

Verilog Bare Essentials (for beginners)

1. Any signal assigned inside always block should be of type reg. If it is an output of the module, it should be redeclared as reg.
2. Sequential logic is coded using always statement.

Example : always @(posedge clk or posedge reset) begin

```
    if (reset)
        Q <= 0;
    else
        Q <= D;
End
```

Note that reset is a must. If there is no reset signal to initialize the registers, you cannot determine the initial condition.

3. Use non-blocking statements (<=) to code sequential logic and blocking assignments to code combinational logic (=).
4. Procedural assignments can only be made inside an always block (ex: if, switch case)
5. Even if you are coding combinational logic inside always block (which means the always block is not triggered by a clock signal), you need to define the signals as type reg.
6. Continuous assignment statements (those outside always block) should be preceded by the "assign" keyword

```
assign y = a&b;
```

7. You can assign a register variable inside only one always block and not in any other always block or elsewhere.
8. Not every statement is synthesizable (example : initial , loops etc).
9. You can split the code into modules. When instantiating a module, follow this format

```
Module_name instance_name (connections)
```

10. The test bench is another verilog code where you will instantiate your design. The test bench need not be synthesizable.
11. Most importantly, remember that your code should be synthesizable. It is not difficult to visualize how your code will be translated to logic after synthesis.