# Bitvis VHDL Conventions – Quick Reference

## General

- 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 no underscore where English may use hyphen)
- Single char pre/post underscore are not allowed other than for reserved prefixes or suffixes (or if dictated by ext. IP)
- Make lots of good comments (Why, not detailed what)
- Use positive logic only (i.e. not cs_n)
- Use positive naming when feasible (enable rather than disable)

## Lowercase / Uppercase

Lowercase to be used for all code apart from:

- Uppercase for constants defined in Entities and Packages
- Uppercase for Generic Constants
- Uppercase for enumerated literals

## Code Layout

- Use sectioning and space to increase readability
- Use indent of 2 and a line length of 130
- Indent wrapped lines by 4 – or align on common elements.
- Use a tabular layout, aligning groups of statements

## Package naming

| | |
|---|---|
| Module or entity specific | *uart_pkg,*<br>*uart_pif_pkg* |
| General packages | *verif_methods_pkg* |
| VIP variants | *uart_bfm_pkg*<br>*uart_executor_pkg* |

## Configuration naming

| | |
|---|---|
| Module configuration | *uart_cfg,*<br>*uart_cfg_altera* |
| Testbench configurations | *uart_tb_cfg* |

## Entity & Instance naming

| | |
|---|---|
| Module entity: Stand-alone function | *uart* |
| Submodule entity: Either functional name or suffix on module-name | a) *baudrate_ctrl*<br>b) *uart_rx* |
| Instance: Prefix by i[#]_ as default, but use functional name if more informative | a) *i_uart, i3_fifo*<br>b) *i_cmd_queue* |
| Testbenches: a) TB for uart<br>b) TB for uart_rx<br>c) TB for testing RX in uart | *a) uart_tb,*<br>*b) uart_rx_tb*<br>*c) uart_tb_rx* |
| VIP variants | *a) uart_vvc* |

## Architecture naming

| | |
|---|---|
| Functional / RTL (and hierarch w/ RTL) | *rtl* |
| Behavioural | *behave* |
| Testbench architecture (simple) | *e.g. func, corner, tx*<br>*rx_test* |
| Testbench architecture (using harness)<br>(When splitting into test-bench, -harness and -cases) | *tb_<name>*<br>*th_<name>*<br>*tc_<name>* |
| Special purpose (e.g. netlist, low power, device specific | Use any meaningful name |

## File Naming

| | |
|---|---|
| Default: <most primary unit in file>.vhd | *uart.vhd*<br>*uart_tb.vhd* |
| Additional arch. (no entity) | *uart_behave.vhd* |
| Packages | *uart_pkg.vhd*<br>*verif_methods_pkg.vhd* |

## Library naming

| | |
|---|---|
| Company dedicated library | *bitvis_irqc*<br>*bitvis_vip_axi* |
| Project dedicated library | *<company_name>_<project-name>[_<module-name>* |

## Type range restrictions

| | |
|---|---|
| Vector (any kind) | *N downto M, (M=0 or justify other)* |
| 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*<br>*ONLY use when really well understood.*<br>*Always restrict range.*<br>*Never use for primary I/O (for synthesis)* |
| Enumerated, Boolean, Records | *Use anywhere, but*<br>*Never use for primary I/O (for synthesis)* |

## Prefixes – for Signal, Var., Const., etc

| | |
|---|---|
| signal | *<name>  ( no prefix)* |
| global signal (def. in pkg) | *global_<name>* |
| variable (def. in process. NOT a register) | *v_<name>* |
| variable (def. in process. Intended register) | *vr_<name>* |
| variable (formal param. subroutine)) | *<name>  ?v_<name>??* |
| variable (def. in subroutine) | *v_<name>* |
| variable (def. in protected type) | *priv_<name>* |
| shared variable (unprotected) | *shared_<name>* |
| protected variable | *protected_<name>* |
| constant (def. in package) | *C_<NAME>* |
| constant (def. in entitiy) | *C_<NAME>* |
| constant (formal param. subroutine) | *<name>* |
| constant (def. in subroutine) | *C_<NAME>* |
| generic constant | *GC_<NAME>* |
| enumeration literals | *[TYPE_]<NAME>*<br>*S_NAME  (for FSM)* |
| alias | *<name>* |

## Register naming  (see RegWiz for details)

| | |
|---|---|
| Address constants (locally) | *C_ADDR_<reg-name>*<br>*E.g. C_ADDR_ERROR_FLAGS* |
| Address constants (centrally/top-level) | *C_ADDR_<module-name>_<reg-name>*<br>*E.g. C_ADDR_UART1_ERROR_FLAGS* |
| Register signal (normally inside record to/from core) | *[[a]<access_type>_]<reg-name>*<br>*a:if auxiliary (i.e. no flops in pif)*<br>*E.g. rw_ier, awt_icr, ro_error_flags* |

## Prefixes – for other language elements

| | |
|---|---|
| process | *p_<name>* |
| procedure | *<name>* |
| function | *<name>(min 1 param)* |
| type  a) default<br>b) 1-D natural array<br>c) multi-D natural array<br>d) any other array | *a) t_<name>*<br>*b) t_<name>_vector*<br>*c) t_<name>_array*<br>*d) t_<name>_<name2>* |
| generate | *g_<name>* |
| loop label (optional) | *l_<name>* |

## Suffixes

NOTES: Intended purely as extra info for signals/variables/const
Multiple suffixes → in alphabetical order.  (e.g. _i_n)

| | |
|---|---|
| internal version of entity output (when needed) | *_i* |
| active low (avoid) | *_n* |
| asynchronous | *_a* |
| synchronized | *_s#* |
| delayed (i.e. all in same clock domain) | *_d#* |
| pipeline stage | *_p#* |
| differential pair | *_dp & _dn* |

## Fixed names (in functional alphabetical order)

| | |
|---|---|
| rw, ro, wo | *access types (read/write, read only, write only)* |
| ack | *acknowledge* |
| addr | *address* |
| c2p, p2c | *core to pif, pif to core (inside a module)* |
| clk | *clock* |
| cnt | *count(er)* |
| ctrl | *control(ler)* |
| dest | *destination* |
| din, dout | *data in, data out* |
| ena | *enable* |
| idx | *index (normally counting from 0)* |
| irq | *interrupt* |
| lsb, lsw | *least significant bit/word* |
| msb, msw | *most significant bit/word* |
| num | *number (of), (do not use 'no')* |
| pif | *processor interface* |
| ptr | *pointer* |
| rd, wr | *read, write* |
| rdy | *ready* |
| re, we | *Read enable, Write enable* |
| rst, arst | *Reset (rst: synchronous, i.e. not immediate)*<br>*(arst: asynchronous reset, i.e. immediate)* |
| src | *source* |
| sync, async | *synchronous, asynchronous* |
| tb, tc, th | *prefixes for test-bench/harness/case* |
| tmp | *temporary* |