Verilog Naming & Formating Guide

1 General Rules

Type	Convention	Example
Case Style	Use `lower_snake_case` for signals, variables, and ports	`data_write`
Constants / Parameters / Defines	Use `UPPER_SNAKE_CASE`	`STATE_IDLE, DATA_WIDTH, ADDR_SIZE`
Modules	Use lowercase, with underscores for clarity	`alu, uart_tx, fifo_ctrl`
Files	File name matches module name exactly	`alu.v, fifo_ctrl.v`
Testbenches	Append `_tb.v` to the module name	`alu_tb.v`
Instances	Use short lowercase names that describe the function	`alu0, fifo_a, mux_core`
Packages / Include Files	Use descriptive lowercase names	`defines.vh, params.vh`

2 Ports and Signals

Type	Convention	Example
Clocks	Use `clk` or `clk_<domain>`	`clk, clk_core, clk_slow`
Resets	`rst` or `rst_n` (for active-low)	`rst_n, rst_n`
Enable Signals	End with `_en`	`wr_en, rd_en, tx_en`
Inputs	End with `_i`	`data_i, addr_i`
Outputs	End with `_o`	`data_o, ready_o`
Active Low Signals	End with `_n`	`reset_n, cs_n`

3 Internal Signals / Registers

Type	Convention	Example
Registers	End with _r	`count_r, data_r`
Wires	End with _w	`count_w, next_state_w`
State Machines	Prefix with `st_`	`st_idle, st_busy, st_done`
Next-state Signals	End with _next	`state_next, count_next`
Arrays	End with `_arr`	`register_arr, mem_arr`
Temporary	End _tmp	`sum_tmp, carry_tmp`

4 Hierarchical Naming

Type	Convention	Example
Top-level module	Reflects project or subsystem	`cpu_top, rca_top`

5 Miscellaneous

Category	Convention
Comments	Use `//` for short comments and `/**/` for long comments
Include Files	Use `.vh` for definitions
Macros	Use clear all-caps names

6 Code Formatting and Indentation

Aspect	Convention
Indentation	Use **4 spaces per indentation level**. Do not use tabs.
Line Length	Keep lines under **80 characters** when possible. Break long statements into multiple lines.

Aspect	Convention
Block Alignment	Align `begin` and `end` with the corresponding `if`, `case`, `for`, or `always` statement.

```
// Good
always @(posedge clk) begin
    if (enable_i) begin
        data_r <= data_next;
    end
end

// Bad
always @(posedge clk) begin
if (enable_i) begin
data_r <= data_next;
end
end
```

Spacing	Place one space after commas and around operators (e.g., `a = b + c;`). Do not insert spaces before commas or semicolons.
Braces and Keywords	Write `begin` and `end` on separate lines to clearly mark code blocks.

```
// Good
if (enable_i) begin
    data_r <= data_next;
end

// Bad
if (enable_i) begin data_r <= data_next;
   end
```

Port Lists	Use one port per line in module declarations, aligned vertically for readability. Example:

```
module alu (
    input  wire [7:0] a_i,
    input  wire [7:0] b_i,
    output wire [7:0] sum_o
);
```

Signal Declarations	Group related signals together and order them as: inputs, outputs, internal wires, and registers.

Aspect	Convention

Comment Placement	Place brief comments on the same line for short explanations, or above the code block for longer notes. Example:
	`// Increment counter on rising clock edge`
	`always @(posedge clk) begin`
	`  count_r <= count_r + 1;`
	`end`

Blank Lines	Use blank lines to separate logical sections of code (e.g., declarations, state machines, assignments).

File Header	Each file should begin with a comment header including module name, description, author, and date. Example:
	
	`/*******************************`
	`*`
	`* Module: alu.v`
	`* Project: SimpleCPU`
	`* Author: Ahmed Hassan (ahmed.h@example.com)`
	`* Author: Lina Omar (lina.o@example.com)`
	`* Description: 8-bit Arithmetic Logic Unit`
	`*`
	`* Change history:`
	`*  01/10/25 { Created initial version (Ahmed)`
	`*  03/10/25 { Added subtraction and logical ops (Lina)`
	`*  10/10/25 { Cleaned up formatting and comments (Ahmed)`
	`*`
	`*************************/`