Verilog Cheatsheet

Prepared by: Garima jangid

1.Basic structure

```
module module_name (input1, output1, ...);
  input input1;
  output output1;
  reg r1;
  wire w1;
  // Design logic here
endmodule
```

2. Data Types

Type	Description
wire	Represents a connection (combinational)
reg	Holds value (for procedural blocks)
integer, real	Used for simulations

3. Always Block

```
always @(posedge clk) begin
  // Sequential logic
end
always @(*) begin
  // Combinational logic
end
```

4. Operators

Type	Symbols
Arithmetic	+, -, *, /, %
Logical	&&, `
Bitwise	&, `
Relational	==, !=, >, <
Shift	<<, >>
Ternary	cond ? true : false

5. Timing Control (Simulation only)

```
#10;    // wait for 10 time units
@(posedge clk);  // wait for clock rising edge
```

6. Control Structures

```
if (condition)
  statement;
else
  statement;

case (sel)
  2'b00: out = a;
  2'b01: out = b;
  default: out = 0;
endcase

for (i = 0; i < 4; i = i + 1)
  statement;
```

7. Modules & Instantiation

```
// Define
module adder(input [3:0] a, b, output [4:0] sum);
  assign sum = a + b;
endmodule

// Instantiate
adder a1(.a(x), .b(y), .sum(out));
```

8. Initial & Testbench

```
initial begin
  clk = 0;
  forever #5 clk = ~clk;
end
initial begin
  // stimulus
  rst = 1; #10;
  rst = 0; #100;
  $finish;
end
```

9. File I/O (for simulation)

```
$readmemb("data.txt", mem);
$display("Value = %b", val);
$monitor("At %t: val = %d", $time, val);
```

10. Bit Selection & Concatenation

```
a[3]    // Select bit 3
a[3:0]  // Select bits 3 down to 0
{a, b}  // Concatenate a and b
```

11. Useful Directives

```
`timescale 1ns/1ps
`define WIDTH 8
`include "defs.v"
```

12. Blocking vs Non-blocking

Type	Symbol	Usage
Blocking	=	Executes sequentially (combinational)
Non-blocking	<=	Executes in parallel (sequential)

```
// BAD EXAMPLE (in sequential logic)
a = b;
b = a;  // Wrong!

// GOOD EXAMPLE
a <= b;
b <= a;
```

13. Parameters & localparam

```
parameter WIDTH = 8;      // Can be overridden at
instantiation
localparam MAX = 255;     // Constant, cannot be
overridden

reg [WIDTH-1:0] data;
```

14. Generate Block (Parameterized Modules)

```
genvar i;
generate
  for (i = 0; i < 4; i = i + 1) begin : gen_block
    and a1 (out[i], in1[i], in2[i]);
  end
endgenerate
```

15. Memories (ROM, RAM)

```
reg [7:0] memory [0:255];  // 256 x 8-bit memory

// Write
always @(posedge clk)
  memory[addr] <= data_in;

// Read (combinational)
assign data_out = memory[addr];
```

16. Testbench Essentials

```
module tb;
  reg clk, reset;
  wire [3:0] out;

  counter uut (.clk(clk), .reset(reset), .out(out));

  initial begin
    clk = 0;
    forever #5 clk = ~clk;
  end

  initial begin
    reset = 1; #10;
    reset = 0;
    #100;
    $finish;
  end
endmodule
```

17. State Machine Example

```
typedef enum logic [1:0] {IDLE, LOAD, EXECUTE,
DONE} state_t;
state_t state, next;

always @(posedge clk or posedge rst) begin
  if (rst)
    state <= IDLE;
  else
    state <= next;
end

always @(*) begin
  case (state)
    IDLE:    next = LOAD;
    LOAD:    next = EXECUTE;
    EXECUTE: next = DONE;
    DONE:    next = IDLE;
  endcase
end
```

18. Signed vs Unsigned Arithmetic

```
reg signed [7:0] a, b;
wire signed [8:0] result;

assign result = a + b;
```

19. System Tasks (Simulation Control)

Task	Purpose
$display(...)	Print once
$monitor(...)	Continuously print changes
$dumpfile(...)	VCD file for waveform
$dumpvars(...)	Dump variable data
$finish	End simulation

```
initial begin
  $dumpfile("wave.vcd");
  $dumpvars(0, tb);
end
```

20. Macros & Includes

```
`define WIDTH 8
`include "my_defines.vh"

reg [`WIDTH-1:0] data;
```

21. Conditional Compilation

```
`ifdef DEBUG
  $display("Debug Info: value = %d", value);
`endif
```

22. Package-like include Files (optional)

In pure Verilog (not SystemVerilog), use include for constants/macros:

```
// file: constants.vh
`define CLK_PERIOD 10

// file: design.v
`include "constants.vh"
```