02 · Verilog HDL & RTL Basics

Module skeleton

```
module name #(parameter int W = 8) (
  input  logic         clk, rst_n,
  input  logic [W-1:0] d,
  output logic [W-1:0] q
);
endmodule
name #(.W(16)) u_name (.clk(clk),
.rst_n(rst_n), .d(d), .q(q));
```

Nets and variables

Type	Driven by	Note
`wire`	`assign`, port	multiple drivers resolve
`reg`	`always`, `initial`	not a register by itself
`logic`	either, one driver	use in new code

Values: `0 1 X Z`. Compare with `===` / `!==` in testbenches to catch X and Z.

Operators

Kind	Operators
Bitwise	`& | ^ ~ ~^`
Logical	`&& || !`
Reduction	`&a |a ^a` (one bit out)
Shift	`<< >>` logical, `<<< >>>` arithmetic
Concat / repeat	`{a, b}` · `{4{2'b10}}`
Case equality	`=== !==` (X and Z match exactly)

Three kinds of hardware

Hardware	Code	Assign
Simple comb.	`assign y = …;`	continuous
Comb. with if/case	`always_comb` / `always @*`	`=`
Flip-flops	`always_ff @(posedge clk)`	`<=`

Race-free rules

1. Sequential: `<=` only.
2. Combinational: `=` only, default every output first.
3. One variable, one `always` block.
4. Never mix `=` and `<=` on a variable.
5. Testbench drives inputs on `negedge` or with `<=`.

Scheduling regions

Active → Inactive (`#0`) → NBA (`<=` updates) → Observed (assertions) → Reactive → Postponed (`$strobe`, `$monitor`).

`$display` after `<=` shows the old value; `$strobe` shows the new one.

FSM template

```
always_ff @(posedge clk or negedge rst_n)
  if (!rst_n) state <= IDLE; else state <=
next;
always_comb begin
  next = state;
  case (state) IDLE: if (go) next = RUN;
endcase
end
```

Testbench toolkit

Need	Code
Clock	`always #5 clk = ~clk;`
Wait cycles	`repeat (n) @(posedge clk);`
Wait for signal	`wait (!busy);`
Print	`$display("%0d %h %b", …)`
Random	`$urandom`, `$urandom_range(lo,hi)`
Stop	`$finish;`
Waves	`$dumpfile("d.vcd"); $dumpvars;`

UART 8N1 frame

Idle 1 → start 0 → D0 … D7 (LSB first) → stop 1. Each bit lasts `CLKS_PER_BIT` clocks; sample in the middle.

Remember it like this

Idea	Picture
Verilog	a building plan, not a recipe: every line exists at once
Module / ports	a charger: you only see the plug and socket
Instance	one LEGO brick placed in a bigger build
wire / reg	a pipe / a bucket
X / Z	smudged reading / unplugged cable
`=`	passing a note along a row
`<=`	a class photo: everyone frozen at one click
Latch	a door with no spring
Testbench	a car on a garage test bench
UART monitor	read each bit in the middle of its box, LSB first