules' at the beginning

Allowed abbreviations

ack	acknowledge
addr	address
c2p, p2c	Signals from core to pif, or pif to core respectively (e.g. inside a module with records between PIF and core)
clr	clear
cmd	command
cnt	count(er) (Actual count value. cd idx)
ctrl	control(ler)
dest	destination
din, dout	data in, data out

Allowed abbreviations

ena	enable
err	error
idx, idx1	index (Use idx when first element 0, otherwise idx1 when first element is 1)
irq	interrupt / interrupt request
lsb, lsw	least significant bit/word (for byte use lsbyte)
msb, msw	most significant bit/word (for byte use msbyte)
num	number (of), (do not use ‘no’)
pif	processor interface
ptr	pointer
rd, wr / rena, wena	read, write (Either set may be used, depending on scenario)
rdata, wdata	data in, data out (same as ‘din, dout’, but used in different scenarios)
rdy, vld	ready, valid
src	source
sync, async	synchronous, asynchronous.red
tb, th, tc	Suffixes for test-bench/harness/case
tmp	temporary
tx, rx	transmit/recieve (Use more explanatory name when any misunderstanding is possible. E.g. use uart_1_tx, tx_moduleA2moduleB, tx_uart1_to_uart3)